Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# TRABAJO FINAL DE CARRERA

**TÍTULO DEL TFC: Implementation And Verification Of A Lunar Mission Subsystems (II)**

**TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Sistemas de Telecomunicación**

**AUTOR: Andrés Petilo Pérez**

**DIRECTOR: Joshua Tristancho Martínez**

**FECHA: 26 de marzo de 2010**

**Título:** Implementation And Verification Of A Lunar Mission Subsystems (II)

**Autor:** Andrés Petilo Pérez

**Director:** Joshua Tristancho Martínez

**Fecha:** 26 de marzo de 2010

Resumen

Este Trabajo Final de Carrera (TFC) es la extensión de otro TFC con el mismo nombre. En el presente documento se amplia el trabajo iniciado por el ya consolidado grupo PicoRover del equipo Team FREDNET, participante en el Google Lunar X-Prize.

El proyecto se realiza en colaboración con otros investigadores de múltiples nacionalidades y disciplinas por lo que requiere formar parte de un equipo en un proyecto real de exploración lunar.

En este trabajo se realizan tres partes bien diferenciadas entre ellas, pero no por ello menos importantes para el progreso de los vehículos, tanto del Rover lunar como del alunizador.

La primera parte está centrada en los protocolos y estándares de los procesos de verificación y validación de la misión lunar del grupo Team FREDNET así como la generación de la documentación del Rover lunar llamado PicoRover.

La segunda parte está dedicada a caracterizar un radio-enlace entre un satélite en Low Earth Orbit (LEO) y una estación de tierra amateur mediante balances de potencia para diferentes casos.

Por último se realiza una implementación preeliminar del sistema de control de actitud del alunizador basado en una placa Field Programmable Gate Array (FPGA) y usando el simulador Moon2.0 de código abierto para probar y validar dicho subsistema. Este último apartado se realiza en colaboración con uno de los grupos de Team FREDNET llamado Lunar Lander y que están construyendo un prototipo de alunizador llamado Mark-I en Texas, Estados Unidos.

**Title:** Implementation And Verification Of A Lunar Mission Subsystems (II)

**Author:** Andrés Petilo Pérez

**Director:** Joshua Tristancho Martínez

**Date:** March, 26th 2010

## Overview

This bachelor final work is a continuation of another TFC (Trabajo Final de Carrera) which has the same name as this. It is an improvement of the well consolidated PicoRover group from the Team FREDNET, contestant for the Google Lunar X-Prize.

This project is performed in collaboration with other international and interdisciplinary researchers, it means, belongs to a real team of lunar exploration. In addition, working team, reports and decision taken are required skills.

This work consists in three differentiated parts but, at a time, each one is important for the advancement of the Lunar Rover and Lunar Lander vehicles.

First part is dedicated to verification and validation process for the Team FREDNET lunar mission as well as documents generation for the so called PicoRover Lunar Rover.

Second part is focused in a narrow link between a Low Earth Orbit (LEO) satellite and an amateur ground station. For this reason, a link-budget is done.

Finally, third part is a preliminary implementation of an attitude control system of the Lunar Lander based on a Field Programmable Gate Array (FPGA) board; using the open source Moon2.0 simulator in order to test and validate this subsystem. This last section is done in collaboration of one of the Team FREDNET groups called Lunar Lander Group as a part of one of the Lunar Lander prototypes, the Mark-I, in Texas, USA.

# ÍNDEX

# 1 INTRODUCTION

We are taken part of Google Lunar X Prize[1] competition being active members of Team FREDNET[2].

The past year, the Rover Group which is part of the Team FREDNET, proposed a kind of rover competition based on the idea of parallel development. This idea only could be implemented in an Open Source based team like Team FREDNET. Other teams have to focus their efforts due to the reduced number of components while Team FREDNET has a constant flow of involved people. The first competitor was Joerg Schnyder who developed from the scratch the so called Wheeled Rover Vehicle 1 (WRV1). This design has four straight wheels the body rover is divided by the steering unit. The second competitor was Tobias Krieger who developed and made the so called Just another lunar rover (JALURO). This rover is based on two wheels and the body is in the bottom of them. Finally, the third competitor was Joshua Tristancho who developed the so called Pico-Rover. This rover is based in a single wheel in form of ball self-driven.

The PicoRover group is formed by a local team of students and a teacher from the UPC in Spain. Because is a collaborative competition we take part in some system development in the whole mission. Team FREDNET is organized as a matrix. Each group has the System, the Hardware and the Software department which are represented by Joshua, Enric and Raúl. We have added a fourth member who is in charge of Quality Control represented by Andrés and independent of the PicoRover group by definition.

Following we present the objectives for the PicoRover group reflected in the present Bachelor Final Work. Joshua Tristancho will be in charge of system design and coordinate the PicoRover group. Enric Fernández will be in charge of some hardware like the *Bus CAN-Do Board* and the *PicoSAR micro-RADAR*. Raúl Cuadrado will be in charge of programming some software like the *PicoRover Short-Range Communication System* and the *PicoRover Attitude and Thrust Control*. Finally, Andrés Petilo will be in charge of some documentation like the *System Requirements Document*, the *System Design Document*, the *Program Management Plan* and the *System Engineering Management Plan* and the *Lunar Lander Attitude and Thrust Control*.

# 2 ACRONYMS, ABBREVIATIONS AND DEFINITIONS

| | |
|---|---|
| DAL | Design Assurance Level |
| CEH | Complex Electronic Hardware |
| ASIC | Application Specific Intregrated Circuits |
| PLD | Programmable Logic Device |
| FPGA | Field Programmable Gate Array |
| COTS | Commerial Off The Self |
| CBA | Circuit Board Assemblies |
| LRU | Line Replaceable Units |
| ARP | Aerospace Recommended Practice |
| FHA | Functional Hazard Assessment |
| PSSA | Preliminary System Safety Assessment |
| SSA | System Safety Assessment |
| CCA | Common Cause Analysis |
| | |
| GLXP | Google Lunar X-Prize |
| RV | Reusable Vehicle |
| SSO | Single Spend to Orbit |
| GMT | Greenwich Meridian Time |
| EIRP | Equivalent Isotropically Radiated Power |
| MCU | Main Control Unit |
| LNB | Low Noise Block |
| SNR | Signal to Noise Ratio |
| DSSS | Direct Sequence Spread Spectrum |
| PRN | Pseudo-Random Numerical |
| | |
| SRAM | Static Random Access Memory |
| LE | Logic Elements |
| BGA | Ball Grid Array |
| JTAG | Joint Test Action Group |
| AMD | Advanced Microprocessor Device |
| ACA | Attitude Control Algorithm |
| RTS/CTS | Request To Send/Clear To Send |
| LCD | Liquid Crystal Display |
| LED | Light Emitting Diode |
| CPU | Central Processing Unit |
| RS-232 | Recommended Standard 232 |
| SDK | Software Development Kit |
| SOPC | System On a Programmable Chip |
| HDL | Hardware Description Language |
| EDA | Electronic Device Automation |
| AHDL | Altera Hardware Description Language |
| VHDL | [From Very High-Speed Integrated Circuit (VHSIC)] Hardware Description Language |
| GUI | Graphical User Interface |
| UART | Universal Asynchronous Receiver/Transmitter |
| LLEC | Lunar Lander Engine Control |

| | |
|---|---|
| RLC/RRC | |
| MSTEP | It is a hardware multiplier |
| GERMS | Mnemonic for the minimal command set:<br>Go,<br>Erase flash,<br>Relocate next download,<br>Memory set and dump,<br>Send S-records |
| ROM | Rean Only Memory |
| PIO | Parallel Input/Output |
| TCL | Tool Command Language |
| V&V | Validation and Verification |
| SIL | Software In the Loop |
| HIL | Hardware In the Loop |

# 3 IMPLEMENTATION AND VERIFICATION PROCESS

## 3.1 *Quality control*

To develop an entire project the team must adhere to local, regional, national and international laws and regulations in all aspects for the different tasks. In order to qualify the project the PicoRover Group provides the Quality Control.
The aim of this section is to ensure that the performed work meets the standards. It is important to remark that standards are taken under consideration since the first stage in the development processes to make easier the review task. The basic standards in safety to guide the project are explained below.

## 3.2 *DO-178B*

Software Considerations in Airborne Systems and Equipment Certification is the title given to this document. It is necessary to establish a Design Assurance Level (DAL) which is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. The failure conditions are categorized by their effects on the aircraft and crew and passengers.

- Catastrophic: failure may cause a crash.
- Hazardous: failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.
- Major: failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries).
- Minor: failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change).
- No Effect: failure has no impact on safety, aircraft operation, or crew workload

Now, processes are intended to support the objectives, according to the software level. Processes are described as abstract areas of work in DO-178B, and it is up to the planners to define and document the specifics of how a process will be carried out. That provides a great deal of flexibility in regard to following different styles of software life cycle. However, once an activity within a process has been defined, it is generally expected that the project respect that documented activity within its process. Furthermore, processes (and their concrete activities) must have well defined entry and exit criteria and must show that it is respecting those criteria as it performs the activities in the process.
The intention of DO-178B is not to be prescriptive. Therefore, it allows many possible and acceptable ways.

Software can assist, handle or help in the DO-178B processes. All tools used for DO-178B development must be part of the certification process. Tools generating embedded code are qualified as development tools, with the same constraints as the embedded code. Tools used to verify the code (simulators, test execution tool, coverage tools, reporting tools, etc.) must be qualified as verification tools.
Outside of this scope, output of any used tool must be manually verified by humans.

## *3.3 DO-254*

The objective of this document is to provide guidance for those aspects of the acceptance process for avionic systems impacted by the use of Complex Electronic Hardware (CEH). This guidance is stated to be applicable, but not limited, to the following hardware items:

- Custom micro-coded components, such as Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) and Programmable Logic Devices (PLDs).
- Integrated technology components, such as hybrids and multi-chip modules.
- Commercial-Off-The-Shelf (COTS) components.
- Circuit Board Assemblies (CBAs).
- Line Replaceable Units (LRUs).

Is remarkable the fact that there is a huge potential for the replacement of a microprocessor or obsolete Application Specific Integrated Circuit (ASIC) with an FPGA, or, indeed, by the replacement of an obsolete circuit board with a single FPGA.

In conclusion, the full scope stated in DO-254 is that it can be applied to the development of CEH at all levels of integration from Integrated Circuits to Circuit Board Assemblies and LRUs.

## *3.4 SAE International Aerospace Recommended Practice*

SAE International - "the Engineering Society for advancing mobility - Land, Sea, Air, Space" publish various ARPs (Aerospace Recommended Practice) to aid industry in achieving required standards.

### 3.4.1 SAE ARP 4754

Certification considerations for highly-integrated or complex aircraft systems is the name of the document. It discusses the certification aspects of highly-integrated or complex systems installed on aircraft, taking into account the

overall aircraft operating environment and functions. So, it excludes specific and detailed systems, software and hardware design processes.
Guidelines are intended to provide a common international basis for demonstrating compliance with airworthiness requirements applicable to the systems that integrate multiple level functions and have failure modes with the potential to result in unsafe aircraft operating conditions.

An important part referring to quality control is the validation process model. The process includes specific assessments conducted and updated during system development and interacts with the other system development supporting processes. The primary safety assessment processes are listed below.

- Functional Hazard Assessment (FHA): examines aircraft and system functions to identify potential functional failures and classifies the hazards associated with specific failure conditions. The FHA is developed early in the development process and is updated as new functions or fault conditions are identified.
- Preliminary System Safety Assessment (PSSA): establishes specific system and item safety requirements and provides preliminary indication that the anticipated system architecture can meet those safety requirements. The PSSA is updated throughout the system development process.
- System Safety Assessment (SSA): collects, analyzes, and documents verification that the system, as implemented, meets the system safety requirements established by the FHA and the PSSA.
- Common Cause Analysis (CCA): establishes and validates physical and functional separation and isolation requirements between systems and verifies that these requirements have been met.

This section of the document provides guidance and recommendations covering what safety analyses are most appropriate for each failure condition classification and how to apply the results of the various safety assessment processes at each stage of system development. This includes identifying functional safety requirements and applicable derived safety requirements.
A very graphical representation of which processes would include a real project development is given below.

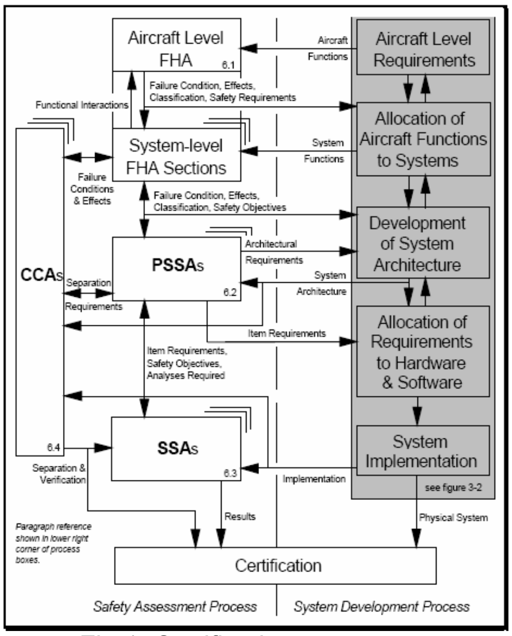

Aircraft Level FHA 6.1
Aircraft Functions
Aircraft Level Requirements
Failure Condition, Effects, Classification, Safety Requirements
Functional Interactions
Allocation of Aircraft Functions to Systems
System-level FHA Sections
System Functions
Failure Conditions & Effects
Failure Condition, Effects, Classification, Safety Objectives
Development of System Architecture
Architectural Requirements
PSSAs 6.2
CCAs
Separation Requirements
System Architecture
Item Requirements
Allocation of Requirements to Hardware & Software
Item Requirements, Safety Objectives, Analyses Required
System Implementation
see figure 3-2
SSAs 6.3
6.4
Separation & Verification
Implementation
Results
Physical System
Paragraph reference shown in lower right corner of process boxes.
Certification
Safety Assessment Process
System Development Process

Fig.1. Certification process

### 3.4.2 SAE ARP 4761

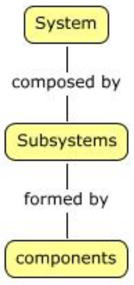
ARP4761 provides "Guidelines And Methods For Conducting The Safety Assessment Process On Civil Airborne Systems And Equipment" it defines processes for using common modeling techniques to asses the safety of a system. It is usually a companion to ARP 4754.

## *3.5 PicoRover LifeCycle*

It is necessary to represent the sequence of steps in a project life-cycle development. It describes the activities and results that have to be produced during product development.

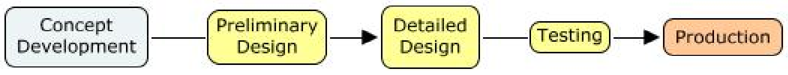
In response to the need of being compliance with the referred standards the group has configured various measures to establish a work plan.

In the resulting design exists a distinction between system, subsystem and component, three levels of hierarchy that are relative to each part of the project.



## 3.5.1 System engineering process

The system engineering development uses a top-down approach. This life-cycle orientation gives strength to the initial definition of system requirements and an interdisciplinary team approach. This process involves a series of steps placed in a logical way based on requirements. So coordination, communication or traceability are remarkable aspects of the model.



- Identify the need: Win the GLXP based on Mission Requirements
- Feasibility study: Can a Picorover win the GLXP?
- Define requirements and design criteria
- Functional analyses and Functional allocation to subsystems
- Identify the preferred system configuration

## 3.5.2 Spiral Process Life Cycle

The Spiral Process Life Cycle is a graphical representation of the system development life-cycle. It represents the main stages to be taken in conjunction with the corresponding deliverables within system validation framework.
The next figure graphically explains the model that the group has decided to take as reference.

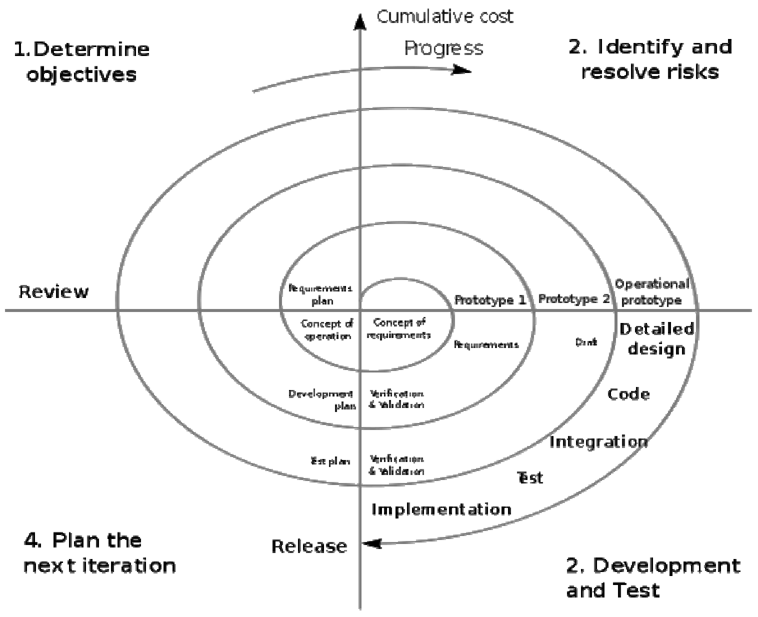


Fig.2. Spiral model

Here is the view of the resulting process being followed:

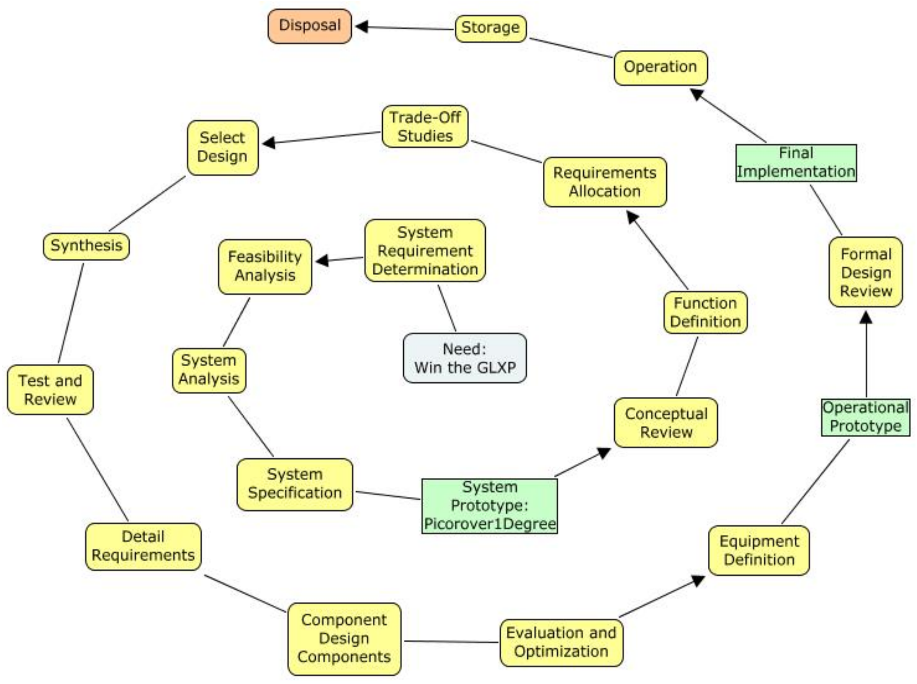


Fig.3. Picorover Spiral process

As it is shown in the figure many steps have been done, these are:

- Need: Win the GLXP
- System Requirement Determination
- Feasibility Analysis
- System Analysis
- System Specification
- System Prototype: Picorover1degree
- Conceptual Review
- Function Definition
- Requirements Allocation
- Trade-Off Studies
- Select Design
- Synthesis
- Test and Review
- Detail Requirements
- Component Design Components
- Evaluation and Optimization
- Equipment Definition
- Operational Prototype: Picorover1.3degrees
- Formal Design Review
- Final Implementation
- Operation
- Storage
- Disposal

### 3.5.3 Program of Documentation

The amount of information obtained during the processes is stored basically on five documents which form the program of documentation. Current versions of documentation are included in the Annexes. It is also available the last version in the Team FREDNET wiki at:
http://wiki.teamfrednet.org/index.php/Picorover_LifeCycle#Program_of_Documentation

#### 3.5.3.1 ConOps Document

The *ConOps* document is focused on the clients giving them answers to why to win and where to dedicate the prize.
http://wiki.teamfrednet.org/index.php/Picorover_ConOps

#### 3.5.3.2 System Requirements Document

The *System Requirements Document* contents the list of the specifications required for the mission systems and subsystems.
http://wiki.teamfrednet.org/index.php/Picorover_Requirements

#### 3.5.3.3 System Design Document

The *System Design Document* is a list with the features of each subsystem, component selection and the preferred configuration. Blueprints and schemes are also included in this document.
http://wiki.teamfrednet.org/index.php/Picorover_Design

#### 3.5.3.4 Program Management Plan

The *Program Management Plan* covers the organization design and role definition for each person in the PicoRover group. Cost analysis is also included in the plan.
http://wiki.teamfrednet.org/index.php/Picorover_Program_Management_Plan

#### 3.5.3.5 System Engineering Management Plan

The *System Engineering Management Plan* defines the manufacture of products and its operation processes.
http://wiki.teamfrednet.org/index.php/Picorover_Engineering_Management_Plan

# 4 N-PRIZE LINK BUDGET

## *4.1 N-Prize reviews*

The N-Prize is a free challenge, in force since November 18, 2008, to launch a satellite into orbit complying stern budgets in exchange for a cash prize. It is aimed at amateurs and intendeds to encourage creativity, originality and inventiveness in the face of severe restrictions.

The N-Prize offers two cash prizes in two categories, each of £9,999.99. One prize (the "reusable vehicle" or "RV" category) will be awarded to the first entrant to complete the challenge using a partially or wholly reusable launch system whose total cost of the launch vehicle exceeds £999.99.The other prize(called the "single-spend-to-orbit", or "SSO" category) will be awarded to the first entrant to complete the challenge with a total launch cost equal or less than £999.99, even if part or all of the launch vehicle is recovered. Like this, no single entry may win both prizes.

The prizes in each category will be available for entrants whose satellites complete his 9th orbit before 19:19:09 (GMT) on the 19th September 2011.

Safety is entirely the responsibility of the entrants and it is assumed that are aware of all risks to themselves, to third parties and to property, whether on the ground, on water, in the air or in space, arising through preparation for, participation in, or after completion of the N-Prize Challenge. To disregard of safety could be a possible reason for exclusion.
In reference to legalties, the entry needs to be compliance with all necessary regulations though the N-Prize organisers do not require proof of compliance with them. So any penalties incurred as a result of failure to comply with relevant local, national or international regulations are entirely a matter for the entrants.

- For the purposes of this challenge, the following definitions apply:
- The 'Launch Site' is defined as the area from which the launch takes place.
- The 'Launch Equipment' is defined as all items of hardware required for the launch, but which remain on the. Any items which are not required after a time 24 hours prior to the launch are not considered to be part of the launch equipment.
- The 'Launch Vehicle' is defined as all items of hardware that leave the ground, including any fuel or other consumables.
- The 'Satellite' is defined as the part of the launch vehicle that enters orbit around the Earth. If more than one item enters orbit must be defined, prior to launch, which item is to be considered the 'satellite'.

The artifact must have a mass of between 9.99 and 19.99 grams, including the weight of any propellant or fuel. It may include shielding or fuel that takes its weight over the 19.99 gram limit, but orbits will not count toward the 9 orbit target until such over-weight items have been jettisoned or consumed. As noted, other items may enter orbit with the satellite, but must not remain attached to it. Nor may the satellite be dependent upon the co-orbiting items in any way during the nine qualifying orbits.

The satellite must complete a minimum of 9 orbits of the Earth, after its separation from or consumption of any items or consumables which put its weight over 19.99 grams. The orbits need not be regular, nor do they need to be at a constant altitude. No part of any orbit may be lower than 99.99 km above the surface of the earth and any method of attaining orbit is acceptable.

Must be able to provide evidence that the satellite has completed a minimum of 9 orbits of the Earth. The costs of providing this evidence do not form part of the budget, except for the costs of any equipment mounted on the launch vehicle to enable detection. There is no need to observe or track the satellite throughout its orbit, as long as sufficient data is collected to confirm that 9 orbits have taken place. Note also that proof may be required that a detected signal originates from the satellite itself.

The budget for each launch is calculated differently for the SSO and RV categories. Our project is focused in the SSO, for this category the maximum cost of the launch, £999.99, should be enough to conduct a repeat of a successful mission, assuming that no part of the launch vehicle is recovered. The money is entirely the responsibility of the entrant and must cover the following:

The cost of the launch vehicle, including the satellite itself, and any fuel, gases or other materials which it carries.
Any items of the launch equipment that could not be re-used for a second identical launch.

The cost that would be incurred for refurbishing, refilling, re-testing or otherwise preparing any launch equipment or any aspect of the launch site and any manufacturing costs for any parts of the launch vehicle; or for any parts of the launch equipment that would require replacement if a second identical mission were to be carried out.

Any failed attempts can be considered prototype development, and hence do not count towards the budget.

Items which need not be covered by the budget include prototyping costs; launch equipment or the launch site; licence fees, permissions etc; charges made for attendance by safety personnel; legal costs; medical costs; insurance costs; fines, penalties or loss of earnings arising from any cause whether prior to, during or after the mission; travel costs of people associated with the mission.

Remark that donations of hardware are permitted but will be judged on a case-by-case basis and entrants may not take benefit from other aerospace projects or 'share' the cost of common hardware. If entrants seek sponsorship, the money that is spent on the launch vehicle and non-reusable launch equipment will be considered part of the budget. The organisers also reserve the right to exclude those whose primary objective seems to be to promote or advertise.

## *4.2 Link budget*

It is compulsory to establish communication with the satellite to accomplish with requirements and guarantee that all is working fine. So, the team has decided to implement a transmitter on board to be able to have some kind of monitorization.

Obviously, the more restrictive link is the one referred to the uplink due to the mass and budget restrictions of the satellite, so this is the considered direction of the link in the next budget.

The radio link budget follows the Friis equation which looks like:

$$P_{RX} = EIRP - L_{FS} - L_{MIS} + G_{RX} - L_{RX} \qquad dBm$$

$$EIRP = P_{TX} + G_{TX} - L_{TX} \qquad dBm$$

$$L_{FS} = 10 * \log(4 * \pi * R / \lambda) dB$$

The idea is to reuse some Picorover devices, already available, such as PicoSAR or eZ430 MCU in order to minimize costs and effort. Then, some parameters will be defined by the existing machinery. The set parameters to consider are:

| | | |
|---|---|---|
| EIRP | 10.6 | dBm |
| $L_{MIS}$ | 3 | dB |
| $G_{RX}$ | 70 | dB |
| $L_{RX}$ | 3 | dB |

The EIRP is obtained from the output power which is able to give the eZ430 MCU (0dBm) and the gain which PicoSAR is able to offer (10.6dB, transmission losses included).

In reception, when the arriving signal is captured by the antenna it goes through the Low Noise Block (LNB) whose function is to amplify but also to down-convert the signal to a lower frequency (specifically in L-band frequency) whether the frequency was 10GHz (it is much easier too to amplify a signal working in L-band than in X-band). The more common gains for the current equipment usually range from 60 to 70 dB, so the maximum value is to be considered for the present budget.

Whereas, in mismatch and reception losses typical values of 3dB are choosen.

Now, the only parameter missing in the equation are the free space losses. Those depend on frequency and distance. Because PicoSAR is designed for working at 2.4 and 10 GHz we will take these values as reference and will considerate different distances that could match with different orbit position of the satellite. The resulting table is shown below:

| | 5 | 50 | 100 | 200 | 500 | 1000 | 2000 | 3000 | km |
|---|---|---|---|---|---|---|---|---|---|
| 2,4 | 114,03 | 134,03 | 140,05 | 146,07 | 154,03 | 160,05 | 166,07 | 169,59 | dB |
| 10 | 126,42 | 146,42 | 152,44 | 158,46 | 166,42 | 172,44 | 178,46 | 181,98 | dB |
| GHz | dB | dB | dB | dB | dB | dB | dB | dB | |

Hence, we can apply Friis equation to calculate the power received in the Earth station:

| | 5 | 50 | 100 | 200 | 500 | 1000 | 2000 | 3000 | km |
|---|---|---|---|---|---|---|---|---|---|
| 2,4 | -39,43 | -59,43 | -65,45 | -71,47 | -79,43 | -85,45 | -91,47 | -94,99 | dB |
| 10 | -51,82 | -71,82 | -77,84 | -83,86 | -91,82 | -97,8418 | -103,86 | -107,38 | dB |
| GHz | dB | dB | dB | dB | dB | dB | dB | dB | |

Now, it is easy to obtain the Signal to Noise Ratio (SNR):

$$SNR = P_{RX} / P_N \qquad dB$$

$$P_N = 10 * \log(k * T * B) \qquad dBW$$

For the noise power two main values for the temperature must be taken into consideration. The first one is the equivalent for a beam pointing to the empty sky and the second corresponds to the equivalent noise temperature of the earth. Both cases work with a bandwidth of 1MHz:

| | 1 | MHz |
|---|---|---|
| 3 | -133,83 | dB |
| 290 | -113,977 | dB |
| K | dB | |

The resultant SNR in downlink considering a bandwidth of 1MHz:

| | 5 | 50 | 100 | 200 | 500 | 1000 | 2000 | 3000 | km |
|---|---|---|---|---|---|---|---|---|---|
| 2,4 | 94,40 | 74,40 | 68,38 | 62,36 | 54,40 | 48,38 | 42,36 | 38,84 | dB |
| 10 | 82,01 | 62,01 | 55,99 | 49,97 | 42,01 | 35,99 | 29,97 | 26,45 | dB |
| GHz | dB | dB | dB | dB | dB | dB | dB | dB | |

Note that, the resulting power values come from a signal that has not been modulated. Typical acceptable values for a correct reception range between 8 and 12dB, which means that the signal would be correctly received for the different situations exposed in the table. This results, let space for a margin that could be necessary to fight fading or other losses caused by negative atmospheric conditions. The margin could reach a value of, for example, 15dB. Like this even the worst case (3000km with a carrier of 10GHz) would continue being correctly received.

## 4.3 Direct Sequence Spread Spectrum (DSSS)

DSSS is a digital modulation technique where transmissions are the combination of the data stream with a higher rate Pseudo Random Numerical (PRN) sequence via XOR function, thereby spreading the energy radiated of the original signal into a much wider band. The fact that uses direct sequence means it does not implement frequency hopping (jump from frequency to frequency). This codding modulation would be useful to ensure that the received signal is coming from our satellite.

The IEEE 802.11 Standard specifies an 11 chip PRN sequence (Barker code). The value of the autocorrelation function for the Barker sequence is 1, -1, or 0 at all offsets except zero, where it is 11. This makes for a more uniform spectrum, and better performance in the receivers.

At the receiver, the pseudo random code is used to de-spread the received data. It is during this process that the matched filter rejects unwanted signals because it has low cross correlation with other sequences likely to interfere. By careful selection of the PRN sequence, the matched filter could provide rejection to multipath signals too.

Three different configurations could be set for both frequencies. The first one and the second, both use a Barker sequence of length 11 the difference between them is that in one, the minimum data rate available is considered, while in the second is the highest rate. The last case considers the maximum processing gain that can be obtained from the eZ430 MCU device. The parameters are contained in the table below:

| | 2.4 | 2.4 | 2.4 | 10 | 10 | 10 | GHz |
|---|---|---|---|---|---|---|---|
| Rb | 1.2 | 45.454 | 1.205 | 1.2 | 45.454 | 1.205 | kbps |
| Rc | 13.2 | 500 | 500 | 13.2 | 500 | 500 | kcps |
| Gp | 10.4 | 10.4 | 26.18 | 10.4 | 10.4 | 26.18 | dB |

Whether no interference is coming from the satellite at the same frequency from other users with other codes and no multipath signals are considered because of the use of a directive antenna in the ground station the equivalent $E_b/N_0$ ratio is easily calculated:

$$E_b/N_0 = G_P * (P_{RX} / P_N)$$

Again the more restrictive way is the downlink and its respective noise powers for the different sets are the following:

| | | | |
|---|---|---|---|
| Pn | 1,2 | 45,454 | kbps |
| 3 | -163,038 | -147,254 | dB |
| K | dB | dB | |

Note that only two cases are referred for the three possible situations this is produced because after the de-spreading at the receiver, the bandwidth at baseband would be restored to the original one and the first case and third have got almost the same data rate (minimum available).

The SNR are:

| SNR | 5 | 10 | 100 | 200 | 500 | 1000 | 2000 | 3000 | km |
|---|---|---|---|---|---|---|---|---|---|
| 2,4 | 123,61 | 103,61 | 97,59 | 91,57 | 83,61 | 77,59 | 71,57 | 68,05 | 1,2kbps |
| 2,4 | 107,83 | 87,83 | 81,81 | 75,79 | 67,83 | 61,81 | 55,79 | 52,27 | 45,454kbps |
| 10 | 111,22 | 91,22 | 85,20 | 79,18 | 71,22 | 65,20 | 59,18 | 55,65 | 1,2kbps |
| 10 | 95,43 | 75,43 | 69,41 | 63,39 | 55,43 | 49,41 | 43,39 | 39,87 | 45,454kbps |
| GHz | dB | dB | dB | dB | dB | dB | dB | dB | |

The resulting $E_b/N_0$ are equal to the previous SNR plus the processing gain:

| Eb/No | 5 | 10 | 100 | 200 | 500 | 1000 | 2000 | 3000 | km |
|---|---|---|---|---|---|---|---|---|---|
| 2,4 | 134,01 | 114,01 | 107,99 | 101,97 | 94,01 | 87,99 | 81,97 | 78,45 | 1,2kbps |
| 2,4 | 118,23 | 98,23 | 92,21 | 86,19 | 78,23 | 72,21 | 66,19 | 62,67 | 45,454kbps |
| 10 | 121,62 | 101,62 | 95,60 | 89,58 | 81,62 | 75,60 | 69,58 | 66,05 | 1,2kbps |
| 10 | 105,83 | 85,83 | 79,81 | 73,79 | 65,83 | 59,81 | 53,79 | 50,27 | 45,454kbps |
| | dB | dB | dB | dB | dB | dB | dB | dB | |

Finally, with these values which largely complies with the requirements could be known the Bit Error Ratio for every modulation such as BPSK, QPSK, etc.

# 5 FPGA DEVICE ANALYSIS

## *5.1 FPGA device description*

The Field-Programmable Gate Array (FPGA) is an integrated circuit containing mixes of a wide range of embedded configurable peripherals and components such as SRAM, microprocessors, transceivers, I/O pins and connectors, diodes, buttons, diplays and logic blocks or routing. In essence, an FPGA consists of programmable logic elements (LEs) and a hierarchy of reconfigurable interconnects that allow the LEs to be physically connected and to perform from the simplest logic gates to complex combinational functions.

## *5.2 Justification of use of an FPGA*

The team has thought of implementing the device, mainly, because of the strength to deal with problems relative to embedding systems designed to do specific task. This is largely due to the modular and extremely flexible nature of an FPGA. As long as it has enough resources, it could be used in replacement of any logical function that, for example, an Application-Specific Integrated Circuit (ASIC) could perform but with a series of advantages that the team has judged as determinants.

A powerful tool like this accelerates prototyping processes since it incorporates typical useful modules to develop integrated circuits for signal processing systems or aerospace. Obviously, since the subsystems to work converge in a single device, it also carries a shorter time required to launch the desired system and a lower cost of researching, developping, designing and testing.

Also, instead of being restricted to any predetermined hardware function, an FPGA allows to program the product features and functions and reconfigure hardware for specific applications also after the product has been realized. Then, an FPGA lengthens the life-cycle either to adapt the product over time or to mitigate the risks of obsolence.

## 5.3 *FPGA specifications*

This section contains an overview of the components mounted on the board and a brief description of the features for those which have been used.

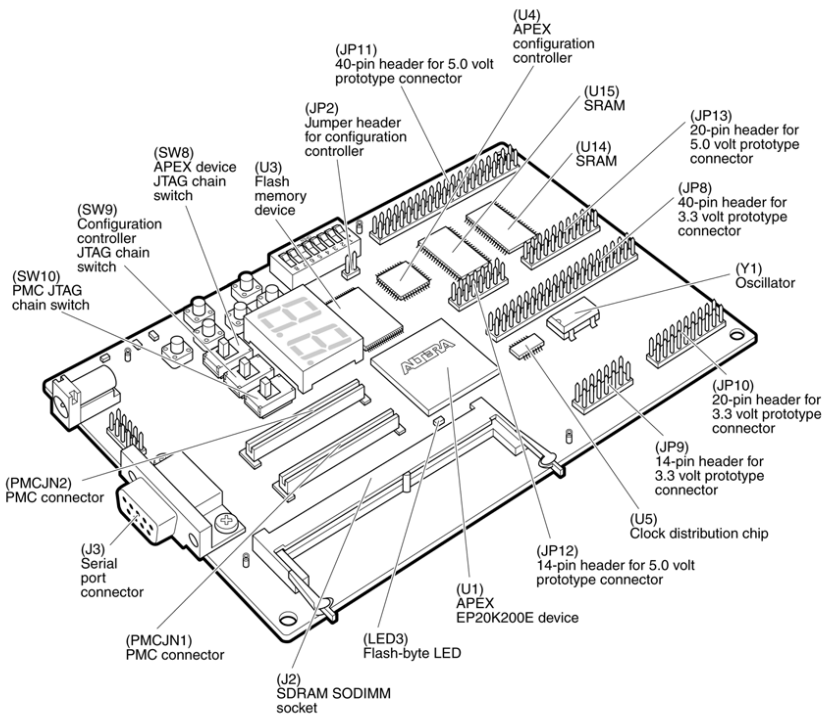


Fig.4. FPGA components

### 5.3.1 The APEX 20K200EFC484 Device

It is a 484-pin BGA package. Typical system module often occupies around 25 percent of the logic on this device.

| | |
|---|---|
| Maximum system gates | 526,000 |
| Typical gates | 211,000 |
| LEs | 8,320 |
| ESBs | 52 |
| Maximum RAM bits | 106,496 |
| Maximum macrocells | 832 |
| Maximum user I/O pins | 382 |

The board provides two separate methods for configuring the APEX device:

- A JTAG connection that can be used with Quartus II software via download cables.
- A configuration controller that configures the APEX device at power-up from hexout files stored in the flash memory.

## 5.3.2 Flash Memory Chip

AMD AM29LV800BB is a 1 Mbyte flash memory chip which is connected to the APEX device so that the Nios processor can use the flash as readable memory and non-volatile storage. That functionality must be useful for data storing or configuration files that implement the ACA32 reference design.

## 5.3.3 SRAM Chips

They are 256 Kbyte (64 K x 16-bit) asynchronous SRAM chips. They are connected to the APEX device so they can be used by the processor as zero-wait-state memory. The two 16-bit devices are used in parallel to implement a 32-bit wide memory subsystem.

## 5.3.4 Serial Port Connector

It is a standard DB-9 serial connector. The transmit (TXD) from the board, receive (RXD) by the board, clear to send (CTS) and ready to send (RTS) signals use standard high-voltage RS-232 logic levels so the buffer accepts 3.3V signals to and from the APEX device. It can support onchip debug peripheral via a serial Y cable. Both pinout sets are shown in the image.

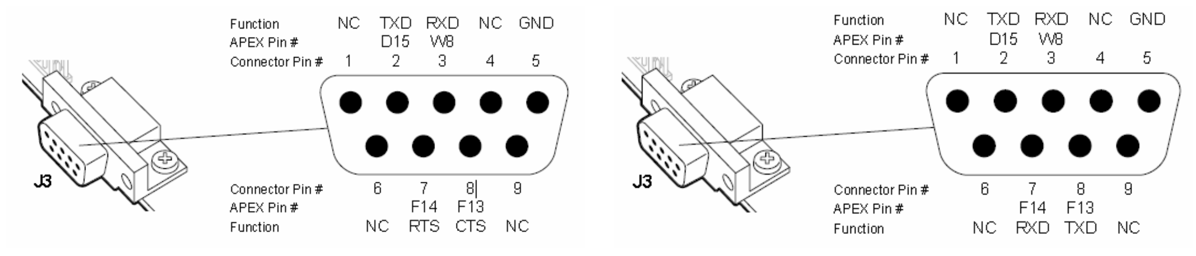


Fig.5. Serial port connector

## 5.3.5 Configuration Controller

The configuration controller is an Altera EPM7064 PLD. It comes factory-programmed with logic that configures the APEX device from data stored in flash on power-up.

In the first instance, the configuration controller will attempt to load the APEX device with user-configuration data. If this process fails the configuration controller will then load the APEX device with factory configuration data. Both are expected to be stored at fixed locations in flash memory.

| | | |
|---|---|---|
| 0x100000 – 0x17FFFF | 512 Kbytes | Nios instruction and nonvolatile data space. |
| 0x180000 – 0x1BFFFF | 256 Kbytes | User-defined APEX device configuration data. |
| 0x1C0000 – 0x1FFFFF | 256 Kbytes | Factory-default APEX device configuration. |

With a jumper if short circuit is detected, the configuration controller will ignore the user-configuration and will run the factory-configuration to fight against any valid-but-nonfunctional user configuration.

### 5.3.6 Two-digit 7-segment display

It is connected to the APEX device so that each segment is individually controlled by a general-purpose I/O pin.

### 5.3.7 Switches, buttons and LEDs

Two LEDs are each controlled by an APEX device I/O. Each LED will light-up when the APEX device drives a logic-1 on its controlling output.
Four push-button switches are also connected to an APEX device I/O. The APEX device will see a logic-0 when each switch is pressed.
It must be said that the board uses dedicated switches SW2 and SW3 for the following fixed functions:

- Reset: when SW2 is pressed, a logic-0 value is driven to the power-on reset controller and when it happens, the configuration controller will reload the APEX device from flash memory.
- Clear: when SW3 is pressed, a logic-0 is driven onto the corresponding APEX pin, then the reference CPU will reset and start executing code from its boot-address.

### 5.3.8 Power-supply circuitry

The Nios development board runs from a 9-V, unregulated, center-negative input. On-board circuitry generates 1.8V, 3.3V, and 5V regulated power levels.

- The 1.8-V supply is used exclusively for the APEX device core power source.
- The 3.3-V supply is used as the power source for all APEX device I/O pins such as RS-232. Remark that the total load may not exceed 500 mA.
- The 5-V supply is present on JP12 connector and used, for example, to work with the LCD module. The total load may not exceed 50mA.

### 5.3.9 Clock Circuitry

The Nios development board includes a 33.333 MHz oscillator and a zero-skew buffer that drive the APEX device.

### 5.3.10 Avalon bus

Avalon is an on-chip bus architecture designed for connecting on-chip processors and peripherals together into a system. It is the interface that specifies the port connections between master and slave components, and specifies the timing by which these components communicate. Basic bus transactions transfer 8, 16, or 32 bits.

The Avalon bus is an active, onchip bus architecture, which consists of logic and routing resources inside a PLD. Some principles of the architecture are:

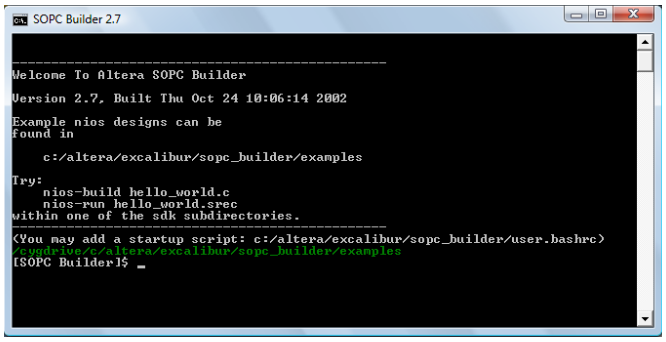
- The interface to peripherals is synchronous to the Avalon clock.
- All signals are active LOW or HIGH. Multiplexers (not tri-state buffers) inside the Avalon bus determine which signals drive which peripheral. Peripherals are never required to tri-state their outputs, even when the peripheral is deselected.
- The address, data and control signals use separate dedicated ports.

## *5.4 Tools for the use of the FPGA*

### 5.4.1 Nios SDK Shell

The Nios SDK Shell command line is developed in the bash environment that provides a UNIX-like environment on the platform. The tool includes most of the commands and utilities in UNIX but, additionally, many Nios-specific utilities for generating and debugging software. The most used commands are:

- *nios_bash*: startup script to set the bash environment for Nios development (bash shell)
- *nios-build*: script that invokes the tools to compile, assemble, and link source code. It ensures the standard C libraries and standard Nios libraries are linked with the user source code, and the associated paths are available. It also generates .dbg files to enable debug.
- *nios-run*: utility for downloading and running a user file by performing terminal I/O.



SOPC Builder 2.7

```
------------------------------------------------
Welcome To Altera SOPC Builder

Version 2.7, Built Thu Oct 24 10:06:14 2002

Example nios designs can be
found in

    c:/altera/excalibur/sopc_builder/examples

Try:
    nios-build hello_world.c
    nios-run hello_world.srec
within one of the sdk subdirectories.
------------------------------------------------
(You may add a startup script: c:/altera/excalibur/sopc_builder/user.bashrc)
/cygdrive/c/altera/excalibur/sopc_builder/examples
[SOPC Builder]$ _
```

Fig.6. Nios SDK shell

### 5.4.2 Quartus II

The Quartus II development software provides a complete design environment for System-On-a-Programable-Chip (SOPC) design. The Quartus II software ensures design entry, processing, and straightforward device programming.

The main Quartus II software logic design capabilities are:

- Design entry using schematics, block diagrams or HDL
- Floorplan editing
- LogicLock incremental design
- Powerful logic synthesis
- Functional and timing simulation
- Timing analysis
- Embedded logic analysis
- Software source file importing, creation, and linking to produce programming files
- Combined compilation and software projects
- Automatic error location
- Device programming and verification

Finally, here is presented the design flow to show a Quartus II project lyfe cycle.

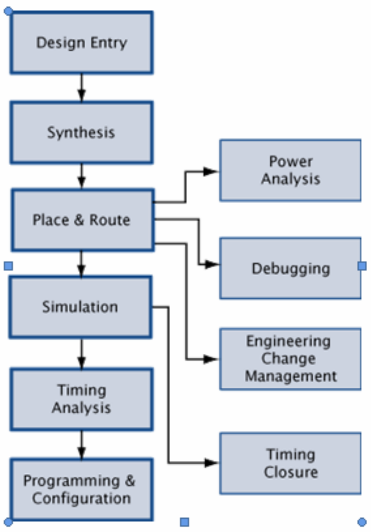


Fig.7. Quartus II design flow

Because of the free license capabilities just the first block has been partially implemented (as it will be explained in the Lunar Lander Engine Control module design). This first stage is explained below.

It can be used used the Quartus II Block Editor, Text Editor, MegaWizard Plug-In Manager, and EDA design entry tools to create the design files in a project. Schematic or block designs can be created with the Block Editor, or AHDL, VHDL, or Verilog HDL designs with the Text Editor. The MegaWizard Plug-In Manager is helpfully to create design files for customized megafunctions. The Quartus II software also supports EDIF Input Files or Verilog Quartus Mapping Files generated by EDA design entry and synthesis tools.

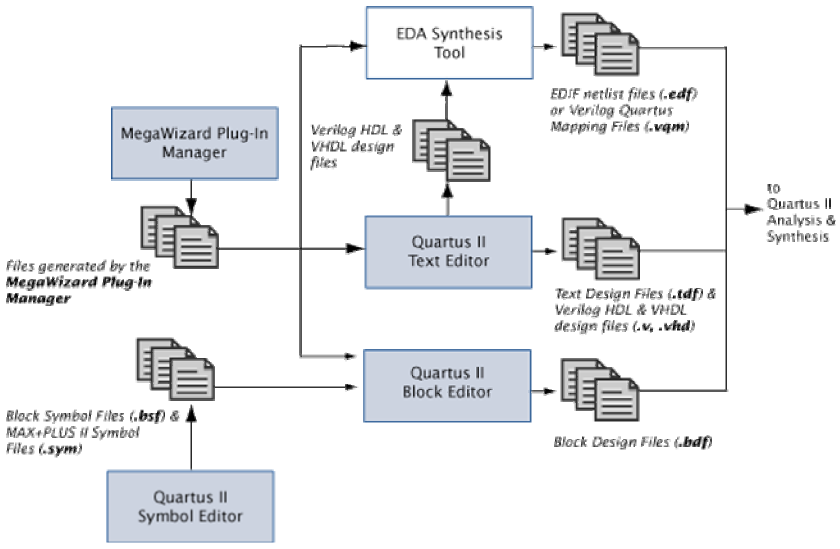
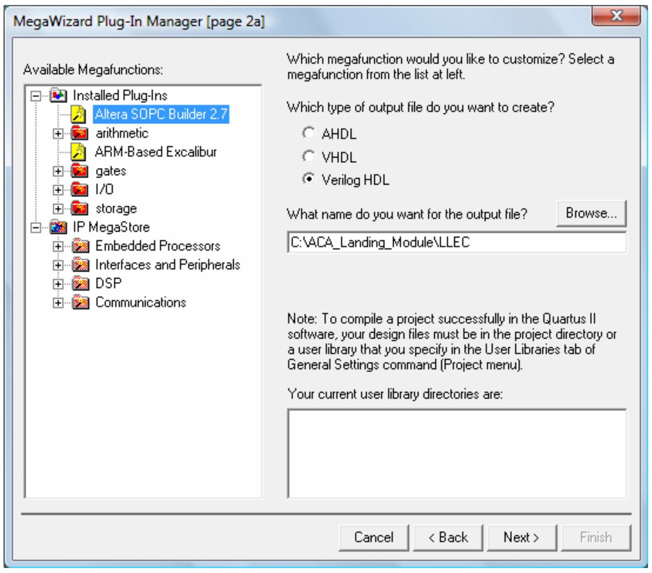

EDA Synthesis Tool
EDIF netlist files (.edf) or Verilog Quartus Mapping Files (.vqm)
MegaWizard Plug-In Manager
Verilog HDL & VHDL design files
to Quartus II Analysis & Synthesis
Files generated by the MegaWizard Plug-In Manager
Quartus II Text Editor
Text Design Files (.tdf) & Verilog HDL & VHDL design files (.v, .vhd)
Quartus II Block Editor
Block Symbol Files (.bsf) & MAX+PLUS II Symbol Files (.sym)
Block Design Files (.bdf)
Quartus II Symbol Editor

Fig.8. Quartus II design entry flow

The Block Editor allows entering and editing graphic design information in the form of schematics and blocking diagrams. The Block Editor reads and edits Block Design Files (.bdf) that contain blocks and symbols that represent logic in the design.

The Block Editor also provides a set of tools that help you connect blocks and primitives together, including bus and node connections and signal name mapping. You can change the Block Editor to display options, such as guidelines and grid spacing, rubberbanding, colors and screen elements, zoom, and different block and primitive properties to suit your preferences.

The MegaWizard Plug-In Manager helps to create or modify design files that contain custom megafunction variations, which you can then instantiate in a design file. These custom megafunction variations are based on Altera provided megafunctions. It runs a wizard that helps to easily specify options for the custom megafunction variations. The wizard is able to use SOPC Builder.



Which megafunction would you like to customize? Select a megafunction from the list at left.

Which type of output file do you want to create?

AHDL
VHDL
[✓ Verilog HDL]

What name do you want for the output file?

C:\ACA_Landing_Module\LLEC

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory or a user library that you specify in the User Libraries tab of General Settings command (Project menu).

Your current user library directories are:

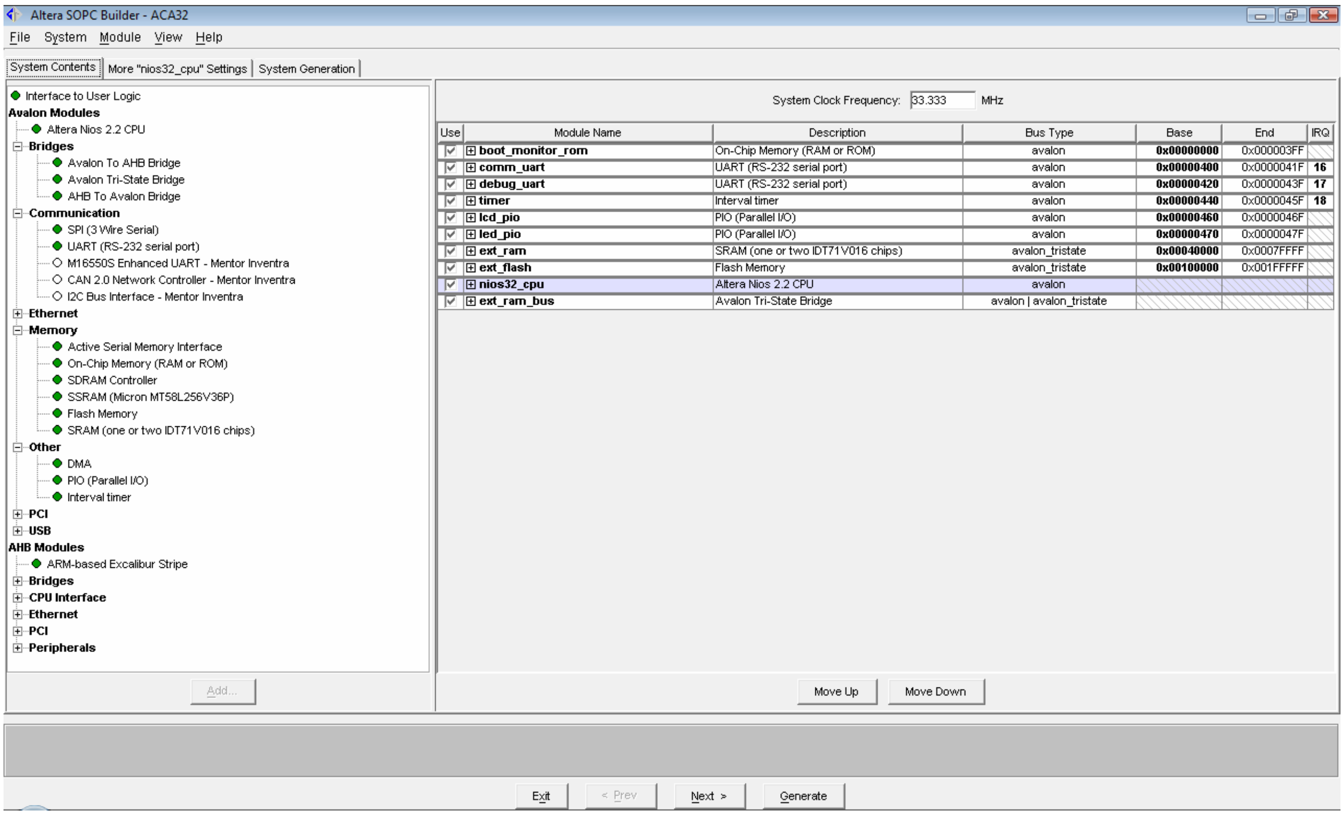
Fig.9. Quartus II MegaWizard plug-in manager

### 5.4.3 System On a Programable Chip Builder

SOPC Builder is an accessible tool through the Quartus II software for composing bus-based systems out of the provided library components. It creates a single system module that instantiates the list of specified components and generates automatically the bus logic on all system components. SOPC Builder provides output files for both synthesis and simulation.

Its two main parts consist on:

- A graphical user interface (GUI) for listing and arranging system components. Each component may also, itself, provide a graphical user interface for its own configuration. The GUI creates the system PTF fil which describes the system.



System Clock Frequency: 33.333 MHz

| Use | Module Name | Description | Bus Type | Base | End | IRQ |
|---|---|---|---|---|---|---|
| ✓ | boot_monitor_rom | On-Chip Memory (RAM or ROM) | avalon | 0x00000000 | 0x000003FF | |
| ✓ | comm_uart | UART (RS-232 serial port) | avalon | 0x00000400 | 0x0000041F | 16 |
| ✓ | debug_uart | UART (RS-232 serial port) | avalon | 0x00000420 | 0x0000043F | 17 |
| ✓ | timer | Interval timer | avalon | 0x00000440 | 0x0000045F | 18 |
| ✓ | lcd_pio | PIO (Parallel I/O) | avalon | 0x00000460 | 0x0000046F | |
| ✓ | led_pio | PIO (Parallel I/O) | avalon | 0x00000470 | 0x0000047F | |
| ✓ | ext_ram | SRAM (one or two IDT71V016 chips) | avalon_tristate | 0x00040000 | 0x0007FFFF | |
| ✓ | ext_flash | Flash Memory | avalon_tristate | 0x00100000 | 0x001FFFFF | |
| ✓ | nios32_cpu | Altera Nios 2.2 CPU | avalon | | | |
| ✓ | ext_ram_bus | Avalon Tri-State Bridge | avalon \| avalon_tristate | | | |

Fig.10. Altera SOPC Builder

- A generator program that converts the PTF file into its hardware implementation. The generator program creates an HDL of the system and then synthesizes it for the selected target device.

SOPC Builder has a set of built-in library components, including a UART, a timer, a PIO, an Avalon Tri-state bridge, and several simple memory interfaces.

All valid library components are represented by a component's class.ptf file which declares and defines all the information about that component so it can be recognized by SOPC Builder.

In addition, a component's library directory may also contain the logic that implements the component, software libraries and drivers that support the component.

In summary when the SOPC Builder generates a design the resulting processes are:

- The system memory map is checked for consistency. Peripheral addresses and interrupt priorities are verified to be unique and to fall within the range of valid entries for the CPU. If not, appropriate errors will be reported.
- A custom software development kit (SDK) is generated for the new system. The SDK consists of a compiled library of software routines for the SOPC design, a Makefile for rebuilding the library, and C header files containing structures for each peripheral.
- An HDL that describes the custom SOPC Module is generated.

## 5.5 Thrusters control

### 5.5.1 Attitude control

Quaternion Λ determinates attitude of a craft relative inertial coordinate system.

$$\Lambda = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \cos(\Theta/2) \\ i_1 \sin(\Theta/2) \\ i_2 \sin(\Theta/2) \\ i_3 \sin(\Theta/2) \end{bmatrix}$$

$i_1$, $i_2$, $i_3$ are body-fixed axes (X, Y, Z),
Θ is an angle between body-fixed and inertial coordinate systems.

$$\dot{\Lambda} = \frac{1}{2}\omega \circ \Lambda = \frac{1}{2}\begin{bmatrix} -\lambda_1\omega_1 - \lambda_2\omega_2 - \lambda_3\omega_3 \\ \lambda_0\omega_1 - \lambda_3\omega_2 + \lambda_2\omega_3 \\ \lambda_3\omega_1 + \lambda_0\omega_2 - \lambda_1\omega_3 \\ -\lambda_2\omega_1 + \lambda_1\omega_2 + \lambda_0\omega_3 \end{bmatrix},$$

For a rigid body subject to external torques the non-linear equation of motion model are given by:

$$I\dot{\omega} + \omega \times I\omega = M ,$$

Where:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

*I* is the moments of inertia tensor, , $I_{xy} = I_{yx}$, $I_{yz} = I_{zy}$, $I_{zx} = I_{xz}$,
*ω* is the body rate,
*ωdot* is the angular acceleration,
*M* is the applied torque.
If we use throttle guidance for thrusters, the thrust of k thruster is
$P_k=(P_k^{nom}+\Delta P_k)*K_{\text{throllte } k}$, k=1..6(?),

where:
$P_k^{nom}$ is a nominal thrust of a *k* thruster,
$\Delta P_k$ is an error of the nominal thrust,
$K_{\text{throllte } k}$ is a coefficient determined by a throttle (1 – full throttle).

Forces from k thruster are calculated thus:

$F_{k\,x}= P_k*\cos(\Delta\psi_k)*\cos(\Delta\theta_k)$,
$F_{k\,y}=-P_k*\sin(\Delta\theta_k)$,
$F_{k\,z}= P_k*\cos(\Delta\psi_k)*\cos(\Delta\theta_k)$,
where $\Delta\psi_k$, $\Delta\theta_k$ are errors of k thruster installation.

Torques (momentum) from thrusters are calculated thus:
$M_k = F_k\times(X_{center\text{-}of\text{-}mass}-X_k)$,
where
$M_k=\{M_{x\,k}, M_{yk}, M_{z\,k}\}^T$, $F_k=\{F_{x\,k}, F_{y\,k}, F_{z\,k}\}^T$
$X_{center\text{-}of\text{-}mass} = \{X_{c\text{-}o\text{-}m}, Y_{c\text{-}o\text{-}m}, Z_{c\text{-}o\text{-}m}\}^T$ are coordinates of center-of-mass,
$X_k=\{X_k, Y_k, Z_k\}^T$ are coordinates of k thruster,
× is crossproduct.

I calculate steering momentum thus:
$M_i=-r_i*\omega dot_i - h_i*\omega_i - k_i*\lambda_0*\lambda_i$, i=1(X), 2(Y), 3(Z) (ωdot isa angular acceleration)

Then we need to choose some thrusters which can create momentum mentioned above.

We know an angle of “moment action”:
$\alpha^*= M_Y/M_Z$
and an angle of an applied force:
$\alpha=\alpha^*-90^o$.

Then we can choose 2 thrusters (if there are 6 thrusters on Mark I as I drew on the picture) which make a positive momentum (thrusters 1 and 2 on the picture) and 2 opposite thrusters which make a negative momentum (thrusters 4 and 5 on the picture):

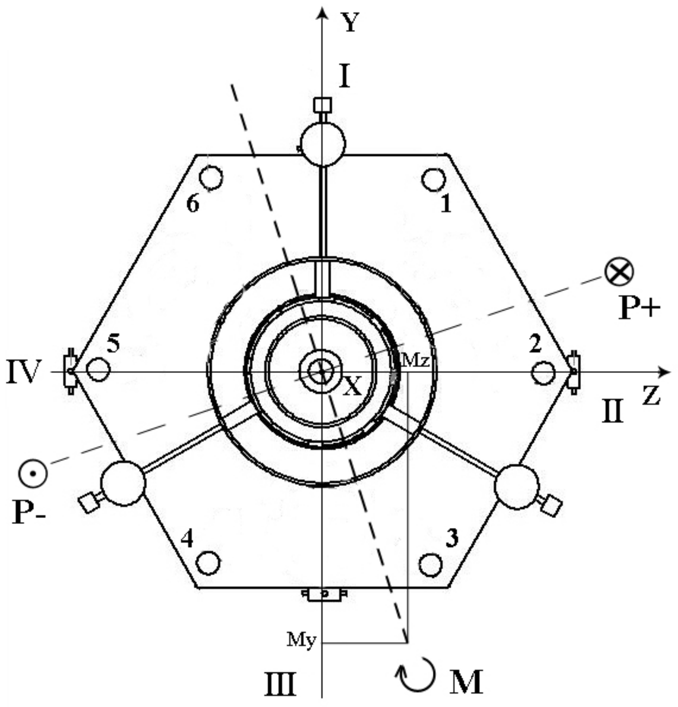


Fig.11. Lunar lander

$M_{Y1} + M_{Y2} = 1/2\ M_Y = 1/2\ (\Delta P_1{*}L_{Z1} + \Delta P_2{*}L_{Z2})$,
$M_{Z1} + M_{Z2} = 1/2\ M_Z = 1/2\ (-\Delta P_1{*}L_{Y1} - \Delta P_2{*}L_{Y2})$,
where $L_{Y1} = Y_{c\text{-}o\text{-}m} - Y_1$, $L_{Z1} = Z_{c\text{-}o\text{-}m} - Z_1$, $L_{Y2} = Y_{c\text{-}o\text{-}m} - Y_2$, $L_{Z2} = Z_{c\text{-}o\text{-}m} - Z_2$
(we'll pretend that $L_{Y1} = L_{Y4}$, $L_{Z1} = L_{Z4}$ and $L_{Y2} = L_{Y5}$, $L_{Z2} = L_{Z5}$).
Hence $\Delta P_1$ and $\Delta P_2$ are
$\Delta P_1 = 1/2\ (M_Y{*}L_{Y2} + M_Z{*}L_{Z2}) / (L_{Z1}{*}L_{Y2} - L_{Y1}{*}L_{Z2})$
$\Delta P_2 = 1/2\ (M_Y{*}L_{Y1} + M_Z{*}L_{Z1}) / (L_{Z1}{*}L_{Y2} - L_{Y1}{*}L_{Z2})$.

So we need to change a thrust of 1,2 and 4,5 thrusters:
$P_1^* = P_1 + \Delta P_1$, $P_2^* = P_2 + \Delta P_2$
$P_4^* = P_4 - \Delta P_1$, $P_5^* = P_5 - \Delta P_2$
Coefficients of throttles are (k=1,2,4,5): $K_{\text{throllte } k}^* = K_{\text{throllte } k} + \Delta P_k/P_k^{nom}$.

I don't know how to look a throttle (something which damp a fuel flow) so I just use a delay of 0.01 second as a throttle model. Also I thought that diapason of throttling can be 1..0.5.

Also I handled a task of craft attitude stabilization as keeping craft vertical orientation Λ relative inertial coordinate system of landing (look at picture).

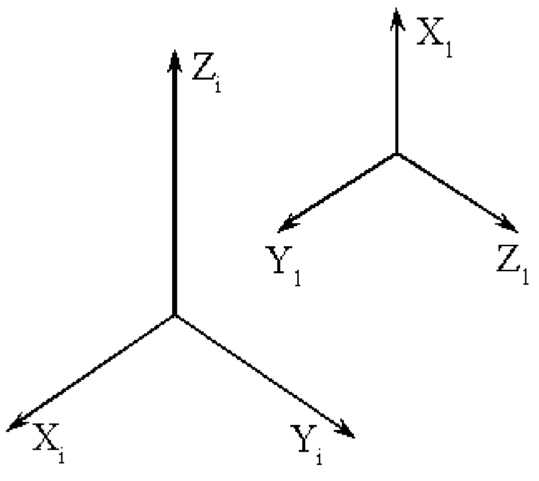


Fig.12. Lunar lander attitude reference system

### 5.5.2 Descent control

Center-of-mass motion is determined by equation
F=ma (2nd Newton's low)
or

$$\dot{V} = a = \frac{F}{m} - g;$$

$$\dot{X} = V \quad ,$$

where
$F=\{F_x, F_y, F_z\}^T$ are applied forces in an inertial coordinate system,
$V=\{V_x, V_y, V_z\}^T$ are craft velocities,
$X=\{x, y, z\}^T$ are craft coordinates,

$g = \{0, 0, 9.8\}^T$ m/s$^2$.
If we know forces in the body-fixed coordinate system $F_{bfcs}$ and orientation of a craft relative the inertial coordinate system $\Lambda$, we can calculate forces in the inertial coordinate system $F_{ics}$:

$$F_{ics} = \Lambda \circ F_{bfcs} \circ \Lambda^{-1}$$

If we want to keep constant vertical velocity $V_{descent}$ during descent, the steering low is:
$P_{ctrl} = -k_V*(V_Z-V_{descent}) -k_a*a_z$
Hence addition thrust is
$\Delta P = P_{ctrl} / 6$ (6 thrusters)
$K_{throllte\ k}^{**} = K_{throllte\ k} + \Delta P / 6$, k=1..6.

## 5.6 Attitude Control Algorithm of the landing module

For several reasons, including a better develop inicialization it has been decided to initially work with the standard 32-bit reference system which was previously implemented and contains a large list of components in his design where are included the required ones.

This section concerns the Lunar Lander Engine Control (LLEC) and the Attitude Control Algorith (ACA) that would control the engine servos in a future.
Remarkable is the fact that this is the first algorithm for the LLEC programmed in C-code, so a much easier Lunar Lander has been considered. So, here are presented the new considerations to make a more basic algorithm:

- Four motors are available.
- Each of them is placed in XY plane and located simetrically to the origin over each X/Y sexiamis.
- The numbering of the engines is “motor n”, where “n” is a number from 1 to 4 given to every engine, starting from positive x-axis and following a counterclockwise rotation.
- Input data are hypothetic angle entrys from accelerometers or targeted angles in both X and Y axis (Z-axis has been neglected just to simplify mathemathics calculations).
- Output data are hypothetic thrust for each of the engines expressed in percentage.
- The current calculation for the thrust is linear to the angle entry.

### 5.6.1 ACA parameters

The parameters the algorithm works with are listed and descripted in the given tables.

- Inputs

| Field | Range | Units | Description |
|---|---|---|---|
| IMU_aX | -180 to 180 | degrees | Inertial angle around X axis (Current) |
| IMU_aY | -180 to 180 | degrees | Inertial angle around Y axis (Current) |
| Target_aX | -180 to 180 | degrees | Commanded angle around X axis (Desired) |
| Target_aY | -180 to 180 | degrees | Commanded angle around Y axis (Desired) |

- Outputs

| Field | Range | Units | Description |
|---|---|---|---|
| Thrust_pX | 0 to 100 | % | Thrust positive X axis pointing towards -Z direction |
| Thrust_nX | 0 to 100 | % | Thrust negative X axis pointing towards -Z direction |
| Thrust_pY | 0 to 100 | % | Thrust positive Y axis pointing towards -Z direction |
| Thrust_nY | 0 to 100 | % | Thrust negative Y axis pointing towards -Z direction |

### 5.6.2 ACA clarifications

Basically, sets the Lunar Lander angle around X axis and Y axis to achieve a target angle for each axis depending on IMU_aX and IMU_aY changing the value of Thrust_pX, Thrust_pY, Thrust_nX and Thrust_nY values.

The sequence diagram of the algorithm is represented in the next image.

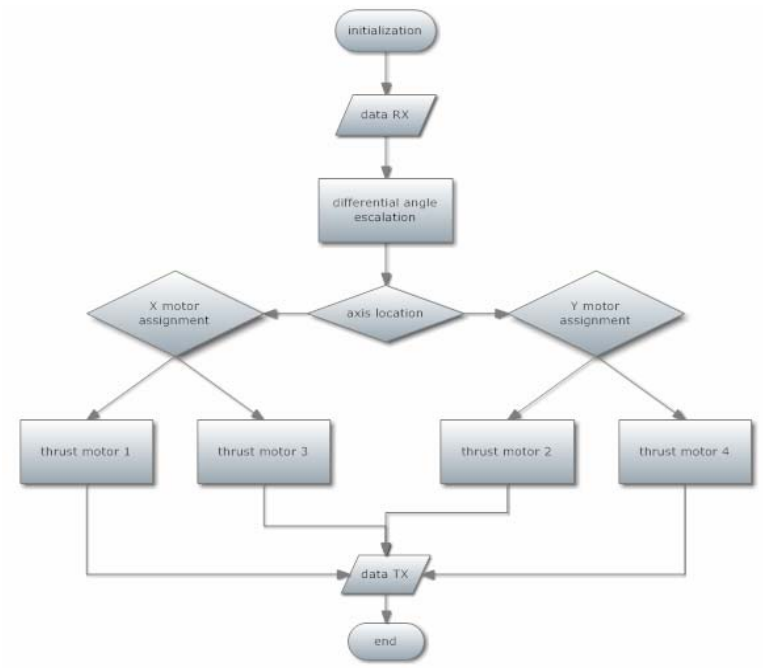


Fig.13. ACA sequence diagram

## *5.7 Landing module system design*

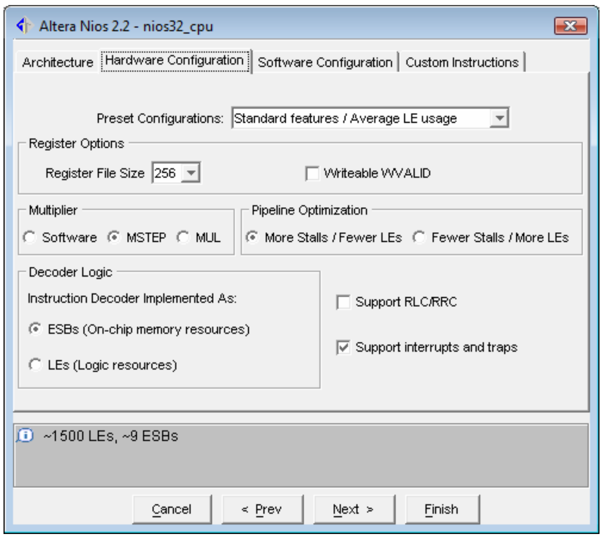
Here is described how is created the top-level Block Design File that contains the Attitude Control Algorithm of the Landing Module schematic. It is also explained how with SOPC Builder has been possible to create the embedded processor, configure system peripherals, and connect these elements to result in the system module.
Before begining to design, a new Quartus II project must have been created and must have been also specified the working directory for the ACA Landing Module project, establishing like this the top-level design entity recognizable for Quartus II and required to start with the system module design.

A complete system module contains an embedded processor and its associated, and required, system peripherals. Those peripherals allow the embedded processor to connect and communicate with internal logic in the APEX device or external hardware on the board. The components and their configurations are detailed below.

### 5.7.1 Altera Nios 2.2 CPU

In first instance, and as it has been commented before, the LLEC module implements an embedded processor in the APEX20KE device. Three proposals of factory-defined CPU configurations are given. The presets are:

- Minimum features
- Standard features
- Full features


Altera Nios 2.2 - nios32_cpu

Architecture | Hardware Configuration | Software Configuration | Custom Instructions

Preset Configurations: Standard features / Average LE usage

Register Options
Register File Size: 256
Writeable WVALID: [ ]

Multiplier: Software [✓ MSTEP] MUL

Pipeline Optimization: [✓ More Stalls / Fewer LEs] Fewer Stalls / More LEs

Decoder Logic
Instruction Decoder Implemented As:
[✓ ESBs (On-chip memory resources)]
LEs (Logic resources)

Support RLC/RRC: [ ]
Support interrupts and traps: [✓]

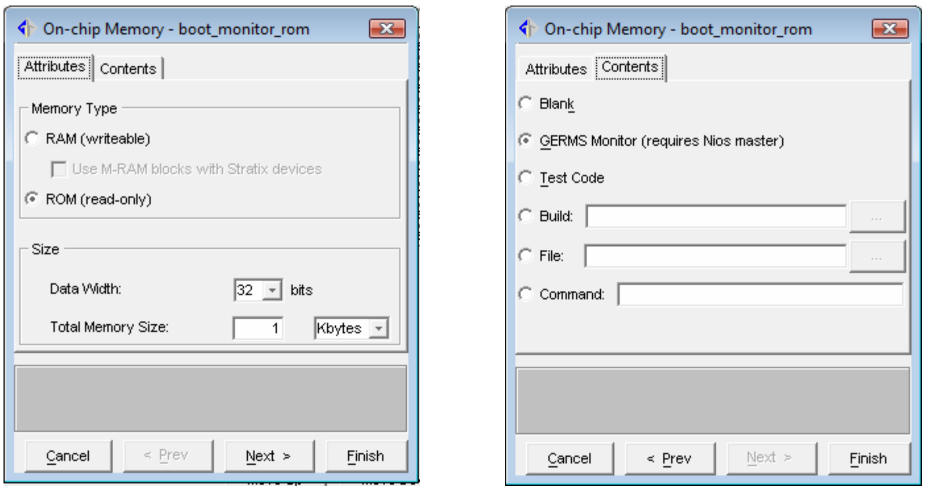
~1500 LEs, ~9 ESBs

Fig.14. CPU

The main differences between them reside on the register file size, the integer multiplier, the availability of supporting interrupts and/or the RLC/RRC protocol and the location of the resources to implement the 32-bit instruction decoder.
For the LLEC module the standard features have been choosen which includes a register file of 256 and the MSTEP multiplier which takes two clock cicles to have a result.

### 5.7.2 GERMS monitor

The GERMS monitor is a boot monitor program that provides basic development facilities for the board. It is included in the default design stored in flash memory. On power-up, the GERMS monitor is the first code to execute and it controls the boot process. Once booted, it provides a way to read from and write to the onboard memories.


On-chip Memory - boot_monitor_rom

Attributes | Contents

Memory Type
RAM (writeable)
Use M-RAM blocks with Stratix devices: [ ]
[✓ ROM (read-only)]

Size
Data Width: 32 bits
Total Memory Size: 1 Kbytes

On-chip Memory - boot_monitor_rom

Attributes | Contents

Blank
[✓ GERMS Monitor (requires Nios master)]
Test Code
Build:
File:
Command:

Fig.15. GERMS monitor

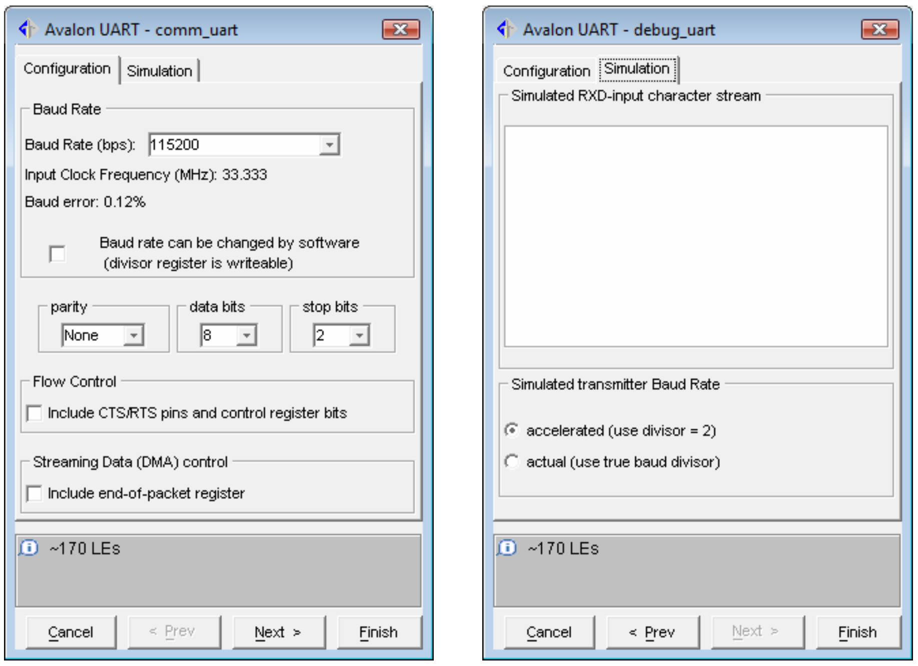
It is implemented as a ROM with 32-bit data width occupying a total of 1 Kbytes. The option to select the GERMS preset appears in the contents tab.

### 5.7.3 Universal Asynchronous Receiver/Transmitter

The UART implements a RS-232 asynchronous transmit and receive logic inside the device. The UART sends and receives serial data over RxD and TxD external pins. Software controls and communicates with the UART through memory-mapped, 16-bit registers.
To comply with RS-232 voltage-signaling, an external level shifting buffer is required between the TxD/RxD I/O pins and the corresponding serial port external connections.
The UART runs on a single synchronous clock input and its internal baud clock is derived from the same as the system clock. Its rate can be fixed or settable via the divisor register which is used, as it name says, to divide clock's period and generate the desired frequency.

Two different designs, but similar, are required for the LLEC module. Both of them work at 115200 bps fixed bit rate (which is the maximum allowed) and have an 8-bit data width with no parity checking, RTS/CTS protocols and end-of-packet register use either.
So the available methods remaining to detect tx/rx process errors are these:

- Through stop bits which split the frames. When a stop bit is not detected, the fe bit in the status register shifts from "0" to "1".
- Overrun detection. That occurs when a character is transmitted/received before being ready to do so.


Avalon UART - comm_uart

Configuration | Simulation

Baud Rate
Baud Rate (bps): 115200
Input Clock Frequency (MHz): 33.333
Baud error: 0.12%
☐ Baud rate can be changed by software (divisor register is writeable)

parity: None
data bits: 8
stop bits: 2

Flow Control
☐ Include CTS/RTS pins and control register bits

Streaming Data (DMA) control
☐ Include end-of-packet register

~170 LEs

Avalon UART - debug_uart

Configuration | Simulation

Simulated RXD-input character stream

Simulated transmitter Baud Rate
● accelerated (use divisor = 2)
○ actual (use true baud divisor)

~170 LEs

Fig.16. UART

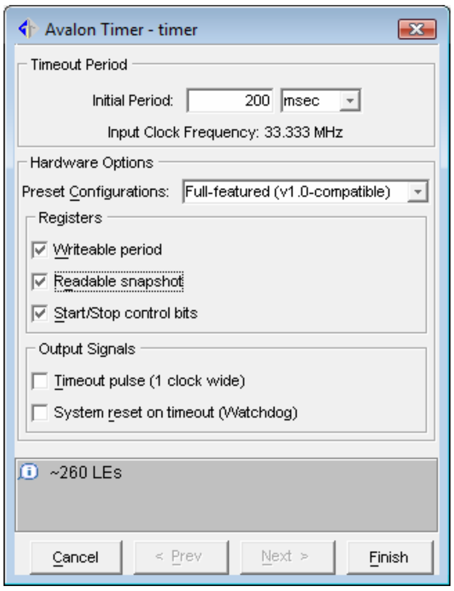
Precisely, the only change from communication UART to debugging is the stop bits number, which are 2 and 1 respectively.

### 5.7.4 Timer

The Timer peripheral is a simple 32-bit interval timer that can be controlled by software. It generates a single interrupt-request output that can be masked by an internal control bit. Because all embedded processor accessible registers are 16 bits wide a 32-bit Nios CPU performs two separate write operations on two 16-bit registers (periodl and periodh) to set a 32-bit downcount value.
Again, as UART also does, the Timer runs off a single master clock input which coincides the same clock provided to the CPU.

A single timer, but full-featured, has been decided. The reason to include only one is the next. Despite requiring one for the input data flow control and a second for the output data (both needed for the Software-In-the-Loop verification) they can be implemented as once if it is taken into consideration that the period of the output data is a multiple of the period of entry.

Full featured stands for a selectable period (whitin the period register possibilities) and snapshot (capture the current timer value) support. It also enables the start/stop bits in the control register to be able to play with the counter.


Avalon Timer - timer

Timeout Period

Initial Period: 200 msec

Input Clock Frequency: 33.333 MHz

Hardware Options

Preset Configurations: Full-featured (v1.0-compatible)

Registers

- [✓] Writeable period
- [✓] Readable snapshot
- [✓] Start/Stop control bits

Output Signals

- [ ] Timeout pulse (1 clock wide)
- [ ] System reset on timeout (Watchdog)

~260 LEs

Fig.17. Timer

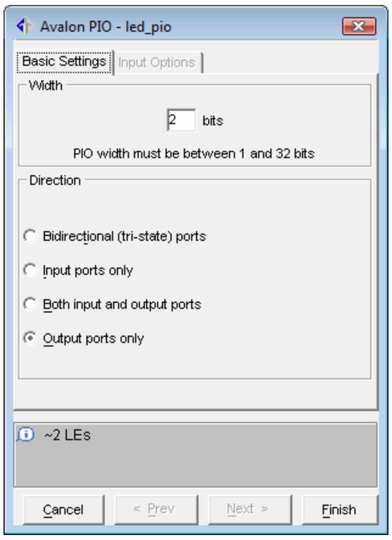
### 5.7.5 Parallel Input/Output

A Parallel Input/Output (PIO) module is a memory-mapped interface between software and the defined logic. The PIO has two contrary applications:

- Providing an interface between software and the defined logic within the same device.
- Providing an interface between software and the peripheral logic that resides outside the device.

Two main different PIO have been used for the module, both as interrupt monitoring.

The first one is implented for the to LEDs onboard and it is as simple as two bits configured with output direction to get their lights on.


Avalon PIO - led_pio

Basic Settings | Input Options

Width: 2 bits

PIO width must be between 1 and 32 bits

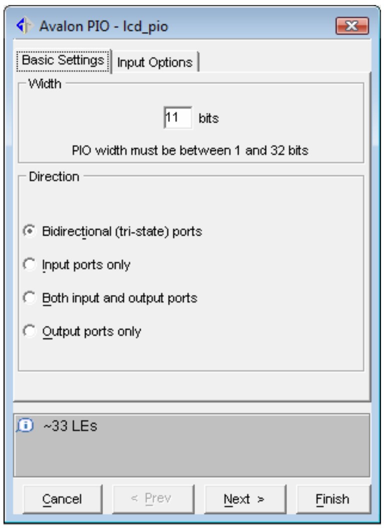
Direction:
- Bidirectional (tri-state) ports
- Input ports only
- Both input and output ports
- [✓ Output ports only]

~2 LEs

Cancel | < Prev | Next > | Finish

Fig.18. PIO LED

Although JP12 connector (detailed before) has 14 pins, the second one is an 11 bidirectional PIO. They are 11 because the pins 1 and 2 are required to provide the display feeding and the third goes to the monitor reset.
Punctuate that because of the rupture third pin (monitor reset) it is short circuited to pin 1, it is to say, that a reset to the LCD, then, is only possible through explicit software deletion.


Avalon PIO - lcd_pio

Basic Settings | Input Options

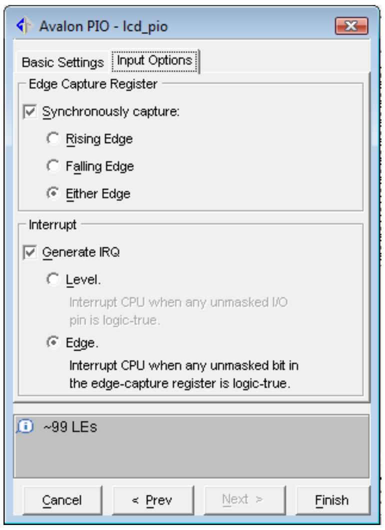
Width: 11 bits

PIO width must be between 1 and 32 bits

Direction:
- [✓ Bidirectional (tri-state) ports]
- Input ports only
- Both input and output ports
- Output ports only

~33 LEs

Cancel | < Prev | Next > | Finish


Avalon PIO - lcd_pio

Basic Settings | Input Options

Edge Capture Register:
- [✓] Synchronously capture:
  - Rising Edge
  - Falling Edge
  - [✓ Either Edge]

Interrupt:
- [✓] Generate IRQ
  - Level. Interrupt CPU when any unmasked I/O pin is logic-true.
  - [✓ Edge.] Interrupt CPU when any unmasked bit in the edge-capture register is logic-true.

~99 LEs

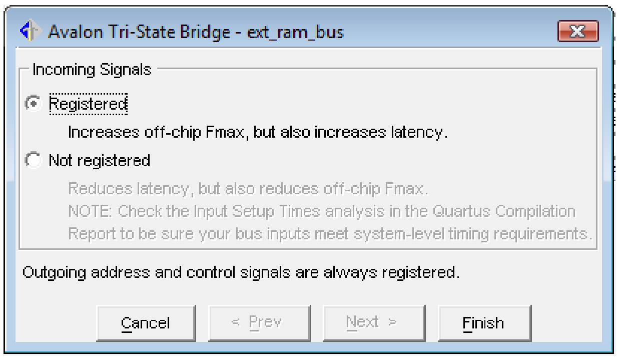
Cancel | < Prev | Next > | Finish

Fig.19. PIO LCD

Finally, comment that despite beeing used only as a status informater the answer to choose bidirectional pins is that wheter it was necessary to monitor some parameter it would be monitorized as well as taken as input for another device and have it all in once.

### 5.7.6 External RAM BUS

For the system to connect on-chip processors and peripherals together, a bridge between the Avalon bus and the buses to which the external memory is connected must be added.


Avalon Tri-State Bridge - ext_ram_bus

Incoming Signals

[✓ Registered]

Increases off-chip Fmax, but also increases latency.

Not registered

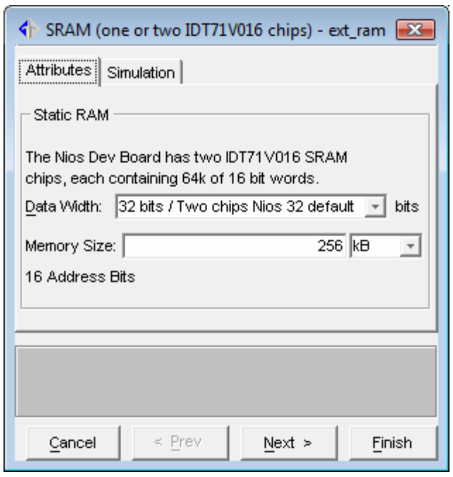
Reduces latency, but also reduces off-chip Fmax.
NOTE: Check the Input Setup Times analysis in the Quartus Compilation Report to be sure your bus inputs meet system-level timing requirements.

Outgoing address and control signals are always registered.

Fig.20. RAM bus

### 5.7.7 External RAM

A dual Static-RAM interface has been choosen to download the C-code. Note that the current .srec file downloaded onto the board to run the Software-In-the-Loop test occupies 121KB, so a 256KB total memory size has been decided, like this, more than enough memory is available.


SRAM (one or two IDT71V016 chips) - ext_ram

Static RAM

The Nios Dev Board has two IDT71V016 SRAM chips, each containing 64k of 16 bit words.

Data Width: 32 bits / Two chips Nios 32 default bits

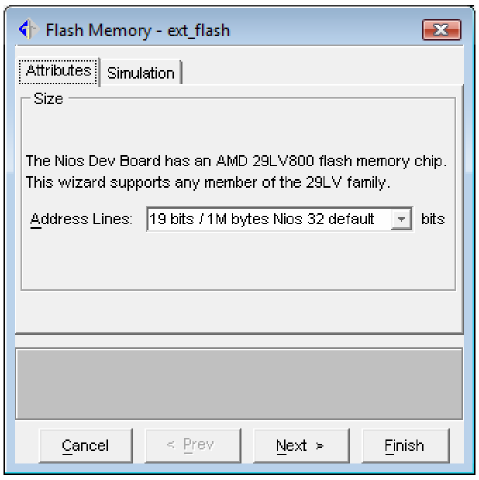
Memory Size: 256 kB

16 Address Bits

Fig.21. SRAM

### 5.7.8 External Flash

For the flash memory the maximum featured preset has been implemented. It offers 1MB of memory which can be used as non-volatile storage or to hold PLD configuration (.hexout files). It is enough memory since, for example, the standard_32_ext.hexout file used for developping both software/hardware-In-the-Loop (a design which largely covers the LLEC module requirements) occupies 505KB.



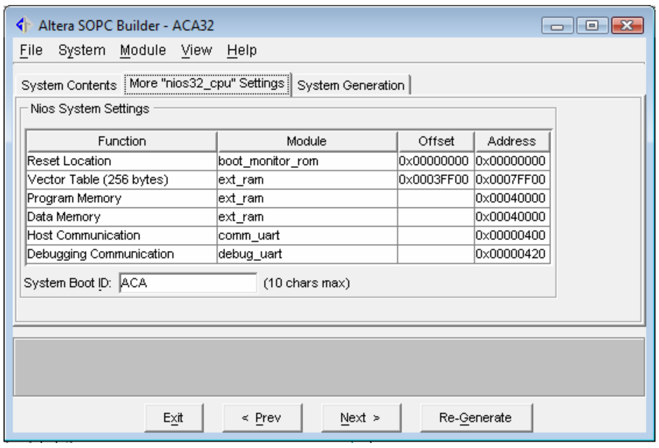
The Nios Dev Board has an AMD 29LV800 flash memory chip. This wizard supports any member of the 29LV family.

Address Lines: 19 bits / 1M bytes Nios 32 default bits

Fig.22. Flash memory

## 5.8 *Lunar Lander Engine module configuration*

After including all the system components, it is required to check that each function is developped by the correct module. To make it easier, a tab with the cpu system settings is provided in the design generating process.



| Function | Module | Offset | Address |
|---|---|---|---|
| Reset Location | boot_monitor_rom | 0x00000000 | 0x00000000 |
| Vector Table (256 bytes) | ext_ram | 0x0003FF00 | 0x0007FF00 |
| Program Memory | ext_ram | | 0x00040000 |
| Data Memory | ext_ram | | 0x00040000 |
| Host Communication | comm_uart | | 0x00000400 |
| Debugging Communication | debug_uart | | 0x00000420 |

System Boot ID: ACA (10 chars max)

Fig.23. SOPC Builder CPU settings

The system generator tab reproduces the SOPC Builder design providing all the SDK headers and libraries required for the selected peripherals, as well as VHDL files to implement the system logic. The resulting design is shown below.

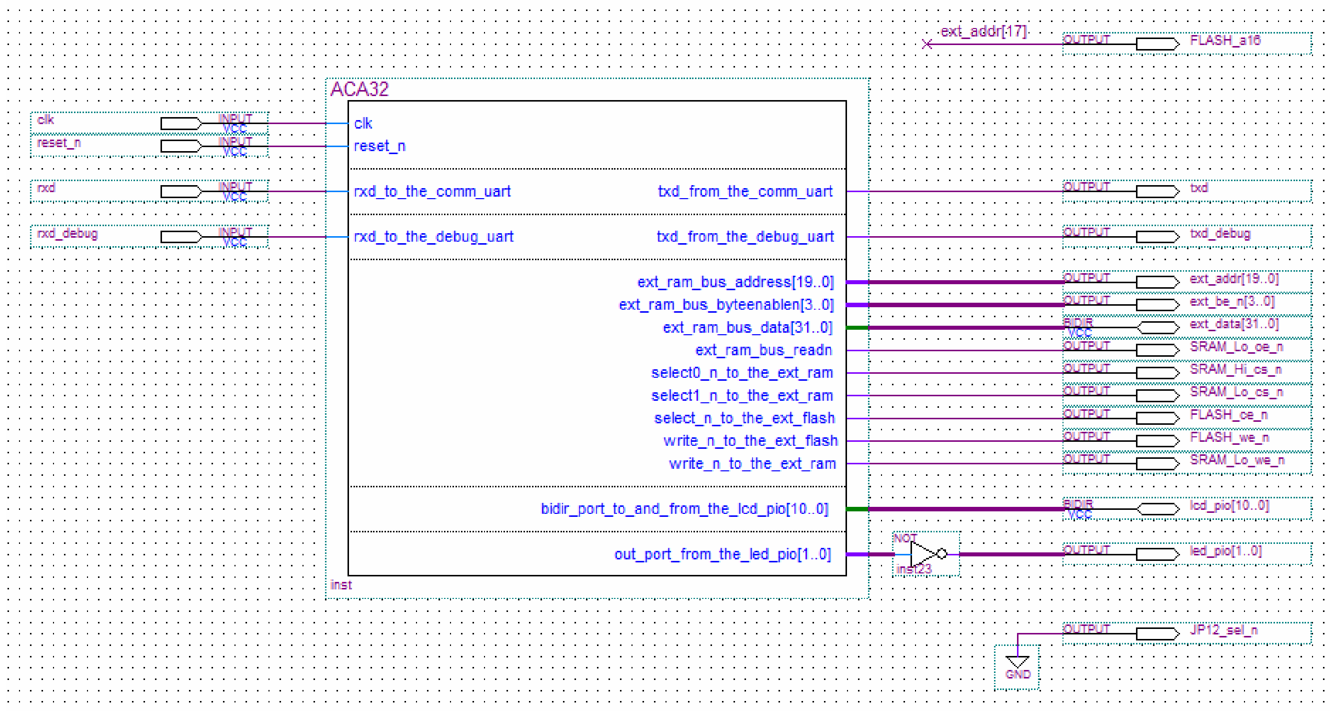
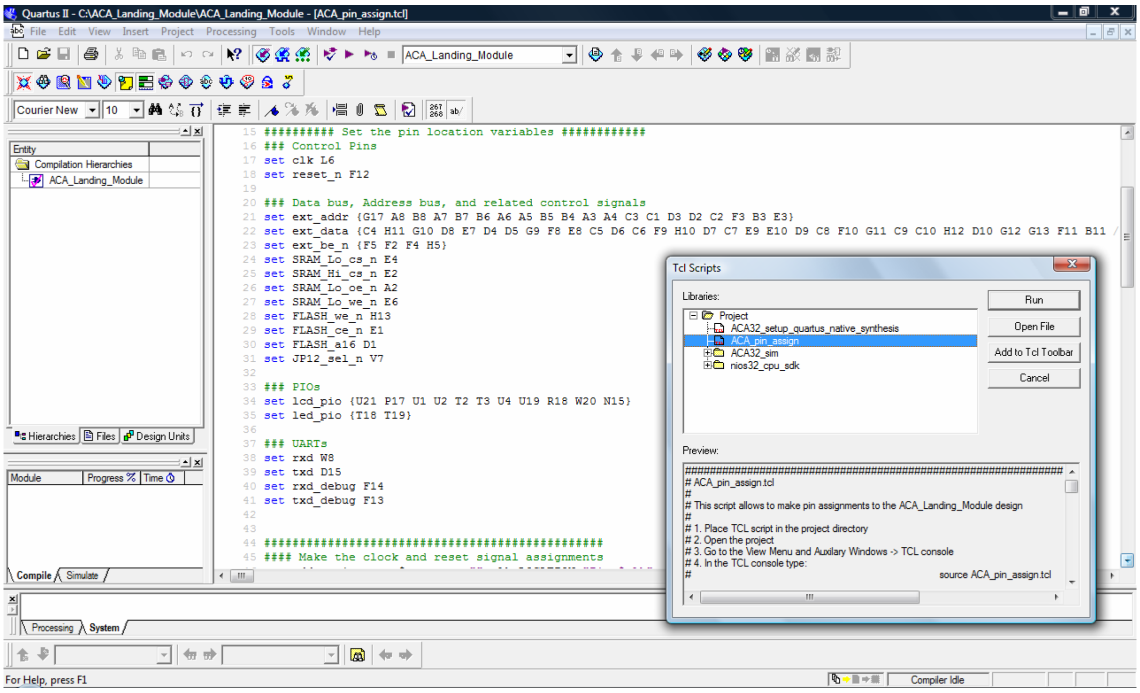


Fig.24. ACA landing module

Finally, after the schematic has been realized it would be necessary to make the pin assignments to enable compilation. The two available methods to do so are:

- Via a Tool Command Language (tcl) file, as it has a console application which is able to execute sequentially the standard commands as well as also the ones from the script.



```
########## Set the pin location variables ############
### Control Pins
set clk L6
set reset_n F12

### Data bus, Address bus, and related control signals
set ext_addr {G17 A8 B8 A7 B7 B6 A6 A5 B5 B4 A3 A4 C3 C1 D3 D2 C2 F3 B3 E3}
set ext_data {C4 H11 G10 D8 E7 D4 D5 G9 F8 E8 C5 D6 C6 F9 H10 D7 C7 E9 E10 D9 C8 F10 G11 C9 C10 H12 D10 G12 G13 F11 B11
set ext_be_n {F5 F2 F4 H5}
set SRAM_Lo_cs_n E4
set SRAM_Hi_cs_n E2
set SRAM_Lo_oe_n A2
set SRAM_Lo_we_n E6
set FLASH_we_n H13
set FLASH_ce_n E1
set FLASH_a16 D1
set JP12_sel_n V7

### PIOs
set lcd_pio {U21 P17 U1 U2 T2 T3 U4 U19 R18 W20 N15}
set led_pio {T18 T19}

### UARTs
set rxd W8
set txd D15
set rxd_debug F14
set txd_debug F13


##############################################
#### Make the clock and reset signal assignments
```

```
######################################################################
# ACA_pin_assign.tcl
#
# This script allows to make pin assignments to the ACA_Landing_Module design
#
# 1. Place TCL script in the project directory
# 2. Open the project
# 3. Go to the View Menu and Auxilary Windows -> TCL console
# 4. In the TCL console type:
#                                     source ACA_pin_assign.tcl
```

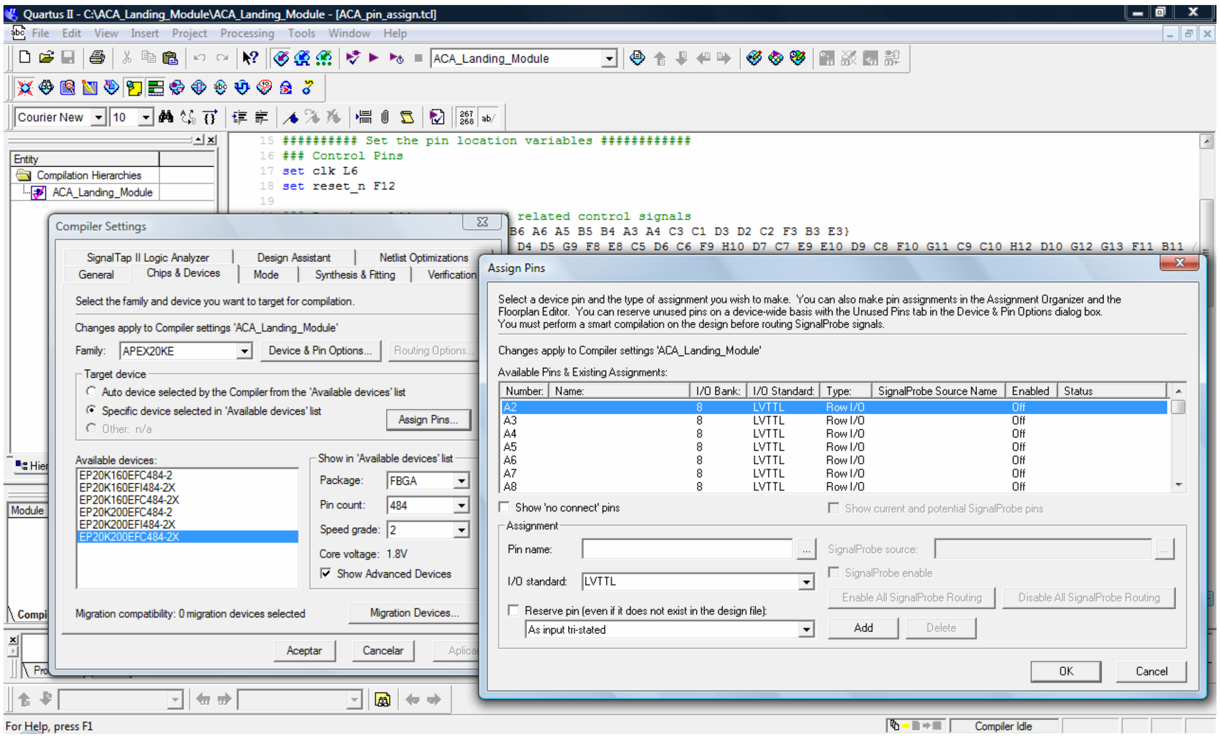
Fig.25. Tcl script selection

The ACA_pin_assign.tcl is the next one.

```
########## Set the pin location variables ############
### Control Pins
set clk L6
set reset_n F12

### Data bus, Address bus, and related control signals
set ext_addr {G17 A8 B8 A7 B7 B6 A6 A5 B5 B4 A3 A4 C3 C1 D3 D2 C2 F3 B3 E3}
set ext_data {C4 H11 G10 D8 E7 D4 D5 G9 F8 E8 C5 D6 C6 F9 H10 D7 C7 E9 E10 D9 C8 F10 G11 C9
C10 H12 D10 G12 G13 F11 B11 B10}
set ext_be_n {F5 F2 F4 H5}
set SRAM_Lo_cs_n E4
set SRAM_Hi_cs_n E2
set SRAM_Lo_oe_n A2
set SRAM_Lo_we_n E6
set FLASH_we_n H13
set FLASH_ce_n E1
set FLASH_a16 D1
set JP12_sel_n V7

### PIOs
set lcd_pio {U21 P17 U1 U2 T2 T3 U4 U19 R18 W20 N15}
set led_pio {T18 T19}

### UARTs
set rxd W8
set txd D15
set rxd_debug F14
set txd_debug F13

#### Make the clock and reset signal assignments
cmp add_assignment $top_name "" clk LOCATION "Pin_$clk"
cmp add_assignment $top_name "" reset_n LOCATION "Pin_$reset_n"

#### Make the external Flash and SRAM assignments
set i 0
foreach {a} $ext_addr {
        cmp add_assignment $top_name "" "ext_addr\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}
set i 0
foreach {a} $ext_data {
        cmp add_assignment $top_name "" "ext_data\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}
set i 0
foreach {a} $ext_be_n {
        cmp add_assignment $top_name "" "ext_be_n\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}
cmp add_assignment $top_name "" "SRAM_Lo_cs_n" LOCATION "Pin_$SRAM_Lo_cs_n"
cmp add_assignment $top_name "" "SRAM_Hi_cs_n" LOCATION "Pin_$SRAM_Hi_cs_n"
cmp add_assignment $top_name "" "SRAM_Lo_oe_n" LOCATION "Pin_$SRAM_Lo_oe_n"
cmp add_assignment $top_name "" "SRAM_Lo_we_n" LOCATION "Pin_$SRAM_Lo_we_n"
cmp add_assignment $top_name "" "FLASH_we_n" LOCATION "Pin_$FLASH_we_n"
cmp add_assignment $top_name "" "FLASH_ce_n" LOCATION "Pin_$FLASH_ce_n"
cmp add_assignment $top_name "" "FLASH_a16" LOCATION "Pin_$FLASH_a16"
cmp add_assignment $top_name "" "JP12_sel_n" LOCATION "Pin_$JP12_sel_n"
```

```
#### Make the PIO pin assignments
set i 0
foreach {a} $lcd_pio {
        cmp add_assignment $top_name "" "lcd_pio\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}
set i 0
foreach {a} $led_pio {
        cmp add_assignment $top_name "" "led_pio\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}

################################
#### Make the UART assignments
cmp add_assignment $top_name "" "rxd" LOCATION "Pin_$rxd"
cmp add_assignment $top_name "" "txd" LOCATION "Pin_$txd"
cmp add_assignment $top_name "" "rxd_debug" LOCATION "Pin_$rxd_debug"
cmp add_assignment $top_name "" "txd_debug" LOCATION "Pin_$txd_debug"
```

- Manually, accessing from compiler settings and introducing the main device model mounted on the board and its specifications. In this case and detailed above is the APEX EP20K200EFC484-2x chip.


Quartus II - C:\ACA_Landing_Module\ACA_Landing_Module - [ACA_pin_assign.tcl]
File Edit View Insert Project Processing Tools Window Help
ACA_Landing_Module
Courier New 10
Entity
Compilation Hierarchies
ACA_Landing_Module
For Help, press F1
Compiler Idle

```
########## Set the pin location variables ############
### Control Pins
set clk L6
set reset_n F12

... related control signals
B6 A6 A5 B5 B4 A3 A4 C3 C1 D3 D2 C2 F3 B3 E3}
D4 D5 G9 F8 E8 C5 D6 C6 F9 H10 D7 C7 E9 E10 D9 C8 F10 G11 C9 C10 H12 D10 G12 G13 F11 B11
```

Compiler Settings

SignalTap II Logic Analyzer | Design Assistant | Netlist Optimizations | General | Chips & Devices | Mode | Synthesis & Fitting | Verification

Select the family and device you want to target for compilation.

Changes apply to Compiler settings 'ACA_Landing_Module'

Family: APEX20KE   Device & Pin Options...   Routing Options...

Target device
- ( ) Auto device selected by the Compiler from the 'Available devices' list
- (●) Specific device selected in 'Available devices' list
- ( ) Other: n/a

Assign Pins...

Available devices:
- EP20K160EFC484-2
- EP20K160EFI484-2X
- EP20K160EFC484-2X
- EP20K200EFC484-2
- EP20K200EFI484-2X
- EP20K200EFC484-2X

Show in 'Available devices' list
- Package: FBGA
- Pin count: 484
- Speed grade: 2
- Core voltage: 1.8V
- [✓] Show Advanced Devices

Migration compatibility: 0 migration devices selected   Migration Devices...

Aceptar   Cancelar   Aplicar

Assign Pins

Select a device pin and the type of assignment you wish to make. You can also make pin assignments in the Assignment Organizer and the Floorplan Editor. You can reserve unused pins on a device-wide basis with the Unused Pins tab in the Device & Pin Options dialog box. You must perform a smart compilation on the design before routing SignalProbe signals.

Changes apply to Compiler settings 'ACA_Landing_Module'

Available Pins & Existing Assignments:

| Number: | Name: | I/O Bank: | I/O Standard: | Type: | SignalProbe Source Name | Enabled | Status |
|---|---|---|---|---|---|---|---|
| A2 | | 8 | LVTTL | Row I/O | | Off | |
| A3 | | 8 | LVTTL | Row I/O | | Off | |
| A4 | | 8 | LVTTL | Row I/O | | Off | |
| A5 | | 8 | LVTTL | Row I/O | | Off | |
| A6 | | 8 | LVTTL | Row I/O | | Off | |
| A7 | | 8 | LVTTL | Row I/O | | Off | |
| A8 | | 8 | LVTTL | Row I/O | | Off | |

[ ] Show 'no connect' pins   [ ] Show current and potential SignalProbe pins

Assignment
- Pin name:
- I/O standard: LVTTL
- [ ] Reserve pin (even if it does not exist in the design file): As input tri-stated
- SignalProbe source:
- [ ] SignalProbe enable
- Enable All SignalProbe Routing   Disable All SignalProbe Routing
- Add   Delete

OK   Cancel

Fig.26. Manual pinout assignment

But only refer to the implemented connexions could not be enough. At the end, also through the compiler settings, it could be compulsory to configure the unused pins in high-impedance mode to emulate an open-circuit, if existing. If not, the design will not run.

## 5.9 Control module validation

One of core areas is verification and validation (V&V). As said in the first pages, V&V is essential for any project to provide quality and robustness of the model system development process. It can be used to check the accuracy of the models and algorithms, to test code generation or to test hardware and/or software interactions.
Remark that in order to ensure that every aspect of the processor code has been tested, it is necessary to derive an appropriate set of input conditions to fully exercise the code. Failure to do so leads to hidden bugs in the module being tested.

Some references are done to the stages of these testing processes. Details for the Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) are given below.

### 5.9.1 Software In the Loop

SIL refers precisely to the kind of testing done to validate the behavior of the main algorithm. The code is intended to be adapted and integrated into the simulator implemented in the FPGA.

For this purpose and to simplify the algorithm validation tasks, the algorithm is incrusted in a C-code program that runs the algorithm using the embedded processor and the required peripherals.

The sequence diagram for the simulation is the following:

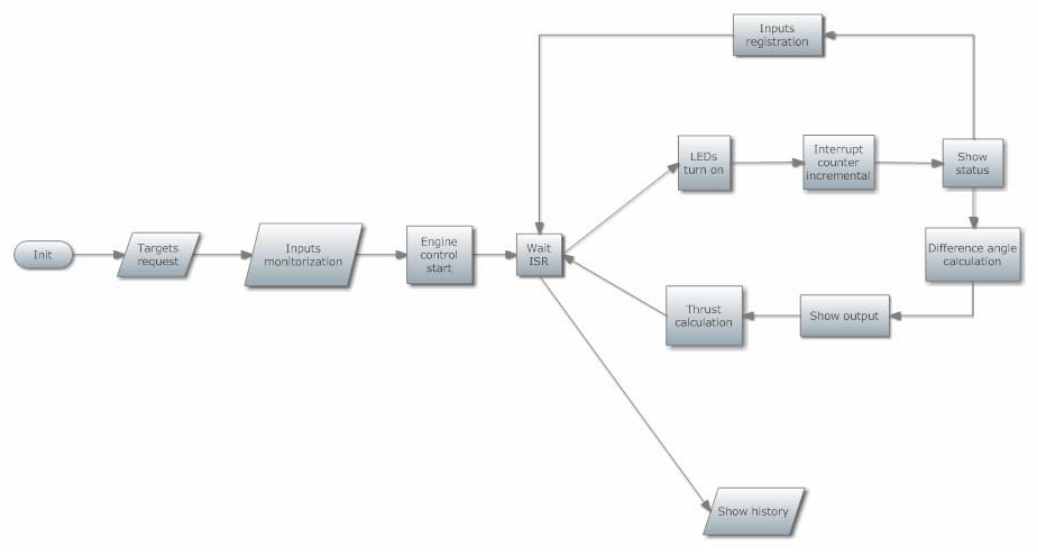


Fig.27. SIL sequence diagram

### 5.9.2 Hardware In the Loop

HIL, for its part, refers to the kind of testing done to study the interactions between the software and the hardware. In a typical HIL setup, one piece of hardware is connected to a software model of the rest of the system.

It would mainly consist on implementing the entire FPGA board inside the Moon2.0 simulator. The ACA module would receive data from the simulator and, after being processed in the embedded CPU, it would transmit the resulting engine thrusts calculations. The communication would be stablished via a UART, already reviewed.

### 5.9.3 Real flight test

Obviously, the last verification for the LLEC module should be to introduce the system in an aircraft flight as it would be implemented with the real components and developping its real function, it is to say, controlling thrust engines to permit a real flight.

# 6 CONCLUSIONS

Working with Team FREDNET has shown me the way aeronautical industry follows, its actual situation and what kinds of matters are actually deeply considered.
Furthermore, it also has taught me learning about self project guiding and to adapt my work according to team necessities. With that I would like to emphasize the variety of topics touched and developed along this TFC.

Also critical for our purposes, and maybe increased, because of my rol inside Picorover team is the fact that team has to show and share the achieved information, as well as explain the pending work to do. In response for that goal Wiki posting procedure is basic to enable collaboration with other Team FREDnet members or to obtain new team or group members in charge of developping new or pending areas, as it has been shown it is a complete and real project with real applicatin.

But, of course, some problems like eventual bad team information or communication insufficient work leaders, collaborators or resources are present into an open project. An example is the thruster engines control which has been just integred to the group information because the member in charge of this part Presented it a week ago or so.

Another example could be the licensing problems which have supposed a big trouble to deal and which are not solved at the moment. In a first instance, they have caused a slower development of the project because it was not able to obtain full capacities from the available tools, such as programming. In a second place, and even more difficult to save, it is the fact that they have also trunked the module design which is not complete now, so it has broken this way of knowledge.

However, a large volume of technical documentation has been dealed and processed and catalogued to produce prototype developing in this case for the hardware, which includes the FPGA board or Quartus II software.
Many disciplines have been used for the development of this project. A member has to work closer to many people of different fields and different countries and develop a unique project. The help of many teachers known through our studies make possible such challenger.
Finally, just to say that we, as a team, are glad by the work which has been already done, but a lot rest of work is to be finished and until final implementation will be reached we will not see our work been applied, although our knowledge is being improved everyday.

# 7 Environmental impact

The whole project will develop over different environment starting from the Earth’s surface and until reaching the moon’s surface, but also through several Earth’s orbits or free-space. So, a lot of environmental reviews must be done at different levels to consider the possible impacts. As an example, at the end of its mission, the Lunar Bus will suffer a controlled crash on the moon.
Remark that coordinates of the areas of disposal devices will be published and catalogued if necessary.

Environment care has been considered for optimizing project process. As it has already been commented in the FPGA justification, the use of the FPGA is a clear example of environmental care as it would permit to use a single device to develop different subsystems inside Team FREDnet organization, as well as embed those belonging to the same system. So, implicitaly, it will reduce the number of peripheral needed and used.

Picorover team always considers optimization of the resources on its procedures in order to reduce the impact on the environment.

# 8 Bibliography

[1] Google Lunar X-Prize
http://www.googlelunarxprize.org/

[2] N-Prize
http://www.n-prize.com/

[3] Team FREDnet
http://www.frednet.com/

[4] Team FREDnet Wiki
http://wiki.teamfrednet.org/index.php/Main_Page

[5] Picorover Wiki
http://wiki.teamfrednet.org/index.php/Portal:Picorover

[6] Lunar Lander Wiki
http://wiki.teamfrednet.org/index.php/Portal:Lunar_Lander

[7] Lunar Bus Wiki
http://wiki.teamfrednet.org/index.php/Portal:Lunar_Bus

[8] Wikipedia
http://en.wikipedia.org/wiki/Main_Page

[9] Altera support
http://www.altera.com/

[10] Altera Nios Documentation
avalon_bus_spec.pdf
nios_board_schematic.pdf
nios_cpu_datasheet.pdf
nios_development_board.pdf
nios_peripherals_reference_manua.pdfl
nios_programmers_reference_32.pdf
nios_software_development_reference.pdf
sopc_builder_datasheet.pdf
pinouts.txt