Student: **Dragos Mihaita Iftimie**

Project: Zynq-7000 FPGA crypto mining profitability

1 – Introduction

The client has access to a big stock of Xilinx Zynq-7000 field programmable gate array and wants to explore the possibility of using it for crypto mining.
The Xilinx's FPGA products fit for both production and research/prototyping environments because they can be programmed via VHDL (a language used for hardware description, it allows fine control of the hardware so it can lead to better resources usage/increased performance at the cost of required programming time and skills) and C language.
Using Xilinx's software, Vivado HLS, a programmer with no FPGA programming background can program an FPGA using C language.
Vivado HLS, given the C code that describes the behaviour to be mapped in hardware, can translate the C code in VHDL.
Is important to notice that in order to achieve that the C code cannot be written using an imperative paradigm, as usually is.
The C code to be written won't be executed in a CPU (line by line) but will be analized by a tool and used to program some hardware; this means that the paradigm and the mental approach are different; this requires non trivial knowledge.
This approach is not optimal because, even though the translation can be controlled through several options, the tool can automatically take decisions that are not fit for the specific project.

Another important aspect about Zynq-7000 is that the board has, in addition to the FPGA, an ARM processor that can communicate with the hardware.
This allows to run an operating system (Petalinux, linux distribution released by Xilinx, highly customizable and has to be compiled from scratch) on the CPU and write in an high level programming language the least CPU-expensive (for example networking and protocol management) tasks while assigning to the hardware the most CPU-expensive tasks (for example the research of the solution for the next block of the blockchain).
This approach avoids waste of hardware of the FPGA for non CPU-intensive tasks, allowing to use all the hardware resources and increase the parallelization of the most expensive tasks.
The CPU can also run a baremetal OS that allows to continuesly run C code; this option is more optimal because there's no delay from a complete OS like Petalinux, but it takes more time to develop.
Since the cryptocurrencies to be studied (see below) are all Bitcoin-like, the interface that the mining function presents (the one to be translated in hardware) is the same for all of them.
The standard API that Xilinx gives to communicate between the CPU and the hardware are "unconfortable and low level"; it's worth it developing some personalized API's using the standard ones in order to communicate with the interface that the hardware exposes.

The client, for each crypto from a list of cryptocurrencies he trusts (Dogecoin, Garlicoin, Groeslcoin, Bitcoin), wants to know if the Zynq can handle the hashing algorithm (some of them may be ASIC resistant), how much electrical energy is needed for the process and how efficient the board is in researching the solution for the next block.
Since a miner runs without ever stopping, 3 full days of mining are enough to understand how many hashes per second the boards can do and how much energy is required.

The boards will get the mining work from a public mining pool server that conforms to the protocol Stratum v1; the software that resides on the CPU and is capable of networking will have to be a Stratum v1 client, will have to be able to decode the work recieved and to assign it to the hardware through the high level API that we developed.
Again, since the cryptocurrencies are all Bitcoin-like, the decoding process is the same; the only difference is a constant in a division (in the operation used to calculate the target the miner has to look for).

The client wants results from us as fast as possible (especially in the cryptocurrencies world time is money); once he has the needed informations, he will decide which crypto to mine (if any is profitable) using native VHDL and a baremetal OS for the CPU in order to achieve the best performances.

3 – Project goals

What the client expects from us is:

- implement the hashing algorithm of each cryptocurrency from the list in VHDL in order to understand if the Zynq-7000 has enough hardware resources to handle it;

- Estimate, for each minable cryptocurrency, how many Watt the FPGA consumes in order to work (the average of the 3 days test);

- Estimate, for each minable cryptocurrency, how many hashes per second the FPGA can calculate (the average of the 3 days test).

Using the power consumption, the hashes per second, the cryptocurrency network difficulty and the value of the cryptocurrency (the last two informations are public) the client will be able to understand if any of the cryptocurrencies is profitable and which one is the most profitable. Based on our research, the client will decide if is worth developing a production Zynq-7000 miner or not.

4 – Project scope

The project scope includes:

Garlicoin, dogecoin, groeslcoin and bitcoin are the cryptocurrencies that need to be explored; this means that their hashing algorithms are the ones to be implemented in hardware.

Developing an high level language solution that can be run on Petalinux OS that manages the mining process.
The solution has to comply with Stratum v1 client protocol in order to recieve work, be able to decode and assign it to the hardware and submit solutions found from the hardware to the server.

The project scope does not include:

Developing a solution that manages the mining process in C that can be run on a baremetal OS.
Implementing the hashing algorithms directly in VHDL, Vivado HLS will be used to translate hardware behaviour written in C to VHDL; this accelerates our research because the implementation of the hashing algorithms is open-source and written in C language.
This means that with small changes to the avaliable open-source solutions we can translate it in VHDL.

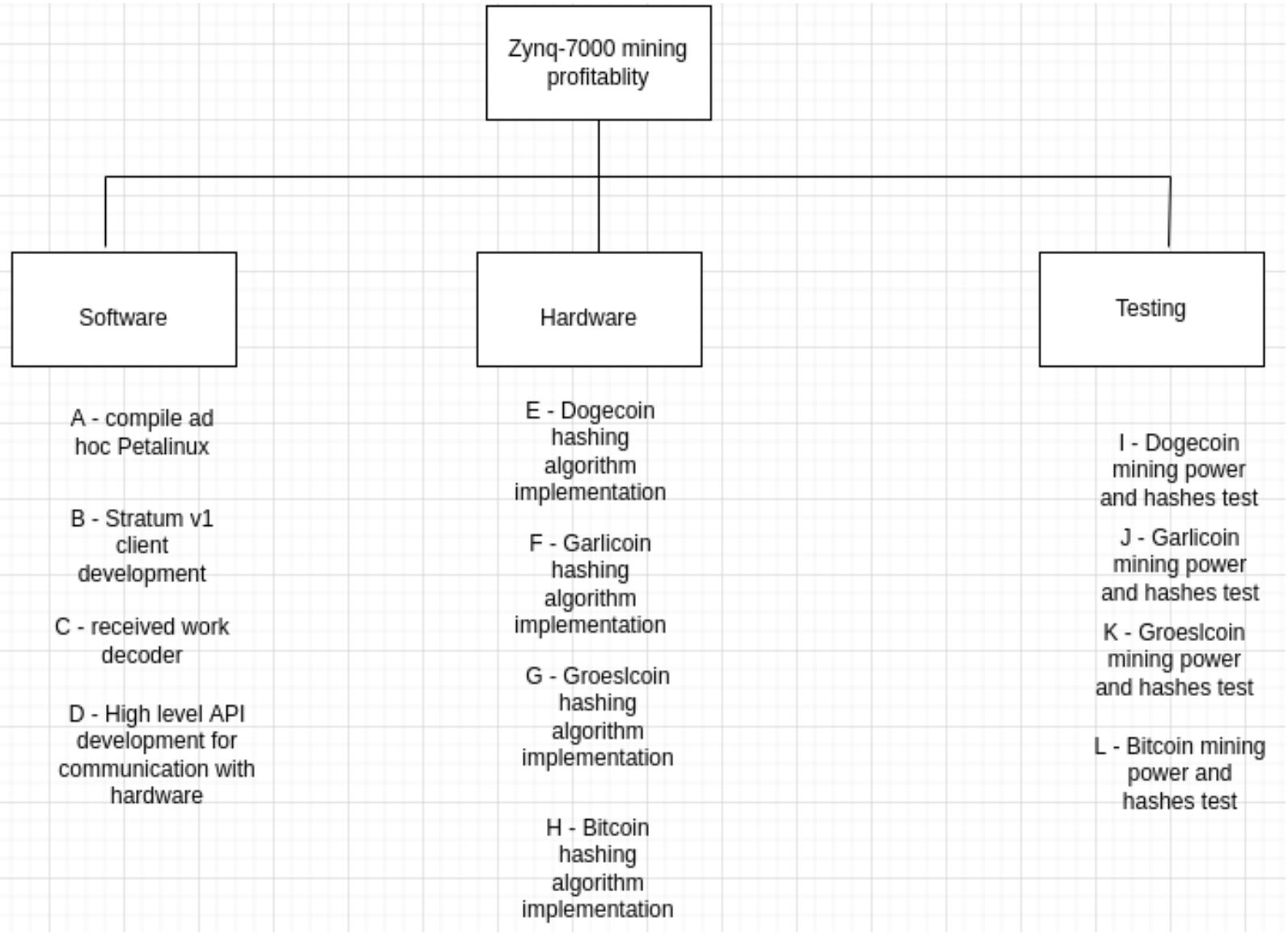
5 – Requirements

The main requirement that have not been well discussed yet is a non functional one: the client wants us to finish our research as fast as possible.
That’s why we are not using optimal solutions, it would take way more time and right now the client just wants to have an idea of which cryptocurrency can be profitable.
A production and optimal development will be held only after our research and if the client thinks is worth it.

6.1 – Deliverables and outcomes

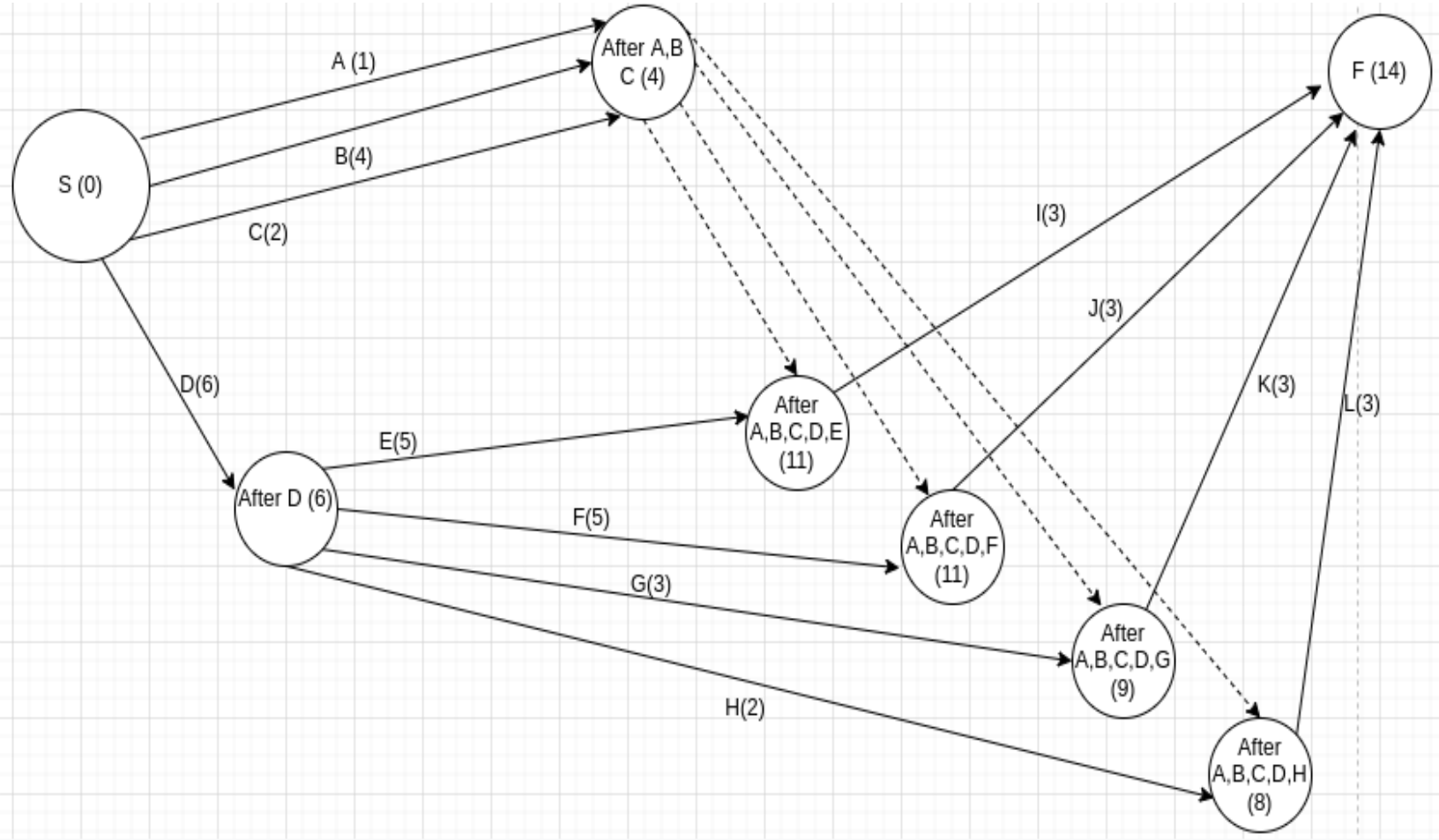
The main deliverables are informations.
The client doesn’t need the software we will develop but just the results of the research, since if he will decide to continue the developing, our software is not optimal.
The client will recieve the list of minable cryptocurrencies, from the list of cryptocurrencies he gave us.
For each cryptocurrency in the list of minable, he will recieve how many watts the FPGA needs to work and how many hashes per second the FPGA can perform.

6.2 – Work breakdown structure



Activity	Complexity (working days)	Sequencing
A	1	-
B	4	-
C	2	-
D	6	-
E	5	After D
F	5	After D
G	3	After D
H	2	After D
I	3	After A,B,C,D,E
J	3	After A,B,C,D,F
K	3	After A,B,C,D,G
L	3	After A,B,C,D,H

7.1 – Network diagram



7.2 – Project schedule and milestones

Team:
Dragos Iftimie – programmer and Xilinx’s tools expert
Daniel Rajer - programmer
Rimantas Jolinas - programmer and Xilinx’s tools expert
Agostino Del Gaudio - programmer and Xilinx’s tools expert

During the weekends the team will not work.
The activities I-J-K-L are not splittable, the tests about the average power consumption and hashes per second must last 3 days.
It’s important to notice that those activities may not happen if the Zynq-7000 cannot handle the hashing algorithm of the respective cryptocurrency.

What	Start	End	Who
First meeting, explain the project, the scope and the goals to the team	April 28 2022	April 29 2022	The whole team, Dragos Iftimie will lead the session
Activity A	May 02 2022	May 02 2022	Daniel Rajer
Activity C	May 03 2022	May 04 2022	Daniel Rajer
Activity D	May 02 2022	May 03 2022	Rimantas Jolinas Dragos Iftimie Agostino Del Gaudio
Activity B	May 05 2022	May 10 2022	Daniel Rajer
Activity E	May 04 2022	May 06 2022	Dragos Iftimie Agostino Del Gaudio
Activity F	May 06 2022	May 10 2022	Dragos Iftimie Agostino Del Gaudio
Activity H	May 04 2022	May 06 2022	Rimantas Jolinas
Activity G	May 09 2022	May 11 2022	Rimantas Jolinas
Activity I	May 11 2022	May 13 2022	Dragos Iftimie
Activity J	May 11 2022	May 13 2022	Agostino Del Gaudio
Activity K	May 11 2022	May 13 2022	Daniel Rajer
Activity L	May 12 2022	May 14 2022	Rimantas Jolinas
Project completed	May 14 2022		

8 – Resources

The team, in order to work, needs basic office furniture (personal computer, electricity, internet connection…).
Other than that the team will need a maximum amount of 4 Zynq-7000 and 4 electrical power consumption measurers; it may also happen that none of the cryptocurrencies are minable so there will be no need for the zynqs and the measurers.
The team will also need some open source code, for example the C implementation of the hashing algorithms.
During the development of the software for activity B and C, the team may look for open source code (widely avaliable online) in order to be faster.
3 (not 4 beacuse Daniel Rajer will not work on it) Xilinx Vivado licenses will be needed in order to perform activities E-F-G-H.

Finally, for the testing part, an external mining pool server will be used to get the work to mine (for example https://aikapool.com/).
We also need some good vents in order to not melt the boards during the tests.

9 – Project expenses and expected benefits

Since our team is specialized in Xilinx products involved projects, we already have enterprise licenses for their software.
This doesn’t mean that this project won’t contribute to the expense that we had years ago (a license can cost up to 4000 dollars) in order to buy the license, just… the actual client won’t pay the whole cost (see the economical concept ‘amortization’); for more details ask the accounting section.
The external pool mining server is free and the client will provide us 4 Zynq-7000 for the project.
We’ll need to buy the power consumption measurers, it’ll cost up to 20 euros each one, and some vents to cool the boards 20-30 euros each one.
The main cost is work. Our team is specialized in that field and has not easy to find competencies. More than that, the team will work hard in order to finish everything as fast as possible, they’ll need to be well payed.

ZYNQ MINING PROFITABILITY

SOFTWARE QUALITY ASSURANCE PLAN

26/05/2022

DRAGOS MIHAITA IFTIMIE

Section 1. Purpose

The purpose of this plan is to define the Zynq mining profitability Software Quality Assurance (SQA) organization, SQA tasks and responsibilities; provide reference documents and guidelines to perform the SQA activities; provide the standards, practices and conventions used in carrying out SQA activities; and provide the tools, techniques, and methodologies to support SQA activities, and SQA reporting.

1.1 SCOPE

This plan establishes the SQA activities performed throughout the life cycle of the Zynq mining profitability project.
Specifically, this SQA Plan will show that the SQA function is in place for this project.
The goal of the SQA program is to verify that all software and documentation to be developed meet all technical requirements. The SQA procedures defined here in shall be used to examine all developed software and documentation to determine compliance with technical and performance requirements.

1.2 DOCUMENT OVERVIEW

In section 2 the organization of the project and its responsabilities are shown.
In section 3 the SQA tasks and planning are shown.

1.3 RELATIONSHIP TO OTHER PLANS

SQA evaluation of the software development processes throughout the life cycle is based on the processes defined in the Zynq mining profitability project charter, reference (a). The SQA Plan is implemented in conjunction with the Zynq mining profitability risk management plan, reference (b).

1.4 REFERENCE DOCUMENTS

a. Zynq mining profitability project charter;
b. Zynq mining profitability risk management plan.

SECTION 2. MANAGEMENT

This section describes each major element of the organization that influences the quality of the software.

2.1 ORGANIZATION

The project is carried out in an enterprise.
Me (Dragos Mihaita Iftimie) I'm both a developer and an SQA team member;
I'm coordinating a group of programmers and above me there's the project management which is indipendent from me.
Its task is to inspect and review our work on regular basis in order to assure the quality of the project.

2.2 RESPONSABILITIES

The project management and me have to assure that the needed quality of the process used for developing is reached.
- Project management:
1) Review and approve this SQA plan.
2) Identifying and ensuring the quality factors to be implemented in the system and software.
3) Implementing the quality program.
4) Supervise, inspect and review on regular basis.

- Me:
1) Ensuring software quality.
2) Developing the test plan.
3) Identifying, implementing, and evaluating the quality factors to be implemented in the system.

- Programmers:
1) Ensure software quality during the development by following the directives.

SECTION 3. SQA PLANNING AND TASKS

The scheduling of SQA tasks is driven by the software development schedule. Therefore, an SQA task is performed in relationship to what software development activities are taking place. One or more SQA tasks can be performed concurrently until a task is completed. A task is considered completed when the required report e.g., SQA Reports, Process Audits Reports, etc. are satisfactorily completed.
The following tasks, requiring coordination and cooperation with the project team, shall be performed by SQA.

3.1 REVIEW SOFTWARE PRODUCTS

Reference (a) identifies all software products to be developed, evaluated and used including the standards or guidelines to be followed.

3.2 EVALUATE SOFTWARE TOOLS

The software needed is the following:
- Xilinx tools in order to obtain the VHDL implementation of the hasing algorithms;
- Xilinx tools in order to get the bitstream, that is used to program the FPGA;
- Xilinx tools in order to compile/run on the CPU petalinux;
- External server to get work from;
- C implementation of the hasing algorithms;
- Stratum V1 client implementation.

The Xilinx tools are the most important external software that we are going to use.
The whole project rely on it; that means that a problem with it may cause the failure of our project.
Luckly our team is experienced in the use of this software and, in case of severe problems, our enterprise licenses assure us an high quality support from Xilinx.

The external server is the most unreliable service that we are going to use.
If we encounter problems with it, we can just change server (there are many avaliable) or, in the worst case, develop our server.
This is not hard but takes time, we just need to run a full node of the crypto (this will give us a getblocktemplate server) and “translate” it to a Stratum server (open source code is highly avaliable online).

The official C implementation of the hasing algorithm allows us to save time.

It is written following an imperative paradigm, our developers will have to modify it in order to follow a descriptive paradigm.
This is a main part of our project and, without an implementation, we cannot go on. Anyways, in case of problems with the code, the papers of the hashing algorithms are freely avaliable online and our team could implement it from scratch, it's just annoing and time consuming.

The Stratum client protocol is a very lightweight protocol.
We will use online code to speed up the project but, again, the protocol is so trivial that our team could develop and test it in less than a working day.

3.3 EVALUATE FACILITIES

For the experimental evaluation the data that we will need is the realtime status of the blockchain; the external server will provide it.
During the testing of our software, we will need some old blockchain blocks in order to verify that we are correctly solving the problem; those data are avaliable through a blockchain explorer (or a full node if we install one).

3.4 EVALUATE PROJECT PLANNING, TRACKING AND OVERSIGHT PROCESSES

The project's quality analysis should be in accordance with reference a which presents the project planning, tracking and oversight.

The project management has to review the plans and identify the appropriate guidelines and standards to meet the project's needs and be sure that all project activities are following the reference a.
In order to do that, the project management will use reference a and check that the processes that have to be carried out in order to finish an activity are performed as planned.
Any change to the plans will require an approval by the project management.

3.5 REVIEW SOFTWARE PROJECT DESIGN PROCESSES

The project design process is presented in reference a.
SQA activities:
- Verify that the project design process is followed and evaluated;
- Using a traceability matrix, verify if the requirements are met;
- Identify corrective actions if the process is not being followed.

3.6 REVIEW SYSTEM AND SOFTWARE REQUIREMENTS FOR QUALITY

ISO-25010 guide the developers and the project management on the requirements needed by the software.
The requirements of the project are:
- Functional suitability: the software must correctly be able to communicate with a pool server and correctly resolve the problem given by it;
- Performance efficiency: we already know that the project will not use 100% of the potential that it has (see reference a about the software on the zynq and the language used to program it), but in order to be able to give reliable informations to the client, we need to be as efficient as we can and consume as less electrical power as we can;
- Reliability: since 3 days tests have to be carried out, the system must be reliable enough to run non stop for that time.

SQA activities:
- Verify that requirements are documented, controlled and traced;
- Review the requirements in order to asses that they are implementable, understandable and consistent;
- Verify that the desired product quality is achievable with the current system requirements

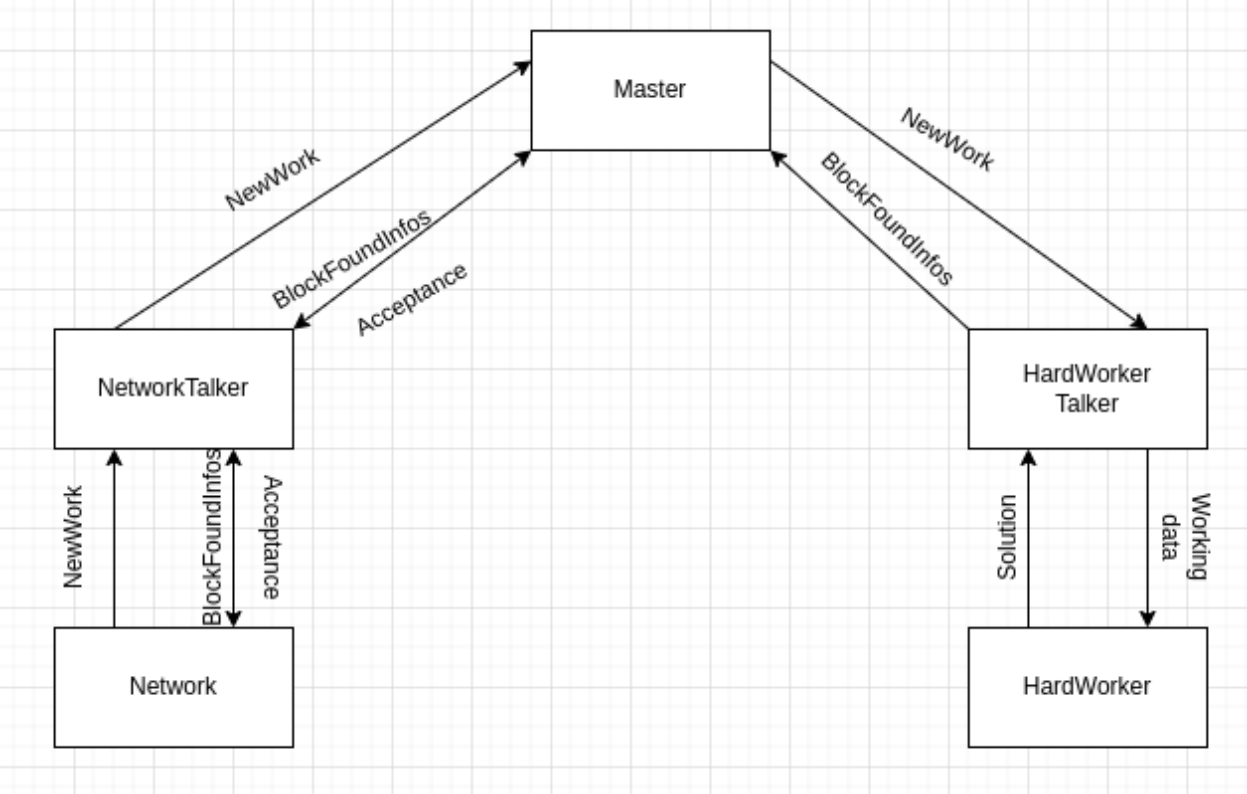
Each change in the requirements must be approved by the project management.

Here we can see a traceability matrix used for that:

Requirement	Degree of fullfillment
Functional suitability	
Performance efficiency	
Reliability	

3.7 EVALUATE SOFTWARE IMPLEMENTATION AND UNIT TESTING PROCESS

After each piece of software (activity) is implemented, it is required to test if it meets the functional requirements and test its integration with other pieces of software.



In this image we can see a simple design of a miner(this should be in reference a but I missed it).
In our case, the NetworkTalker is a stratum v1 client, the master just exchange the informations between the parts/decodes the received work and the HardWorkerTalker contains the API to talk with the hardware.

After the implementation of the stratum client, it has to be tested if it is able to communicate with a stratum server (functionality) and to communicate to the master the needed informations (integration).
After the implementation of the API, it has to be tested if the hardware actually reacts as expected (functionality) and if it is able to serve the master (integration).
Same thing for the development of the hashing algorithms on the FPGA, it has to be tested if it correctly can solve a test-problem (functionality, we already know in advance the solution of the test problem) and if it can successfully communicate it to the HardWorker Talker.

SQA activities:
- Verify that the development and testing process are conducted in conformance with the requirements;
- Verify that all the parts/the whole system is tested and certify the tests;
- Use test reports to document and evaluate each test activity.

3.8 REVIEW METRICS, TESTING METHODS AND VERIFICATION

To mantain the software quality, those metrix are used and mantained:

- Degree of fullfillment of requirements traceability matrix (0-100);
- Number of changes during the execution of the project;
- Degree of fullfillment of the whole system (0-100);
- Number of tests (anticipated vs happened).

Testing methods:

- functional: test if the module or the system behaves as it should. Black box testing of each module (or component) is recommended;
- integration: test if the module (or component) can communicate and serve the other modules of the system;
- system: test if the whole system works as expected and harmoniously.

Verification: is the process of checking if the activities/products of a stage of software development meet the requirements of the previos stages.

- Use a traceability matrix in order to trace all the requirements from the first stage of the project and forward it through the whole project.

3.9 REVIEW DELIVERABLES

The project has no software/hardware deliverables (see reference a)

3.10 REVIEW AND EVALUATE THE CORRECTIVE ACTION PROCESS

Once a problem (it can be software/system problem or non compliance with this plan) is recognized, it will have to be reported to the project management, a correction measure can be attached to the report.
The project management will analize and discuss the problem and, if present, also the proposed correction.
After the discussion, a solution has to come out and in any case the solution must be approved by the project management.

3.11 CONFIGURATION MANAGEMENT PROCESS

The Configuration Management process ensures that selected components of a complete IT service, system, or product (the Configuration Item) are identified, baselined, and maintained and that changes to them are controlled.

3.12 PLAN PROJECT REVIEWS AND AUDIT

The partecipans are me, some members of the project management and the developers.
The project management will assure that the reviews are carried out and I will assure that the changes to be implemented, will be implemented.
The reviews will be carried out during the development (after each activity in reference a is finished) and on the final system.
The process audit will be carried out before the beginning of each activity, in order to assure that the process will be implemented as stated in reference a.
Each review and audit will be documented by using checklists, reports…
After the last activity has been carried out, it will be my responsability to present the finished work to the project management.

Zynq mining profitability

Risk management plan

Dragos Mihaita Iftimie

1 PURPOSE

The purpose of this plan is to document policies and procedures for identifying and handling uncommon causes of project variation (i.e. risk). Risk should be thought of as the possibility of suffering a negative impact to the project, whether it be decreased quality, increased cost, delayed completion, or project failure.

2 RISK IDENTIFICATION

2.1 ANALYSIS OF PROJECT DELIVERABLE

- System unefficiency: if the system is too inefficient, the results may indicate that a certain cryptovalue is not worth mining while it actually is;

2.2 ANALYSIS OF WBS AND PROJECT SCHEDULE

- System breaks during tests: the tests have to be carried out and the system has to run non stop for 3 days, if it is not reliable enough it may break during the tests, forcing the team to go back to development;
- Problem with the Xilinx tools: A bug in the software or any other problem could stop several activities;
- The hardware during the tests may fail: since the tests are quite intensive, the hardware may break.

2.3 ANALYSIS OF POSSIBLE SCOPE CHANGE

- New cryptovalue to test: the client may ask us to add or change a cryptovalue to test;
- Avoid using an OS for the software part: the client may ask us to develop the whole software part in C code, in order to directly run it on the CPU without the OS;
- Design the FPGA directly in VHDL: the client may ask us to not use the C language to program the FPGA but VHDL.

3 RISK DEFINITION

Risk	Category	Source	Trigger	Outcome
System unefficiency	Software	Bad design	It’s not easy to define a trigger, when a crypto is not worth it, it may be because is true or because the implementation is not efficient enough	Our research tells the client that a crypto is not worth mining
System breaks	Software	Bad design	Some software problem makes the miner stop mining	A problem has to be solved and maybe we have to go back to development stage
Xilinx tools problem	Software	Tool problem	Anything that can interrupt our developers from using the tools	Many activities depend on this, the project should be stopped until it is solved
Hardware fail	Hardware	Bad use of hardware	Not taking care of the hardware	The tests cannot be carried out until the problem is fixed or, if not fixable, the client provides new boards
New crypto	Schedule	Client	The client requests that	We will have to implement a new hashing algorithm on the FPGA and carry out the tests

Avoid OS	Schedule	Client	The client requests that	We will have to develop the miner software part in C language
VHDL design	Schedule	Client	The client requests that	Our team cannot do that, we should IMMEDIATLY look for someone that can do it or the project should fail

4 RISK ANALYSIS

This consists in determining, for each risk, the probability of the occurance of a risk and his impact.

The following chart shows risk probability definitions. During risk analysis the potential likelihood that a given risk will occur is assessed, and an appropriate risk probability is selected from the chart below.

Probability Category	Probability	Description
Very High	0.90	Risk event expected to occur
High	0.70	Risk event more likely than not to occur
Probable	0.50	Risk event may or may not occur
Low	0.30	Risk event less likely than not to occur
Very Low	0.10	Risk event not expected to occur

Table 1 – Risk Probability Definitions

RISK IMPACT DEFINITIONS

The following chart shows risk impact definitions across each of the potentially impacted project areas (cost, schedule, scope, and quality). During risk analysis the potential impact of each risk is analyzed, and an appropriate impact level (0.05, 0.10. 0.20, 0.40, or 0.80) is selected from the chart below.

Project Objective	Very Low 0.05	Low 0.10	Moderate 0.20	High 0.40	Very High 0.80
Cost	Insignificant cost impact	< 10% cost impact	10-20% cost impact	20-40% cost impact	> 40% cost impact
Schedule	Insignificant schedule impact	< 5% schedule impact	5-10% schedule impact	10-20% schedule impact	> 20% schedule impact
Scope	Barely noticeable	Minor areas impacted	Major areas impacted	Changes unacceptable to sponsor	Product becomes effectively useless
Quality	Barely noticeable	Only very demanding applications impacted	Sponsor must approve quality reduction	Quality reduction unacceptable to sponsor	Product becomes effectively useless

4.1 QUALITATIVE ANALYSIS

RISK	LIKELIHOOD	IMPACT
System unefficiency	Low	Quality
System breaks	Low	Schedule
Xilinx tools problem	Very Low	Schedule
Hardware breaks	Very Low	Schedule, Cost
New crypto	Probable	Schedule, Scope
Avoid OS	Low	Schedule, Scope
VHDL design	Very Low	Schedule, Cost, Scope

4.2 QUANTITATIVE ANALYSIS

The risk probability and impact matrix shows the combination of risk impact and probability, and is utilized to decide the relative priority of risks. Risks that fall into the red-shaded cells of the matrix are the highest priority, and should receive the majority of risk management resources during response planning and risk monitoring/control. Risks that fall into the yellow-shaded cells of the matrix are the next highest priority, followed by risks that fall into the green-shaded cells.

Probability	Threats				
0.90	0.05	0.09	0.18	0.36	0.72
0.70	0.04	0.07	0.14	0.28	0.56
0.50	0.03	0.05	0.10	0.20	0.40
0.30	0.02	0.03	0.06	0.12	0.24
0.10	0.01	0.01	0.02	0.04	0.08
	0.05	0.10	0.20	0.40	0.80

Table 2 – Risk Probability and Impact Matrix

Risk	Probability	Impact	Score
System unefficiency	0.3	0.8	0.24
System breaks	0.3	0.1	0.03
Xilinx tool problem	0.1	0.8	0.08
Hardware breaks	0.1	0.1	0.01
New crypto	0.5	0.2	0.1
Avoid OS	0.3	0.4	0.12
VHDL design	0.1	0.8	0.08

This is the priority list of the risks:

RED:

System unefficiency

YELLOW:
Avoid OS
New crypto
VHDL design
Xilinx tool problem

GREEN:
System breaks
Hardware breaks

5 RISK MITIGATION, MONITORING AND MANAGEMENT

System unefficiency: we can only mitigate the probability, a good design with efficiency in mind is the key.
We may use a local full node as a server to reduce the network delay, we should design the stratum client in order to be as efficient as possible and also the implementation of the hashing algorithm should use the parallelization that the FPGA offers as much as possible.
It’s kinda hard to monitor/manage this risk, the only thing we can do is a good mitigation

Avoid OS: this comes from the client, there is not much to do about it.

New crypto: this comes from the client, there is not much to do about it.

VHDL design: this comes from the client, if this happens, the project may fail, we should talk with him and be clear that we cannot do that or it will take a lot of effort to do it.

Xilinx tool problem: the only thing we can do is management; if it occures, our Xilinx license assures help from them.

System breaks: The only thing we can do is to mitigate it through code a safe implementation, being aware of what can break a system (eg bad use of memory or file descriptors…)

Hardware breaks: What we can do to mitigate it is to treat hardware properly, by knowing that high temperatures or bad handling can damage it