

Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

Implementation of a PCIe Encryption Module for Ip-over-Satellite Modem with support of Open-Encryption Interface

Bc. Ondrej Ille

Supervisor: Doc. Ing. Jiří Novák
Field of study: Cybernetics and robotics
Subfield: Sensors and Instrumentation
May 2017

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ille** Jméno: **Ondrej** Osobní číslo: **393186**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra měření**
Studijní program: **Kybernetika a robotika**
Studijní obor: **Senzory a přístrojová technika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Implementace PCIe šifrovacího modulu pro satelitní IP modem s podporou rozhraní Open-encryption

Název diplomové práce anglicky:

Implementation of a PCIe Encryption Module for Ip-over-Satellite Modem with Support of Open-Encryption Interface

Pokyny pro vypracování:

1. Navrhněte a implementujte PCIe modul pro HW akcelerované šifrování.
2. Zabezpečte výrobu prototypu, oživte jej spolu se základní deskou a ověřte funkčnost komunikačních rozhraní.
3. Upravte existující ovladače OS Linux pro platformu Cavium MIPS a pokud to bude nezbytné, rozšiřte jejich funkcionalitu.
4. Navrhněte otevřené rozhraní a protokol pro správu klíčů a integrujte jej do protokolového zásobníku.

Seznam doporučené literatury:

- [1] Johnson H., Graham, M.: High-Speed Digital Design, Prentice Hall 1993
- [2] Corbet, J., Rubini, A., Kroah-Hartman, G.: Linux Device Drivers, O'Reily 2005
- [3] Schroder, C.: Linux Networking Cookbook, O'Reily 2008

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jiří Novák Ph.D., K 13138 - katedra měření

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **21.12.2016**

Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce: **30.09.2018**

Podpis vedoucí(ho) práce

Podpis vedoucí(ho) ústavu/katedry

Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to thank all my colleagues from Research and Development Department of ND Satcom for cooperation during the development of this thesis. Namely Artur Hauger and Werner Staerk for hardware development, Bernd Jungbludt for supervision and mental support, Bruno Schrade for advice on Linux drivers and discussions about real-time operating systems and Philipp Schönberger for help with the SW integration. I would also like to thank my supervisor Doc. Ing. Jíří Novák who provided useful consultations during the development of this thesis. I also would like to thank Ing. Vít Záhlava Csc. for precious consultations about PCB layout. Last but not least, I would like to thank my family for support during whole studies.

Declaration

This thesis is an intellectual property of ND SatCom GmbH, Graf-von-Soden-Strasse, D-88090 Immenstaad am Bodensee, Germany. All possible commercial applications are prohibited without agreement of ND Satcom. The redistribution of this thesis is allowed without confidential information stated in Appendix, according to Czech law, 111/1998 §47b. The full version is available only to the closed circle of employees of Czech Technical University who are required to see the thesis for the purpose of its submission and defense. All design materials and source code are confidential to ND Satcom.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 20 May 2017

Abstract

Master thesis deals with the design and implementation of the encryption system for satellite router to the state of functional prototype. The encryption itself is implemented in dedicated CPU with hardware accelerator. The system contains hardware part concerning with design and manufacturing of PCIe module and software part implementing and integrating necessary software modules. The whole design is connected to IP stack of satellite router Skywan 5G and it is ready for promising commercial applications.

Keywords: encryption, PCIe, COMe, Linux, Satellite communications, Skywan 5G

Supervisor: Doc. Ing. Jiří Novák

Abstrakt

Diplomová práca sa zaoberá návrhom a implementáciou šifrovacieho systému pre satelitný router až do stavu fungujúceho prototypu. Šifrovanie samotné je riešené pomocou dedikovaného CPU s hardwarovým akcelerátorom. Systém je realizovaný na úrovni hardware v podobe návrhu a výroby PCIe modulu a tiež na úrovni software v podobe implementácie a integrácie potrebných modulov. Celý návrh je zapojený do protokolového zásobníku IP satelitného routera Skywan 5G a je pripravený na sľubné komerčné použitie.

Kľúčová slova: šifrovanie, PCIe, COMe, Linux, Satelitná komunikácia, Skywan 5G

Překlad názvu: Implementace PCIe šifrovacího modulu pro satelitní IP modem s podporou rozhraní Open-encryption

Contents

Abbreviations	1	4.3 Encryption sequence	65
1 Introduction	5	4.4 C291 CLI	66
1.1 Skywan 5G	5	4.5 Configuration of the SW	66
1.2 Encryption requirements	6	4.6 Porting of the C291 driver	67
1.3 Possible realization	7	4.6.1 Per-CPU variables	67
1.3.1 Internal module	8	4.6.2 Memory allocation	68
1.3.2 External module	8	4.6.3 Reorganisation of structures	69
1.4 Background of Cryptology	9	4.6.4 Real-time kernel	70
Symmetric vs. asymmetric key ciphers	9	4.7 Connection of encryption framework to Linux networking engine	70
Block vs. stream ciphers	9	5 Testing and conclusion	73
1.4.1 Cryptoanalysis	10	5.1 Thermal performance and power consumption	74
1.4.2 Algorithms of interest	10	5.2 Manual encryption tests	74
2 Hardware design	17	5.3 Response time	76
2.1 System level design	19	5.4 Packet forwarding test over TDMA	76
2.1.1 CPU selection	19	5.5 Conclusion	79
2.1.2 System configuration and status	23	Appendix A - Schematic	81
2.1.3 Power supply design	24	Appendix B - Component placement	83
2.1.4 Thermal design	25	Bibliography	85
2.1.5 Clock design	26		
2.2 Tool selection	26		
2.3 Schematic design	29		
2.3.1 Power supply	30		
2.3.2 Power sequencing	34		
2.3.3 Ethernet transceiver	35		
2.3.4 Clock circuits	37		
2.3.5 DDR3	37		
2.3.6 PCIe	37		
2.3.7 Termination resistors	38		
Other parts of the schematic	39		
2.4 Board layout	40		
2.4.1 Board stack-up	40		
2.4.2 Component placement	41		
2.4.3 Transmission line impedance	42		
2.4.4 Routing	43		
2.5 Board manufacturing	46		
2.6 EMC consideration	48		
3 HW/SW Interface	51		
3.1 Power sequencing	52		
3.2 POR configuration	56		
3.3 Initialization sequence	59		
4 SW integration	61		
4.1 Host driver	62		
4.2 Device firmware	63		

Figures

1.1 Skywan 5G	6	4.3 Handshake between the driver and firmware	64
1.2 Internal architecture of Skywan 5G	7	4.4 Memory allocation configuration	69
2.1 High-level diagram of a Encryption board interfaces	18	4.5 Integration of the encryption hook	71
2.2 C29x Block diagram(from [14])	21	5.1 Thermal performance (no air flow - left, with air flow - right)	75
2.3 C29x modes of operation(from [14])	22	5.2 Thermal performance with Ethernet transceiver disabled	75
2.4 C29x reference design	23	5.3 Test set-up with local traffic generation	77
2.5 Power supply chain	25	5.4 Test set-up with traffic generation in PC	77
2.6 ISL95870B internal circuits(from [17])	31		
2.7 V_{CORE} simulation	32		
2.8 V_{1V5} simulation	34		
2.9 V_{2V5} simulation	35		
2.10 V_{3V3} simulation	36		
2.11 COMe module dimensions	40		
2.12 Board stack-up	41		
2.13 Routing of local signals (IO power - left, decoupling capacitors - right)	45		
2.14 Routing of DDR3 (adress signals - left, reference voltage - right)	45		
2.15 Routing of DDR3 data bytes and strokes	46		
2.16 Routing of PCIe (RX pairs - up, TX pairs - down)	47		
2.17 EMC measurement of Skywan 5G	48		
2.18 Board prototype	49		
2.19 Board plugged into the motherboard	50		
3.1 Structure of initialization SW ..	51		
3.2 Commands in Command line application	53		
3.3 SE state machine	54		
3.4 Measurement of power supplies via ADM1069	55		
3.5 Power sequencer initialization sequence	56		
3.6 Power-on reset sequence (from [31])	59		
3.7 HW initialization sequence	60		
3.8 C291 detected on PCIe	60		
4.1 SW overview	61		
4.2 Physical memory layout in C291	63		

Tables

2.1 C29x family parameters(from [14])	21
2.2 Power supply consideration	24
2.3 Details of C291 power consumption	26
2.4 C291 clock requirements	27
2.5 DDR3 bytes connection.....	38
2.6 PCIe lanes connection	38
2.7 Impedance results	43
2.8 Impedance equations	44
3.1 EEPROM scratchpad registers .	55
3.3 POR configuration	57
3.5 POR configuration (continued) .	58
4.1 Driver/Firmware pre-build configuration	67
4.2 Pre-run configuration	68
4.3 Functions implemented by encryption hook	71
5.1 Power consumption of the Encryption board	74
5.2 Response time of AES encryption	77



Abbreviations

ADC Analog to Digital converter.

BAR Base address register. A register in PCI/PCIe for configuration of memory accesses.

BGA Ball Grid Array. A type of Integrated circuit package with ball-shaped pins placed on a bottom side.

BGP Border Gateway Protocol. A routing protocol.

BOM Bill of Materials. A list of all used components on printed circuit board.

CBC Cyclic block chaining

CLI Command Line Interpreter/Interface.

COMe Computer-on-Module. A standard for embedded boards.

CPLD Complex Programmable Logic Device. A non-volatile programmable logic circuit.

CPU Central Processing Unit.

DCR Direct Current Resistance. A parasitic parameter of a real inductor.

DDR Double Data Rate. An interface between a processor and memory chips/modules.

DFM Design for manufacturability.

DMA Direct Memory Access. A method for transferring the data in a system without putting a load on the processor.

DPA Differential Power Analysis. A side channel attack on encryption system.

DRC Design Rule Check.

DVB-S2 Digital Video Broadcasting Satellite. A protocol for high-bandwidth transfer over satellite.

- ECC** Error Correction Code. An approach of adding redundant information to memory to increase error immunity.
- EP** Endpoint. A type of node in PCI Express terminology. More endpoints are allowed on one bus.
- FEM** Finite Element Method. A numerical method for solving partial differential equations
- FIPS** Federal Information Processing Standards. Set of standards by US government.
- FPGA** Field Programmable Gate Array.
- FSM** Finite state machine.
- FW** Firmware
- HW** Hardware
- I2C** Inter-integrated circuit. A protocol for communication of processor and other chips inside a computer.
- IC** Integrated Circuit
- IO** Input/Output
- IP** Internet Protocol
- IP-Core** Intellectual Property Core. A reusable hardware design for digital integrated circuits.
- LAW** Local Access Window. A memory window used to configure physical address map of a C291 processor.
- LUT** Look-up-Table. Basic building block in Field programmable gate array circuits.
- MAC** Medium Access Control.
- MF-TDMA** Multi-frequency Time Division Multiple Access. A method for accessing physical layer by link layer.
- MIPS** Mega Instructions per Second. Unit for measuring the performance of a processor.
- MMU** Memory Managment Unit. A unit inside processor for translation of virtual addresses to physical addresses.
- NDA** Non-disclosure Agreement.
- NIST** National Institute of Standards and Technology.

- NMS** Network Management Service. Skywan 5G application for network configuration.
- OS** Operating System
- OSPF** Open Shortest Path First. A routing protocol.
- PCB** Printed Circuit Board.
- PDE** Partial Differential Equation.
- PCIe** Peripheral Computer Interconnect Express. A standard for connection of peripherals to a processor.
- PKCAL** Public key calculator mode. A mode of a C29x processor.
- POR** Power-on-Reset. A type of reset which always has to be executed after power-up of a processor.
- PWM** Pulse-width Modulation.
- QoS** Quality of Service. A feature of Skywan 5G allowing traffic prioritization.
- RC** Root Complex. A type of node in PCI Express terminology. Only one Root complex is allowed on one bus.
- RF** Radio Frequency
- RNG** Random Number Generator.
- RT** Real Time
- SEC** Security engine from Freescale/NXP
- SFD** Supply Fault Detector. A circuit inside ADM1069 used for detecting over/under voltage on input pins.
- SKMM** Secure key management mode. A mode of a C29x processor.
- SMBus** System Management Bus
- SoC** System-on-Chip
- SO-DIMM** Small Outline Dual Inline Memory Module. A type of memory module used in notebooks and embedded systems.
- SW** Software
- TBD** To be defined. This feature is open for future modifications.
- TLB** Transaction Lookaside Buffer. A fast memory inside processor used for storing page tables by the operating system.
- TSEC** Tripple Speed Ethernet controller.

Chapter 1

Introduction

Security is an important feature in today's world. Every area has a demand for secure handling of data. In communication industry, this demand is extraordinarily high. Secure communication provides a chance for services such as internet banking. Encryption has a rich mathematical background, often difficult to comprehend without deep knowledge of algebra. Engineers insight into this problem can be found in [1] and it is briefly discussed further in this chapter. From engineers point of view, the realization of encryption systems is especially interesting. During the implementation of an encryption system, many aspects have to be considered such as Performance, Security, Power requirements, Cooling, Scalability, Genericity or Level of realization.

The system whose encryption is implemented in this thesis is a satellite router used in traffic control, aviation, border control or military applications. The product can be categorized as "High reliability" device and thus guarantee proper operation is more important than short time to market. Reliability is directly proportional to design time spent. This thesis does cope with the design of prototype (keeping in mind manufacturability) and does not intend to develop the product into the state where it would be ready for market.

1.1 Skywan 5G

The fifth generation of Skywan satellite router is compact communication system providing satellite connectivity with custom MF-TDMA based protocol. An additional DVB-S2 demodulator is integrated within a unit. Each unit offers up to 16 frequency channels. The IP traffic data rates over TDMA are in the range 12-20 Mb/s. The IP level is handled by Linux networking engine with a strong connection to user-space IP stack. Modern routing protocols such as OSPF and BGP are used for route distribution. An extensive QoS system can prioritize the IP traffic.

The whole system is configurable by the database which is accessible via Web-based engine or NetConf protocol. From external interfaces, the device features 4 Ethernet ports, RS232 and coaxial cable for connection of satellite antenna. The mechanical format is intended for installation into a server rack. One unit is shown in Figure 1.1. The exact mechanical specification is not

important for the purpose of this thesis. The main function of the device can be summed up as follow: provide reliable IP-over-Satellite connectivity.



Figure 1.1: Skywan 5G

An internal architecture of Skywan 5G is shown in Figure 1.2. The IP traffic is handled by four core MIPS CPU (Cavium OCTEON CN6130). The device has a six-port Ethernet switch which can be programmed from the CPU. Satellite connectivity is achieved via connection of an external antenna to the Analog Modem which transmits/receives modulated signal at frequencies from 800 MHz up to 2,4 GHz (known as L-Band). The low level part of MF-TDMA protocol is implemented inside the FPGA. The device is equipped with two extension slots in the COMe format[10]. MIPS CPU is running Realtime Linux, kernel version 2.6.

The device is accessible over a set of buttons and display. The primary configuration interface is a Web browser application which provides full configuration features. The whole network can be configured from one node (after setting up default TDMA connectivity) over NMS application. These details are sufficient for the reader of the thesis. Further details about the device are stated in [2, 3].

1.2 Encryption requirements

Up to now, the encryption of the IP packets was left to the customer. The customer was responsible for using the encryption on the application level before the IP traffic entered the Skywan 5G. The main goal of this thesis is to design a simple extension of Skywan 5G, which would encrypt the IP traffic. The encryption has following requirements:

- Encryption throughput capable of encrypting all the IP traffic which is forwarded by the device. The maximal ethernet-ethernet forwarding performance is approximately 65 000 packet per second. Considering the ethernet jumbo frames (whose support is now being developed), it leads up to approx. 560 MB/s. This estimate assumes that packets forwarded from Ethernet to Ethernet would be encrypted. In the first step, only packets forwarded over satellite will be encrypted.
- Choose components which are suitable for FIPS certification of the system. This mainly concerns RNG engines.
- Have a possibility to remove the encryption from the system. Some markets are barely reachable with FIPS certified encryption. FIPS is

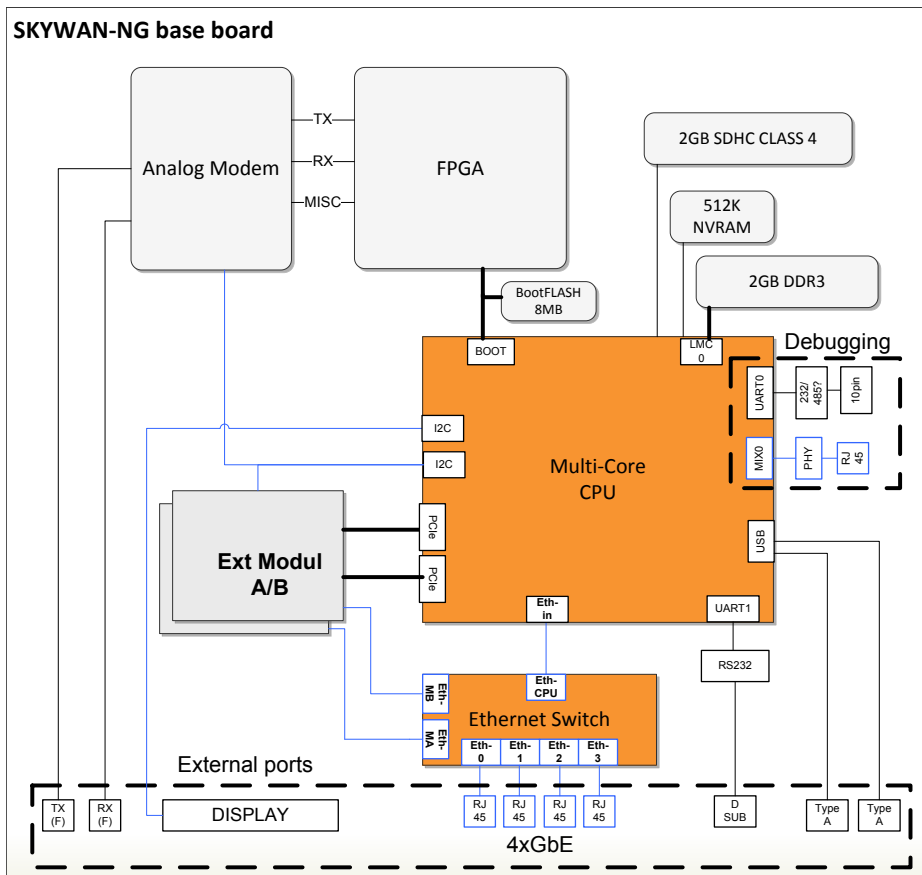


Figure 1.2: Internal architecture of Skywan 5G

not trusted all over the world due to suspicion of predictable (and thus prone to be hacked or spied on) RNG generators. Simply said: Politics play an important role within encryption and cyber security, thus being flexible and neutral is beneficial.

- Have a support of common encryption algorithms and protocols such as AES, RSA, Diffie-Helman, Elliptic curves, Hash functions and others.
- Low power consumption (5-10 W).

1.3 Possible realization

Any form of encryption is computationally demanding operation. Using software which would perform the encryption on the main CPU on Skywan 5G Motherboard is the easiest approach to implement. This solution is unacceptable since the CPU on the motherboard is already handling the IP traffic. The basic load (without IP traffic) of the CPU depends on the configuration of TDMA and QoS software modules. With increasing traffic, the CPU is loaded more up to the point where it is not able to handle all

IP packets, and traffic drops occur. Encryption on the Cavium CPU would necessarily lead to lowering the packet forwarding performance (in packets-per-second) due to high computational requirements (and thus high CPU time). This effect is undesirable, and this kind of realization is not considered.

Another possibility is to use the FPGA on Skywan 5G motherboard and extend the design (which is handling the low-level part of TDMA protocol) with a set of encryption IP cores. The FPGA has a limited amount of LUTs (215 360), and the actual design uses more than half. Future extensions are planned for the FPGA design. These extensions require LUTs which would be used by encryption IP cores, which does not make the FPGA perspective for implementation of encryption.

CPU and FPGA are the only possible computational resources on the Skywan 5G motherboard and both are not suitable for the implementation of encryption. Extension of original Skywan 5G hardware must be considered. As can be seen from Figure 1.2 the Skywan 5G is equipped with two extension slots. One of the extension slots is intended for implementation of custom functionality such as encryption. This solution will be referred to as “internal module” solution. Another approach is to use some of the external interfaces to connect local module which would handle the encryption. This solution will be referred to as “external module” solution.

1.3.1 Internal module

The internal module solution is extension board entirely hidden within Skywan 5G chassis plugged into COMe slot. The SW running on the board would use encryption resources on the extension board to perform the encryption operations. The extension module connector features PCIe interface, which can be used for connection of encryption module to the motherboard. This solution is easy to implement, but it has a limited potential. The whole device is strictly defined, and there is no space for third party additions into the implementation. From the business perspective, it is sometimes important to have flexibility for customers who do not trust the actual implementation of encryption.

1.3.2 External module

The external module can be connected to Skywan 5G by Ethernet. A custom Ethernet-based protocol would be used to communicate with the device. The encryption requests would be sent to the device via this protocol. The encrypted/decrypted data would be sent back by the external module again by Ethernet. This form of implementation is more difficult and time demanding since it requires not only the HW design but also the design of the Encryption protocol. Such a protocol is being developed by ND Satcom and it is intended to be used in combination with Internal module solution, thus creating a hybrid solution (having benefits of both solutions).

The protocol is named Open Encryption Interface (OEI) and it is described in [42]. Once the OEI is designed and supported by Skywan 5G it is possible

to give its details to third party customer who can design private encryption system. The system will be usable with Skywan 5G since all the communication would be according to the OEI. The implementation of this protocol (and its connection to configuration database) is a logical continuation of this thesis.

1.4 Background of Cryptology

Cryptology is a scientific discipline, part of Mathematics, which attempts to hide the content of data or messages. Cryptology can be divided into several disciplines. Two most important are: Cryptography and Cryptoanalysis. Cryptography develops algorithms which are transforming data so that only certain people can find out the original content. The original data are known as plaintext. The transformed data are known as ciphertext. Encryption is the process of changing a plaintext to a ciphertext. Decryption is the process of changing a ciphertext back into a plaintext. Cryptoanalysis is examining possible errors in cryptographic algorithms and it is trying to penetrate the encryption systems. An important part of encryption algorithm is a key which is a secret information (usually known only to a private group of communicants). A key is an additional input (apart from data) into encryption algorithm.

Symmetric vs. asymmetric key ciphers

One of the criteria of the division is by the handling of a key. Symmetric key ciphers refer to set of algorithms which share the same key for encryption, decryption. To keep the encryption secure, this key must be secret known only to the participants of the communication. On the other hand, asymmetric ciphers distinguish between private and public keys. A private key is a secret kept between the participants of the communication. A public key is a key which is available to third parties which can monitor the communication process.

The disadvantage of symmetric key ciphers is that both communicants need to have common secret before the communication. To achieve this, another secure channel must be existing (in the old times a courier distributing highly guarded book, e.g. with keys for each day of the year). This is a weak spot of the whole process. If this private information leaked, anyone could decrypt the ciphertext and obtain the plaintext. The asymmetric key cryptography has algorithms with no prior common secret (the most common is explained below). It is common practice to use asymmetric protocol for private key exchange and use the private key subsequently for symmetric encryption.

Block vs. stream ciphers

Next criteria of the division is by the size of data chunk processed by an encryption algorithm. Since most of the data sizes are variable (e.g. each IP

of this system is to offer support for more encryption algorithms, achieve integration with embedded database and configuration engine and implement OEI as embedded SW module. These topics are beyond the scope of this thesis.

An important fact when choosing an encryption algorithm is public availability and openness. If the implementation is open, many people already attempted to break it and thus more secure it is (if it is still unbroken). A paraphrased motto from [1] says: Cryptography is not about hiding a necklace in a big city and saying: find the necklace without having any other details about the necklace. Cryptography is about locking a necklace in a box, providing all the construction information about the box and being calm that the necklace will stay inside the box!

■ Advanced encryption standard

Advanced encryption standard (shortly AES), originated in Rijndael algorithm, is a symmetric key block cipher. The algorithm was result of public competition from NIST. The winning proposal was submitted by Joan Daemen and Vincent Rijmen in [4]. These days it is one of the most used symmetric encryption algorithms. AES is operating on 128 bits (16 bytes) block size and using key sizes of 128, 192 or 256 bits. The number of iterations (rounds) for one AES calculation is key dependent (10,12 or 14 rounds). AES calculation is matrix based and input block is ordered into 4*4 state matrix:

$$\begin{bmatrix} d_0 & d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 & d_7 \\ d_8 & d_9 & d_{10} & d_{11} \\ d_{12} & d_{13} & d_{14} & d_{15} \end{bmatrix} \quad (1.1)$$

The algorithm is described by following steps:

1. Execute key expansion according to key schedule algorithm. The algorithm description is omitted due to its complexity. The important outcome is that each round operates on a different key which is the output of key scheduling algorithm. Such a key is called round key.
2. Calculate initial round by bitwise XOR of the round key with the respective input byte.
3. Perform calculation of $n - 2$ rounds, where n is the total number of rounds with following steps:
 - a. Byte substitution: Each byte of the state matrix is replaced by another byte from Look-up-Table known as Rijndael S-box (the calculation of the S-box is omitted).
 - b. Shift the matrix rows: Each byte element of the row k is shifted left $k - 1$ times.

- c. Perform multiplication of each column by fixed matrix:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \quad (1.2)$$

Since the elements of the state matrix are 8 bit wide, they are not treated as a decimal number, but as coefficients of a polynomial of order 7. Thus the multiplication of the column (vector) consists of polynomial multiplication and addition. Since the multiplication of polynomials of order 7 and 3 can result to a polynomial of order 10, the modulo of the multiplication result by polynomial $x^8 + x^4 + x^3 + x + 1$ is executed and the coefficients of the polynomial remain one byte wide.

- d. Perform the XOR operation with current round key.
4. Calculate the last round which is the same as rounds in step 3 with leaving out the sub-step (c).

Detailed mathematical background of AES is beyond the scope of this thesis. AES algorithm is standardized by the NIST and it is adapted by the US government for the encryption of classified data. Successful side channel attacks on the implementations of AES exist (using cache timing attack on SW implementation as in [5]), but the cipher is still considered to be safe. The HW implementations use various methods of transforming the data before the encryption to avoid correlation between power consumption, key and plain-text as in [6]. A successful DPA attacks still exist [7].

Diffie-Hellman

Diffie-Hellman refers to asymmetric key encryption algorithm introduced in [8] which provides a mechanism for sharing a secret with no prior common secret. Development of this cipher was considered as the huge step in the field of cryptography. The high-level overview of the algorithm is following:

1. Let's assume two communicants A and B who have no prior common secret. They establish a common integers p and g , where g is primitive root modulo of p . They share these integers over a public channel.
2. Communicant A chooses a secret integer m and calculates $M = g^m \text{ mod } p$ and sends the M to communicant B.
3. Communicant B chooses a secret integer n and calculates $N = g^n \text{ mod } p$ and sends the N to communicant A.
4. Communicant A calculates: $k = N^m \text{ mod } p$.
5. Communicant B calculates: $k = M^n \text{ mod } p$.

6. Now both communicants share common secret integer k . This integer can be used as a key to symmetric key encryption.

By definition modulo operation can be rewritten by adding (subtracting) integer multiple of the divisor:

$$N = g^n \text{ mod } p = g^n + pK \quad (1.3)$$

If we use the equation (3), substitute it into the equation from step 4 and apply binomic formula we get:

$$\begin{aligned} k = (g^n + pK)^m &= \sum_{k=0}^m \binom{m}{k} (g^n)^{m-k} (pK)^k = \\ &g^{nm} + \sum_{k=1}^m \binom{m}{k} (g^n)^{m-k} (pK)^k \end{aligned} \quad (1.4)$$

Note that since all the members of binomic series are multiple of p it is clear that:

$$\left(\sum_{k=1}^m \binom{m}{k} (g^n)^{m-k} (pK)^k \right) \text{ mod } p = 0 \quad (1.5)$$

which gives us:

$$\begin{aligned} k = (g^n + pK)^m &= \\ \left(g^{nm} + \sum_{k=1}^m \binom{m}{k} (g^n)^{m-k} (pK)^k \right) \text{ mod } p &= \\ (g^{nm} \text{ mod } p) & \end{aligned} \quad (1.6)$$

If the other communicant performs the same operation as in step 5 he obtains:

$$M = g^m \text{ mod } p = g^m + pK \quad (1.7)$$

$$\begin{aligned} k = (g^m + pK)^n &= \sum_{k=0}^n \binom{n}{k} (g^m)^{n-k} (pK)^k = \\ &g^{nm} + \sum_{k=1}^n \binom{n}{k} (g^m)^{n-k} (pK)^k \end{aligned} \quad (1.8)$$

$$\begin{aligned}
& (g^m + pK)^n = \\
& \left(g^{nm} + \sum_{k=1}^n \binom{n}{k} (g^m)^{n-k} (pK)^k \right) \text{mod } p = \\
& (g^{nm} \text{mod } p) + \left[\left(\sum_{k=1}^n \binom{n}{k} (g^m)^{n-k} (pK)^k \right) \text{mod } p \right] = \\
& \qquad \qquad \qquad g^{nm} \text{mod } p = k
\end{aligned} \tag{1.9}$$

which proves that both communicants reach the same shared secret k . Note that since the third person in the communication knows M , N , g , p he would need to solve following equations:

$$M = g^m \text{mod } p \tag{1.10}$$

$$M = g^m \text{mod } p \tag{1.11}$$

to be able to calculate k . This problem is known as discrete logarithm problem and there is no known algorithm for its calculation (other than brute-force). If g and p have sufficient length, it is impossible to iterate over all possible options with existing computational resources. It is recommended that p should have approximately 1000 bits and $q \approx \frac{p}{2}$.

■ Hash message authentication code

Hash message authentication code (shortly HMAC) which is used for verification of data integrity and authentication of a message. A hash function creates a small sequence from a bigger sequence, which logically implies that many inputs map to equal outputs. A typical requirement for a hash function is that minor change (e.g. one bit) in the input sequence causes a big change in the output sequence. Typically many bits need to be altered in the message to achieve the same output of hash function from original and altered data. The minimal amount of bits which must be altered in any of inputs (the whole input state space must be considered) to obtain equal output is known as Hamming distance. If fewer bit flips than Hamming distance occurs during the communication, the received hash and the calculated hash will not match and the error is detected. If the hash function is executed on private data (which are exchanged with symmetric encryption), any person in the middle can not produce the same output of the hash function. A match between received and calculated hash thus not only guarantees the data integrity but also the authenticity of the message.

Many hash functions exist, the most known are: SHA1, SHA2, SHA3 or MD5. The HMAC is a combination of such a hash function and secret cryptography key which was standardized in [9]. The main formula for calculation of HMAC function is:

$$HMAC = H(K' \oplus op, H(K' \oplus ip, m)) \quad (1.12)$$

Where H is the hash function, K is the hash key, K' is derived hash key and m is the input message. The security of HMAC always depends on the underlying hash function. Today MD5 and SHA1 are not considered to be secure enough as standalone hash functions. However, the usage of SHA1 in the HMAC format remains in practical use for message authentication.

Chapter 2

Hardware design

As mentioned in 1.3 there are two possible options of encryption system realization. An external module connected over Ethernet or internal PCIe extension module to the motherboard system. PCIe extension module solution was chosen. Two extension slots, A and B, in the format of COMe (Computer-On-Module) standard are available on Skywan 5G motherboard. The COMe standard is described in [10] and it has 220 signal and power pins. The exact pinout is omitted and can be found in the specification. Slot A is used for the implementation of Encryption module. Following interfaces of Skywan 5G Motherboard CPU are routed to the Slot A of COMe connector:

- PCIe Generation 2 with four lanes
- 1Gb Ethernet
- I2C
- UART
- Control and status signals

PCIe is used as the main communication interface between Motherboard and system running on Encryption module. The high-level diagram of all the connected interfaces to the COMe connector is shown in the Figure 2.1.

An important topic when designing an embedded system is the level at which the main features (encryption algorithms) are implemented. As it is common in engineering practice, different aspects are in mutual conflict. Due to this reason, it is important to lay out hardware realization requirements:

- Low power consumption of the system. 5-8 W is required, while the system should not at any case consume more than 10W.
- Encryption throughput of more than 560 MBs. The number is estimated from maximum packet forwarding rate of Skywan 5G (ethernet to ethernet) when considering ethernet jumbo frames.
- Component availability. It is important to use components which are not outdated, and their production is guaranteed for the whole lifetime of the project (5 to 10 years).

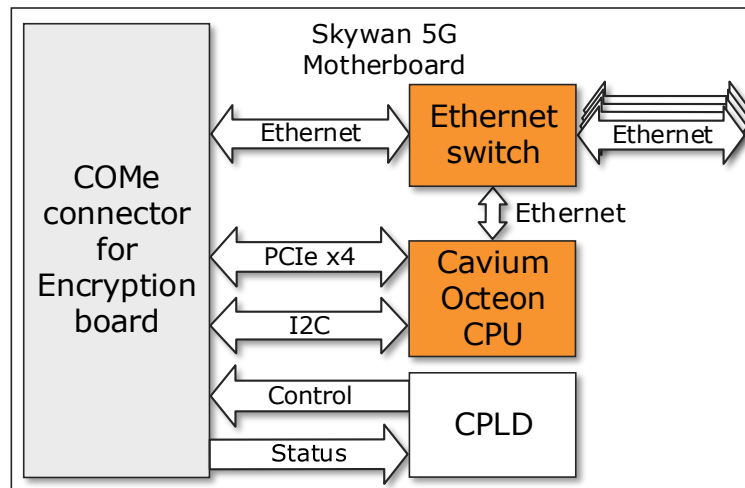


Figure 2.1: High-level diagram of a Encryption board interfaces

- A manufacturing prize per 1000 pieces. If the system is too expensive, it is not rentable for series manufacturing. Intended manufacturing prize of the whole system (board and components) is approximately 200 Euros per board when manufacturing 1000 boards.
- A level of commercial support and availability of integration tools for chosen components.
- A degree of realization difficulty. The products with good support (e.g. drivers, reference designs) are easier to implement.

With these points outlined there are several options available:

SW realization Put CPU on the board and execute the encryption operations either in a firmware of the CPU or as a software module under an operating system. The advantage of this solution is a variety of suitable CPUs and availability of open source encryption libraries in C (e.g. in [11] or [12]). The disadvantage of this approach is the power consumption of such a CPU. CPUs with enough MIPS to calculate the encryption in SW can have higher power consumption than required.

HW realization with custom chip There is an option to use a chip which only implements encryption. One such a chip is [13]. Since this chip is not available anymore and no similar chips were found it was decided not to use this option.

FPGA implementation The idea of implementing the encryption operations in FPGA with either SoC or set of open source IP cores is the most optimal from the power vs. performance point of view. However, HW design would involve not only PCB design but also the design of SoC and its integration with PCIe interface. This additional design effort makes this task implausible within the time dedicated for this thesis.

CPU with HW Acceleration Using a CPU with dedicated HW block for encryption acceleration. Since encryption is HW accelerated this solution has low power consumption and simple usage since there is no need to design own SoC. The disadvantage of this approach is usually the price. Application specific CPUs are commonly more expensive than general purpose CPUs since the target market is smaller.

Based on these points it was decided to use a CPU with HW accelerated encryption.

■ 2.1 System level design

Before drawing the schematic and designing the layout, it is important to design the product on a system level. System level design includes connection of interfaces, power supply design, and clock tree design. It is important to think about system initialization, proper reset, and configuration. Particular components must be selected to realize all the features. The outcome of this step is a group of block diagrams and tables which are introduced in each of the following sub-sections. The overall block diagram of the board is shown in Appendix A, Page 2. Following rules mainly drive this part of the design:

- Maximize the components re-usage from the motherboard of Skywan 5G. If some functionality is required, it is better to choose component which is already used on Motherboard, rather than using new component. This design rule minimizes the number of entries in Bill of Materials (BOM). With fewer entries in the BOM, the manufacturing is cheaper. With a higher amount of pieces from the same component the price per piece drops.
- Use reference materials and designs. It is not probable to make an error when using a component exactly as in reference design. It is common to rely on reference designs in engineering practice today.
- Avoid using deprecated and discontinued components. Once a finite supply of these components is sourced out, the device will not be manufacturable anymore!
- Use generic approach. It is better to implement a functionality available within a component and not use it during operation (not populate the component, resistor, etc.) than not implement it at all. This namely concerns the interfaces which are available in CPUs such as DDR3 or Ethernet. With this approach, there is a room for future development and extension of features without HW re-design.

■ 2.1.1 CPU selection

As mentioned above, CPU with HW accelerated encryption will be used. After research of available components in the CPU market, following options were considered:

- CAVIUM Nitrox PX family. This CPU family fits by encryption parameters, but after communication with the manufacturer, it was decided not to use this CPU since it is outdated and it is not recommended for new designs.
- CAVIUM Nitrox III family. This CPU family is intended for a high-end server application. CNN3510-C5 is the cheapest CPU from this family. This CPU costs approx. 300 USD/piece when bought 1000 pieces. A single piece of the CPU costs over 500 USD. The reference design of this CPU costs 25 000 USD. The power consumption of this CPU is no less than 13 W which is beyond the required maximum. Due to these reasons, it was decided not to use this CPU.
- Intel CPUs with a support of AES Instruction set. An appropriate CPU here is Intel Atom which is intended for low power applications. The usage of CPU from Intel requires NDA, acquiring one is a long-run process in case of Intel. A design of Intel-based system was considered infeasible within the duration of this thesis.
- C29x CPU family from Freescale Semiconductors (acquired by NXP Semiconductors). It is intended for server applications, and it has a wide selection of products. The overall prize/performance/power ratio of this CPU family is very lucrative. Furthermore, NXP Semiconductors offer reference design materials for free.

A C29x family of CPUs was chosen. This family of CPUs contains three products: C291, C292, C293. The last digit in the name of the CPU indicates the number of SEC engines (encryption accelerator from Freescale). The CPU has Power Architecture e500-v2 from IBM. The basic parameters of the CPU family are displayed in Table 2.1. Based on the encryption throughput and power consumption it was decided to use C291. The prize per one CPU within 1000 pieces batch is 112 USD. The block diagram of the CPU and the encryption algorithms supported by SEC are shown in Figure 2.2. More details are in [14]. The CPU contains true random number generator based on thermal noise which is NIST certified. This feature is important due to possible FIPS certification of the whole system. Furthermore, AES accelerator claims to be DPA resistant. C291 is available in convenient BGA package with 783 pins. The core frequency of C291 does not reach GHz region, which is also a critical aspect. Faster CPUs would generate EMI at L-Band frequencies, and it might disturb the satellite connectivity of the whole system.

The CPU can operate in two modes: Public key calculator (PKCAL) and Secure Key Management Module (SKMM). In PKCAL the CPU works as a remote endpoint to a standalone system for encryption offload. In this mode most of the peripherals are disabled, and there is only small firmware running in the core of the CPU. There is no OS running on the CPU. In the SKMM the device should be running Linux OS, and it can be used for receiving Ethernet traffic and performing encryption operations on it. CPU makes use

Table 2.1: C29x family parameters(from [14])

	C291	C292	C293
CPU frequency [MHz]	667	1000	1200
SEC frequency [MHz]	267	333	400
DDR frequency [MHz]	800	1067	1200
Typical power (65 °C) [W]	5	9	18
2048 bit private key operations	8461	17587	31689
Bulk encryption (AES-HMAC SHA-1 for SSL or Ipsec) [Gb/s]	6	9	12

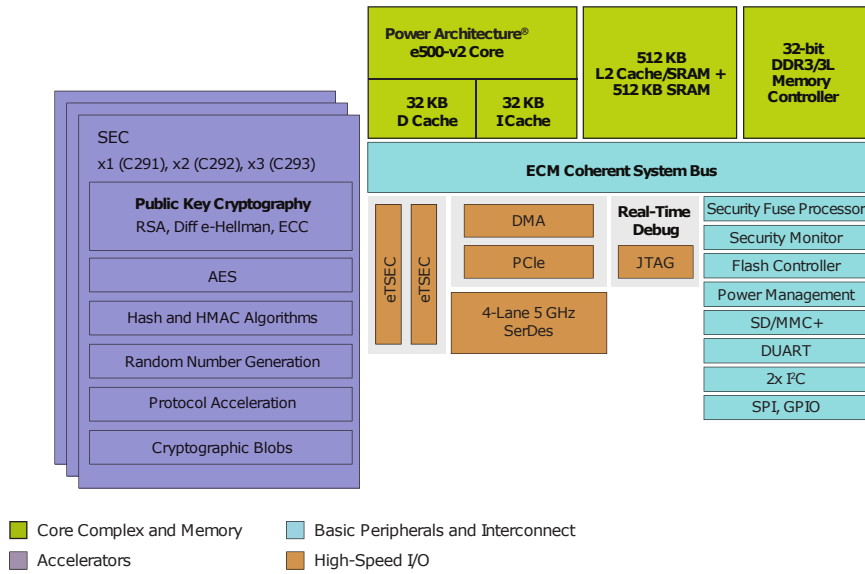


Figure 2.2: C29x Block diagram(from [14])

of external DDR memory in this mode. The comparison of applications in both modes is shown in Figure 2.3. It is intended to use the CPU in PKCAL mode. To keep SKMM for future, all the interfaces must be implemented on the PCB! If used in PKCAL, it is possible not to populate several components to reduce the manufacturing costs.

C291 is equipped with following peripherals/interfaces:

- PCIe MAC controller. It is intended to use this controller as the main communication interface with the Cavium CPU on Skywan 5G Mother-board.
- 5 Ghz, 4 Lane SerDes (Serialize-deserialize) engine which can be used as physical layer transceiver for PCIe.
- DMA controller with external signals. This controller will be unused in the design. DMA operations are well supported by PCIe bus without a

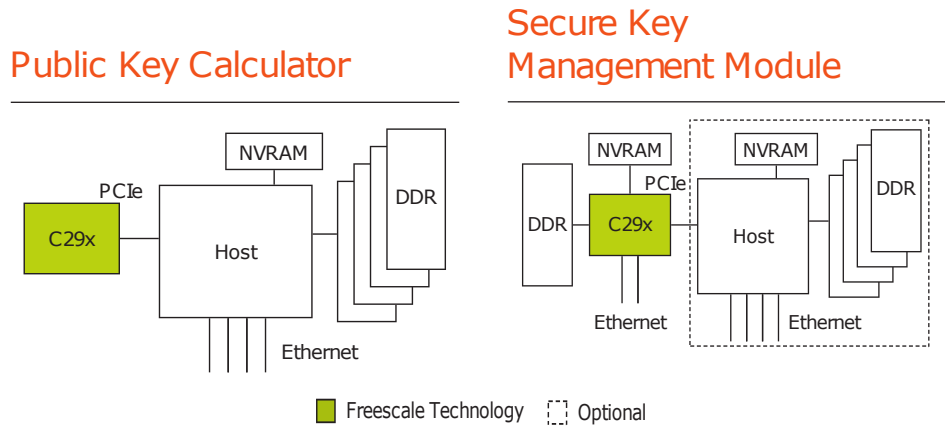


Figure 2.3: C29x modes of operation(from [14])

need of an external DMA engine. The C291 features bus mastering, so it can copy the data to/from motherboard without burdening Cavium CPU.

- eTSEC (Enhanced Tripple speed Ethernet Controller) offering 10 Mb/100Mb/1Gb Ethernet versions.
- 32-bit DDR3 memory controller with ECC. It is used only in SKMM module. This interface is the most difficult from the design point of view, but it was decided to implement it. Running the CPU in SKMM (booting Linux) is impossible without external memory.
- I2C. There are two I2C controllers each can be Master or Slave on I2C bus.
- SDHC controller. During operation in SKMM, CPU needs non-volatile storage for booting an OS. Flash card in MicroSD slot can be used for this purpose.
- Two UART interfaces. First UART interface will be routed to COMe connector. The second one can be used as debugging interface, and it will be routed to an external connector.
- SPI interface is unused. The original intention was to connect an external RNG based on the thermal noise of resistor. Since designing an analog RNG is a complex topic it was decided not to implement it.
- IFC (Integrated flash controller) interface is dedicated for connecting external flash memory. Flash memory can be used as Linux boot location when running in SKMM mode. To reduce the board costs, it was decided not to implement this interface. In the case of running SKMM microSD slot will be used as a boot location. Though microSD throughput is lower than IFC throughput, it is sufficient for booting Linux.

This CPU is well prepared for integration into commercial systems. The manufacturer offers reference design in the format of PCIe card (with C293). The schematic of this design can be downloaded from NXP website. The device can be bought for 990 USD which makes it cheap in comparison with CAVIUM reference designs. One such card was obtained, and it is shown in Figure 2.4. Software drivers, Linux ported to C291, host drivers for PKCAL mode, hardware layout, and other design documents are provided on CD.

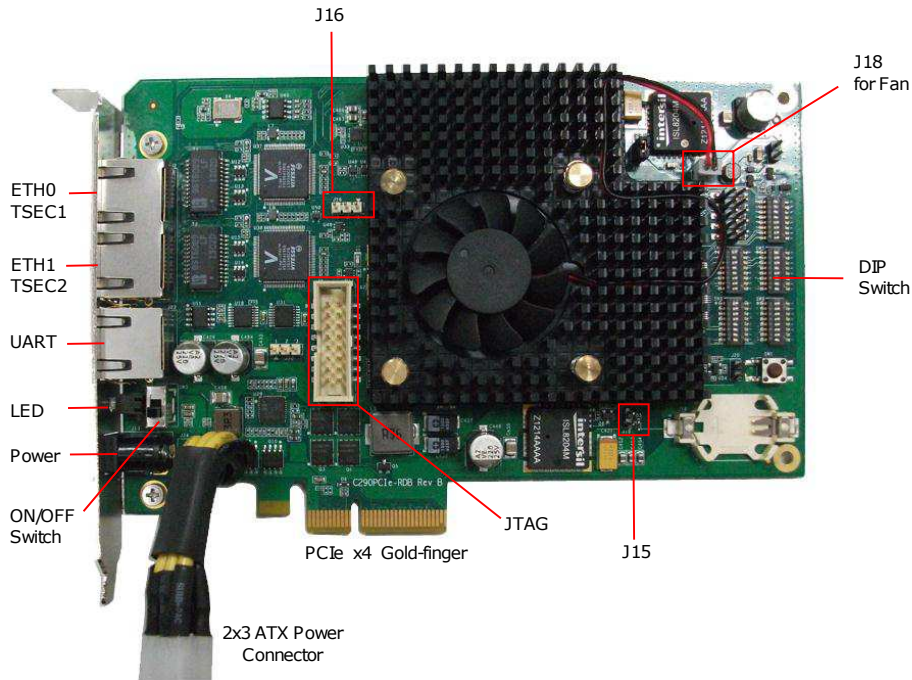


Figure 2.4: C29x reference design

2.1.2 System configuration and status

Digital boards often have non-volatile CPLD with design loaded during manufacturing. It provides board initialization, configuration, and test functions. A CPLD is on the motherboard of Skywan 5G and it has dedicated control and status signals for connecting the Extension slot. Due to this reason, there is no CPLD on the Encryption board. However, C291 CPU still needs to be configured. This configuration is called POR (Power-on-Reset) configuration. During the hard reset of CPU (pulling H_RESET_N input low) the CPU latches logical level at several pins. Based on these values, CPU is configured. At the time of Hard reset, all output pins are in a state of high-impedance. Connecting a weak pull-up or weak pull-down resistor then determines the logical level at reset. This approach is used in the reference design where pull-up or pull-down is selected via DIP switch. On the encryption board it is necessary to have the CPU reconfigurable from SW of Skywan 5G. Due to this reason, 40 bit IO expander is used. Serial resistor between the output of IO

expander and configuration pin is used to realize weak pull-up or pull-down function. If a pin is later driven to a different voltage by CPU, the device will not be damaged. IO Expander circuit is a slave device on the I2C bus, and it is connected to COMe connector. This way the CPU on the motherboard (I2C master) can configure the C291. The whole process is further described in Chapter 3.

2.1.3 Power supply design

Since the C291 is the main component on the encryption board, the design of the power supply chain is driven by its requirements. It is important to realize that the board contains only digital components, there is no analog part on the board (except temperature sensor). This fact simplifies the design of power supply chain. In analog design, it is necessary to be careful about noise and use linear voltage controllers instead of DC-DC converters or switched controllers. The main attempt when designing the power supply chain is to follow reference design and design of the motherboard. The reusability of components makes the design more flexible for future “mass” production.

Table 2.2: Power supply consideration

Name	Voltage [V]	C291 consumption [W]	Other consumption [W]	Overall consumption [W]
V_{CORE}	1,0	8,194	0	8,194
V_{GVDD}	1,5	0,81	2	2,81
V_{VTT}	0,75	-	-	-
V_{3V3}	3,3	0,15	approx. 1	1,15
V_{2V5}	2,5	-	0,5	0,5
V_{1V5}	1,5	0,24	-	0,24

The 12V power supply available from COMe connector from a motherboard powers the whole Encryption board. No other power supply (like external connectors) will be used! The COMe specification states the maximal power consumption of the module to be 121W. This is far above the design goal, and there is no need to worry about the current capacity of COMe connector.

The first step of power supply design is to list all the supply voltages which will be used in the design and estimate maximal power consumption. Although the C291 is the main component, the consumption of other ICs is not negligible! The Table 2.2 shows the estimate of power consumption of each power supply.

The second step is to create a voltage chain and define how are the voltages going to be generated. The designed supply chain is displayed in Figure 2.5. Most important ICs are further discussed in 2.3. As mentioned above it is common to use switching controllers in digital boards. Switching controllers have higher efficiency than linear controllers, but produce more

noise. Since all voltages listed in Table 2.2 are lower than the 12V supply voltage from the motherboard, step-down converters are used. Supply voltages for CPU core and DDR memory have the highest current consumption. This leads to controllers with external switching transistors connected in a bridge configuration.

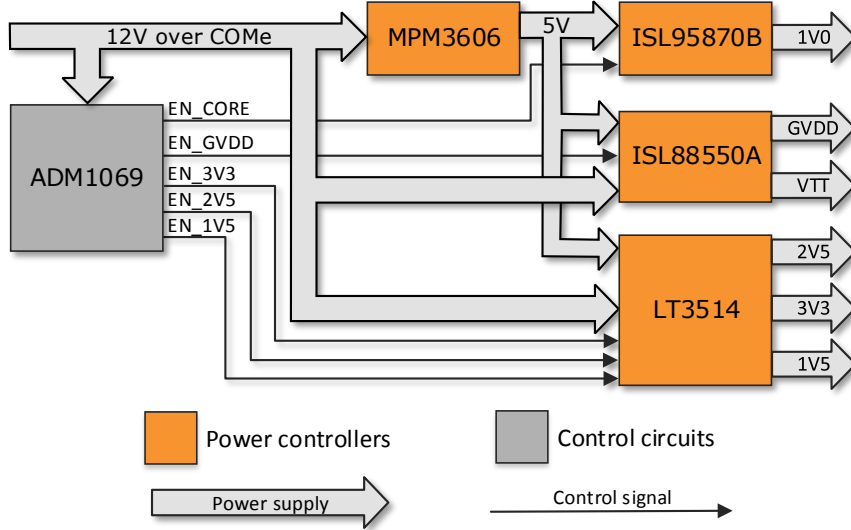


Figure 2.5: Power supply chain

2.1.4 Thermal design

The datasheets of ICs usually state quiescent power, which is power consumed by a circuit just to maintain its state without performing any operations (no gates switching). Thus it is good to consider this information as required minimum and keep a reserve in the thermal design. Luckily the C291 documentation provides more accurate information about power consumption. The power is measured with Dhrystone algorithm running on the core, DMA engine in operation, and the SEC engine executing encryption. The average power is measured with 90% activity. The maximal power is measured with 100% activity.

The design must fulfill the requirements for SKMM although it will be used in the PKCAL mode. Furthermore, power consumption at highest temperature must be considered. The power consumption details are shown in Table 2.3. It is intuitively clear that the BGA package can not dissipate this amount of power without a heatsink. The maximal thermal resistance of a heatsink can be calculated:

$$P_{DIS} = \frac{T_j - T_A}{R} = \frac{105 - 45}{12} = 5W \quad (2.1)$$

$$P_{DIFF} = P_{MAX} - P_{DIS} = 9,54 - 5 = 4,54W \quad (2.2)$$

$$R_{HSMAX} = \frac{T_J - T_A}{P_{DIFF}} = 13,2 \frac{^{\circ}C}{W} \quad (2.3)$$

where R is a Thermal resistance of BGA package, P_{DIS} is Maximal power dissipated by BGA package, T_j is Maximal junction temperature, T_A is Ambient temperature, P_{MAX} is Maximal overall power consumption. The calculation was performed with the assumption of 4 layer PCB and no air flow! Since the final board has ten layers and the air flow above Extension Slot A in the Skywan 5G is not negligible, this calculation was taken only as guideline and heatsink with 16 °C/W was selected. Available heatsinks with lower thermal resistance were too expensive.

Table 2.3: Details of C291 power consumption

Part of C291	Power consumption [W]
Average core power at 65 °C	5,206
Maximal core power at 105 °C	8,194
DDR3 power at 800 Mhz	0,81
PCIe power at x4 5GT/s	0,24
eTSEC power at 3.3V	0,07
Power of other peripherias	0,08
Maximal overall power consumption	9,54

2.1.5 Clock design

The clocking in the design can be divided into two parts: clocking of C291 and clocking of PCIe. The C291 clocking requires three separate clocks: System clock, DDR clock, and TSEC clock.

The second part of the clock design is the design of PCIe reference clock. In the terminology of PCIe standard, one of the devices always has to provide PCIe reference clock (100 or 125 Mhz). The clock is usually generated by a device which is Root-Complex (RC) in the PCIe system. It is not a rule, but a general habit. It is not the case in our system. Extension slot Ext A, used for Encryption board does not have the reference clock provided from the motherboard. Thus PCIe clock generator must be placed on the Encryption board. This architecture is used since the Extension slot A should be used for connection to stand-alone systems which would provide the reference clocks. All the clock signals and clocking requirements are listed in Table 2.4.

2.2 Tool selection

Schematic design and design of PCB is challenging task which requires complex software tools. Before selecting the particular tool it is important to lay down tool requirements:

- The Design of multilayer boards with at least ten copper layers. The motherboard is designed with ten copper layers.

Table 2.4: C291 clock requirements

Clock signal	Required frequency [MHz]	Duty cycle [%]	Slew rate [V/ns]	Peak period jitter [ps]
System clock (SYS_REFCLK)	66-100	40-60	1-4	+150
DDR clock (DDR_REFCLK)	66-100	40-60	1-4	+150
TSEC clock (TSEC1_REFCLK)	125	47-53	3.33	+150
PCIe clock (PCIE_CLK_CPU_P/N)	100 or 125	45-55	-	60

- Support for routing of differential pairs. This feature is important for routing of PCIe and Ethernet interfaces. Maintaining single-ended and differential impedance of differential pair is critical. The differential impedance depends on the distance of conductors. A slight mismatch in the distance between conductors within a pair can lead to catastrophic signal integrity.
- Length matching of single ended traces, differential pairs and deskewing of differential pairs. These features are most crucial for routing of parallel interfaces where all signal lines must have same electrical length. Designing DDR3 memory interface without the length matching feature is practically impossible.
- Generation of BOM (Bill of materials) and proper manufacturing data (Gen-Cad, Gerber, and drilling data).

Many tools are available from open-source, up to professional tools. The motherboard of Skywan 5G is designed with Orcad Cadence. It would be the best to design it with the same tool. However, license to this software is expensive (approximately 5000 EUR) and it was decided not to use it. A similar situation is with Altium Designer from Mentor Graphics. Since the board layout of Skywan 5G was outsourced, whole license would have to be bought. Possibly the tool would remain unused for a long time after finishing the Encryption board. Thus the regular support fee is the main reason why neither Altium nor Orcad was used.

An open source tool, KiCad [15], was chosen. It is a rapidly developing tool, and it supports all above-listed features. At the time of the design, version 4.0.4 was available. This tool supports scripting in Python which is a benefit in comparison to other programs. Each design file is stored in a structured text file which makes the design easily understandable. KiCad is multiplatform software so the same design can be used in many operating systems. A disadvantage of this tool is a lack of schematic and footprint

libraries. The online database does offer only basic footprints. Since the time scope of this project is relatively long, all schematic components and footprints were designed from scratch. The tool offers following subprograms:

- Electronic schematic editor - Design of schematic.
- Schematic library editor - Design of components in the schematic.
- Printed circuit board editor - Design of boards layout and generation of manufacturing data.
- PCB footprint editor - Design of IC footprints and mechanical objects.
- Gerber viewer - Review of generated manufacturing data in Gerber format.
- Bitmap to Component - Generator of decoration images in non-copper layers from Bitmaps.
- PCB Calculator - Tool for impedance calculations of transmission lines.
- Worksheet layout editor - Editor of page outline patterns that can be used across the design.

After design and manufacturing of the board, the tool was evaluated and compared with professional tools. The tool fulfilled the expectations and did not put any significant obstacles into the design process. However, the tool has many bugs causing freeze or crash of the program.

The performance of OpenGL implementation of routing tool is weak. When the design is complex, and it contains many components, it often freezes (mostly for actions like select and move track). Sometimes it is even more beneficial to delete the track and route it again since the tool is able to do it more fluently.

Several features are missing in the tool (in Comparison to Altium and Orcad), which makes the design process obscure and unnecessarily long. The estimate is that the design process was prolonged by at least by 50 hours due to all these flaws. Following notes were reported to KiCad community in online bug-tracker:

- Missing zone DRC. It is not possible to override global DRC rules with zone specific rules which make it difficult to route the signals out from a BGA package. This was especially problematic in case of DDR3 interface! The global design rules must be temporarily changed for the initial phase of routing and then modified for routing outside of BGA. Furthermore, the “Shove track” tool does not perform well as long as two tracks already corrupt actual DRC rules at some point (typically under the BGA).
- The length matching tool offers only one-time meander creation. Once the meander was created its length can not be tuned anymore. It consists of tiny lines which are not grouped into some “meander object”. If one

wants to change the meander size, the whole meander must be erased (segment by segment), the line must be re-routed and length-matched again, which is very impractical.

- The management of differential pairs is catastrophic. The tool does not provide a way how to drag the differential pair together. One can either erase the pair (or its segment) and route it again, or drag each track individually, risk changing the distance between the pair members and violate the differential impedance! Additionally, the DRC check of differential pairs reports thousands of errors if the distance within a pair is the same as minimal track clearance. Luckily, the last problem can be fixed by lowering the minimal track clearance by 0.001 mm, since the problem exists only due to incorrect operation with floating point arithmetic in the tool.
- The layout designer can not design custom via stack-pad. This option makes it impossible to route high-speed boards operating at frequencies above 10 GHz.
- Missing FEM simulation. Design guidelines recommend performing FEM simulations after finishing the layout.

2.3 Schematic design

The design of schematic for digital boards is a simple task these days. After the specification of requirements on System level (discussed in the previous sub-chapter), the functionality must be implemented with either ICs or discrete components. Although the CPU is chosen at this point, the selection of the remaining components is not negligible. Choosing a wrong component can cause failure of the whole project. It is unusual to use discrete components for realization of some functionality (e.g. discrete power supply) since it is too expensive to manufacture the device with discrete components. Most of the functionality is thus realized with dedicated integrated circuits. The connection of the ICs is simple due to recommended applications which are available in the datasheets. However one must keep many issues and design approaches in mind. Following list names only a few of them:

- Be careful about recommended and limiting values of each input and output pin of every IC. Although exceeding the recommended values is not often critical, it is not good to use circuit permanently beyond the recommended values. Exceeding the maximal ratings is often critical and leads to damaging the circuit.
- Always check the thresholds of digital logic levels when connecting two digital ICs together. The minimum voltage for detection of logic 1 and the maximal output voltage of logic 0 must be fairly distant. This situation is getting even worse with noise! Noise between digital circuits

can cause wrong interpretation of logic level. This problem is in detail explained in [16].

- Beware of live-insertion effects. These include connection of IO pins of un-powered ICs to signals in logic one. If this effect occurs, the clamping diode in IO pin of unpowered IC will open, and the logic pin will be loaded with the power consumption of the whole chip. In the worst case, it can damage the circuit. This fact is especially important at I2C bus since it is the main method of accessing the Encryption board during its initialization.
- Use generic approach as much as possible. If a chip can be configured by pull-up or pull-down resistor, then it is good to implement both options and populate only one of the resistors. Leaving the board configurable without a need of redesign is practical. The only change needed, is a population of a different component.
- Use plenty of 0 Ohm resistors as switches. Sometimes it is convenient to have an option to disconnect certain functionality, and the best way to do it is not to populate 0 Ohm resistor, instead of cutting a wire if the resistor is not there.

■ 2.3.1 Power supply

■ Core power supply

The core power supply is used to power the core of C291, system bus, SEC engines and main logic of peripherals. Its implementation is displayed in Appendix A - page 10. The used voltage controller is ISL95870B from Intersil. It is step-down buck PWM regulator intended for Mobile GPUs and Mobile memories. The output stage is driving two transistor half-bridge. SI4420BDY and SI4116DY transistors are used in the output stage, due to low channel resistance in turned on state. It is important to select an inductor with low DC resistance, due to high current flowing through it. A nice feature of this circuit is available at RTN pin. This pin is used to balance the voltage offset of the ground plane. Feedback from the output voltage is connected via the voltage divider (feedback attenuator). Part of the ISL95870B circuitry is shown in Figure 2.6. The voltage controller regulates the voltage on the FB pin to be equal to the reference voltage. The reference voltage is set to 0.5 V via pull-up resistors on VD0/VD1 pins.

Following voltage divider is used to set the output voltage :

$$V_{FB} = V_{CORE} \frac{R_{1011}}{R_{1011} + R_{1013}} = 0,5 \quad (2.4)$$

where V_{FB} is the voltage at feedback pin, $V_{CORE} \approx 1V$ is the output voltage and R_{1011}, R_{1013} are resistors in the voltage divider (marked as in Appendix A). The overcurrent protection is set with resistors R_{1018} and R_{1019}

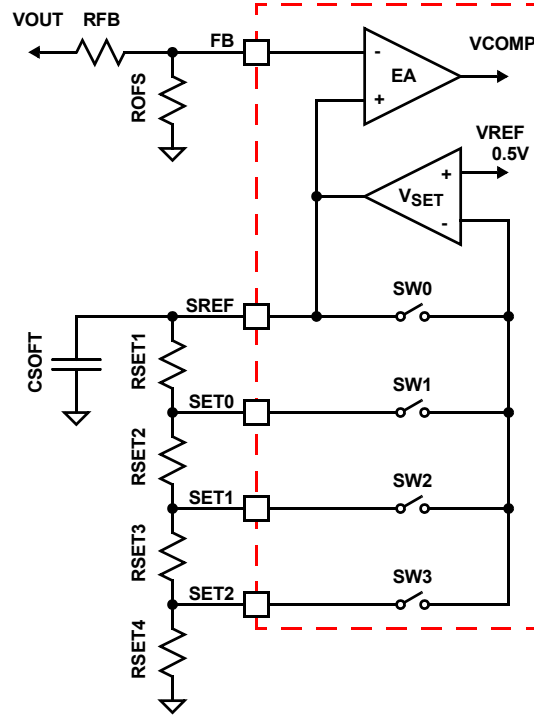


Figure 2.6: ISL95870B internal circuits(from [17])

it is calculated according to the datasheet [17]:

$$R_{1018} = \frac{I_{OC} \cdot DCR}{I_{OCSET}} = 12,353K\Omega \rightarrow 12,1K\Omega \quad (2.5)$$

where $I_{OC} = 10A$ is the overcurrent that triggers the shutdown of the controller, $I_{OCSET} = 8,5\mu A$ is the current sunk by OCSET pin, and $DCR = 10,5m\Omega$ is the resistance of the inductor L_{1001} . Switching frequency of this controller is selectable, and it is set to 500 kHz. The soft start capacitor is being charged until V_{SREF} reaches reference voltage (0.5V). It allows linear voltage ramp in time given by calculation:

$$t_{SS} = \frac{V_{SREF} C_{1001}}{I_{SS}} \approx 1,38ms \quad (2.6)$$

where t_{SS} is the time to reach the full output voltage, $I_{SS} = 17\mu A$ is the charging current and $C_{1001} = 47pF$ is the value of soft start capacitor. The minimal inductance of coil on the output of transistor stage is calculated and inductor with similar inductance is selected:

$$L_{1001} = \frac{V_{CORE} \cdot \left(1 - \frac{V_{CORE}}{V_{IN}}\right)}{f_{SW} I_{P-P}} = 1,3889\mu H \rightarrow 1,5\mu H \quad (2.7)$$

where $V_{IN} = 12V$ is the supply voltage, $f_{SW} = 600kHz$ is the switching frequency, $I_{P-P} = 1,1$ is the relative current ripple. The operation of voltage controller with its exact implementation was simulated in Spice-based

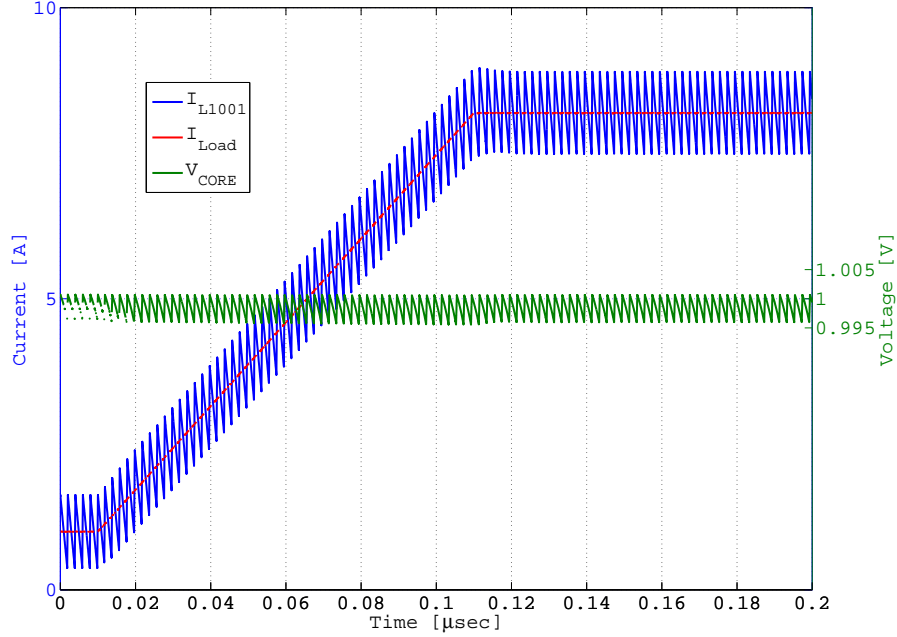


Figure 2.7: V_{CORE} simulation

simulator ISim from Intersil. The results are shown in Figure 2.7. A transient simulation under change of load current from negligible current consumption up to maximal possible power consumption of C291 (8.19 A) was executed. From the Figure it is clear that maximal ripple on V_{CORE} (approximately 5 mV) is in the recommended interval for C291 Core power supply (30 mV).

■ DDR power supply

The DDR3 voltages include V_{GVDD} and V_{TT} . V_{GVDD} is the main power for DDR3 module and DDR3 circuitry inside C291. V_{TT} is the termination voltage for the DDR3 signals. To generate both voltages at the same time ISL88550A controller is used. Its implementation is shown in Appendix A - page 11. It is step-down buck converter with external switching transistors. Both upper and lower transistors are in a single package (SI4816DY). Apart from the supply voltages, the controller provides also reference voltage V_{REF} for the detection of a logic level in the DDR3 memory chips. Switching frequency is set to 600 KHz. According to [18], the feedback pin V_{FB} is regulated to constant 0.7 V and the output voltage (with ripple voltage neglected) is calculated from:

$$V_{GVDD} = V_{FB} \left(1 + \frac{R_{1118}}{R_{1119} + R_{1120}} \right) = 1,5001V \quad (2.8)$$

where $V_{GVDD} = 1,5V$, $V_{FB} = 0,7V$ and R_{1118} , R_{1119} , R_{1120} are resistors as in Appendix A. The inductance of output inductor L_{1101} is calculated and

appropriate inductor is selected:

$$L_{1101} = \frac{V_{GVDD} \cdot (V_{IN} - V_{GVDD})}{V_{IN} \cdot f_{SW} \cdot I_{MAX} \cdot LIR} = 3,648\mu H \rightarrow 3,3\mu H \quad (2.9)$$

where $V_{IN} = 12V$ is the input voltage, $f_{SW} = 600kHz$ is the switching frequency, $I_{MAX} = 2A$ is the maximal consumption of memory module, $LIR = 0,3$ is the ripple current ratio. The overcurrent resistors and the output capacitor are overtaken from Skywan 5G design. The operation of DDR3 controller was not simulated since Spice model of this part was not available from Intersil.

IO power supply

IO Voltages include V_{3V3} for IO Operation, V_{1V5} for PCIe and V_{2V5} for Ethernet operation. All the voltages are generated with triple step-down buck converter with integrated switching stage. This architecture requires a minimal amount of external components. An interesting feature of this IC is soft start feature (SS1, SSS2, SS3 pins). The pin is a current source as long as its voltage is below 1V. A capacitor should be connected to this pin. When the circuit operation is started, the connected capacitor is continuously charged until it reaches 1V. The voltage on this pin controls the ratio of the output voltage in comparison to the final output voltage. If the pin is tied below 100 mV, the according output is disabled. A simple transistor stage was constructed on SS pins, which allows control of each output stage separately. It is not possible to reach this via ENA pin since it is shared for all outputs. The implementation of this circuit is shown in Appendix A - page 12. According to [19], each feedback pin is regulated to $V_{REF} = 800mV$ and the output voltages are calculated from:

$$V_{1V5} = V_{REF} \left(\frac{R_{1205}}{R_{1207}} + 1 \right) = 1,4926V \quad (2.10)$$

$$V_{3V3} = V_{REF} \left(\frac{R_{1210}}{R_{1208}} + 1 \right) = 3,3295V \quad (2.11)$$

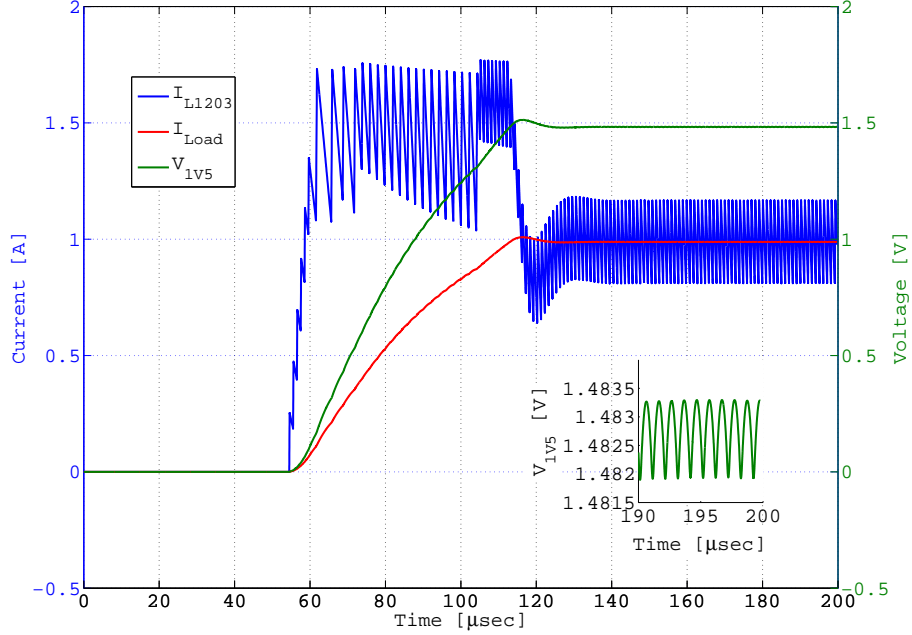
$$V_{2V5} = V_{REF} \left(\frac{R_{1206}}{R_{1209}} + 1 \right) = 2,4838V \quad (2.12)$$

where $R_{1207} = R_{1208} = R_{1209} = 10,5K\Omega$, $R_{1205} = 9,09K\Omega$, $R_{1210} = 33,2K\Omega$, $R_{1206} = 22,1K\Omega$. The inductances and capacitances of output stage components are calculated:

$$L_{1203} = \frac{V_{1V5} + V_D}{f_{SW}} = 3,7851\mu H \rightarrow 4,7\mu H \quad (2.13)$$

$$L_{1204} = 2 \frac{V_{3V3} + V_D}{f_{SW}} = 7,4590\mu H \rightarrow 6,8\mu H \quad (2.14)$$

$$L_{1202} = \frac{V_{2V5} + V_D}{f_{SW}} = 2,8838\mu H \rightarrow 3,3\mu H \quad (2.15)$$

Figure 2.8: V_{1V5} simulation

$$C_{1206} = \frac{33}{V_{1V5} \cdot f_{SW}} = \left(\frac{33}{1, 5.1} \right) = 22,109 \mu F \rightarrow 33 \mu F \quad (2.16)$$

$$C_{1207} = \frac{132}{V_{3V3} \cdot f_{SW}} = \left(\frac{132}{3, 3.1} \right) = 39,645 \mu F \rightarrow 47 \mu F \quad (2.17)$$

$$C_{1208} = \frac{33}{V_{2V5} \cdot f_{SW}} = \left(\frac{33}{2, 5.1} \right) = 13,286 \mu F \rightarrow 22 \mu F \quad (2.18)$$

where $L_{1203}, L_{1204}, L_{1202}$ are output inductors, $C_{1206}, C_{1207}, C_{1208}$ are output capacitors, $V_D = 0,4V$ is the voltage on the catch diode and $f_{SW} = 1MHz$ is the switching frequency. The inductance of main inductor L_{1201} is calculated from equation:

$$L_{1201} = \frac{V_{IN} \cdot \frac{5}{V_{IN}+5}}{2 \cdot f_{SW} \cdot \left[0,3 \cdot \left(1 - 0,25 \cdot \frac{5}{V_{IN}+5} \right) - \frac{I_{3V3} V_{3V3} + I_{2V5} V_{2V5} + I_{1V5} V_{1V5}}{50 \cdot V_{IN}} \right]} \quad (2.19)$$

$$= 6,7802 \mu H \rightarrow 10 \mu H$$

where $V_{IN} = 12V$ is the supply voltage and $I_{3V3} = 2A, I_{2V5} = I_{1V5} = 1A$ are maximal currents of each output. The power controller was simulated with LTSpice tool from Linear Technology under highest current load available for each output. The results are shown in Figures 2.8 , 2.9 and 2.10

2.3.2 Power sequencing

Since C291 in SKMM has power sequencing requirements, it is necessary to turn on the power supplies in particular order. ADM1069 is used as

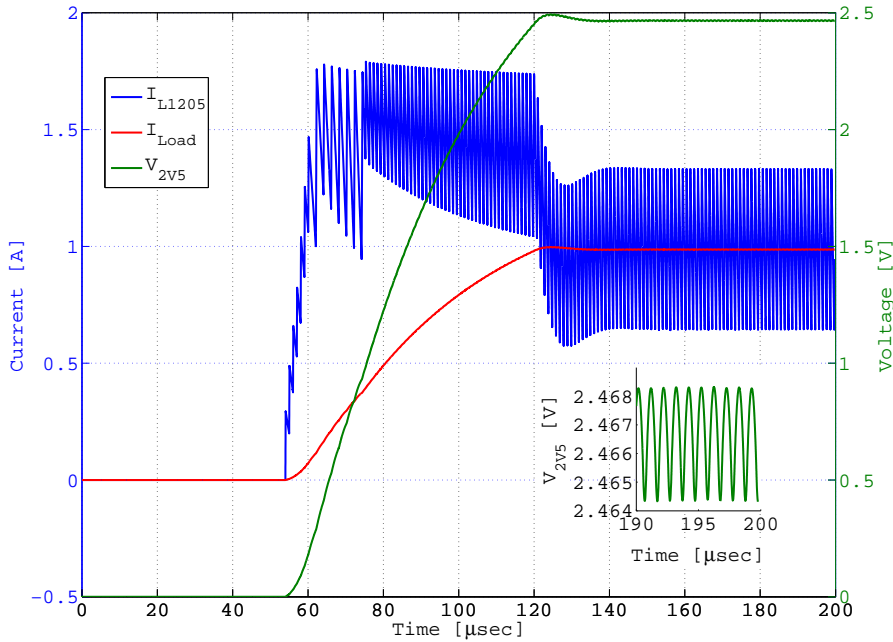


Figure 2.9: V_{2V5} simulation

sequencing circuit for the whole supply chain. It is a powerful chip with many features, such as programmable FSM for sequencing, AD converter, supply fault detection and supply margining. The circuit is accessed as a slave device on SMBus (compatible with I2C bus). This interface is connected to I2C interface coming from the motherboard. SW running on the motherboard can thus control the power sequencing on the Encryption board. Programmable outputs of ADM1069 drive ENA pins of each voltage controller. The power supplies are connected to input pins of ADM1069 for supply supervision. ADM1069 features an EEPROM where the whole configuration can be stored at first power-up or at manufacturing time. At every next power-up, the configuration is automatically downloaded from EEPROM and written to registers controlling the functionality. Thus the chip can be programmed to run the power up sequencing immediately after the input 12V power is available. More on this chip can be found in [20]. The implementation of the chip is shown in Appendix A - page 13.

2.3.3 Ethernet transceiver

As can be easily observed from Figure 2.1, the C291 CPU is connected to the motherboard via 1Gbit Ethernet (via COMe connector). The C291 contains TSEC (Tripple-speed Ethernet controller) which is used for this connection. A simple Reduced-Gigabit Media Independent Interface is used as communication standard between the Ethernet transceiver and C291. Since the

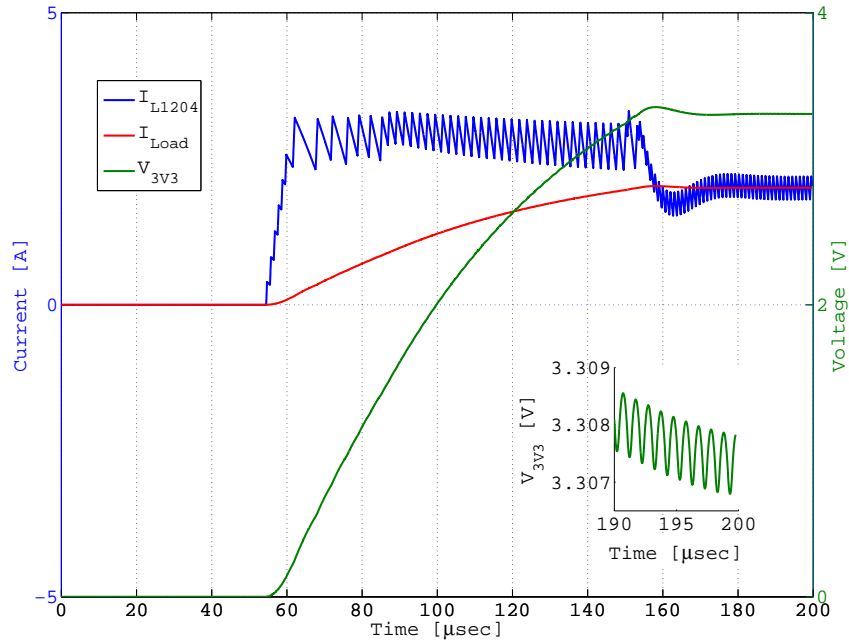


Figure 2.10: V_{3V3} simulation

motherboard of 5G is equipped with single port Ethernet transceivers, the same chip was chosen in the Encryption board. It is 88E1116R Ethernet transceiver from Marvell. It was not impossible to obtain the datasheet from Marvell due to NDA issues, so the chip is connected exactly as on the motherboard. Several configuration pins are connected via resistor configuration which makes it possible to reconfigure the chip just by populating different components. The implementation of this circuit is shown in Appendix A - page 5. Since Ethernet link is between C291 CPU and Ethernet switch on Skywan 5G motherboard, there is no need for using transformer coupling as it is common in the Ethernet designs. Instead, a cheaper and more compact capacitive coupling was chosen.

After the bring-up and manufacturing of prototypes (refer to 2.5) it was discovered that Ethernet transceiver consumes up to 4W. During this time the datasheet was already available, and a design error which forced the transceiver to sink big amount of current was discovered. This design error was fixed in second HW revision of the board and can be seen in the revision log in Appendix A - page 1. For the development time, a temporary HW workaround was introduced. The increased power consumption can be seen from Figure 5.1.

■ 2.3.4 Clock circuits

The design of clocks on the Encryption board is overtaken from the reference board. Three separate oscillators are used on the board. Note that these are not crystals! Crystal is just a passive piezo element which needs amplification circuits to provide a clock signal. If crystals are connected to an IC, it contains these amplification circuits. However, crystals must be placed close to the IC, due to noise immunity and minimum load by the capacitance of PCB trace. It is not possible to put crystal so close to the BGA package to satisfy these conditions. Thus oscillator circuits must be used. Clock signals are sensitive to noise which creates clock jitter due to power supply drifts. Simple LC filter is used on the power supply of every oscillator to minimize the noise. The implementation of oscillators is shown in Appendix A - page 8.

The PI6C557-05B IC from Pericom generates the PCIe reference clock. It is a clock generator IC used for clocking of HCLS or LVDS interfaces. Since clocks are often related to high EMI, it offers configurable spectrum spread.

Maintaining correct topology of routed clock signals during board layout is crucial. The proper topology is described in device datasheet [21]. The implementation of this circuit is shown in Appendix A - Page 4.

■ 2.3.5 DDR3

An implementation of DDR3 schematic can vary from easy up to a very tough task. To simplify the implementation and reduce manufacturing cost, soldered-down implementation was not used. Standard SO-DIMM DDR3 module was selected instead. SO-DIMMs are tackling the problems with routing topology (fly-by topology) and signal termination. Thus there are no terminators on the board, and the schematic design is reduced to a connection of signals with matching names. During the board layout phase, it was discovered that routing of byte lanes is impossible with a direct connection (first byte of SO-DIMM to the first byte of C291). The bytes were reorganized and routed as in Table 2.5. The DDR3 standard allows this implementation as long as the first bit of each byte remains in its place! This condition is important due to write leveling mechanism. Since the SO-DIMM socket is 64-bit (8 bytes) and the DDR3 controller of C291 is 32-bit (4bytes + ECC byte), 3 bytes of SO-DIMM remain unused. These require special treatment which is explained in 2.3.7. The implementation of DDR3 can be found in Appendix A - page 2. More about DDR3 can be found in [22].

■ 2.3.6 PCIe

Schematic design of PCIe is also an easy task. One must not forget to place the 100 nF capacitors on the TX pairs to have proper AC coupling as defined by the PCIe standard. During the layout phase it was discovered that direct pair routing is not possible due to the placement of COMe connector and C291 on different sides of the board. C291 supports byte lane switching on PCIe. The final lane connection is shown in Table 2.6. C291 supports the

Table 2.5: DDR3 bytes connection

SO-DIMM byte	C291 byte
1	4
2	3
3	2
4	1
ECC	0
-	5
-	6
-	7

byte lane switching only when full link width (x4) is used! Thus the actual PCIe layout gives the limitation that x4 link width has to be always used. Otherwise the link training mechanism will fail! The implementation of PCIe is shown in Appendix A - Page 4.

Table 2.6: PCIe lanes connection

C291 lane	COMe (Cavium) lane
1	4
2	3
3	2
4	1

2.3.7 Termination resistors

During the design of high-speed digital boards, one must distinguish between lumped and distributed circuits. Since propagation velocity of an electric signal is always finite, lumped circuit is only an abstraction which is usable for lower frequencies. If the circuit is fast enough, effects like reflections and ringing must be considered. Rule of a thumb for distinguishing the lumped and distributed circuit states: If the electrical length of rising edge is more than six times longer than the dimension of the circuit, it can be considered as lumped circuit. If a trace does not satisfy this rule, it must be viewed as a transmission line, and its impedance must be controlled. If there is an impedance change along the transmission line, reflections will occur. Maintaining the impedance of a PCB trace is a layout problem, and it is discussed in the “Board layout” section. However, if the input/output impedance or receiving/transmitting pin is not matching the impedance of PCB trace, reflections will also occur. To avoid this problem termination resistors must be placed.

First of all, we must think which digital interfaces we have to consider for termination. The fastest bus on the board is PCIe which is far beyond the lumped/distributed margin. The PCB trace of PCIe must have its impedance

maintained. From C291 datasheet, we can see that the PCIe pins are already internally terminated inside the C291. Additional external termination would have an undesirable effect.

The second fastest interface is DDR memory bus. This bus is also suitable for termination. Since we use the SO-DIMM module, it is not required to add additional terminators, because termination resistors are present on the SO-DIMM module. However, this statement is not entirely true in the case of Encryption board. Since the SO-DIMM module is 64 bit wide, it has 8 data bytes available. Each of the bytes is connected to separate memory chip. The C291 is equipped with 32-bit DDR3 memory controller, which has 5 data bytes (4 data + ECC). Thus 3 data bytes of SO-DIMM remain unconnected.

Let's now consider the operation of DDR3. The address and control signals are shared by each chip on the module in fly-by topology. This means that it is not possible to activate only some of the chips and leave others inactive during the access. The only way how to achieve this is to get a special module with several chips unpopulated. This fact does not cause a problem during the write cycle because C291 is driving the data bytes and unused bytes remain undriven. The problem appears during the read cycle. Control signals activate the chips corresponding to unconnected data bytes. These chips actively drive the data bytes and strobes.

If the unused data bytes of SO-DIMM were left floating, reflections could occur. If constructive reflection occurs, the voltage will reach higher values than during usual operation. This would lead to higher crosstalk and affection of data bytes which are in use! We can see that it is necessary to source terminate the unused data bytes and strobes towards VTT (Termination voltage) with $50\ \Omega$ resistors (the recommended impedance of data bus for DDR3).

Next interfaces suitable for termination are the RGMII interface and Ethernet. The output and input impedance of Ethernet transceiver makes it unnecessary to terminate the Ethernet differential pairs. RGMII interface is also suitable for termination. The termination which is selected in this case is end-termination on both TX and RX lines. Since the input impedances of C291 and Ethernet transceiver are different, different resistors are chosen on TX and RX lines. The input impedance of a pin is a serial combination of termination resistor and input resistance of the pin. The values of termination resistors are overtaken from the reference design. The remaining interfaces such as SDHC, I2C or UART does not need to be terminated.

■ Other parts of the schematic

The schematic includes other minor parts. An I2C temperature sensor ADT7461ARMZ is implemented to provide monitoring of C291. Status LEDs were added to indicate the most important status signals of the CPU. A small EEPROM is placed on the I2C bus. Both UART interfaces are routed, first to the COMe connector and second to a header for debugging. The only non-volatile storage (apart from EEPROM) which is suitable for booting the CPU e.g. in SKMM mode is the microSD card usable with SDHC interface.

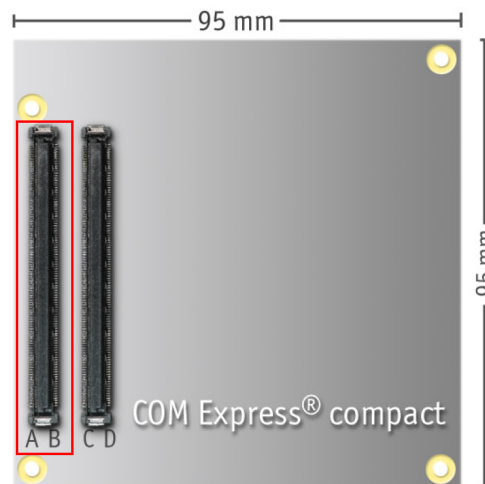


Figure 2.11: COMe module dimensions

2.4 Board layout

PCB layout is one of the most complex tasks from HW design, and it requires a lot of time. Due to this reason large projects are placed and routed by specialists, layout engineers. This project is already big enough to be routed by a specialist. However, it was decided that PCB layout will be part of the design, with the purpose of gaining the layout experience. The board layout took approximately 150-200 hours of work (placement and routing). Before start of the layout, the footprints for each component were created. Some of the footprints were overtaken from public libraries (e.g. DDR3 SO-DIMM socket, microSD slot). After the design of footprints, each schematic component was assigned with a specific footprint. This design step was overlapping with schematic design. Several components were available in multiple packages. When choosing the particular package (footprint) many aspects (size, price, availability, usage in original/reference design) were considered. After the design of footprints, board outline was drawn based on the COMe standard. The dimensions of the board along with the used connector are shown in Figure 2.11.

2.4.1 Board stack-up

Board stack-up refers to a order of copper layers, a thickness of copper layers, core or prepreg thickness, material permittivity, and treatment of bottom and top layers. It was decided that the same stack-up as on the motherboard of Skywan 5G will be used. This leads to simpler manufacturability from the same manufacturer and lower production costs. The stack-up consists of 10 copper layers, solder mask and golden coating on top and bottom layers. Silk-screen layers are not used in the design. The thickness of the stack-up is 1,632 mm and it is demonstrated in Figure 2.12.

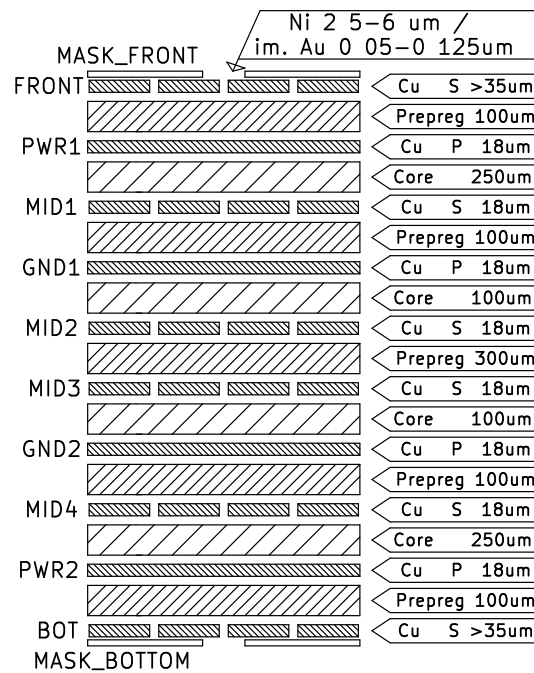


Figure 2.12: Board stack-up

2.4.2 Component placement

Before the start of routing, components were placed on the board. Placement of the components was driven by many constraints such as space limitations, interface accessibility, EMI considerations. The result of the placement is available in Appendix B. C291 is the main component, and it is placed on top. As discussed in 2.1.4, C291 needs to be cooled by a heatsink. The space between the 5G motherboard and encryption PCB is roughly 8 mm. Fitting the heatsink on bottom side was impossible. The size of heatsink makes it impossible to place DDR3 socket on the same layer as C291. Because of this reason SO-DIMM socket is placed on the bottom side. A socket with a height of 4 mm was chosen to fit between the board and motherboard.

The main EMI concern is crosstalk from power supply parts to the rest of the board. Due to this reason power supply chain is implemented in the upper side of the board, while rest of the design is spread across the board. Components belonging to each power controller are grouped as close as possible, to avoid current loops with a large area, thus maximizing the noise immunity according to Faraday's Law of Induction. Minimizing the current loops also affects noise radiation according to Ampers law, which is even more important.

The C291 is placed in the middle of the board allowing the interfaces connections from all sides. The C291 is rotated by -90° to make all the interfaces easily connectible. COMe connector is placed according to the standard.

■ 2.4.3 Transmission line impedance

As discussed earlier, the characteristic impedance of transmission lines must be maintained for fast signals. Characteristic impedance of loss-less line is given by formula:

$$Z_0 = \sqrt{\frac{L}{C}} \quad (2.20)$$

where Z_0 is characteristic impedance, L is inductance of the trace, C is capacitance between trace and nearest ground/power plane. In the case of Encryption board following interfaces must be routed by maintained impedance:

- DDR3 Memory. The impedance recommendations and layout guidelines are available in [23].
- PCIe. The impedance recommendations and layout guidelines are available in [24].
- Ethernet. The impedance recommendations and layout guidelines are available in [25].

The remaining interfaces are slow, and their transmission line impedance is not critical. However, to have the design well organized, other interfaces are routed with a single-ended impedance of 50Ω . The characteristic impedance depends on the trace thickness, width, distance from the nearest ground/power plane, layer in which is trace routed and the permittivity of isolation material between the copper layers. There are two main types of PCB transmission lines: Microstrip and Stripline. Microstrip refers to a transmission line on top or bottom layer with capacitance towards one reference plane. Stripline refers to a transmission line in the inner layer between two power plane layers. The equation 2.20 does provide an intuitive insight into transmission line impedance. However, it does not calculate the impedance from physical parameters (thickness of PCB trace and material). When calculating the transmission line impedance one can use analytical approach and derive the equations analytically, which is complex and it requires deep knowledge of electromagnetic field theory. Such derivations are beyond the scope of this thesis. An alternative is to use quantitative approach and estimate the equation from electromagnetic field simulations (Simulators use numerical methods like FEM). The output of this method is usually simple equation relating physical dimensions and transmission line impedance. There are several sets of estimated equations and online tools which can be used to calculate the impedance. Most of the tools have proven to be unreliable, including the native KiCad tool. The equations from [26] are used since they match the calculations from Skywan 5G layout. The equations for impedance calculations are listed in Table 2.8. The calculated results are listed in Table 2.7. In both tables Z_0 refers to single ended impedance and Z_{DIFF} refers to differential impedance within a pair.

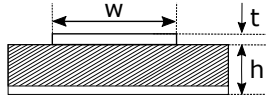
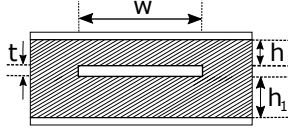
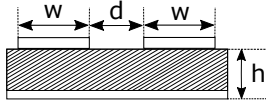
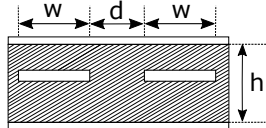
Table 2.7: Impedance results

Signal group	Layer	Z_0 [Ω]	Z_{DIFF} [Ω]	Required Z_0 [Ω]	Required Z_{DIFF} [Ω]	w [mm]	d [mm]
DDR3 Address	Mid1	49	-	50	-	0,1	-
DDR3 Address	Top	48	-	50	-	0,15	-
DDR3 Clock	Mid1	49	83	40-50	75-95	0,1	0,1
DDR3 Data	Mid4	49	-	50	-	0,1	-
DDR3 Strobe	Mid4	49	83-90	50	75-95	0,1	0,1
DDR3 Data	Mid3	49-51	-	50	-	0,1	-
DDR3 Strobe	Mid3	49-51	81-91	50	75-95	0,1	0,1
PCIe	Top/ Bot- tom	53	94	55	82	0,125	0,15
PCIe	Mid3	51	88	55	92	0,1	0,175
PCIe	Mid4	49	89	55	92	0,1	0,175

2.4.4 Routing

The KiCad autorouter was not used, and all signals were routed manually. This approach takes more design time but it provides a better overview of the whole board. Additionally, autorouter has no support for routing of high-speed interfaces, which require special attention. Each interface has set of rules (trace impedance, track clearance etc.) which must be followed. These rules are available within the specification of given interface (refer to 2.4.3), manufacturer's application note or professional layout seminar available online. These sets of rules are usually strict, and it is impossible to create a design without corrupting any of the rules (e.g. keeping track clearance under BGA package). The specifications usually do not mention which rules can be corrupted and how much. It is possible to use a FEM simulator to find out whether the particular layout will be functional or not (e.g. whether crosstalk between conductors will not corrupt signal integrity). This approach requires expensive software (FEM simulator) and it is not viable for this project. Most of the design guidelines recommend performing simulations. As already mentioned, it is not possible to meet all the design guidelines due to physical limitations (package type, board size, vias, etc.). This is where the intuition and experience step in. Due to this reason, an experienced layout engineer is a valued employee, even more, an expert in sub-areas such as RF

Table 2.8: Impedance equations

Topology	Demonstration	Impedance equation
Microstrip		$Z_0 = \frac{87}{\sqrt{\epsilon_r + 1.41}} \ln \left(\frac{5.98h}{0.8w+t} \right)$ (Valid for $\frac{w}{h} \in (0.1, 3)$)
Stripline		$Z_0 = \frac{80}{\sqrt{\epsilon_r}} \ln \left(\frac{1.9(2h+t)}{0.8w+t} \right) \left(1 - \frac{h}{4h_1} \right)$
Differential pair in microstrip		$Z_{DIFF} = 2Z_0 \left(1 - e^{-2.9\frac{d}{h}} \right)$
Differential pair in stripline		$Z_{DIFF} = \frac{174}{\sqrt{\epsilon_r + 1.41}} \ln \left(\frac{5.98h}{0.8w+t} \right) \left(1 - 0.48e^{-0.96\frac{d}{h}} \right)$

design, analog design, high-speed design or low power design. A useful starter into a design of digital boards is in [16, 27]. These books were irreplaceable sources of information during the implementation and helped with issues such as ground loops, crosstalk, terminations or reflections.

The routing was executed in following order:

1. Routing of local signals.
2. Routing of high-speed buses like DDR3, Ethernet and PCIe.
3. Routing of global connections and slower buses.
4. Design of power and ground planes

■ Routing of local signals

At this point of the design, C291 pins were routed to vias (under BGA package). Power supply controllers and surrounding components were connected. An example of IO power supply controller is shown in Figure 2.13. Other power supply controllers were routed in a similar manner. The datasheet of each power supply controller contains the recommended layout (e.g. in [19]), which was used as a reference during the design. Connections of decoupling capacitors, pull-up and pull-down resistors were also part of this design step. An example of such decoupling capacitor connection under the BGA package of C291 is shown in Figure 2.13.

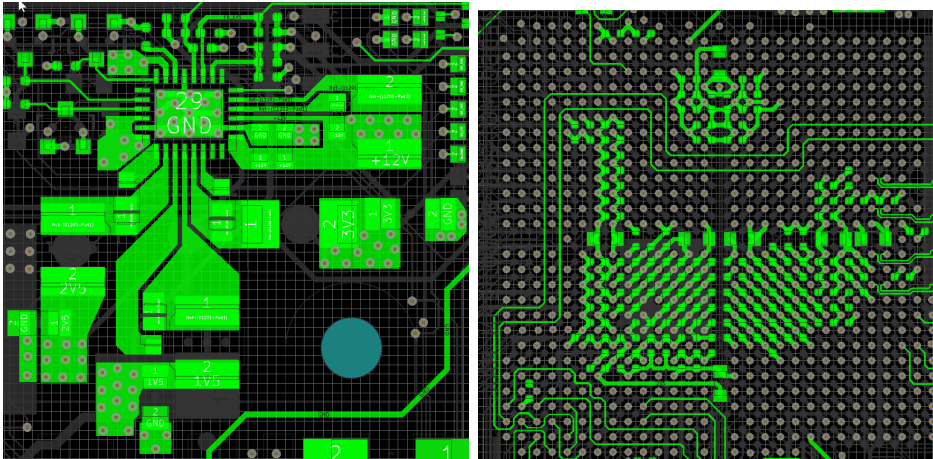


Figure 2.13: Routing of local signals (IO power - left, decoupling capacitors - right)

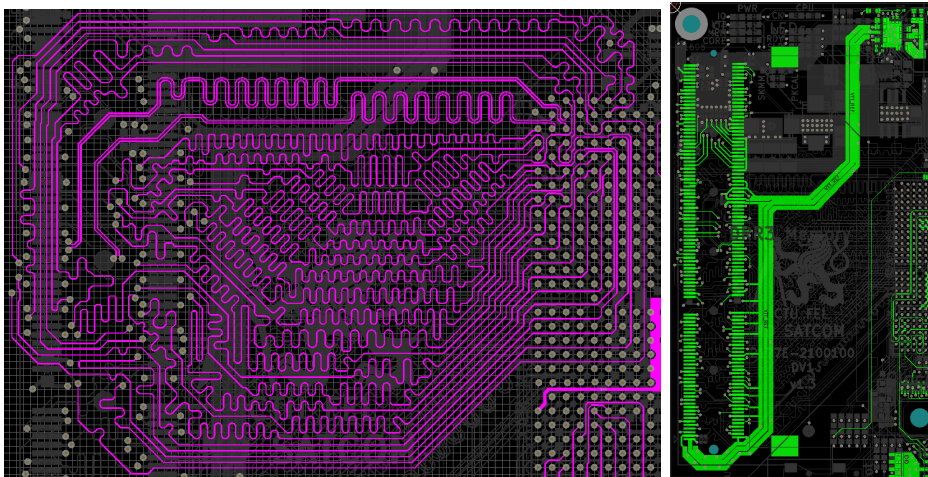


Figure 2.14: Routing of DDR3 (address signals - left, reference voltage - right)

■ Routing of high-speed buses

DDR3 was routed to the SO-DIMM socket as first. Address signals were routed out from BGA and then routed to pins of the connector. All the address and control signals were length matched to 74 mm (control signals are further described in the DDR3 standard). These signals were routed in Mid 1 layer and are shown in Figure 2.14. As next data bytes and appropriate byte strobes were routed and length matched to 54 mm in layers Mid 3 and Mid 4. An example is shown in Figure 2.15. The routing of a reference voltage and termination voltage was left at the end, and it is shown in Figure 2.14. Note that reference voltage is surrounded by ground traces to increase noise immunity. According to [28] this is one of the main pitfalls in DDR3 design.

The PCIe was routed as next. This interface is the most critical interface since it is using the highest frequency signals (up to 5GT/s). Each signal pair is routed with adequate separation from surrounding signals and pairs. Each

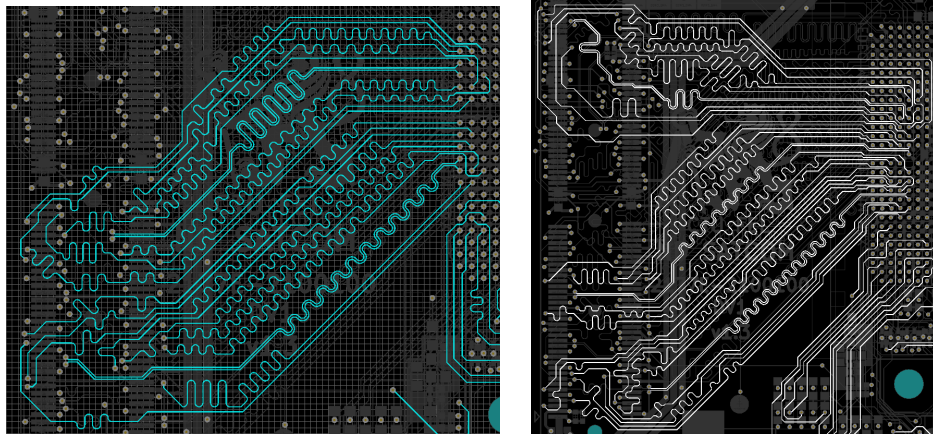


Figure 2.15: Routing of DDR3 data bytes and strobes

pair is deskewed at near end, in each layer. Due to applied lane swapping, the routing is straightforward. Routing of PCIe lanes is shown in Figures 2.16.

Ethernet interface was routed in the last step. Since this interface is of no importance in the current use of the encryption board, its layout demonstration is omitted.

■ Routing of global connections

At this point I2C, UART, SDHC, interrupts, configuration and status signals were routed. There is no need for length matching or strict impedance control on the slower buses. However, all the signals were routed by 50 Ohms impedance just to keep order in the design. This step would be alternatively routable by Autorouter.

■ Design of power and ground planes

As can be observed from the board stack-up there are two power supply planes and two ground planes on the encryption board. In this step, copper zones were created in each layer to distribute all the supply voltages. When using copper zones it is important not to have plane voids over high-speed tracks! Such a topology would corrupt the impedance of the transmission line and lead to unwanted reflections. The ground planes are divided only into two parts: DDR ground and rest of the design. Since there are only two power plane layers and six supply voltages, each power plane is divided into more copper sections, each section covering one supply voltage.

■ 2.5 Board manufacturing

After finishing the design and resolving all DRC errors, five prototypes were manufactured. A German company MSC Technologies did manufacture the

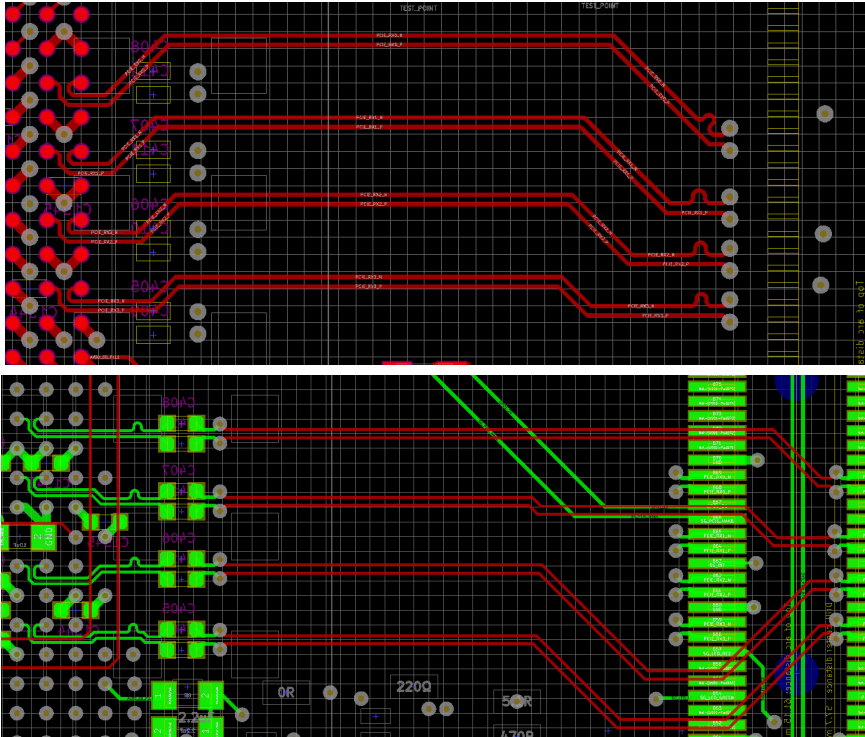


Figure 2.16: Routing of PCIe (RX pairs - up, TX pairs - down)

prototypes. MSC Technologies is a producer of embedded products, therefore it can take care of ordering the components, manufacturing the PCBs (over another external company) and populating the board. The feedback from the company was a valuable asset for the author of this thesis. Several errors were discovered in the first version of manufacturing materials. After fixing these mistakes and few iterations with manufacturing data, the board prototypes were ready. This step of the project took up to 6 weeks due to different technical and organizational aspects. All issues in the first design of the board were processed into a separate document which provided input for next hardware revision (which is being manufactured at the time of thesis submission). The manufacturing of prototype is more flexible than series manufacturing, and small mistakes in any part of the design can be manually fixed. The manufacturing of series requires no mistakes since manual fixing in 100s of boards is costly. The design practices which must be followed are known under abbreviation: DFM (Design for Manufacturing). Some of the mistakes which were tolerated in the manufacturing of prototype are:

- A slight mismatch in footprint. For example, microSD connector footprint was not exactly matching the prototype. It was possible to populate the connector by hand, but it would be unacceptable for the automated population.
- Using THT UART connector. The prototype does contain a through-hole connector for UART. Usage of THT connectors is not recommended

for series manufacturing since it is more expensive than surface mount technology.

The board manufacturing formats did include gerber files of all the copper layers, solder paste, solder mask, epoxy layer, placer file, BOM, drilling file and “GenCad” file for the testing of the board. The prototype and motherboard (in operation with prototype) are shown in Figures 2.18 and 2.19.

2.6 EMC consideration

The Encryption board does not have to satisfy any EMC rules on its own. The board will always be used with Skywan 5G. The Skywan 5G meets Class A requirements from [44]. The proper EMC performance of Skywan 5G is achieved by 2 mm thick aluminum chassis. It is better to use a robust chassis than risk possible HW redesign due to EMC problems. EMC performance of Skywan 5G with the Encryption board was not measured during this thesis, and it will be measured at the time of EMC certification. The robust chassis provides significant reserve (refer to Figure 2.17) in EMC performance without the Encryption board. Due to this reason, EMC measurement with the Encryption board was not executed, and it is assumed that there will be no problems with EN55022 certification.

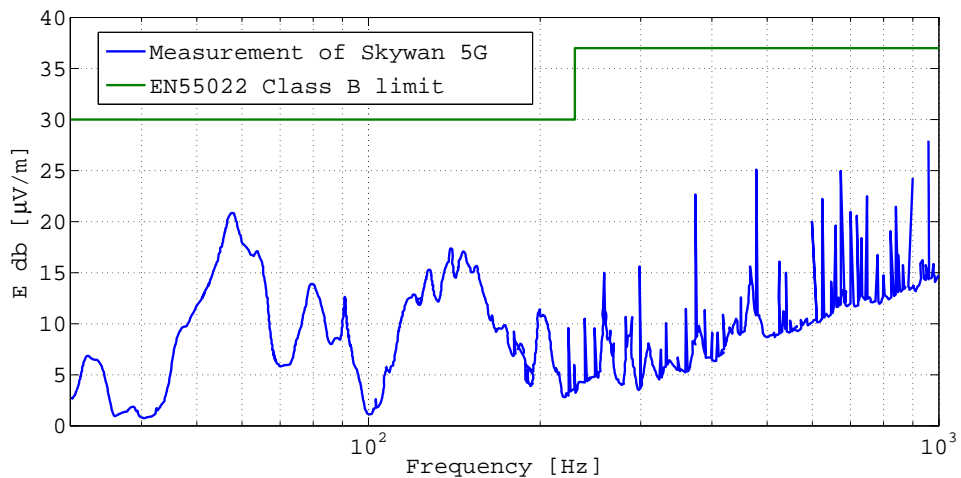


Figure 2.17: EMC measurement of Skywan 5G

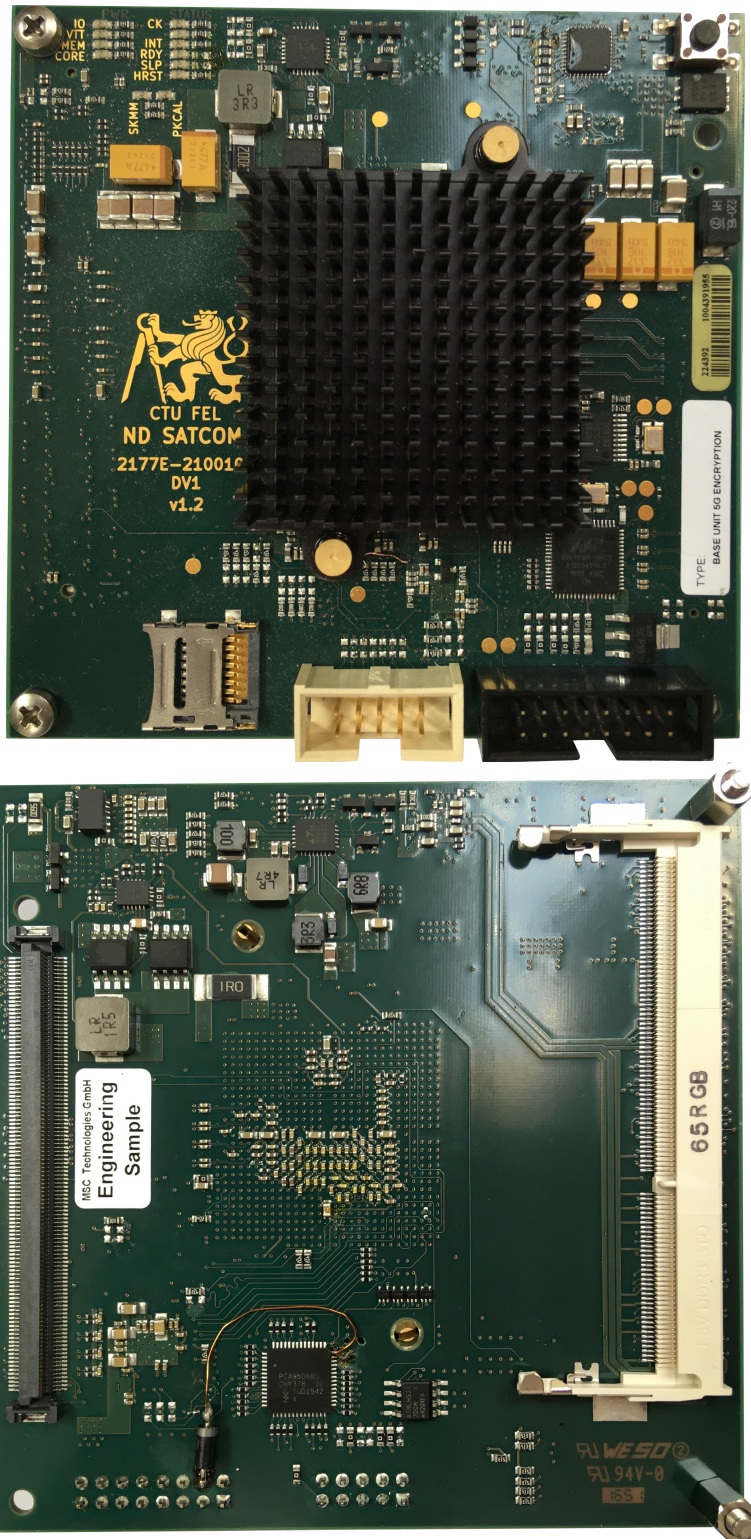


Figure 2.18: Board prototype

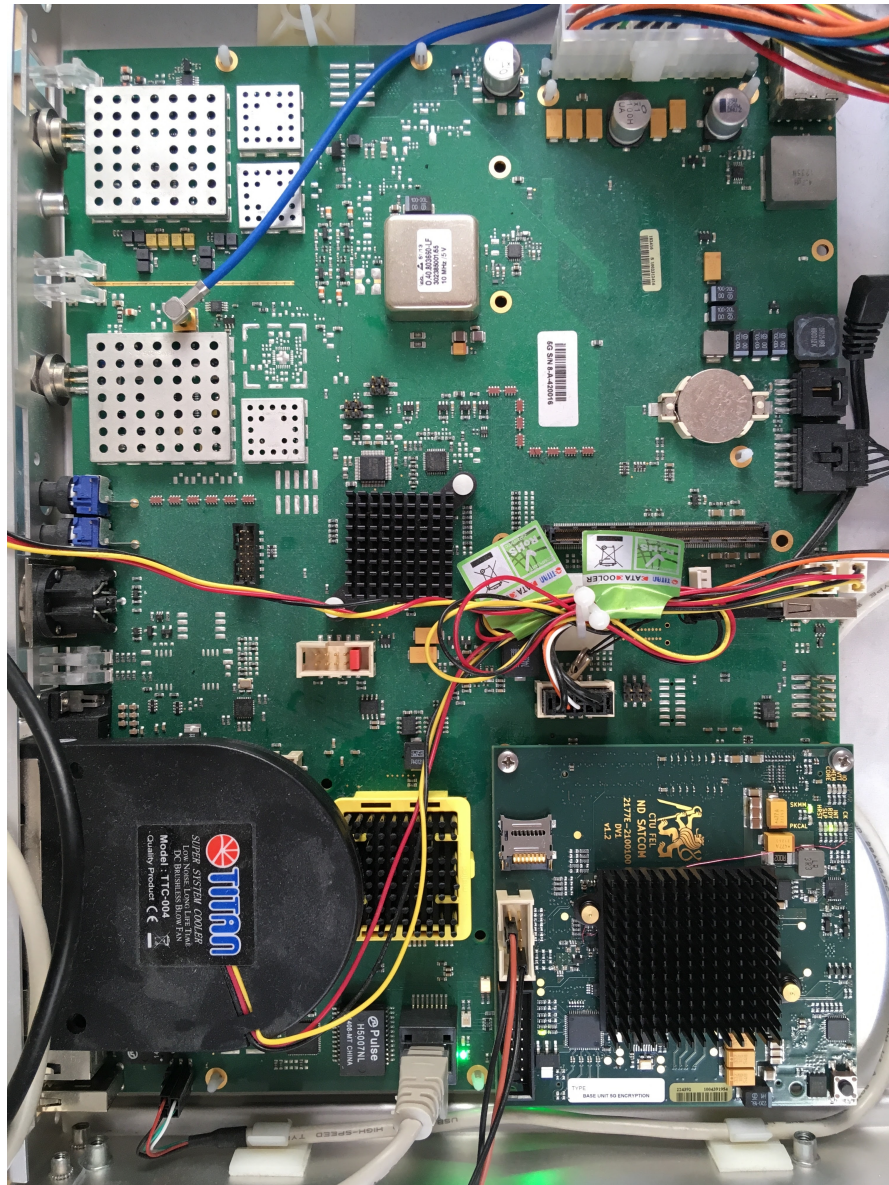


Figure 2.19: Board plugged into the motherboard

Chapter 3

HW/SW Interface

For booting the C291 properly, it was necessary to implement the drivers for the ADM1069 power sequencer and PCA9505/06 IO Expander. As mentioned earlier these circuits handle Power sequencing and POR Configuration. Since both circuits are accessible as Slave devices on the I2C bus, it is possible to use Linux kernel driver [29] and access the devices from I2C Master under Linux on Cavium CPU. The driver is convenient since it handles the low-level implementation of I2C protocol such as generation of start condition and stop condition. It exposes a device file which supports system calls such as “write” or “read”. This interface is universal, but it does not allow to use the full potential of both devices. ADM1069 supports SMBus extension of I2C protocol, and it offers many special commands such as EEPROM Page erase, Block read, Block write or Set address command. More on the special SMBus commands of ADM1069 can be found in [8]. These special commands cycles are not achievable with simple “read” or “write” system calls. It is necessary to use “ioctl” system call with “I2C_SMBUS” macro and provide the read/write buffer with the SMBus frame according to circuit specification. Additionally, ADM1069 supports voluntary Packet error correction (PEC) for each access. This option was also implemented via custom “ioctl()” calls. However, it is disabled in the final implementation of communication due to problematic behavior.

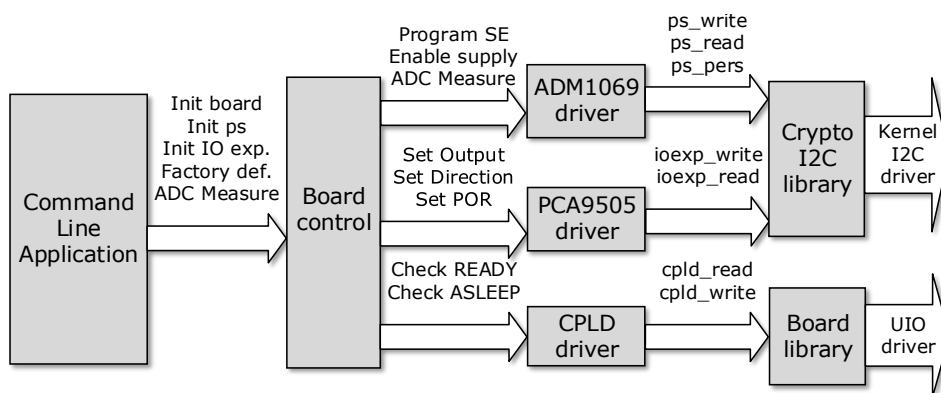


Figure 3.1: Structure of initialization SW

The application level drivers are implemented in Linux userspace. The structure of the initialization SW is shown in Figure 3.1. All the functions for interaction with I2C kernel driver are grouped in one source file `I2C_crypto.c`. Since some of the special commands contain multiple I2C frames (like set address and subsequent block write, set address and subsequent page erase), it is important to execute them without interruption by another frame to the same device. The results can be unexpected then. Due to this reason, each access function is locking a mutex. All the accesses must be executed over the implemented command line tool (not from another process) to be effective! Above the I2C library, each circuit has its dedicated wrapper of register-level functionality. Each register is described by a C structure to create hardware abstraction. With bitfields, all structures are implemented to match the register formats in the circuits. A pointer to such a structure is directly passed to the I2C access function without the need of bit masking. However, one must take care of byte endian conversions with functions such as `htole64` or `le64toh`.

The application functionality is abstracted to “Board control” library. On top of the drivers and “Board control” library a command line application is implemented, allowing simple manipulation with both circuits as well as CPLD on the 5G motherboard from Linux terminal. The screenshot from the terminal is shown in Figure 3.2, showing all the available commands. The tool can be started with “gui” argument starting the application or executed from Bash script with the same arguments as the commands in the application. The most important command is “brd init” which runs the Power sequencing and applies POR configuration. Other commands can execute standalone functionality and were used during debugging. The whole application is built with framework for Skywan 5G SW, and it is inserted into SW repository

3.1 Power sequencing

ADM1069 offers a programmable Sequencing engine (SE) which can control circuit outputs and provide supply supervision. A simple Supply fault detectors (SFDs) can be programmed on each input. Such a supply fault detector can be programmed to generate a warning if an input is above or below a certain threshold. The output of SFD can then condition state transitions in the SE. When using this method, SE configuration can detect if something went wrong and prevent incorrect functioning or even board damage. More on the exact functionality of SE can be found in the device datasheet. State transition diagram of programmed SE is shown in Figure 3.3. It implements a required power sequence as is defined in the datasheet of C291. The configuration of SE, input pins, output pins and ADC is located in a separate file (`power_seq_config.c`) and is described by constant arrays whose memory layout is matching the memory map inside the ADM1069.

ADM1069 features an additional multiplexed 8-channel ADC. Round-Robin planning circuit controls measurement on the ADC channels. The intention of this ADC is purely for debugging and voltage readback. The ADC is 12

```

Terminal - oille@skylx101:~
File Edit View Terminal Go Help
root@192:/nds/crypto#
root@192:/nds/crypto#
root@192:/nds/crypto#
root@192:/nds/crypto# ./crypto_brd_tool gui

Welcome in Shell tool for Encryption board configuration.
Type <help> to obtain list of available commands
>help
brd init force      Same as 'brd init' but PS EEPROM configuration is always applied!
brd init           Encryption board init function (Init PS + apply POR config)!
ps config get      Print Power sequencer EEPROM configuration info!
ps dest eeprom     Set the Power sequencer access destination to EEPROM
ps dest ram        Set the Power sequencer access destination to RAM
ps config set input Load the input pins configuration into Power sequencer
ps config set output Load the output pins configuration into Power sequencer
ps config set adc   Load ADC configuration into Power sequencer
ps config set se    Load the Sequencing engine configuration into EEPROM
ps eepr erase       Erase power sequencer EEPROM content!
ps eepr fd         Write all zeroes to Power sequencer EEPROM!
ps init           Run only Power sequencer initialization!
ps se check        Read Sequencing engine state in loop until final state is reached
ps adc single      Single measurement of supply voltages!
ps adc cont        Continous measurement of supply voltages
ps voltage         Turns on/off individual power supplies
ioe por get        Read the actual POR configuration in IO Expander!
ioe por set        Write the POR configuration to the IO Expander!
ioe hreset on      Activate the C291 Hard reset
ioe hreset off     Deactivate the C291 Hard reset
ioe 3stbufs on     Activate 3-State buffers
ioe 3stbufs off    Deactivate 3-State buffers
cpld get status    Read the CPLD status registers
cpld get gpio      Read the CPLD GPIO registers
quit              Exit the tool

```

Figure 3.2: Commands in Command line application

or 16 bits depending on averaging settings. As part of the driver, a voltage readback sequence was implemented. Two of the commands in Figure 3.2 are dedicated to measuring the voltage on each channel (`ps adc single`, `ps adc cont`). The result of such a measurement is shown in Figure 3.4. The voltage is calculated inside the driver (`power_seq.c`), based on input voltage range:

$$V = \frac{ADC\ value}{Range} \times Attenuation \times V_{REFIN} \quad (3.1)$$

Where *ADC value* is read from ADC registers, *Range* is 4096 for 12 bit and 65536 for 16 bit, *Attenuation* depends on voltage range on the input pin, $V_{REFIN} = 2,048V$.

Figure 3.5 shows the initialization sequence of the ADM1069. During the first power-up of the board, the configuration of Inputs, Outputs, SE and ADC is written into the EEPROM of ADM1069. This approach is less complicated and cheaper than programming the EEPROM during the manufacturing. The EEPROM has a limit of 10 000 erase/write cycles, thus it would be unacceptable to erase and write the configuration during every boot! One thus needs to “save” the information about the EEPROM content. According to the datasheet, the EEPROM contains a scratchpad segment (between addresses 0xF900 and 0xFAFF), which is perfect for this purpose. A set of registers was defined in the scratchpad segment, giving the details about the design which is stored in the EEPROM. These registers are then used during system initialization to determine whether the EEPROM contains a valid configuration of ADM1069. It is up to the application to update these

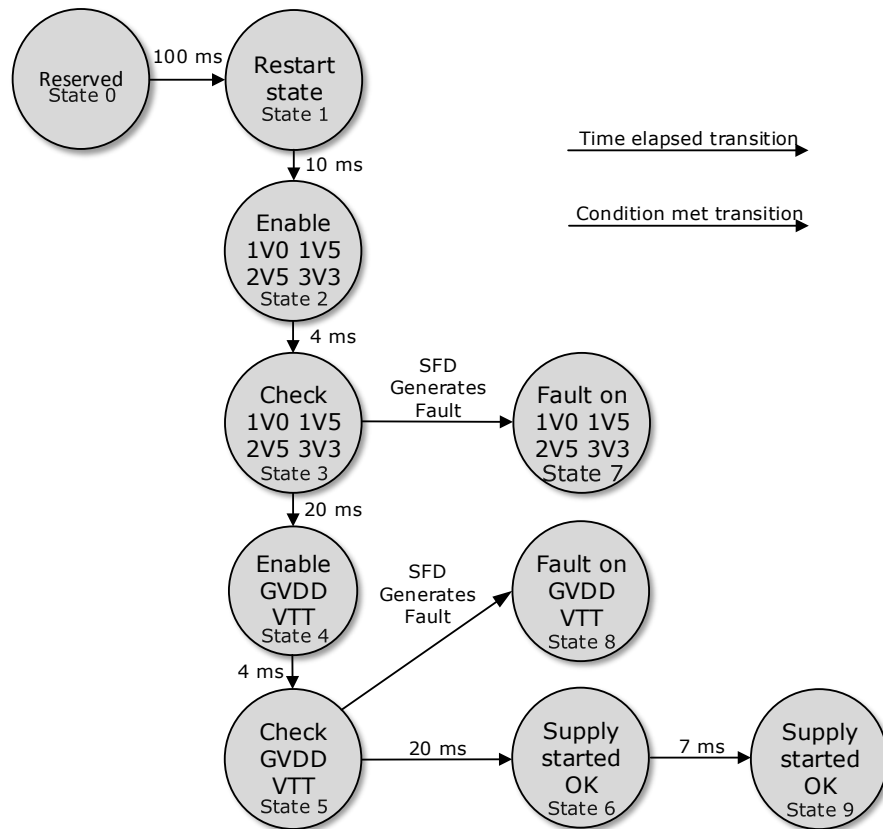


Figure 3.3: SE state machine

registers once the EEPROM is programmed. Note that ADM1069 comes with EEPROM containing 0x00 in each byte. Thus EEPROM needs to be first erased (each byte in 0xFF) and then programmed! The selection of the VALID bit polarity must be such that new circuit will be considered as a circuit with VALID bit inactive. The disadvantage is that once the EEPROM is erased this bit is in logic 1. Therefore SW reading the VALID bit would assume there is a configuration stored in the EEPROM. SW implementation relies on this effect, and it erases EEPROM only if a new configuration must be written to it. If there is an error during erase, SW takes care of setting the EEPROM into a factory default state (0x00 in the whole EEPROM). The registers containing the information about the design are shown in Table 3.1. Once the EEPROM is programmed, the configuration is downloaded into RAM memory and registers which control the actual outputs of the power sequencer. During every next boot-up, this step is done automatically by ADM1069. The meaning of the register bits from table 3.1 is following:

VAL Valid bit giving the details about the configuration in the EEPROM of ADM1069.

SE Sequencing engine is used in current configuration (Informational bit only).


```

Terminal - oille@skylx101:~
File Edit View Terminal Go Help
root@192:/nds/crypto#
root@192:/nds/crypto# ./crypto_brd_tool gui

Welcome in Shell tool for Encryption board configuration.
Type <help> to obtain list of available commands
>ps adc single

V CORE:      0.99 V
V DD:        1.51 V
V TT:        0.74 V
12V:         21.45 V
3V3:         3.29 V
2V5:         2.51 V
1V5:         1.48 V
>

```

Figure 3.4: Measurement of power supplies via ADM1069

Table 3.1: EEPROM scratchpad registers

Register name	Address	Bit							
		7	6	5	4	3	2	1	0
EEPRCFG _STAT	0xF941		OUT	IN	DAC	ADC	BB	SE	VAL
EEPRCFG _MIN	0xF942					MINUTE			
EEPRCFG _HOUR	0xF943					HOUR			
EEPRCFG _DAY	0xF944					DAY			
EEPRCFG _MONTH	0xF945					MONTH			
EEPRCFG _YEAR1	0xF946					YEAR1			
EEPRCFG _YEAR2	0xF947					YEAR2			

BB Black-box recording is used in the current configuration (Informational bit only, ADM1169 version only).

ADC Analog to digital converter is used in current configuration (Informational bit only).

IN Inputs are used in current configuration (Informational bit only).

OUT Outputs are used in current configuration (Informational bit only).

MINUTE Minute when the configuration was written.

HOUR Hour when the configuration was written.

DAY Day of the month when the configuration was written.

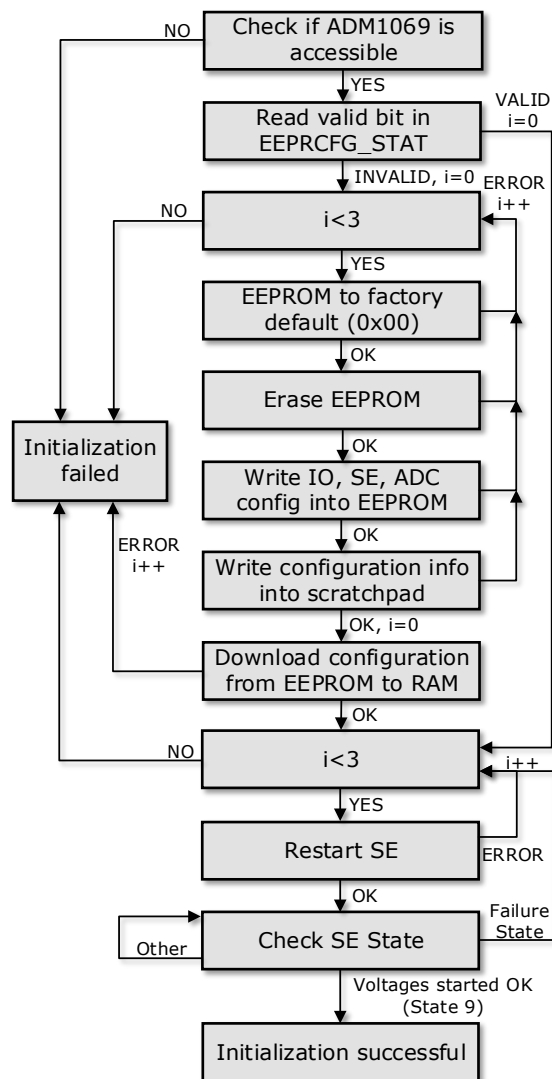


Figure 3.5: Power sequencer initialization sequence

MONTH Month when the configuration was written.

YEAR1 Two highest decimal numbers of year when the configuration was written. (e.g. 2016 -> 20)

YEAR2 Two lowest decimal numbers of year when the configuration was written. (e.g. 2016 -> 16)

3.2 POR configuration

As mentioned earlier the POR configuration is realized via 40 bit IO Expander PCA9505. Additionally, IO Expander pins are connected to control signals to the CPU. The most important signal is H_RESET_N. Pulling this signal low executes the POR. The C291 reset sequence is shown in Figure 3.6.

The application first pulls the H_RESET_N and TRST_N pins low, then it applies the POR configuration on the other pins. Then it deactivates the H_RESET_N after a small delay. The IO expander must also enable the output of three-state buffers (shown in Appendix A - page 15). These buffers are present since some of the POR pins operate at 2V5 during normal operation. During POR these pins operate on 3V3. If a pin is operating at 2V5, it is no longer desired to have even weak pull-up to 3V3. An indication of successful reset is on the READY pin of C291. This output is connected to the LED (LED711) and the CPLD on Skywan 5G motherboard (via COMe connector). LED indicating that C291 is reset can be seen in Figure 2.18, where the Encryption board is in operation. The POR configuration is shown in Table 3.3.

The POR configuration for PKCAL mode contains several necessary settings apart from CPU mode itself. PCIe must be set to EP mode. The CPU must be set to so-called Boot Hold-off mode which prevents the core from booting immediately after the reset. It gives a time to the External host (RC on PCIe) to load the firmware into internal SRAM of CPU. The clocking domains of C291 are also configured via POR. The device contains several clock domains. The SoC is clocked by CCB (platform clock), which is derived from SYSCLK input. CCB clock is set to 320 Mhz. This clock is used for most of the peripherals and internal memories. The e500 core has its PLL which locks on the CCB clock. The core is running at 640 Mhz. Another clock domain is DDR clock. Although the DDR is not used at the moment, its frequency is set to 800 Mhz.

Table 3.3: POR configuration

POR Config name	Value	Signal meaning	Setting
cfg_sys_pll	0x0	Ratio between SYSCLK input and CCB clock (platform clock)	4:1
cfg_sys_speed	0x1	SYSCLK frequency threshold	above 66 MHz
cfg_core_pll	0x4	Ratio between the e500 core clock and CCB clock	2:1
cfg_core_speed	0x0	Core speed configuration input	Core clock frequency is greater than or equal to 600 MHz and less than 1001 MHz

Table 3.5: POR configuration (continued)

POR Config name	Value	Signal meaning	Setting
sw_cfg_romloc	0x7	Boot ROM location (unused in the PKCAL mode)	SDHC (ready for boot from microSD card)
cfg_plat_speed	0x1	Platform speed configuration input	Platform clock frequency is greater than or equal to 320 MHz and less than 401 MHz
cfg_ddr_pll	0x0	Clock ratio between 100Mhz OSC clock input and DDR complex clock	8:1
cfg_ddr_speed	0x0	DDR complex speed configuration input	DDR data rate is less than 967 MHz
cfg_boot_seq	0x3	Boot sequencer configuration options	Boot sequencer is disabled. No I2C ROM is accessed. This is the default setting.
cfg_cpu_boot	0x0	CPU boot configuration inputs	CPU boot hold off mode. The e500 core is prevented from booting until configured by an external master
cfg_io_ports	0x0	Different I/O ports active on the SerDes	PCIe-x4 (5 GHz)
cfg_host_agt	0x0	Host/agent reset configuration input	Endpoint on PCIe interface
cfg_sb_dis	0x1	Secure boot	Disabled
cfg_svr	0x1	System version	C291
cfg_gpinput	TBD	General-purpose POR configuration vector to be placed in GPPORCR	TBD
cfg_eng_use	0x3	To be used in the future to control functionality	Default operation
cfg_chip_mode	0x1	PKCAL/SKMM mode	PKCAL (SKMM for debugging with UART)

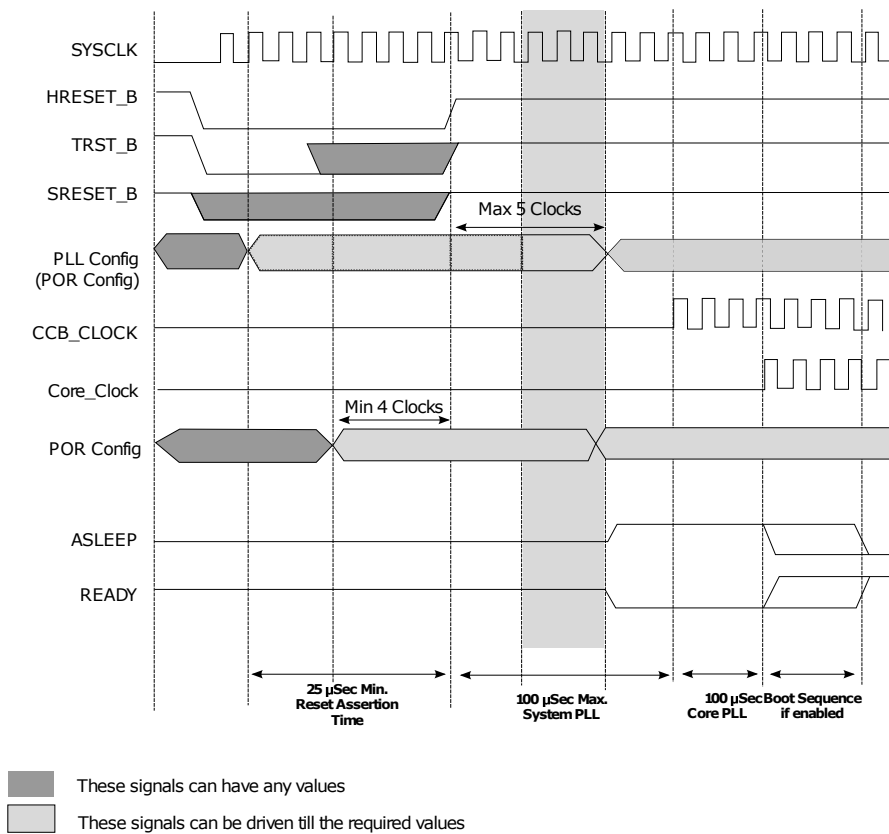


Figure 3.6: Power-on reset sequence (from [31])

3.3 Initialization sequence

Once the power is up and POR is executed, C291 is ready for PCIe enumeration. This process is fully described in [30]. The enumeration is performed by PCIe Root Complex (RC). MIPS CPU on the 5G motherboard is the RC. The enumeration involves identifying the EPs in the system and configuring them. It usually runs during boot of OS. Linux kernel contains PCI/PCIe framework which connects HW specific operations to an abstract layer. The enumeration process is then simplified to a function call which is commonly executed during “arch_init” phase of kernel initialization. Since the device is not properly reset at boot time (reset is executed from the Command line application), enumeration would not be successful and the device would not be detected. Due to this reason, PCIe enumeration is deferred. In the implemented system, PCIe kernel driver is modified to have the enumeration available on IOCTL system call. This system call is executed on the custom device file which is registered at boot time. With this approach, the enumeration can be executed once the whole board is initialized with the Command line application.

Since the device is accessed over PCIe, the HW initialization ends once the PCIe device can be seen in the system. From this point on, the SW takes

on. The complete HW initialization sequence is shown in Figure 3.7. After the sequence is executed, the device appears in the system and it is possible to detect it by “lspci” command on Cavium as in Figure 3.8. The whole initialization sequence is implemented in a Bash script which operates on the Command line application executed with argument “brd init” and on the device file controlling the PCIe enumeration.

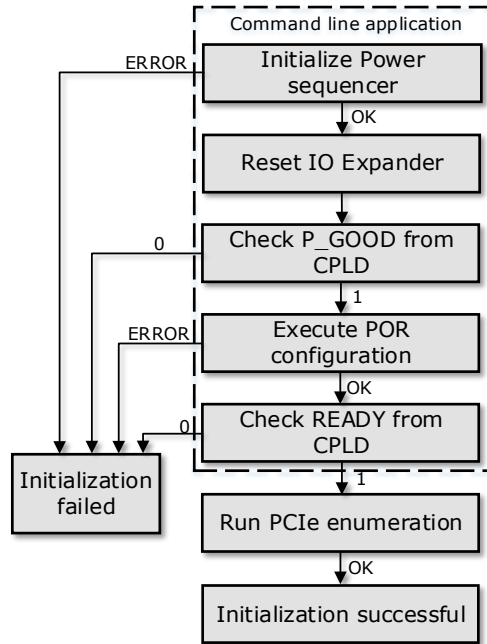


Figure 3.7: HW initialization sequence

```

Terminal - oille@skyfx101:~
File Edit View Terminal Go Help
root@192:/etc/init.d#
root@192:/etc/init.d# lspci -v
01:00.0 Power PC: Freescale Semiconductor Inc Device 0803 (rev 10) (prog-if 01)
  Flags: fast devsel
  Memory at 11b00f0000000 (32-bit, non-prefetchable) [disabled] [size=1M]
  Memory at 11b00f0100000 (32-bit, prefetchable) [disabled] [size=1M]
  Capabilities: [44] Power Management version 3
  Capabilities: [4c] Express Endpoint, MSI 00
  Capabilities: [88] MSI: Mask- 64bit+ Count=1/16 Enable-
  Capabilities: [100] Advanced Error Reporting
root@192:/etc/init.d#
  
```

Figure 3.8: C291 detected on PCIe

Chapter 4

SW integration

The software support for the C29x family of CPUs is good. The reference design used during the development contains complete source code (in C) of software required to use the C291 as an Encryption endpoint in the PKCAL mode. Without this support, it would be impossible to integrate the encryption features within this thesis. There are two main parts of the software: Host driver and Device firmware.

The driver was inserted into Linux kernel repository which is maintained by the development team of Skywan 5G. It is running on the host system (MIPS Cavium OCTEON CPU). The firmware is built separately with the NXP SDK, and it is running on the encryption board (C291 from NXP Semiconductors). These two CPUs communicate via PCIe. The communication is defined on abstract channels called "rings". Two types of rings are available: command rings and application rings. Command ring is used for debugging, and it offers commands such as ping device, get information about the device, or reset SEC engine. The application rings are used for enqueueing the encryption requests and handling the encryption responses. The overview of software operation is shown in Figure 4.1.

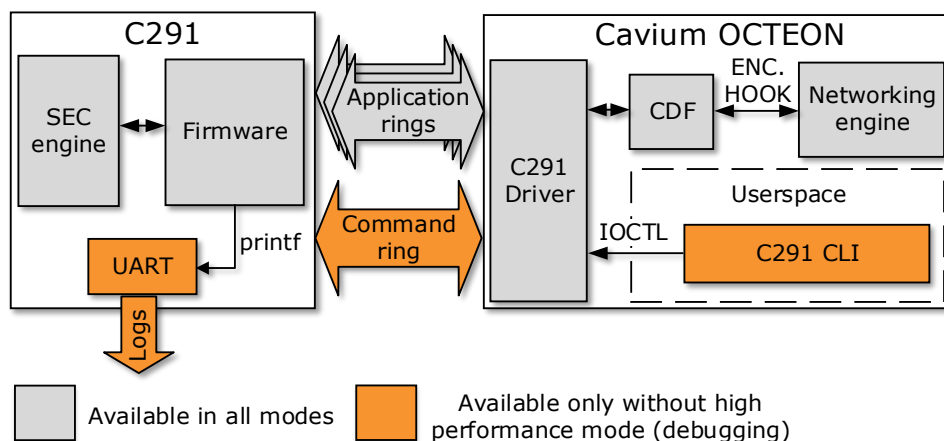


Figure 4.1: SW overview

The SW design is inherited from previous families of security processors such as P4080 which contained the same encryption accelerator as the C29x

family. The SW is written for two CPU architectures: PowerPC (both master CPU and CPU with encryption accelerator) and x86 (for master CPU only). The SW is intended for Linux kernel version 3.0.57. Porting the SW (host driver) to the MIPS architecture kernel version 2.6 has some peculiarities which are further discussed in this chapter.

4.1 Host driver

The host driver is a Linux kernel driver which connects the PCIe device and Linux Crypto Dev Framework. It is compiled as a kernel module, and it is loaded during the boot-time of the CPU. From Figure 3.8 we can see that the C291 as PCIe Endpoint provides two Base Address Registers (BARs). By default (after reset) the memory accesses to the first BAR are mapped to the CCSR registers (refer to [31]) of C291. CCSR registers are used to configure the device. Each peripheral has registers located in the CCSR registers region. With this approach, the device can be controlled over PCIe. The driver first sets up Local access windows (LAWs) which configure the physical address map of the C291. The internal 512 KB SRAM and L2 cache are mapped to single 1 MB long memory location placed between physical addresses as is shown in Figure 4.2. The second BAR is mapped to this memory location, and the driver copies the firmware into this memory region. The firmware also operates in this region and stores the details of rings, commands and encryption requests here.

The last page of SRAM is defined as boot window of C291 (located at address 0xFFFFF000). The core of C291 executes the first instruction from physical address 0xFFFF FFFC. From the mapping of SRAM and L2 Cache in Figure 4.2 it is clear that boot window overlaps with memory mapping inside the C291. Due to the POR configuration, C291 is in boot-hold off mode. Therefore, the core is not fetching any instruction until it is released by an external access. After LAWs are set and the firmware binary is loaded into SRAM/L, the driver releases the core for execution by clearing CPU_EN bit in ECM_EEBPCR register. During the build of the firmware, one has to make sure that the first instruction will be placed on the physical address 0xFFFF FFFC. A “reset_vector” attribute in the linker script is used for this purpose. More details of boot process can be found in [31].

The Crypto Dev Framework (CDF) is Linux kernel API (refer to [33]) which provides a uniform way of encryption for user-space or kernel applications. The applications which require encryption can use set of generic functions without worrying about the actual implementation of the encryption algorithms. Thus applications can make use of HW accelerators such as C291. The driver registers encryption algorithms in this framework and implements functions which encrypt the data in the C291 (or rather move the data into C291 and move it back after the encryption). The CDF provides two types of requests: synchronous and asynchronous. The synchronous encryption request moves the data into C291 and polls on memory descriptors for the completion of the encryption/decryption operation. The asynchronous move

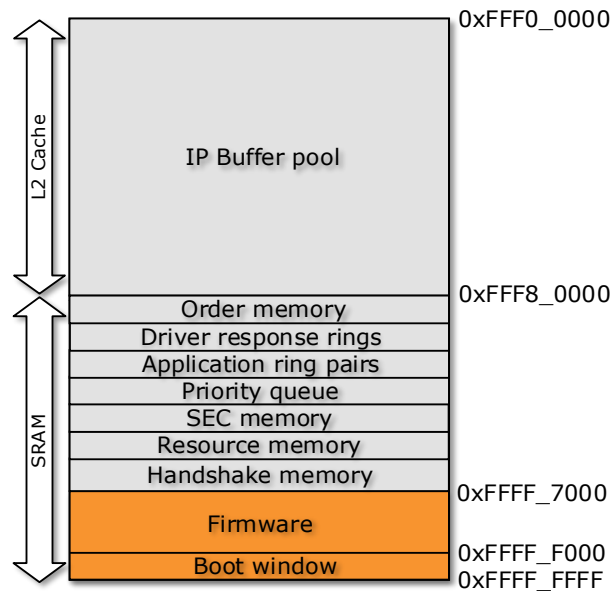


Figure 4.2: Physical memory layout in C291

the data and register a callback function. The callback function is called by the CDF to notify the caller about finishing the encryption operation. Thus the synchronous functions exit only after the encryption is finished and asynchronous functions exit immediately.

An important aspect of data transfer is its efficiency. Even the most powerful CPUs can achieve poor performance if they spend most of the time copying the data between various memory locations. The driver can natively interact with two types of DMA transfers. First one is the usage of native CPU DMA engines (this approach is unused at this point of the project as explained in 4.5). The second type is the PCI DMA framework which makes use of the Bus mastering capability of C291. Shortly, the encryption function (in Cavium CPU) provides a destination (local DRAM of Cavium) and source address (in the internal C291 SRAM) for C291, then it gives a command for the C291 to transfer the data and receives an interrupt once the transfer is done. This approach is further described in 4.3. The majority of DMA support remains unused and is a topic for further development.

4.2 Device firmware

The firmware is based on the opensource bootloader U-Boot, ported to the PowerPC architecture. The first part of the firmware is written in assembly language (as most bootloaders), and it takes care of necessary core initialization such as setting up MMU and TLB, configuring CPU branch predictor, setting up caches and reconfiguring the LAWS so that all the peripherals could be used by OS. The Uboot was modified for the purpose of C291 firmware. After initial setup, firmware loop is called instead of initialization function of OS. The firmware is written in C, and the call of

the initialization function creates a border between the assembly and C. The advantage of using U-BOOT is that there is no need to implement the low-level core setup since it is already implemented for PowerPC e500v2. Additionally, U-boot provides libraries with drivers for standard peripherals which are also part of other NXP CPUs.

After the firmware starts running, device firmware executes a handshake procedure between driver and firmware. It is run at every boot-up of the device. The handshake procedure is shown in Figure 4.3 and it is the first communication between the driver and firmware. It manages initialization of the communication rings inside the C291 and initialization of random number generator. The Handshake is executed via Handshake memory inside the C291 memory (refer to Figure 4.2).

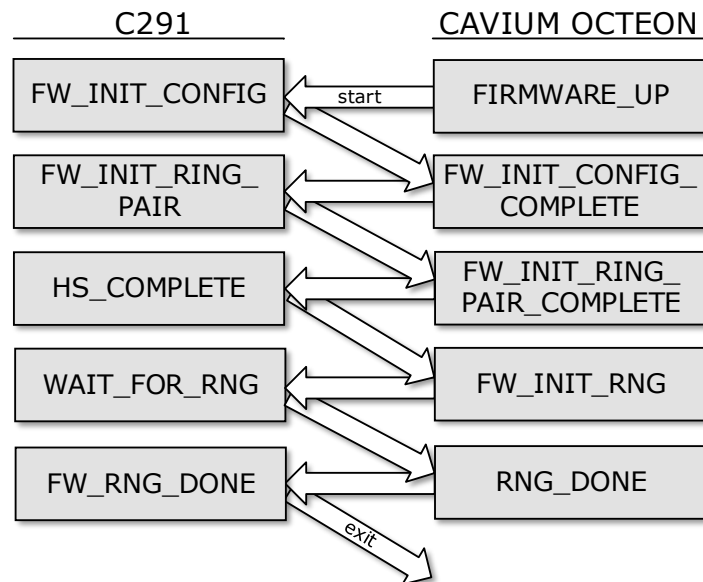


Figure 4.3: Handshake between the driver and firmware

After performing the handshake, the firmware is ready to accept the encryption requests from the driver. The logs from the firmware are provided on UART output of the C291 (when compiling with `DEBUG_PRINT` config). Custom UART/USB device was used via “gtkterm” Linux application to display the logs. Logging was disabled in the released version of the firmware and firmware was modified to achieve bring-up with logging. Once the debugging options were turned on, the constant character buffers of log messages (usually stored in assembly literal pools which are accessed by an immediate operand of the instruction) caused exceeding of the original firmware size (7 pages - 28 KB) and errors during linking. Respective linker script of Uboot (attribute `SYS_TEXT_BASE`) was updated, and firmware size was increased to 9 pages - 36 KB.

4.3 Encryption sequence

The driver and firmware interact in the processing of the Encryption and Command requests. The driver can operate on the memory of C291 and Firmware can also operate on the memory of Cavium since the memory allocated for the driver is plain (for reasons refer to 4.6.2). Following steps describe the processing of encryption (decryption is similar) request::

1. Application invokes CDF encryption function (e.g. “crypto_ablkcipher_encrypt”).
2. CDF calls function for encryption algorithm provided by the C291 driver (e.g. “fsl_ablkcipher_encrypt”).
3. The function propagates the key into the device (if needed) and copies the data (and metadata) into Input Buffer pool in the C291. A PCI DMA mapping is created on a single Output Buffer from Output Buffer pool in Cavium memory. Thus the Cavium will not interfere with this memory until it obtains the encrypted data. The data transfer is alternatively executable by DMA engine inside the Cavium. Finally, driver increases the counter of added requests (in C291 memory).
4. Firmware loop is polling on status counters of ring pairs. If there is a mismatch between the number of processed requests (“rp->cntrs->jobs_processed”) and the number of added requests (“rp->r_s_c_cntrs->jobs_added”) the firmware reads the encryption request from IP Buffer pool, enqueues it to SEC engine and updates counter of processed requests. If no, firmware attempts to dequeue finished encryption requests instead.
5. SEC engine executes the encryption on the data which are present in Input Buffer pool.
6. Firmware which is polling on the finished encryption request (step 4) recognizes that encryption operation was finished. Firmware copies the data (and metadata) over PCIe into Output Buffer in Memory of Cavium.
7. Firmware raises an MSI interrupt.
8. Interrupt service routine (ISR) processes MSI interrupt, and posts response processing on a work queue.
9. Response ring is checked for completed requests and response is parsed. The driver now recognizes that encrypted data are in Output Buffer in the local memory of Cavium.
10. PCI DMA mapping is released, CDF is notified about the completion of the encryption operation, and the callback function is called.

Table 4.1: Driver/Firmware pre-build configuration

Config name	Actual setting	Description
SDK_PATH	-	Local path to the installation of the NXP SDK.
P4080_EP	n	If the SW/FW is build for P4080 CPU
C293_EP	y	If the SW/FW is build for C29x family of CPUs
XXX_PRINT	y,y,y	Specifies whether debug, info or error logs are enabled (XXX stands for DEBUG, INFO, ERROR)
CONFIG_FSL_C2X0_HASH_OFFLOAD	n	If hash support should be compiled
CONFIG_FSL_C2X0_SYMMETRIC_OFFLOAD	y	If symmetric ciphers support should be compiled.
RNG_OFFLOAD	n	Random number generator support should be compiled in.
USE_HOST_DMA	n	Use the DMA engines.
HIGH_PERF_MODE	y	High performance mode. Disable command ring support.
VIRTIO_C2X0	n	Support virtualization.

4.6 Porting of the C291 driver

4.6.1 Per-CPU variables

First of the problems is the usage of “per-cpu” variables. A “per-cpu” variable is allocated for every CPU in multiprocessor (multicore) system. Since there is only one core accessing this variable, there are performance benefits in the memory access. The cache coherency problem practically does not exist since cache controllers do not have to synchronize the values between the caches of each core.

The framework for defining these variables is different in kernel versions 3.0 and higher than in older kernels [36]. The old framework makes it impossible to define an attribute of a structure which will be separate per each core of the multicore system. Several structures handling the processing of requests in the communication rings were implemented in this manner. It was possible to extract the attributes into separate variables or convert the attributes to

Table 4.2: Pre-run configuration

Property name	Description
depth	Number of requests which can be queued into the ring at the same time
affinity	Preference which core of multicore system should be bound to processing of this ring.
priority	Priority relative to the other rings. Rings with higher priority are processed before the lower priority ones
order	Processing of the requests in the ring should be in order.

standard “not per-cpu” ones. Since the driver always binds processing of a ring to the particular core, the synchronization of accesses to these structures is not an issue. The second approach was chosen, and these attributes were modified to standard variables (not “per-core”) and performance drawbacks were admitted.

4.6.2 Memory allocation

A major modification was the implementation of memory allocation for the driver. During the load of the driver module, the driver allocates memory for communication with the firmware whose size depends on the number of defined application rings. The default size is approximately 500 KB. The driver was using “pci_alloc_consistent()” function which ended up calling “kmalloc” with attribute “GFP_ATOMIC”. The name of the function gives a hint that the allocated memory is physically contiguous. The memory must be physically contiguous since there is no IOMMU which would handle the translation of non-contiguous logical address into the address on the PCIe bus and thus in C291.

Due to fragmentation it often happens that there is no physically contiguous block of desired size available. Allocation of memory fails in this case. One can use aggressive allocation and try it again, which is not only unreliable, but it can also result in a deadlock [34]. An alternative which is available in newer kernel versions is CMA (Continuous memory allocator) patch, which is an extension of the memory allocator. In a case of the allocation failure, it moves pages into different page frames (physical addresses) and thus it creates a physically contiguous block of sufficient size.

Instead, a simple solution was chosen. Memory was allocated at boot time (in Linux initialization function “start_kernel” with “alloc_bootmem” function). At system boot time all the physical memory is available, and fragmentation can not occur. Pointer to kernel buffer was exported to via “EXPORT_SYMBOL” macro and used by the driver. Since the configuration of the rings is unknown at boot time (it is given by pre-run configuration file), the exact amount of needed memory is also unknown. A sufficient reserve was considered in the allocation of boot time memory (1MB). The allocation was

added to the pre-build kernel configuration [37]. The boot time allocation and size of allocated memory can be set at kernel build time via KConfig framework as shown in Figure 4.4.

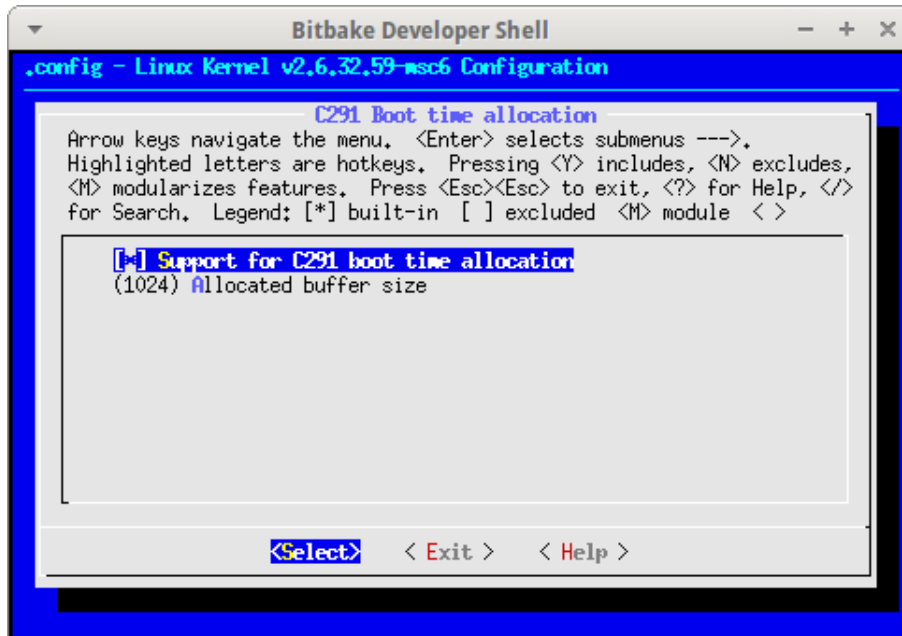


Figure 4.4: Memory allocation configuration

4.6.3 Reorganisation of structures

A relatively minor modification is the reorganization of memory structures which are shared by driver and firmware. Such a structures are defined separately in the driver and firmware (but having the same attributes). Having same attributes does not guarantee that copying the structure between memories on two architectures will be correct. Compilers might decide to insert padding between struct elements. The insertion of a padding is architecture and word size dependant. To have memory layout equal on two CPUs (the structs are copied between systems by “memcpy” calls or DMA engines) attribute “__packed” must be used ([38]). This approach is widely used within the driver and firmware. However, this attribute introduces one problem. The “__packed” attribute might cause that e.g. uint64 element is placed on address offset 0x4 (from the beginning of struct). Accessing such an element might cause unaligned access in the hardware. Different architectures have different handling of unaligned accesses. The original driver platforms x86 and PowerPC have the transparent handling of unaligned access which only results in a minor performance penalty. On MIPS architecture the HW generates an exception which loads/stores the element by bytes (using “load byte”, “store byte” instructions) and causes a bigger performance penalty. To avoid this situation explicit padding was added into packed structures, and attributes are word aligned.

4.6.4 Real-time kernel

The Linux running on MIPS Cavium CPU is patched with PREEMPT_RT patch [39]. This patch modifies the scheduler of Linux to be suitable for real-time applications. An important characteristic of any real-time system is the response time to an internal or external event. Usually, the hard-core real-time systems have a time limit in which this event has to be processed. If the limit is exceeded the information are lost. The event can be an HW interrupt from an external device or another thread with higher priority getting into runnable state. Since part of the TDMA protocol is realized in MIPS Cavium, the Skywan 5G is a hardcore real-time system. If the basic time quantum of TDMA (referred to as “slot”) is not processed in certain time, the information will get lost. Linux scheduler with RT_PREEMPT patch achieves the maximal response time as low as $46\mu s$ in presence high IO-bound workload [40]. Such a response time is sufficient for TDMA implementation. The approach to achieve the low response time is fairly simple. The kernel is completely preemptive, that means that a task can be pre-empted by another higher priority task even if its basic time quantum for execution has not elapsed yet. Even software interrupt handlers can be preempted. This approach increases the number of context switches and thus reduces the computational performance of the CPU.

Apart from being preemptive, there are also other differences to standard kernel. Spinlocks which does not give up the CPU in a standard kernel, can give up the CPU on RT kernel. To achieve the behaviour of classic spinlock on RT Linux one must use “raw_spinlock”. Since the driver was originally not used on the real-time system it is using spinlocks which can sleep on RT: The overall effect of using the sleeping spin-locks was that the kernel was reporting a bug: “Scheduling while atomic” during processing of responses on communication rings. All the spinlocks were converted to raw spin-locks, which resolved this bug.

4.7 Connection of encryption framework to Linux networking engine

To encrypt IP packets handled by Linux on Cavium OCTEON a simple encryption middleware in the Linux kernel was developed. The middleware (called “Encryption hook”) is using the Encryption board via CDF and is interfacing to the Linux networking engine. Linux networking engine is a complex part of Linux kernel which is available from first versions of Linux [35]. It supports many protocols on each level of ISO/OSI model, and it has a very convenient transition between the layers of ISO/OSI. Linux operating system today runs on a majority of servers and many smaller networking devices such as switches or routers. This makes the Linux networking engine one of the most used source codes in the world.

The networking engine in the kernel was modified to interface with the encryption hook in cooperation with a development team of ND Satcom.

The implementation is compatible with both forms of realization: Ethernet module and PCIe module. The basic structure which contains a packet (either Ethernet or IP) is “struct sk_buff” (socket buffer). The aim of the encryption hook is to provide a set of functions which are called from Networking engine for encryption and decryption of socket buffers and management functions for key manipulation. All functions implemented in the encryption hook are shown in Table 4.3.

Table 4.3: Functions implemented by encryption hook

Function name	Description
ipenc_init	Initialize encryption hook
ipenc_exit	Finalize encryption hook
ipenc_setkey	Set AES key
ipenc_setiv	Set initialization vector
aes_encrypt_sync_skbuff	Encrypt skb synchronously
aes_encrypt_async_skbuff	Encrypt skb asynchronously
aes_decrypt_sync_skbuff	Decrypt skb synchronously
aes_decrypt_async_skbuff	Decrypt skb asynchronously
hmac_set_key	Set key for HMAC
hmac_hash_sync	Calculate hash synchronously
ipenc_aes_test	Execute AES encryption test
ipenc_aes_rtt_measure	Measure response time of a single AES encryption
ipenc_hash_test	Execute hash test

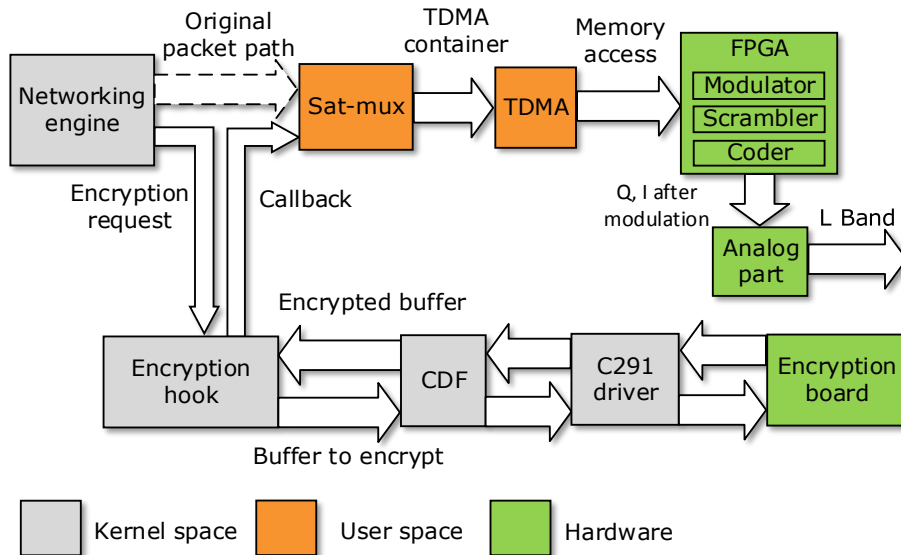


Figure 4.5: Integration of the encryption hook

The encryption in Skywan 5G is executed on all packets which are being forwarded over TDMA. Formally the TDMA is visible in the Linux network-

ing via registered networking interface (“struct net_device”). Each packet that is forwarded to the TDMA interface is processed by a kernel function “ip_finish_output2” (apart from other functions which are further described in [35]). At this point, the asynchronous encryption is requested, and a callback function is provided. After the encryption, IP packet is transferred to userspace into Satmux process which encapsulates the packets into TDMA containers. The container is passed to the TDMA process which takes care of a high-level implementation of TDMA protocol. TDMA process copies the containers into the FPGA via memory mapped memories. The low-level part of TDMA protocol (scrambling, coding, modulation, and others) is implemented in the FPGA. On the output of the FPGA Q and I elements of the digitally modulated signal are processed by high-speed DACs and up-converted to L-Band frequencies in the analog part of the design. The high-level overview of this process is shown in Figure 4.5. Received packets are processed in reverse order. Further details of the architecture are beyond the scope of this thesis.

Chapter 5

Testing and conclusion

The testing and debugging of the whole system are always part of the design process. Although previous parts of this thesis include immediate results, this chapter aims to provide a complete overview of the system bring-up with reference to the concrete sections.

The first part of the testing did involve the simulations of power controllers during the schematic design phase. The outcomes are shown in 2.3.1. The layout phase did not contain any simulations (due to unavailability of FEM simulator) although this was recommended [28, 23] for interfaces such as DDR3 and PCIe. Even without the simulator, the PCIe interface (the fastest interface by far) has proven to be reliable. The manufacturing of the board did include several complications which are discussed in 2.5. After the prototypes had been manufactured, the board was brought up.

The first power up of the board damaged the board power supply due to a communication error with the manufacturer. The manufacturer did populate all the components, even components which were not supposed to be populated. Due to this reason short-cut damaged the power supply. After removing unwanted components and exchanging the power supply, the board was finally brought up. The power supply chain was checked via ADM1069 as shown in Figure 3.4. The implementation of the drivers from 3 was executed on circuit samples connected to the I2C connector of Skywan 5G and debugged with USB logic analyzer with I2C bus analyzer from Saleae [41]. The power up sequence and POR configuration resulted in successful enumeration on PCIe bus as it is shown in Figure 3.8.

The bring-up of the firmware up to handshake procedure did involve debugging in the SKMM mode with the debug prints over the UART. The driver and firmware did contain errors which were difficult to find such as: using UART prints from firmware even in the PKCAL mode which caused the FW to stuck in an infinite loop, mistakes in the handshake procedure or problems in the processing of encryption requests. The SW was working very well in the state as provided by NXP. Different possible configurations (debug prints, high performance mode) from Table 4.1 did reveal many problems. The problems were fixed, and the SW was brought to proper operation.

From the available interfaces the operation of I2C, PCIe and UART was verified. The Ethernet interface communication was observed via the TX

LED on the Skywan 5G. The set-up of the Ethernet link at the end of the auto-negotiation process was observed via “Link up LED”. The Ethernet transceiver did function as expected (link was set-up) even with the design error mentioned in 2.3.3. Testing with Ethernet traffic was not executed, and it is intended for future. The DDR3 interface operation was not verified since external DRAM was not needed for execution of the FW (the internal memory was sufficient). The testing of SDHC slot is also planned for future since booting from microSD card is unused at this point.

5.1 Thermal performance and power consumption

The board power consumption was measured with the increased power consumption of Ethernet transceiver (reasons are explained in 2.3.3) as well as with Ethernet transceiver disabled, and it is shown in Table 5.1. Apart from the Ethernet transceiver which was fixed in second HW revision, the power consumption of the board is in the expected range. Note that the board power consumption differs when operating in the PKCAL and SKMM. This fact is achieved by turning of the clocks of unused peripherals in the PKCAL mode by clock gating technique.

The thermal performance was measured with the infrared camera. Two cases were considered: No air flow (Skywan 5G cover removed) and with estimated air-flow inside the Skywan 5G. Note that since the Skywan 5G is encapsulated in several different chassis, the required cooling fan depends on the geometry of chassis. An additional thermal measurement is shown in Figure 5.2 verifying the thermal performance with the Ethernet transceiver disabled.

Table 5.1: Power consumption of the Encryption board

Mode	Ethernet transceiver	Power consumption with no FW running [W]	Power consumption with FW running [W]	Power consumption during encryption [W]
SKMM	enabled	8,2	11,2	11,8
PKCAL	enabled	8,2	10,8	11,4
PKCAL	disabled	4,2	7,0	7,8

5.2 Manual encryption tests

After implementation of the “Encryption hook”, the functionality of the encryption functions shown in 4.3 was verified manually via commands in C291 CLI application. AES encryption with 256-bit key was executed on random packets and compared with the results from reference implementation

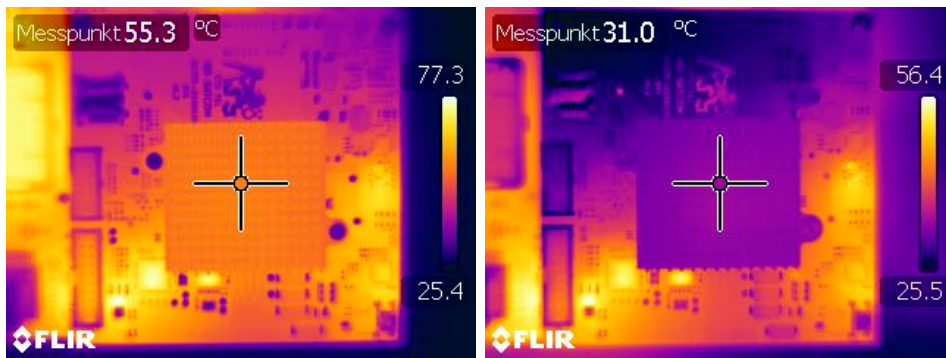


Figure 5.1: Thermal performance (no air flow - left, with air flow - right)

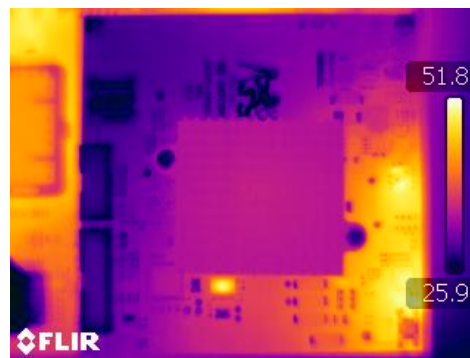


Figure 5.2: Thermal performance with Ethernet transceiver disabled

of AES. The AES is used in the CBC mode. Following encrypted packet was exported from kernel logs (note that the key and initialization vector are chosen to be trivial to demonstrate the encryption):

Key: ABCDEFABCDEFABCDEFABCDEFABCDEFAB

Initialization vector: ABABABABABABABAB

Buffer before encryption:

```
2E 0B E6 5D 50 B1 5C 86 36 BF 36 4C 38 9B 34 B3 2A FB 14 6E
89 42 66 EE 93 33 5B 33 1D FC 81 59 97 8F E8 85 74 9C 2F F1
6D 1A F1 7F 40 57 5C 19 A0 7D 0D E1 26 E9 5B B8 CC DB D9
CD F6 18 33 77 81 A1 87 DD 57 C9 20 C9 05 86 12 98 B8 51 55 E9
C4 37 95 E0 41 78 C5 60 CE 62 79 0D 6D DE 8B 58 E4 56 C8 EC
F1 00 50 D4 42 14 5C DC F6 70 65 05 A6 27 90 7C 47 6E E4 02 85
6C B4 13 5C 7A 7D 20
```

Buffer after encryption:

```
D7 5A BA 22 BC 25 EA 4B 12 FA 9E 5E 72 3F C5 DB 84 6B 80
B9 E3 F5 0F DD 36 B2 EC B7 4F E9 9B 94 35 10 CD B1 37 20
CA 88 B1 88 95 4C FE E0 87 DA 38 E6 82 F9 E9 6D 43 71 E1 B1
4D 16 D2 D2 2B 0C 4A EF 7A B8 AB 99 06 6A F4 34 C1 92 D3
E8 9F AC AD A7 0D 98 F3 9F 7A 1C 8E 32 DB D1 CC AA F7 4E
3B 76 0B 84 02 4A D3 19 50 B0 72 82 D0 56 09 48 54 BB 5B D5
BB C7 D3 96 A4 77 FB 0F 2A 48 88 23
```

Since the AES encryption is symmetric, a semi-automated way of testing was implemented in C291 CLI. A random IP packet was generated and back-up of this packet was created. Encryption and decryption were executed on the packet and the outcome of decryption was compared with a back-up packet. On a match between data before encryption and data after decryption the test passed. The tests ran repeatedly to reveal possible memory leaks in the implementation of the “Encryption hook” or C291 driver.

5.3 Response time

Integration of the encryption into the satellite system requires basic characterisation of the system. A key property of any data processing system is the amount of time it takes to perform the elementary operations. As explained in 4.7 the connection of driver to the networking engine provides functions for encryption of IP packets. A simple measurement was executed which did help with overall system balancing. The results are shown in Figure 5.2. The results shown in the table can not be used to calculate overall encryption throughput since there was always only one packet “on-the-fly”, and the test engine was waiting for completion of a single encryption request. The measured data include the overall round-trip time including following delays: posting the encryption request, transfers over PCIe, processing of the encryption request by the firmware, encryption in HW itself, and response time of RT Linux scheduler.

The response time of RT Linux scheduler (and its jitter) is comparable with the overall time to complete the encryption request [40]. Each measurement was averaged over $3 \cdot 10^5$ samples to filter out the jitter of each response. Note that the measurements were taken in the release configuration build of the software, with command rings disabled and with prints over UART disabled! Since the FW in C291 is running without OS, debugging prints on the UART are blocking the execution on the CPU. The measurement with the UART prints turned on was off by orders of magnitude (encryption requests took approximately 40 ms). Kernel debug logging on the Cavium was not compiled in (DEBUG_PRINT configuration) since measurements with debug logging on Cavium also had a negative effect on the response time. At the time of measurement with debug logs compiled in the “syslog” process and “klog” process consumed up to 70% of one CPU time.

5.4 Packet forwarding test over TDMA

Since the main purpose of Skywan 5G is to forward IP packets over satellite, the Encryption system developed in this thesis was tested in this context. After the “encryption hook” had been integrated into the Linux networking engine, a simple TDMA network with one frequency channel was set-up in the laboratory. Two cases were considered: IP traffic generated locally inside Skywan 5G (Figure 5.3) and IP traffic generated in PC and forwarded by

Table 5.2: Response time of AES encryption

IP Packet size [bytes]	Mean time to complete AES encryption request [ms]
128	0,0662
256	0,0663
512	0,0693
1024	0,0720
2048	0,0747
4096	0,0898
8192	0,1213
9216	0,1280

Skywan 5G from Ethernet (Figure 5.4). The first case is trivial, but it has a side effect. Local generation of traffic also consumes CPU time which might affect the test result. In the second case, static routing was used to direct the traffic from Ethernet to TDMA and vice versa. Since satellite setup-up is expensive and impractical while debugging, a direct connection with coaxial cable is sufficient for verification of the communication and it is commonly used in testing of the Skywan 5G.

The first part of the testing was executed via simple “ping” utility. The advantage of “ping” utility is that it verifies encryption as well as decryption. If either encryption or decryption contained an error, the ping utility would not detect response to ping packets. The packet is generated locally, encrypted, passed to the TDMA, transmitted, received in the other modem passed to the networking engine and decrypted. The device responses with ping reply and the whole process is repeated from the other modem.

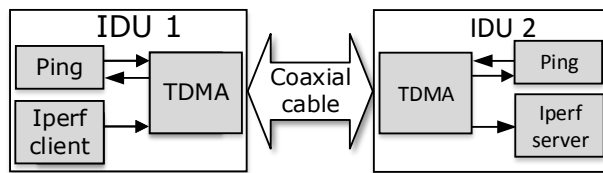


Figure 5.3: Test set-up with local traffic generation

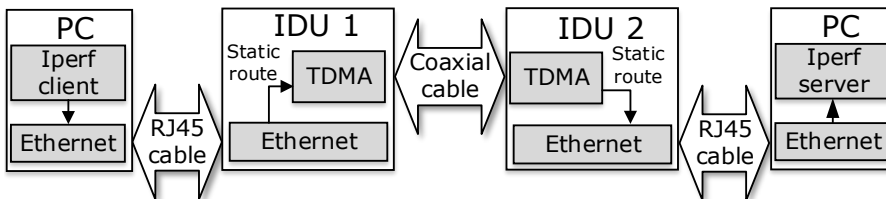


Figure 5.4: Test set-up with traffic generation in PC

Furthermore, the system was stressed to the point of high traffic load over TDMA. An “iperf” Linux utility for generation of UDP packets was used. One modem did set-up an iperf client and the other modem iperf server. A proper operation was verified up to the bandwidth of 10 Mbits/s. In the case of local generation of traffic, no packets were dropped neither reordered. While forwarding IP traffic from Ethernet, few drops did occur. The same effect also occurred without the encryption, and it is caused by overloading the Ethernet card of traffic generating PC, thus the test environment itself causes this problem. More sophisticated methods are being used by ND Satcom for proper testing of Ethernet to TDMA forwarding. This basic method was sufficient for characterization of the system and verification of required functionality. The results of the test with locally generated traffic were following:

```
local 20.0.0.10 port 5001 connected with 10.1.1.2 port 48438
0.0- 1.0 sec 1.19 MBytes 9.98 Mbits/sec 0.958 ms 0/ 849 (0%)
1.0- 2.0 sec 1.19 MBytes 10.0 Mbits/sec 1.124 ms 0/ 850 (0%)
2.0- 3.0 sec 1.19 MBytes 10.0 Mbits/sec 0.974 ms 0/ 851 (0%)
3.0- 4.0 sec 1.19 MBytes 10.0 Mbits/sec 0.960 ms 0/ 851 (0%)
4.0- 5.0 sec 1.19 MBytes 9.98 Mbits/sec 0.976 ms 0/ 849 (0%)
5.0- 6.0 sec 1.19 MBytes 10.0 Mbits/sec 0.955 ms 0/ 851 (0%)
6.0- 7.0 sec 1.19 MBytes 10.0 Mbits/sec 0.941 ms 0/ 851 (0%)
7.0- 8.0 sec 1.19 MBytes 10.0 Mbits/sec 0.935 ms 0/ 850 (0%)
8.0- 9.0 sec 1.19 MBytes 10.0 Mbits/sec 0.935 ms 0/ 851 (0%)
9.0-10.0 sec 1.19 MBytes 9.98 Mbits/sec 0.926 ms 0/ 849 (0%)
0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec 0.934 ms 0/ 8504 (0%)
0.0-10.0 sec 1 datagrams received out-of-order
```

The results of the test with traffic generated in PC were following:

```
local 20.0.0.10 port 5001 connected with 10.2.2.10 port 54276
0.0- 1.0 sec 1.18 MBytes 9.92 Mbits/sec 0.064 ms 126/17834 (0.71%)
1.0- 2.0 sec 1.18 MBytes 9.90 Mbits/sec 0.068 ms 189/17869 (1.1%)
2.0- 3.0 sec 1.18 MBytes 9.92 Mbits/sec 0.051 ms 138/17848 (0.77%)
3.0- 4.0 sec 1.18 MBytes 9.94 Mbits/sec 0.059 ms 130/17874 (0.73%)
4.0- 5.0 sec 1.18 MBytes 9.90 Mbits/sec 0.083 ms 179/17850 (1%)
5.0- 6.0 sec 1.18 MBytes 9.90 Mbits/sec 0.066 ms 175/17857 (0.98%)
6.0- 7.0 sec 1.18 MBytes 9.93 Mbits/sec 0.058 ms 141/17870 (0.79%)
7.0- 8.0 sec 1.18 MBytes 9.91 Mbits/sec 0.063 ms 141/17831 (0.79%)
8.0- 9.0 sec 1.19 MBytes 9.94 Mbits/sec 0.059 ms 122/17877 (0.68%)
9.0-10.0 sec 1.18 MBytes 9.90 Mbits/sec 0.048 ms 159/17829 (0.89%)
0.0-10.0 sec 11.8 MBytes 9.91 Mbits/sec 0.085 ms 1499/178571
(0.84%)
0.0-10.0 sec 1 datagrams received out-of-order
```

These tests proved the functionality of the Encryption board and its SW integrated into the Skywan 5G and moved the design into the state of a working prototype. A further (automated) tests for IP forwarding performance

are being executed by the development team of ND Satcom at the time of thesis submission.

■ 5.5 Conclusion

The purpose of this thesis was to design an encryption system from concept to the realization of a working prototype. This task was full-filled, and the development of this product will continue in the Research and Development Department of ND Satcom. All the points from the assignment were met:

- Hardware was designed, implemented and manufactured.
- The board was brought up and control software was implemented.
- The driver was ported to MIPS CPU and extended. The firmware build environment was created.
- Encryption was tested and integrated into Linux networking stack.

The future usability of the whole system was a great motivation, and it is the reason why does the whole design have professional quality. This fact is obvious in HW (proper revisioning, reconfigurability via population of different components or processing of feedback from manufacturer) as well as SW (generic approach, proper coding style and re-usage of existing SW).

Errors did also occur during the development (e.g. the Ethernet transceiver) due to various reasons. SW errors were easy to fix during the debugging. Workarounds were applied on HW errors, and these errors were fixed in the second HW revision which is being manufactured at the time of thesis submission. More HW revisions are common in the development of complex electronic systems. A responsible approach to detection and repairment of errors was the crucial part of design process.

The implemented system is in a stable state. The encryption feature will be extended from SW point of view and included in the next release of Skywan 5G at the end of 2017. The first series of 50-100 pieces will be manufactured in September of 2017. The quality of the implemented solution is achieved by the employment contract from ND Satcom which allowed to invest much more time into the development than the time recommended by “ECTS” for a master thesis.



Appendix A - Schematic

Appendix A is not available in the public version of the thesis. The schematic of the Encryption board is confidential to ND Satcom



Appendix B - Component placement

Appendix B is not available in the public version of the thesis. The component placement on the Encryption board is confidential to ND Satcom.



Bibliography

- [1] Applied Cryptography: Protocols, Algorithms and Source Code in C, B. Schneier, Wiley, 978-1119096726
- [2] SkyWAN NG, Hardware Architecture, ND Satcom
- [3] SKYWAN 5G, System Architecture, ND Satcom
- [4] AES Proposal: Rijndael, Joan Daemen, Vincent Rijmen, 1999
- [5] Cache timing attacks on AES, D.J. Berndstein, Department of Mathematics, Statistics, and Computer Science (M/C 249) The University of Illinois at Chicago
- [6] An implementation of DES and AES secure against some Attacks, M.L. Akkar and C. Giraud, Workshop on cryptographic hardware and Embedded systems CHES 2001, volume LNCS2162, pages 309-318, Springer-Verlag 2001
- [7] Experimental Implementation of 2ODPA attacks on AES design with flash-based FPGA Technology, N. Masmoudi, L. Bossuet, A. Ghazel,
- [8] New Directions in cryptography, W. Diffie, M. Hellman, IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976
- [9] HMAC: Keyed-Hashing for Message Authentication, RFC2104, H. Krawczyk, M. Bellare, R. Canetti, 1997
- [10] COM Express Module Base specification, PCI Industrial Computer Manufacturers group, 2005
- [11] GNU Cryptographic library, www.gnu.org/software/libcrypt
- [12] LibTomCrypt Library, <https://github.com/libtom/libtomcrypt>
- [13] STM7007, Single-chip hardware accelerated encryption engine for computer and peripherals applications, Data brief, STMicroelectronics, 2011, <https://www.digchip.com/datasheets/parts/datasheet/456/STM7007-pdf.php>

- [14] Freescale C29x Family of Crypto Coprocessors, Fact-Sheet, Freescale, <http://www.nxp.com/assets/documents/data/en/fact-sheets/C29XFAMFS.pdf>
- [15] KiCad, A cross platform and Open source Electronic Design automation suite, <http://kicad-pcb.org/>
- [16] High speed digital design, Handbook of Black magic, H. Johnson, M. Graham, Prentice Hall, 1993
- [17] PWM DC/DC Controller with VID Inputs for Portable GPU Core-Voltage Regulator ISL95870, ISL95870A, ISL95870B, Intersil, 2013
- [18] ISL88550A, Synchronous Step-Down Controller with Sourcing and Sinking LDO Regulator, Intersil, 2013
- [19] LT3514, Triple Step-Down Switching Regulator with 100% Duty Cycle Operation, Datasheet, Linear Technology, www.linear.com/LT3514, 2013
- [20] ADM1069, Super sequencer with margining control, Datasheet, Analog Devices, 2016
- [21] PI6C557-05B, PCIe 3.0 Clock Generator with 4 HCSL Outputs, Pericom, 2011
- [22] JEDEC DDR3 SDRAM Specification, JEDEC Solid State Technology association, 2010
- [23] Hardware and Layout Design Considerations for DDR3 SDRAM Memory Interfaces, Freescale semiconductors, 2013
- [24] Board Design Guidelines for PCIe™ Architecture, Zale Schoenborn, PCI Express Electrical WG, PCI-SIG, 2004
- [25] COM Express Design Guide, PCI Industrial Computer Manufacturers Group, 2009
- [26] Online impedance calculator, <http://www.multek.se/engelska/engineering/pcb-structures-2>
- [27] High speed signal propagation Advanced black magic, H. Johnson, M. Graham, 2003
- [28] DDR3 Design Considerations for PCB Applications AN111 , J. Burnett, Freescale Technology forum, 2009
- [29] Linux I2C dev interface, <https://www.kernel.org/doc/Documentation/i2c/dev-interface>
- [30] PCI Express® 2.0 Base Specification, Revision 0.9, 2006
- [31] C29x Crypto Coprocessor Family Reference Manual, Freescale semiconductors, Rev. 2, 2014

- [32] C29x Crypto Offload User Guide, Freescale semiconductor, 2013
- [33] Linux Kernel crypto API, <http://www.chronox.de/crypto-API/crypto/index.html>
- [34] Linux Device Drivers, J. Corbet, A. Rubini, and G. Kroah-Hartman, Third Edition, O'Reilly, 2005
- [35] Understanding Linux networking Internals, Ch. Benvenuti, O'Reilly, 2005
- [36] Linux kernel map, 8.5 Per-CPU variables, <http://www.makelinux.net/ldd3/chp-8-sect-5>
- [37] The Linux Documentation project, Kernel configuration, <http://www.tldp.org/HOWTO/SCSI-2.4-HOWTO/kconfig.html>
- [38] GNU C Compiler, Type attributes, <https://gcc.gnu.org/onlinedocs/gcc-3.3/gcc/Type-Attributes.html>
- [39] Real time Linux Wiki, https://rt.wiki.kernel.org/index.php/Main_Page
- [40] A Comparison of Scheduling Latency in Linux, PREEMPT RT, and LITMUSRT, F. Cerqueira, B. B. Brandenburg, Max Planck Institute for Software Systems (MPI-SWS)
- [41] Saleae Logic Analyzer, Logic 8, <https://www.saleae.com/>
- [42] SKYWAN 5G Open encryption interface, ND Satcom
- [43] DESCHAL Project, R. Verser, M. Curtin, J. Dolske, <http://www.interhack.net/projects/deschall/>
- [44] European standard EN 55022, Information technology equipment, Radio disturbance characteristics, Limits and methods of measurement, CENELEC, 2006