“FREJA”

A programmable board for TFC system components testing

Andrea Borga

ABSTRACT

The *TFC (Timing and Fast Control)* system of the LHCb experiment is getting close to the pre-production phase. The need for a tool able to make detailed and precise tests of all the boards is for that reason essential. The TFC test board (*Freja*) is a general purpose board based on an FPGA capable to produce, receive and process stimuli from and to the TFC boards and return test results via a dedicated user interface. The test principle in based on “predicted event comparison” (what we get is what we expect?).

The fact that Freja is absolutely general purpose makes it much more than a simple prototype tester board. Its flexibility allows it to be used in future application for instance as a powerful tool for production and commissioning testing as well as a monitor board during the whole life time of experiment.

Prepared By: A. Borga, Politecnico di Torino, Torino, Italy

Project supervisor: R. Jacobsson, CERN, Geneva, Switzerland
LHCb Online Group

Academic supervisor: D. Trinchero, Politecnico di Torino, Torino, Italy
Dipartimento di Elettronica

INDEX

Per tutto ciò che è
incrocio perfetto
fra arte e scienza

varrà ancora la pena
alzarsi domani.

1 Introduction

«What are we? Where do we come from? Where are we going? »

Those could be the usual starting questions of a scientific treatise, interrogations that mankind's curiosity always tried to answer, sometimes with good results but often without any sure certainties.

This introduction doesn't aim at the explanation of those tricky queries, it focus instead on giving an answer of a more simple question that can still turnout to be complex due to the hundreds of "faces" that the subject of analysis can show: «What is CERN? And what do people do there? ».

An obvious answer that is related with the previous questions could be: «Well… At CERN people try to find the answers to the above questions! ». Too vague maybe. «It is the place where the WEB was born than! », this is true, but internet is just a small invention comparing to CERN's final goal.

Let's proceed with order then, and try to reply stating facts.

CERN was founded in 1954 just after the end of the Second World War by a group of scientists with the aim to study and reveal the building blocks of matter, the forces that bind nature and, not least, to gather people with a common love and passion for science, innovative technologies and, of course, particle physics.

To discover and study matter physicist use particle colliders: like children try head-on colliding cars to see how much they can resist, scientists "play" with atomic and sub-atomic particles at high energies to see what "comes out".

By experimental experience it was possible to prove the existence of the quarks, much smaller than the bricks of Bohr's atom model which are electrons, protons or neutrons; but studies at CERN also revealed the existence of particles which don't exist in the actual universe… how is this possible? There are no mysteries apparently, the physicists found a way to also "travel back in time" using particles accelerators: even if absent in the present days' nature and space, matter created and studied at very high energy existed in the universe at its early stages when it was no older than a fraction of a second.

The more energy that is put in the collisions the more far back in time physicists can look: the *LHC (Large Hadron Collider)*, CERN's new particle accelerator that will start working in 2007, will operate at an energy of 7TeV. At the energy of LHC, a small-scale reconstruction of the universe at a time of 10^{-10} second after its birth will be analyzed.

Greedy machines, the four detector of the experiments (Atlas, CMS, Alice and LHCb), installed in the 27km under ground ring will observe from different point of view what happens at the collision points.

The Universe started out 13.7 billion years ago as an extremely hot, dense and homogenous soup of energy and particles. The energy was continuously converted into pairs of matter and antimatter. As the matter and antimatter collided they annihilated each other recreating energy. Hence there was a perfect balance between matter and antimatter. After the big bang the universe started its expansion and cooling. These phenomena provoked a series of drastic changes in its composition which led matter to take over antimatter. Nowadays then, we live in a "matter dominated" world, but why did this happen? The *LHCb (Large Hadron Collider Beauty)* experiment **[3]** will try to find the explanation to this preference of nature by operating measurements on particular particles, the beauty quarks.

Like their predecessors, LHCb and all the other experiments require ultra-modern technology to operate in the most efficient way. The Web for example was invented in 1992 at the time of the *LEP (Large Electron Positron collider)* experiment to exchange and share, in a completely new fashion, information and data between the physicists working at CERN. As history tell us, the idea of the project turned out to be so powerful that it was then shared and extended to the whole world.

Engineers and computer scientists work in close contact with physicists to help with their knowledge the development of what can sometimes appear to be "crazy thoughts" or "science fiction ideas" only.

Researchers are working hard trying to make dreams come true, seeking answers to millenary questions. "Maybe there is just the hand of a god behind all that… who knows…", though, what has been discovered up to now need strong confirmations, no body has the arrogance to state that man is capable to create something that never existed. Scientists are sceptic by definition and for that reason will keep patiently looking for reasonable explanations.

And then… we will see…

2 Me and my work at CERN

I came to CERN in June 2003 after three years of Telecommunication engineering at the Politecnico di Torino to start a technical studentship focused on electronics design. My high school background in electronics allowed me to start learning with good basis while the engineering approach, learnt at university, helped on facing problems in a more structured way… but still, more than one year "of CERN" taught me many things about ultra-modern electronics both theoretical and practical. Beside the pure scientific knowledge that I gathered since the day I came, CERN gave me the opportunity to work with international people in an international environment contributing to open my mind toward many cultures and realities completely different from mine.

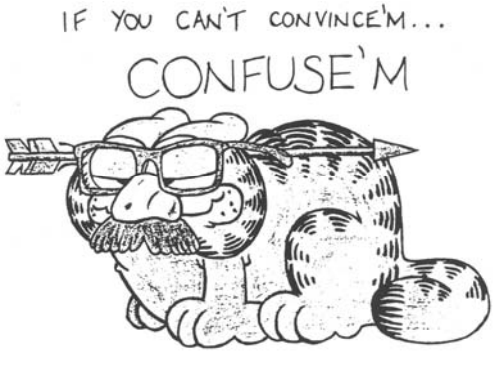
When I first arrived I spent a few weeks taking over the work of Ramy Abdel-Rahman, which drew the first sketches of the schematics of the test board. Together with my supervisor Richard Jacobsson, lots of revisions, modifications and improvements were implemented before the layouting process started. The board has been routed taking into consideration the base lines of 9U VME board standard and the precious suggestions of Zbigniew Guzik who designed all the others TFC boards.

I also gave support and supervision during the whole PCB manufacturing and mounting process to improve the board according to the indications and needs of the workshop personnel.

The development of the firmware and control software has been done in parallel with the board debugging in a bidirectional manner: test software was developed to debug hardware and hardware has been used to debug software. The integration of the board inside the system architecture started from the development of self-test routines to debug the hardware and then moved step by step toward the whole chain testing incrementing the functionalities of which the board is capable.

Once software and hardware were sufficiently developed many prototype and pre-production tests were done in the lab giving me the possibility to spend time measuring and "hacking" on all the TFC boards, building small devices to improve the tests and developing other tiny ideas which led to satisfying results.

Two versions of Freja were produced: a first "final prototype" including two PCB manufactured and one mounted, and "final version", with two PCB both mounted, differing from the first one by little bug fixes and small improvements. Those four boards will provide the necessary functionalities to the TFC system for the whole experiment life.



3 TFC System Overview

3.1 The LHCb detector

The LHCb detector will be installed about 100m underground in the cavern of LHC Pit 8, around one of the collision points of the LHC collider.

The detector is structured as a sequence of sub-detectors organized like a lying pyramid. In total it measures 20m in length and 10m in height. The sub-detectors are based on different technologies and detection principles in order to reconstruct fully the particle reaction; in particular they provide information about the structure of the collision through particle tracking, the particle energies and momenta, and their identities. The information comes out of the detector as electronic signals which are recorded by the data acquisition system.

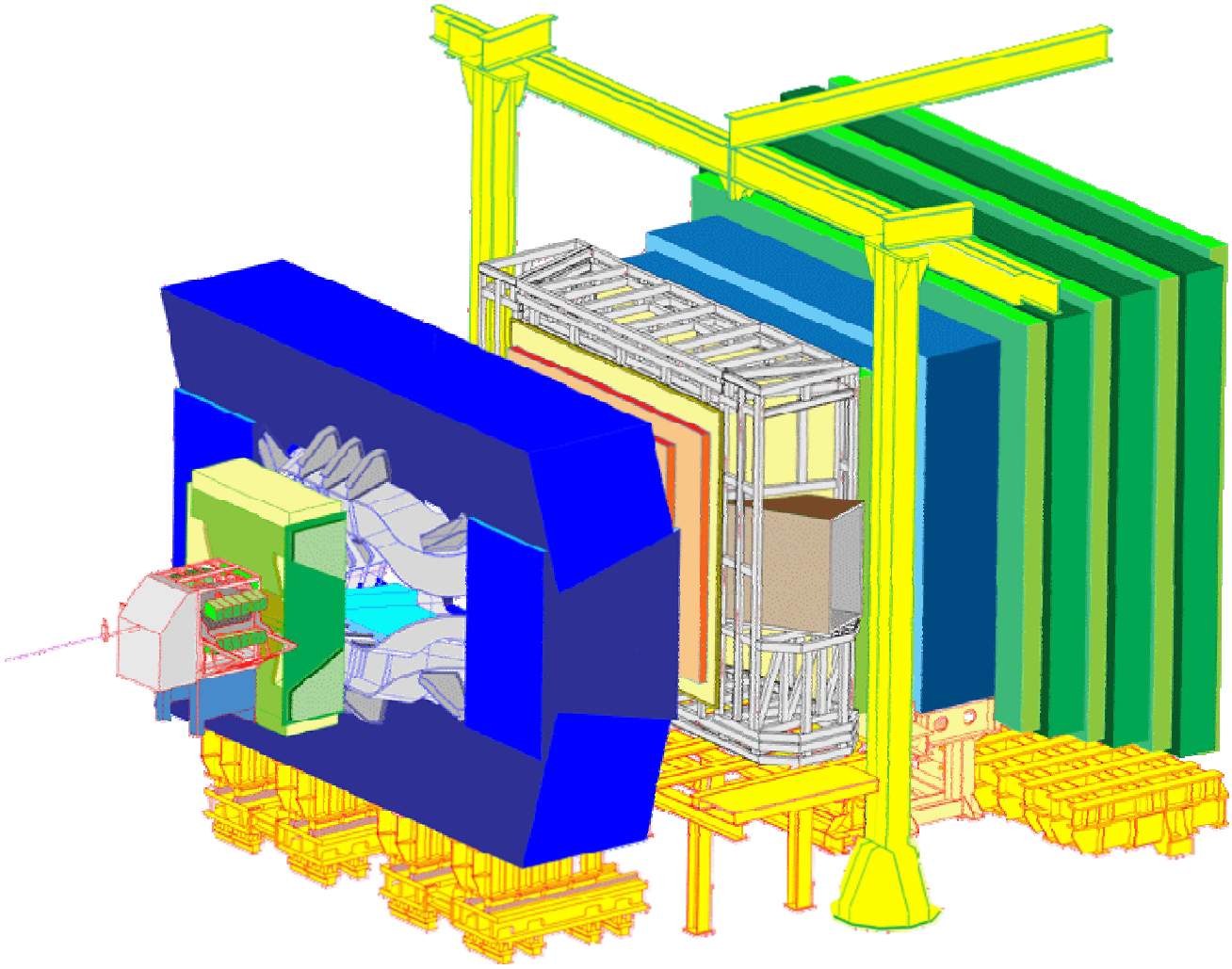


Figure 1: Schematic view of the LHcb detector.

3.2 LHCb readout system

The figure below (**Figure 2**) shows a simplified picture of the entire LHCb readout system architecture **[4]**.

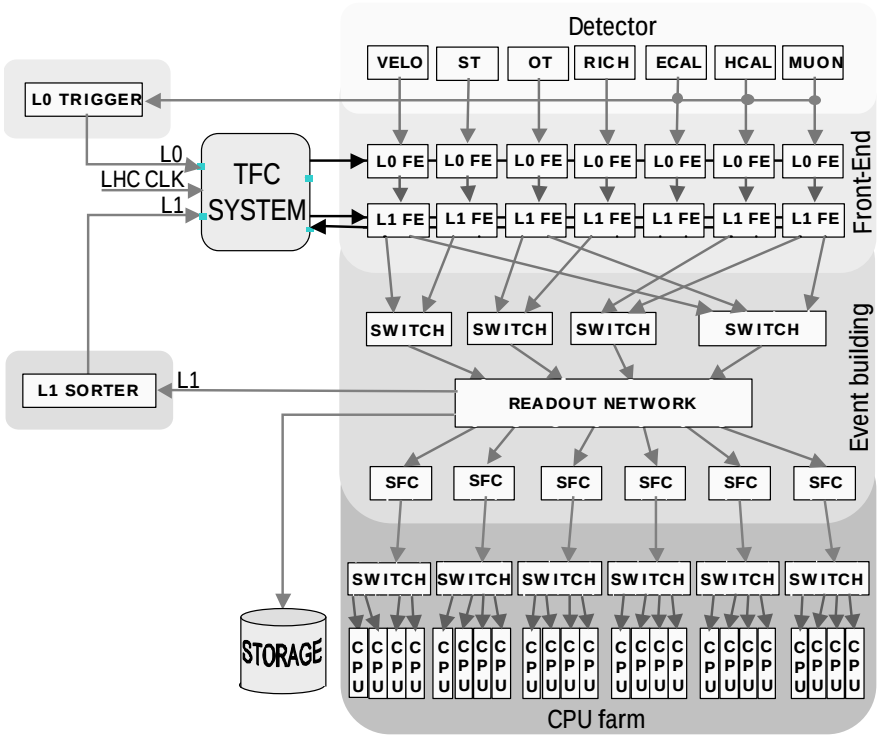


Figure 2: A simplified picture of the LHCb readout system architecture

Collisions taking place inside the detector are read out from the Front End electronics at a rate of 40 MHz. The big amount of data is too heavy to handle entirely and lots of events are not of physics interest; for those reasons the readout system features two levels of high-rate triggers: a Level 0 (L0) trigger that brings down the physics interaction rate of 10 MHz to an event accept rate of maximum 1.1 MHz, and a Level 1 (L1) trigger with an accept rate of maximum 40 kHz. The L0 trigger processing is carried out in a dedicated hardware module whereas the L1 trigger processing takes place in the CPU farm.

The architecture of the Front-End (FE) electronics reflects this two level structure in that it consists of a L0 part and a L1 part. The L0 Front-End (L0 FE) electronics samples the signals from the detector (at a rate of 40 MHz) and stores them during the L0 trigger processing (4 μs). The event data are subsequently de-randomized before being handed over to the L1 FE electronics. The L1 FE has two channels to the event building network, one of which is used to transmit event data to the L1 trigger processing and the other which is used for the complete readout after the L1 trigger decision.

Thus, upon receiving event data from the L0 FE, the L1 FE electronics sends a part of the data over the event building network for the L1 trigger processing and buffers the complete event data during the L1 trigger latency (58 ms). Upon receiving a positive decision the L1 FE de-randomizes the events, zero-suppresses the data, and finally sends the complete event data to the CPU farm for the High Level Trigger processing.

3.3 The TFC System architecture

The *Timing and Fast Control (TFC)* system is responsible for controlling the LHCb readout by distributing timing, trigger and synchronous commands to the LHCb front-end electronics **(Figure 3)**. It is different from the equivalent systems of the other LHC experiments in that it has to support the two levels of high-rate triggers. As the name itself suggests the system is responsible for:

1. **Timing**: the system must provide a clock, with minimum jitter, for means to archive timing alignment of the front-end electronics.
2. **Fast control**: provides trigger control and distribution and data routing. The system must also incorporate functionality to prevent buffer overflows in the entire readout chain, and provide means of different types of auto-triggering for tests and calibration.

In order to simplify the implementation of a partitionable system, the TFC master ship of a configurable ensemble of front-end electronics is centralized in one module: the Readout Supervisor **[5]**.

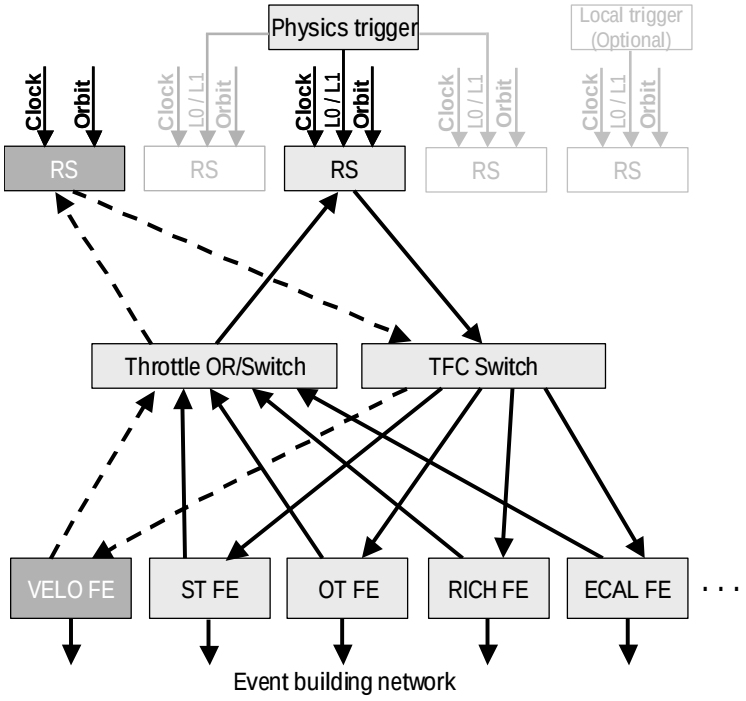


Figure 3: The logical layout of the TFC architecture showing an example of partitioning.

The sub-system VELO FE is driven by the leftmost RS triggered internally.
The other sub-systems are driven by the RS in the centre connected to the LHCb trigger system.
The other RS' are unused.

For separate local runs of sub-systems a programmable patch panel, the TFC Switch, allows associating sub-systems to the different optional Readout Supervisors: they may thus be configured to sustain completely different timing, triggering, and control, and can also be connected to local trigger sources.

The TFC Switch distributes in parallel the information from the Readout Supervisors to the Front-End electronics of the different sub-systems. The information transmitted by the Readout Supervisors to the Front-End electronics via the TFC Switch and the TTC distribution network consists specifically of:

1. The LHC reference clock at ~40 MHz as received from the LHC timing generators. This is the master clock of all the electronics.
2. The two levels of high-rate trigger decisions (L0 and L1).
3. Commands resetting event related counters in the Front-End electronics used to identify the accepted events and to check synchronisation.
4. Commands resetting the Front-End electronics in order to prepare it for data taking or to recover from an error condition.
5. Calibration commands activating specific calibration systems in the Front-End electronics or in the sub-detectors.
6. IP/Ethernet addresses assigning the CPU farm destinations to the L1 trigger data and the full event readout.

If the physics trigger rate gets abnormally high or data congestion occurs in the event building network, there is a potential risk of overflow in the buffers of the Front-End electronics. In order to prevent this, the Readout Supervisor controls the trigger rates according to the status of the buffers. Whereas the status of the fast buffers can only be known by emulating them centrally in the Readout Supervisor, slower buffers are monitored locally. In case they are monitored locally, imminent overflows are signalled via a dedicated throttle network. The Throttle Switch feeds back the buffer overflow warning signals from the slower buffers in the readout to the appropriate Readout Supervisor.

Figure 4 shows a detailed view of the TFC architecture. The Throttle ORs function as concentrators of buffer overflow warning signals from the FE electronics and make a logical OR of the signals within the same sub-system. The TTC modules in **Figure 4** are all standard components of the CERN TTC system (see **Section 3.7**).

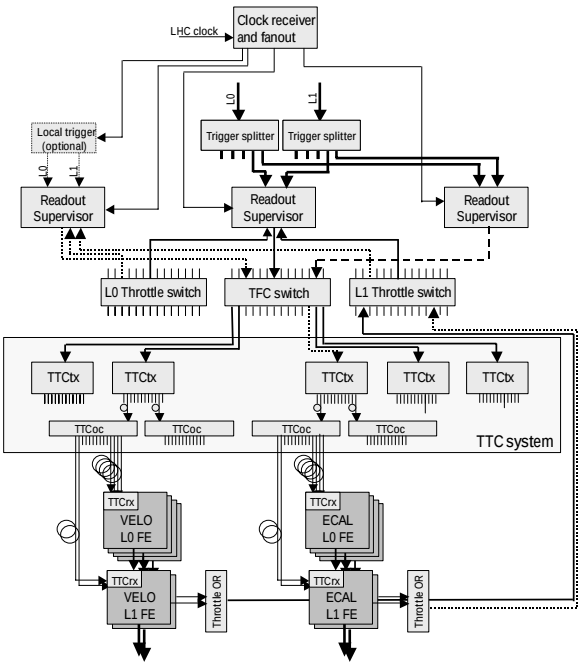


Figure 4: Overview of the TFC architecture.

Summarizing, the TFC controls the readout part of the system accomplishing three main tasks using three main boards:

1. **Control, monitoring and test runs**: control is done by *one* of the Readout Supervisors in the pool and monitoring and test runs can be done by the others.
2. **Partitioning**: implemented via the TFC switch with the aim of easing the accomplishment of different task at the same time (e.g. physics taking and detector debugging), focused monitor testing on detector sub-parts without stopping a run.
3. **Feedback**: done by the Throttle Switch in order to give back-pressure to maintain the readout speed below the throughput limit of the Data AcQuisition system (DAQ).

In the next paragraphs a more detailed description of each sub-part is given.

3.4 Readout Supervisor: "ODIN"

The Readout Supervisor is a complex board responsible for a multitude of functions (**Figure 5**).

The speed requirements and the multifunctionality of the readout supervisor necessitate optimal technological solutions. At the same time the logic must be modifiable to support extensions or changes in the running modes.

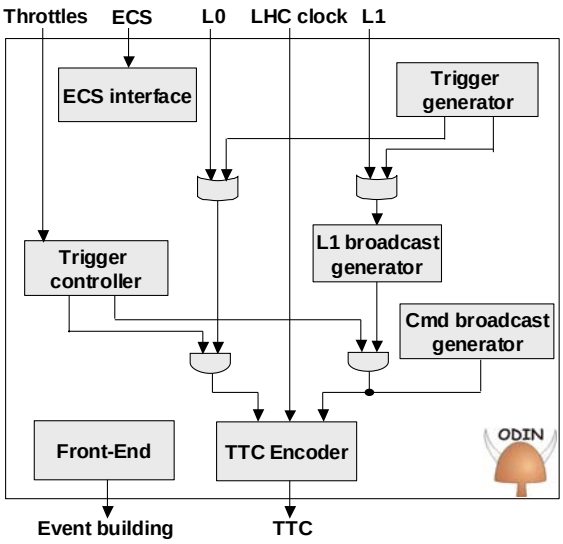


Figure 5: Simplified logical diagram of the Readout Supervisor showing the basic functions.

The TTC encoder circuit incorporated in each Readout Supervisor receives directly the LHC clock and the LHC orbit signal via a TTC machine interface (TTCmi). The clock is distributed on the board in a star fashion and is transmitted to all synchronous destinations via the TTC system.

The Readout Supervisor receives the L0 trigger decision from the central L0 trigger Decision Unit (L0DU), or from an optional local trigger unit, together with the Bunch Crossing ID. In order to adjust the global latency of the entire L0 trigger path to a total of 160 cycles (4 μs), the Readout Supervisor has a pipeline of programmable length at the input of the L0 trigger. Provided no other changes are made to the system, the depth of the pipeline is set once and for all during the commissioning with the first timing alignment.

The Bunch Crossing ID received from the L0DU is compared to the expected value from an internal counter in order to verify that the L0DU is synchronized.

For each L0 trigger accept, the source of the trigger (3-bit encoded) together with a 2-bit Bunch Crossing ID, a 12-bit L0 Event ID (number of L0 triggers accepted), and a "force bit" is stored in a FIFO. The force bit indicates that the trigger has been forced and that consequently the L1 trigger decision should be made positive, irrespective of the L1 physics trigger. The information in the FIFO is read out at the arrival of the corresponding L1 trigger.

The RS receives the L1 trigger decision with a 2-bit Bunch Crossing ID and a 12-bit L0 Event ID. If the force bit is set the decision is converted to positive. The 3-bit trigger type and 2 bits of the L0 Event ID are subsequently transmitted as a short broadcast according to the format in **Table 1**.

The Readout Supervisor controls the trigger rates according to the status of the buffers in the system in order to prevent overflows. Due to the distance and the high trigger rate, the L0 FE buffer occupancy cannot be controlled in a direct way. However, as the buffer activity is completely deterministic, the RS has a state machine to emulate the occupancy. This is also the case for the L1 FE buffers. In case an overflow is imminent the RS throttles the trigger, which in reality is achieved by converting trigger accepts into rejects. The slower buffers and the event-building components feed back throttle signals via the dedicated throttle network to the RS (**Figure 4**). Data congestion at the level of the High Level Trigger farm is signalled via the Experiment Control System (ECS) to the onboard ECS interface, which can also throttle the triggers. For monitoring and debugging, the RS has history buffers that log all changes on the throttle lines.

The RS provides several means for auto-triggering. It incorporates two independent uniform pseudo-random generators of L0 and L1 triggers according. The RS also has a unit running several state machines synchronized to the LHC orbit signal for periodic triggering of a single or a specified number of consecutive bunch crossings (timing alignment), triggering at a programmable time after sending a command to fire a calibration pulse, triggering at a given time on command via the ECS interface etc. The source of the trigger is encoded in the 3-bit L1 trigger qualifier.

The RS has also the task of transmitting various reset commands. For this purpose it has a unit running several state machine, also synchronized to the orbit signal, for transmitting Bunch Counter Resets, Event Counter Resets, L0 FE electronics reset, L1 + L0 FE electronics reset, L1 Event ID resets etc. The RS can be programmed to send the commands regularly or solely on command via the ECS interface.

The RS transmits the IP/Ethernet destination for the L1 event data and for the complete readout as long broadcasts.

The transmission of the various broadcasts is handled according to a priority scheme. The Bunch Counter and the Event Counter Reset have highest priority. Any clashing broadcast is postponed until the first broadcast is ready (L1 trigger broadcast, IP/Ethernet destination) or until the next LHC orbit (reset, calibration pulse, and all miscellaneous commands).

The RS keeps a large set of counters that record its performance and the performance of the experiment (dead-time etc.). In order to get a consistent picture of the status of the system, all counters are sampled simultaneously in temporary buffers waiting to be read out via the onboard ECS interface. The RS also incorporates a series of buffers analogous to a normal Front-End chain to record local event information and provide the Data AcQuisition (DAQ) system with the data on an event-by-event basis. The "RS data block" contains the "true" bunch crossing ID and the Event Number, and is merged with the other event data fragments during the event building.

3.5 TFC Switch: "THOR"

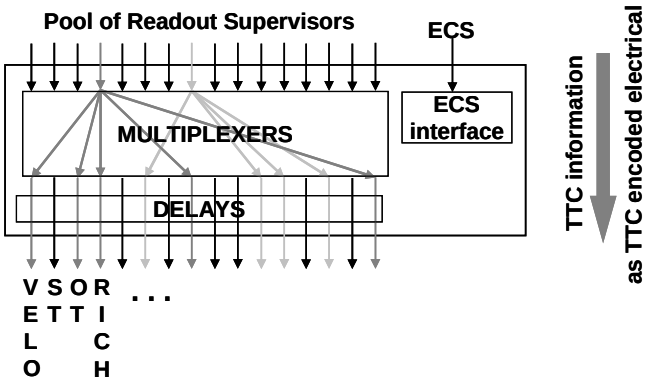


Figure 6: Diagram of the TFC switch.

As mentioned before, the TFC Switch (**Figure 6**) realises the partitioning of the TFC system. It is a programmable patch panel (not a switch in the network sense) that allows distribution of the synchronous information to the different parts of the Front-End electronics.

From the architecture of the TFC system, it follows that the Front-End electronics that is fed by the same output of the TFC Switch is receiving the same timing, trigger and control information. The connectivity provided by the board is not necessarily one-to-one: the TFC Switch should allow setting up several partitions, by associating a number of partition elements (e.g. sub-detectors), to several Readout Supervisors in order to accomplish different tasks. For example while the main RS is controlling the detectors for data taking, the optional Readout Supervisors have the possibility of running separately on other detector parts for test and debugging purposes.

The TFC Switch has been designed as a 16x16 switch and thus allows the LHCb detector to be divided in 16 "atomic" sub-systems. To increase the partition granularity an option exists whereby four TFC Switches are deployed in order to divide the LHCb detector into 32 sub-systems. The TFC switch configuration setup is done remotely by software via the control system interface.

3.6 Throttle Switch and Throttle OR: "MUNIN" and "HUGIN"

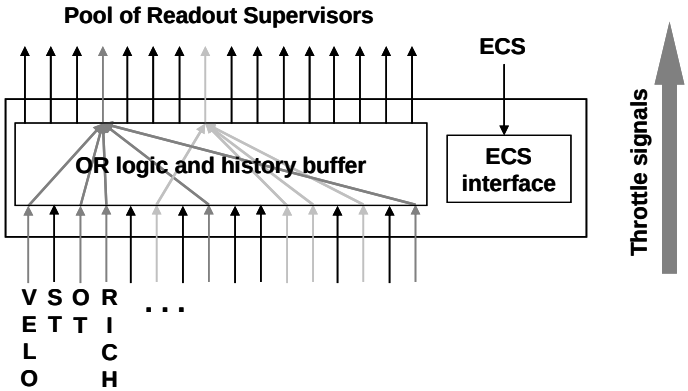


Figure 7: Diagram of the Throttle Switch.

Opposite to the data flow provided through the TFC switch, a Throttle Switch (**Figure 7**) has been designed with the aim of providing backward paths of throttle signals (in case of imminent buffer overflows) from the end buffers in the L1 front end electronics to the appropriate Readout Supervisor. The functionality of this module is specular to the one used in the TFC switch: several input signals are OR'ed to produce a single output signal.

The module has both 16 electrical and optical inputs and 16 electrical outputs. Besides providing the ORing and the routing of the throttle signals the Throttle Switch also trace the behaviour of all input and output signals with a good time resolution.

In addition to the Throttle Switch, a Throttle OR will be designed to group throttle lines belonging to the same partition elements. It is identical to the Throttle Switch in all aspects except that it ORs 20 L0 and 20 L1 throttle inputs and transmits then on a L0 and a L1 output.

Those components too are software configurable via the control system interface.

3.7 TTC Distribution System

The TFC distribution network is based on the RD12 *Trigger, Timing and Control (TTC)* system: a CERN standard optical network used by all four LHC experiment **[7]**. The TTC system distributes information optically on two serial channels:

- *Channel A*: transmits the LHCb L0 trigger decisions to the Front-End electronics in the form of an accept/reject signal at 40 MHz.
- *Channel B*: transmits framed and formatted broadcasts, including Hamming code. Two types of broadcasts are available: 16 bit frames which have 8 bits of user information, so called short broadcasts (**Table 1**), and 42 bit frames which have 16 bits of user information (8 bit data/8 bit address), so called long broadcasts (**Table 2**). It is used for several functions like: transmission of the commands to reset the Bunch Counters (BCR) and the Event Counters (ECR) in the Front-End electronics and the trigger systems; transmission of the L1 trigger decision and of the Front-End control commands (e.g. calibration pulse triggering).

Table 1: Summary of the short broadcasts in LHCb.

	7	6	5	4	3	2	1	0
L1 trigger	1	Trigger type			L0 EvID		0	0
Reset	0	1	R	L1 EvID	L1 FE	L0 FE	ECR	BCR
Calibration	0	0	0	1	Pulse type		0	0
Command	0	0	1	Command type			0	0

Table 2: Summary of the long broadcasts for the IP destination assignments in LHCb.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L1 IP destination	1	0	0	Flush	R	R	Ethernet/IP address									
HLT IP destination	1	0	1	Flush	R	R	Ethernet/IP address									

The two channels are time division multiplexed (TDM) and bi-phase mark encoded before being converted to an optical signal. The bi-phase signal also allows transmitting the clock with low jitter. The encoding is done in the Readout Supervisor with the *TTC Readout Supervisor (TTCrs)* module and the electrical to optical conversion is done by the *TTC Transmitter (TTCtx)* modules, which have 14 high-power transmitters. The optical fan-out *TTC Optical Couplers (TTCoc)* allows distributing the signal to 32 destinations, which means that one *TTCtx* can drive up to 448 destinations. The *TTC Receiver (TTCrx)* ASIC reconstructs the 40 MHz clock and converts the encoded signal into the user information. The TTCrx also provides means to adjust the timing of the TTC information in order to time-align all Front-End boards. A version of the TTCrx, mounted on a mezzanine *(TTCrm)*, is also available if an easier access to the chip is needed, like for example, for debugging purposes.

Another TTC module, the *TTC machine interface (TTCmi)*, interfaces the local experiment TTC system with the accelerator timing system. It's responsible of providing the LHC clock and the LHC orbit signal.

4 Testing methodologies

4.1 Introduction to hardware testing

Although standard testing methodologies were initially developed for software debugging, the definitions described below can fit perfectly hardware testing.

Two main categories of test setups are widely used:

1. **Black box testing**: functional testing.

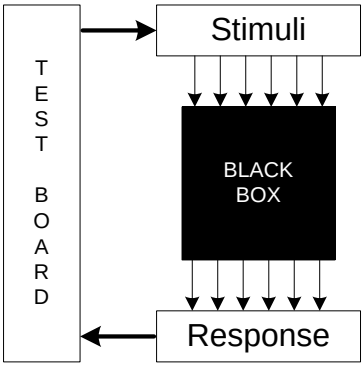


Figure 8: Black box configuration.

Black box focuses on purely functional testing. Test routines are written knowing only the system requirements, the inputs and the outputs (the box is "closed"). The internal structure and implementation are therefore ignored by the tester. To give a better hardware view, these kinds of test are called **"in the crate"**.

2. **Glass (white) box testing**: focused testing.

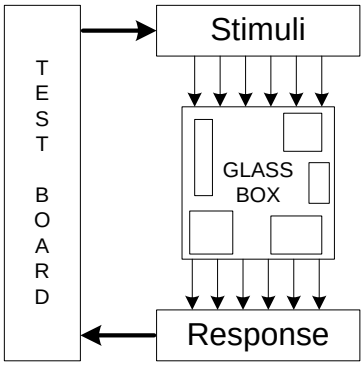


Figure 9: Glass box configuration.

Glass box testing applies a more direct strategy by focusing on testing the implementation. The internal structure and composition is accessible (the box is "transparent"). Knowing it, the tester can try to find out which is the best testing approach. Information about the system can be also used to test particular features or to obtain specific results. Again for a good hardware analogy these tests are called "**on the table"**.

White box testing can be split in other subcategories:

a. *Static:* the system is not running. This is the case for onboard measurements to detect for examples bad contacts between pins and pads, transmission lines malfunctioning, etc.
b. *Dynamic:* this is "common" testing. It involves probing the system while it's running.

Testing rarely only involves one of the methods: a complete verification is achieved only by a good combination of both.

4.2 Testing theory

In a mathematical form testing can be expressed in the following way:

$$R = f(S,C)$$

A *Response (R)* of a system is the result of a *function (f)* fed with a *Stimuli (S)* and *Configurations (C)* selected by the user inside a multi-dimensional space. **Figure 10** shows a response of the system in a test space with one stimulus and one configuration:

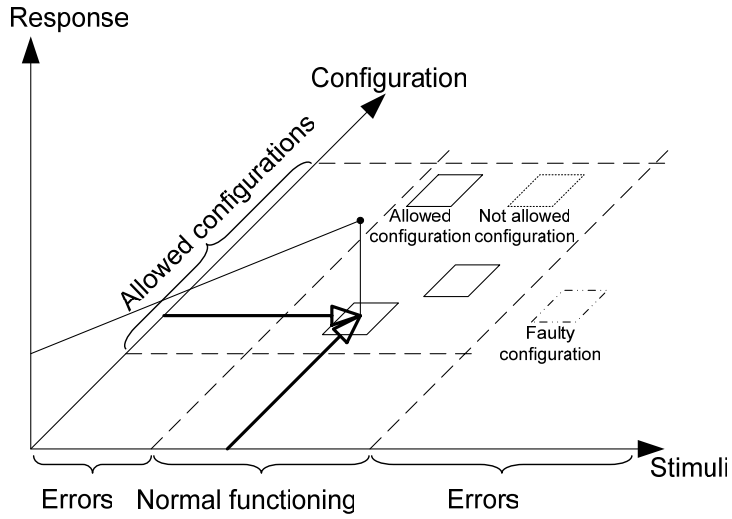


Figure 10: Three dimensional plot of the system test space.

The stimuli are selected according to the type of test that is performed. The test can either be constructive or destructive: stimuli can be within the range of correct values to test normal running behaviour or the system can be exposed to faulty stimuli to test error recovery.

A stimulus can be either purposive (periodic) or random to ensure coupling and decoupling of the system from its previous stimuli.

The configuration axis is scanned in discrete steps: it is in fact often useless or not possible to setup the board in certain combination of configurations.

According to the allowed configurations and the chosen range of stimuli, which are often selected according to the configuration in use, it is possible to define existence regions for the testing function which are represented by the rectangles on the above plot.

It is very important to define the test (Stimuli and Configurations) carefully to optimize the test procedure, not to end up testing all combination.

4.3 TFC test setup

With **Section 4.2** in mind the aim has been to implement a test bench which has a unit to produce stimuli and receive responses. In order to make verification the unit emulates functionalities of the system in test. A control system configures and sets up the test procedures.

The test unit, called Freja, of the TFC test bench is the subject of this report. All its functionalities and a detail description are following in the next chapters.

Two kind of test are made for the TFC:

1. Single board test.

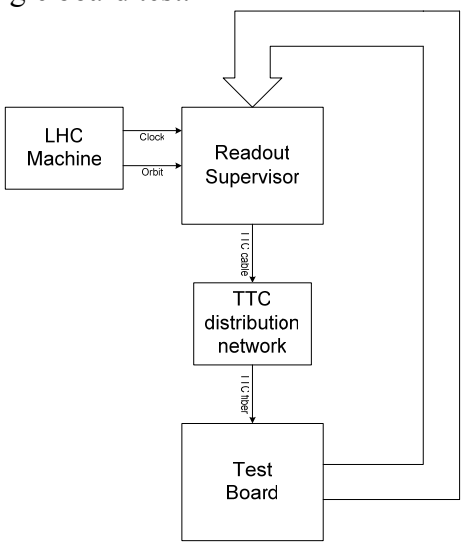


Figure 11: Single board test configuration.

Since the Readout Supervisor is the most complex board of the TFC system it requires focused tests. Referring to the device description made in **Section 3.4** the test will concentrate on debugging the following features:

- L0 trigger path
- L1 trigger path
- Buffer level control
- IP destination broadcasting

2. Full TFC system testing.

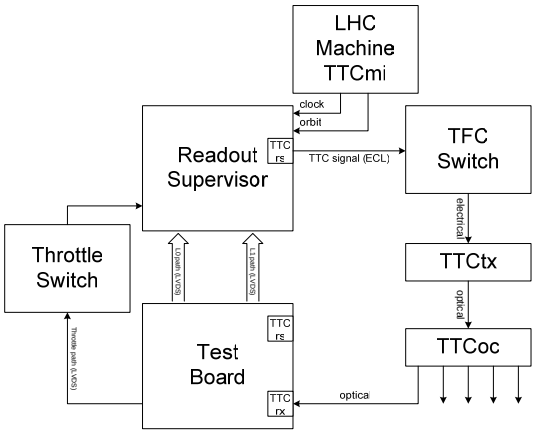


Figure 12: Full TFC system test configuration.

The test board emulates both the L0 and L1 decision units (L0 trigger and L1 trigger in **Figure 2**) sending decisions to the Readout Supervisor; the decisions are then transmitted to the Front-end electronics (L0 and L1 FE emulated by the test board) via the whole TFC chain: TFC switch and TTC distribution network. The test board can speed up the decision sending in order to overflow the buffers in the Readout Supervisor or in the Front End: this technique is used to check the throttle backward path and the buffer emulation (via the throttle switch).

The aim of Freja is to perform black box testing of all the functionality of the boards in the setups listed above. As mentioned before black box testing and glass box testing are complementary and black box testing will help the testers to discover bugs which will be subsequently debugged using glass box testing.

For example: a test aim is to check if a data bus is clean, complete (no bits missing), well driven and arbitrated and pass all the way trough its designed path. A black box test done with the test board will consist in feeding data pattern and driving the control bus lines of the target board via a known bus input and receive them back from a known bus output. Afterwards the board will compute the received signal (checks the data integrity, delays, etc) and decide if it's correct or not. If the result is correct the test is over and successful. If not one must switch to glass box tests: measurements can start from the path check to make sure that signals are propagated properly all the way down, if the path is interrupted somewhere the test focus then on the chip or line that causes interruptions, if the signals shapes are ruined but none on the chip driving has an odd behaviour, the test can move to neighbouring parts of the board. A careful combination between the two tests and a bit of method and intuition will always drive a tester to a solution.

4.4 Application of Freja

Freja is a general purpose board which allows it to be used in the TFC system in several applications for a very long time. Many different purposes were conceived and studied already in the early stages of the conception. Here after they are discuss in more details.

4.4.1 Prototype testing

Prototyping phase is the most critical part of all. The complete set of features of the TFC has to be tested extremely carefully in order to lunch the production of a 100% working product. Prototype testing involves tests on parts that will not be changed (except in case of future development and improvements), like the hardware. Another point to concentrate on during prototype testing is to find a good set of "well known" problems that will ease life on production testing: for example if a particular component follows a difficult mounting technique (like dense, impedance matched connectors, fine pitch ICs, etc) and problems are discovered during prototype testing, it will be important to check carefully it on all the produced boards. Error recovery testing is fundamental as well in this phase: it's important to know how the system reacts and recover from certain error conditions, how long it takes and if afterwards it can be considered stable again.

4.4.2 Pre-production and production testing

This phase makes use of all the experience gathered during prototype testing. Test bed contains a full set of testing and debugging tools (see following chapters) in order to speed the procedures.

4.4.3 Commissioning

"You have a board perfectly working on your lab table… you bring it down to the pit, plug it into the crate… and suddenly everything goes wrong!" this is a nightmare that very often scares technicians, engineers and physicists and the reason why a test board can be very useful to test board-to-board functioning and compatibility during commissioning. In situ architecture and installation tests aim at solving these kinds of problems: Freja will perform the standard test routines, developed in the lab environment, on the TFC boards in the barracks of Pit 8 (LHCb installation site). Since the TFC installation will take place long before the installation of the detectors with their electronics, Freja will allow the TFC system to be tested without the final front-ends.

4.4.4 Experiment monitoring

Freja will be an independent implementation of a Front End electronics to perform system tests together with idle Readout Supervisors. The test board will also perform simple TTC system monitoring, tuning alignment between detectors and cross checking of the connectivity of the entire TFC system.

5 TFC test board: "FREJA"

5.1 Overview of the board

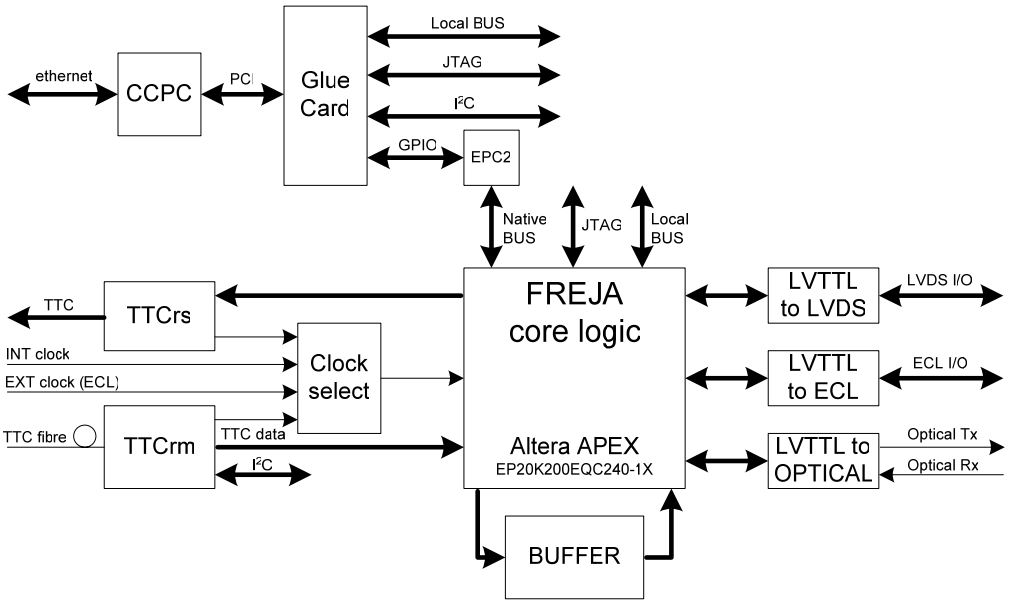


Figure 13: A block diagram of the TFC test board.

The board consists of seven main blocks with different functions and using different electronics:

1. **Control interface [Section 5.2 and Section 5.3]:** a Credit Card PC is accessible via remote connection in an Ethernet network. From the CCPC the hardware is accessible via a 33MHz PCI bus. An interface card (Glue card) translates the PCI bus to the proper bus formats to reach specific devices on board (such as FPGAs, memories, etc). Four types of conversions are made by the Glue card:
   a. *JTAG:* dedicated lines for the FPGA programming and boundary scan of devices.
   b. *I^2C:* standardized serial bus used to configure small devices (e.g.: serial ID E^2PROM, I^2C port, TTCrx).
   c. *LBUS:* the PLX Local Bus is a 32-bit bus used to configure, control and monitor the functions implemented in the FPGA.
   d. *GPIO:* general purpose input/output lines. Used in Freja to drive JTAG lines to program the Altera configuration device (EEPROM) for the FPGA.
2. **External general purpose electric I/O [Section 5.4]:** to interface the test board with the rest of the system in order to make tests, it's important to have as many input/output ports as possible. Two different electrical transmission techniques are used to transmit data over the system: single ended ECL (Emitter Coupled Logic) and differential LVDS (Low Voltage Differential Signalling).
3. **Optical transmitter and receiver:** the optical I/O interface of Freja is used in a standalone mode to test the optical path of the throttle network (Munin) and to transmit throttle signals when Freja acts like a front end board.
4. **FPGA [Section 5.5]:** together with an external FIFO buffer this is the core part of the board. A programmable logic device with 168 IO ports is used to perform all functions of Freja, drive and receive test signals, perform calculation and checks on them and make results available to the user interface. A device configuration EEPROM is implemented to speed up the programming process of the device at start-up.

5. **Clock sources:** in order to run the board in different modes a clock selection multiplexer, accessible via I^2C, has been implemented giving the possibility to select between:
   a. *Internal clock:* 40MHz quartz generated clock.
   b. *TTCrm clock:* clock generated by the TTC system (experiment official clock).
   c. *TTCrs clock:* internal or external clock coming from the TTCrs mezzanine.
   d. *External clock:* allows the board to run with an external clock source.
6. **TTCrs:** TTC encoder and transmitter.
7. **TTCrm:** optical TTC system receiver.

5.2 Credit Card PC

5.2.1 The ECS interface and the CCPC

The electronics boards in LHCb will all be controlled from the LHCb *Experiment Control System (ECS)*. In order to access the actual board resources, an ECS interface will be located on each board. The ECS interface is connected to the control network and performs directly all programming, configuration, control and monitoring of each electronics board.

Three ECS interfaces have been proposed for different applications: the *SPECS (Serial Protocol for the Experiment Control System)* and the *ELMB (Embedded Local Monitor Board)* will be used for electronics boards situated in the radiation area close to the experiment; for the electronics situated in the counting rooms behind the shielding wall, the ECS interface is based on a commercial small (credit-card size) embedded PCs used to provide the necessary local intelligence on electronic boards **[12]**. They are connected to the central ECS via a conventional Ethernet.

The core of the *Credit-Card PC (CCPC)* is a SM520PC Smart Module produced by Digital-Logic, Inc **[13]**. This module comprises a PC-on-a-chip also known as micro-controller, the Elan520 from AMD, an Ethernet interface and a Flash RAM. It can run any standard PC operating system and runs Linux in LHCb. The CCPC is equipped with a 33 MHz PCI bus which is used for all access to the board logic.

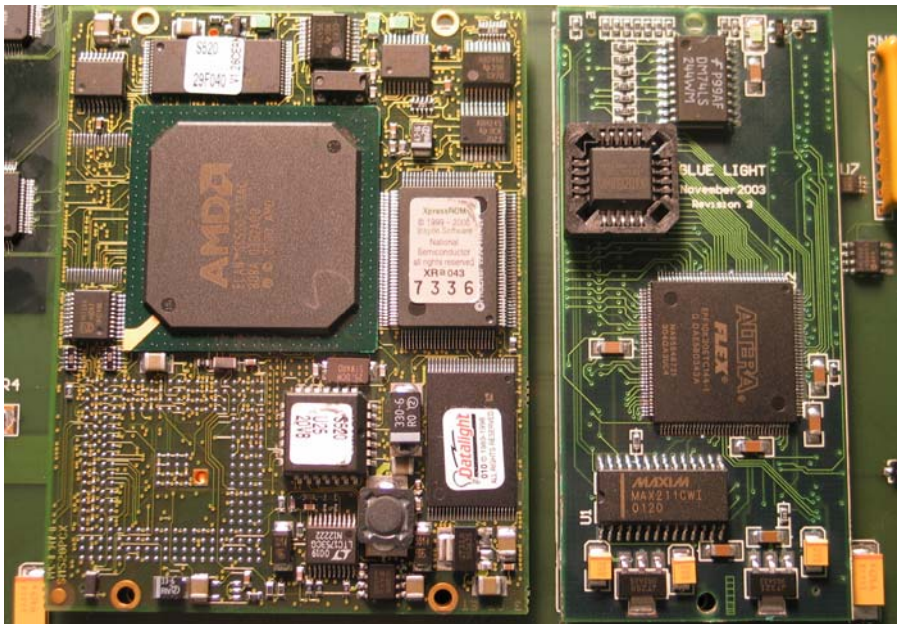


Figure 14: Picture of the "Glue light" board (right) together with the Credit Card PC (left).

5.2.2 PCI Interface

Currently by far the most popular local I/O bus, the *Peripheral Component Interconnect (PCI)* bus, was developed by Intel and introduced with the version 1.0 in 1992. Since then several updates (revision 2.0, revision 2.1, version 2.2 **[17]** and PCI-X **[18]**) have been made to improve not only the bus speed but also the overall performance (bus arbitration, transmission, etc). It was geared specifically to fifth- and sixth-generation systems (e.g. Intel Pentium and Pentium II), although the latest generation 486 motherboards were already using it as well. Like its predecessor, the *VESA (Video Electronics Standards Association)* Local Bus, PCI 1.0 was a 32-bit bus running at a maximum frequency of 33 MHz. Since then its wide usage in electronic devices led this technology to nowadays' extreme edges.

The PCI bus is used to interface devices requiring fast access to each other and/or system memory and that can be accessed by the processor at a speed close to its internal bus. A key feature of PCI is the capability to transmit data in bursts (a single address phase is followed by two or more data phases) whose length is set by the master. The target device receives information about the transaction type and the start signal, but not the transfer length: while the master makes a burst of data transactions, it signals the target device when the last word transfer occurs.

The typical PCI device includes a complete peripheral adapter encapsulated within an IC package or implemented on an expansion card: the most famous interface device is the PLX 9030 whose functions have inspired the development of the firmware for the Glue card (see **Section 5.3**).

Reflected-wave switching

At the time when bus speeds were low (below 1 MHz) the characteristics of transmission lines were neglected by designers and the choice of the power of the driving device was selected according to the electric characteristics (in particular the input capacitance) of the devices connected to the lines. Using buses with clock frequencies higher than 25 MHz, traces acts as transmission lines and the electrical characteristics of the trace must also be taken into account solving the equation that defines the characteristics of the output driver: an impedance, varying between 50 and 110 Ohm, is presented to the driver trying to drive a voltage change onto the line and also imposes a time delay in the transmission of the voltage change along the trace.

An old method used to eliminate the annoying effect of the line impedance is called *Incident-Wave Switching*. It involves the use of strong output drivers with the capability of switching close to the driving point of the device (point of incidence). The main disadvantage of this method is that connecting several devices on one trace increase immediately the driving current required. This limits the packing of drivers in chips and increases the heat on board. Incident-wave switching also requires termination on the lines.

Since this technique turned out to be inefficient, another was introduced for PCI called *Reflected-Wave Switching*: no termination is required and the reflected wave is used as an advantage. A carefully selected, relatively weak output driver is used to drive the signal line partially towards the desired logic state. The driver only has to drive the signal line partially towards its final state, rather than completely (as a strong incident-wave would). No inputs along the trace will sample the signal until the next rising-edge of the clock.

When the wavefront arrives at the unterminated end of the bus, it is reflected back and doubled. Upon passing each device input again during the wavefront's return trip down the trace, a valid logic level registers at the input on each device. The signal is not sampled, however, until the next rising-edge of the PCI clock. Finally the wave front is absorbed by the low-impedance within the driver. This method cuts driver size and lower by half the current needed using incident-wave witching technique.

Diode terminations are integrated inside devices to limit reflections and correct the propagation

delay. Even though Reflected-wave switching has improved a lot the signal characteristic in PCI, it is always better to swap to more versatile bus techniques as soon as possible. With the last statement in mind the Glue card has been developed.

Bus signals

The PCI interface requires a minimum of 47 pins for a one-target device to handle data, addressing, interface control, and system functions. Two additional pins, routed directly to a bus arbiter, are required in the case of multi-master configuration for each device that intends to act as a master on the bus.

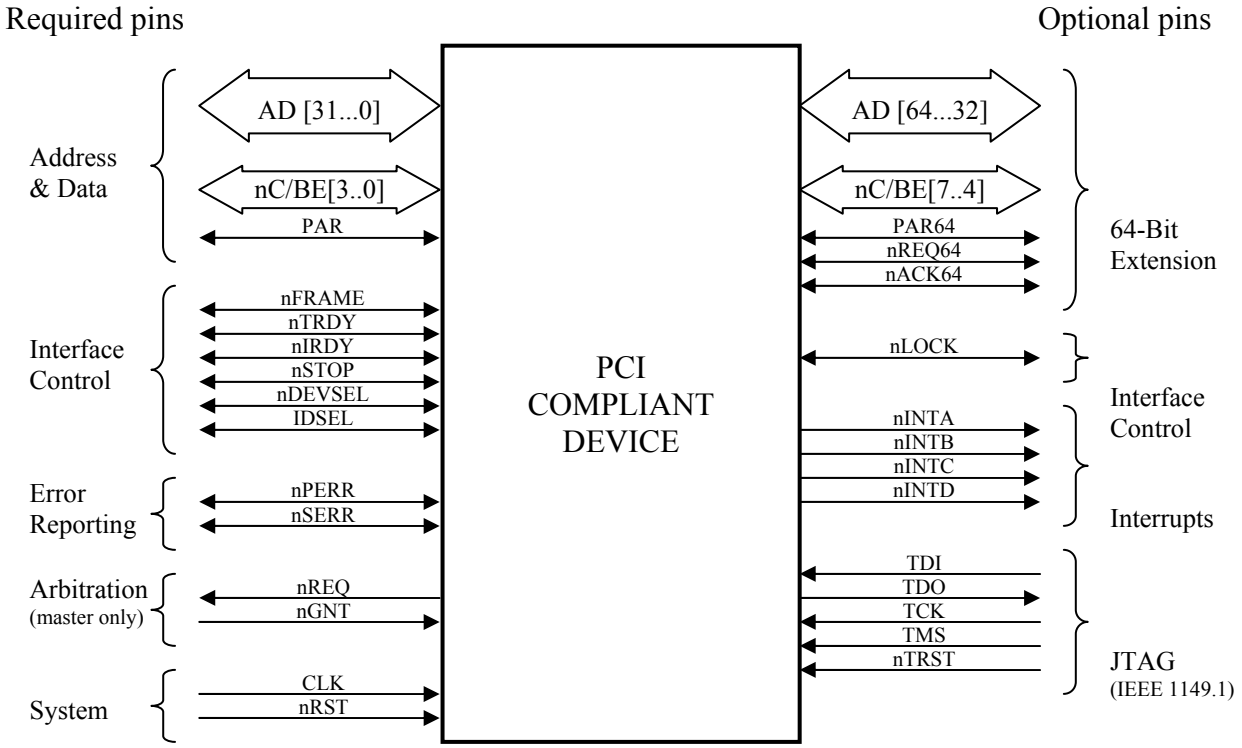


Figure 15: PCI bus signal organization.

As shown in the figure above PCI pins are organized in different groups. Below a more detailed description of the most important pin's function:

1. **System pins:**
   a) *CLK:* provides timing for all transaction on PCI and is an input of every PCI device. The CCPC can operate up to 33 MHz or 66 MHz (according the specification of rev 2.2); though it must be kept in mind that newer specification, like PCI-X foresee operations of devices at a frequency up to 133 MHz.
   b) *nRST:* PCI's reset line.
2. **Address and Data pins:**
   a) *AD [31...0]:* Address and Data are multiplexed on the same PCI pins. A bus transaction consists of an address phase followed by one or more data phases. As mentioned before PCI supports both read and write bursts.

   b) *nC/BE [3...0]:* Bus Command and Byte Enables are multiplexed on the same PCI

pins. During the address phase the pins are used to define Bus Commands (the type of transaction the bus is requesting). In the data phase they are used as Byte Enables (determine which bytes are valid in the bit stream on the bus).

c) *PAR:* Parity is even parity across AD [31...0] and C/BE [3...0]. Parity generation is required by all PCI devices.

3. **Interface Control pins:**
   a) *nFRAME:* Cycle Frame is driven by the current master to indicate the beginning and duration of an access.
   b) *nIRDY:* Initiator Ready indicates the initiating devices' ability to complete the current data phase of the transaction. During write mode it indicates that valid data is present on the address bus. During read mode it indicates that the master is prepared to accept data.
   c) *nTRDY:* Target Ready indicates the target devices' ability to complete the current data phase of the transaction. During a write operation it indicates that the target is prepared to accept data. During a read operation it indicates that valid data is present on the bus.
   d) *nSTOP:* Stop indicates that the current target is requesting the master to stop the current transaction.
   e) *nLOCK:* Lock indicates an atomic operation to a bridge that may require multiple transactions to complete.
   f) *IDSEL:* Initialization Device Select is used as a chip select during configuration read and write transaction.
   g) *nDEVSEL:* Device Select, when actively driven, indicates that the driving device has decoded its address as the target of the current access. As an input it indicates whether any device on the bus has been selected.
4. **Arbitration pins:**
   a) *nREQ:* Request indicates to the arbiter that this device desires use of the bus. This is a point-to-point signal.
   b) *nGNT:* Grant indicates to the device that access to the bus has been granted. This is a point-to-point signal too.
5. **Error Reporting pins:**
   a) *nPERR:* Parity Error is only for the reporting of data parity errors during all PCI transaction. It is mandatory to use this control signal because the devices driving the bus must assume that the receiver will check the correctness of the information and flags back in case of errors. It can be neglected in case of Special Cycles (a broadcast message to one ore more PCI devices).
   b) *nSERR:* System Error is for reporting address parity errors, data parity errors during a Special Cycle and critical errors (usually even catastrophic) other than parity.
6. **Optional pins:**
   a) **64-Bit Extension:** PCI bus can be extended up to 64-bits. This requires more control logic of course. The PCI extension is not analyzed in detail because for TFC purposes the 32-bit wide bus is sufficient.
   b) **Interrupt pins:** 4 interrupt lines are reserved on the PCI bus to allow devices to request attention from its device driver.

5.3 Glue Card

Since PCI is a complex and difficult bus to handle (require special attention when routing due to tight timings and many control line) a small device is used to convert the PCI bus after a short distance on board to more simple ones; the conversion unit is the Glue Card (the name comes from the fact that glue logic consists of a simple circuit used to interface different devices, like the CCPC to the board logic) implemented in a "*light*" version by TFC system **[19]**. The mezzanine is entirely based on a single FPGA which is interfaced with the PCI bus of the CCPC and emulates a PLX 9030 (an ASIC chip that produces a simple PLX local bus) **[14]**.

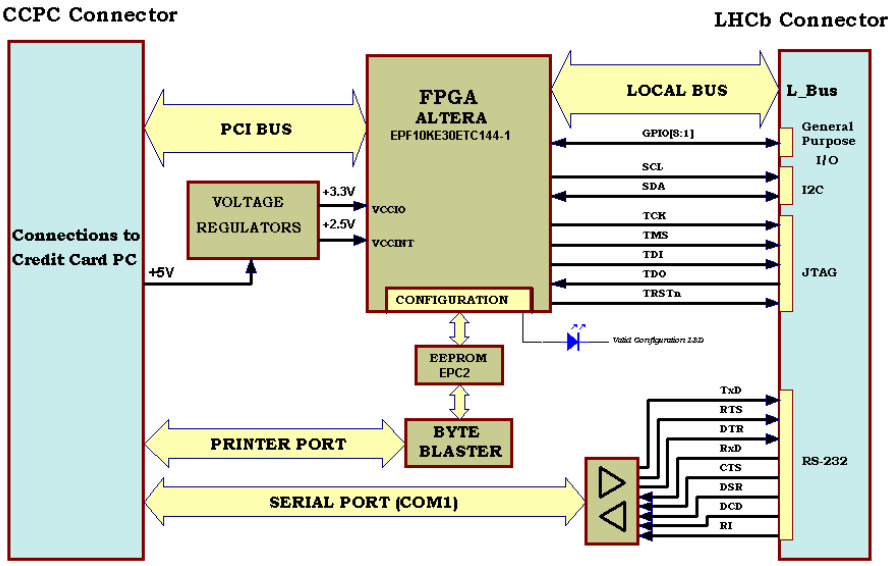


Figure 16: Block diagram of the Glue card.

As mentioned before the Glue Card translates the PCI bus into different bus protocols (JTAG, I^2C and Local Bus) allowing access to many different types of devices. Next each bus type is described in more details.

5.3.1 PCI slave

The PCI interface implemented in the FPGA of the Glue Card is compliant with the PCI Local Bus Specification Revision 2.2 and emulates a PLX 9030. The target supports a 33 MHz and 32-bit PCI bus.

Table 3: Implemented PCI bus commands.

CBEN [3 ... 0]	Bus Command Cycle	Note
0110	Memory Read	fully implemented
0111	Memory Write	fully implemented
1010	Configuration Read	fully implemented
1011	Configuration Write	fully implemented
1100	Memory Read Multiple	substituted by Memory Read
1110	Memory Read Line	substituted by Memory Read
1111	Memory Write and Invalidate	substituted by Memory Write

Table 3 summarizes the PCI bus commands that are supported by the Glue Card. **Table 4** presents the PCI Configuration Registers. These registers are accessed by the commands "Configuration Read" and "Configuration Write". Only the shaded fields are implemented. Read accesses to the other fields always return zero. The Configuration Registers can be accessed either in byte, word or double word transfers. The first two Base Address Registers (BAR0 and BAR1) are reserved for system use. BAR2 holds the base address of the local resources described below. **Table 5** shows the values assigned by default to the Configuration Registers.

Table 4: PCI Configuration Registers. Shaded registers have been implemented.

Address	31 ... 24	23 ... 16	15 ... 8	7 ... 0
00h	Device ID		Vendor ID	
04h	Status Register		Command Register	
08h	Class Code			Revision ID
0Ch	BIST	Header Type	Latency Timer	Cache Line Size
10h	Base Address 0 (BAR0)			
14h	Base Address 1 (BAR1)			
18h	Base Address 2 (BAR2)			
1Ch	Base Address 3 (BAR3)			
20h	Base Address 4 (BAR4)			
24h	Base Address 5 (BAR5)			
28h	Cardbus CIS Pointer			
2Ch	Subsystem ID		Subsystem Vendor ID	
30h	Expansion ROM Base Address			
34h	Reserved			
38h	Reserved			
3Ch	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line
40h - FCh	Reserved			

Table 5: Default settings of the PCI Configuration Registers.

Name	Offset	Value
Vendor ID	00h	1172h (Altera ID)
Device ID	02h	0001h
Command Register	04h	Read/Write bits: Bit 1 – MEM_ENA Bit 6 – PERR_ENA Preset bits: Bit 7 – STEP_IMPL (forced to HIGH) Other bits hardwired to LOW
Status Register	06h	Updated Read/Write bits: Bit 27 – TAR_ABORT (Set when no slave response) Bit 31 – DET_PAR_ERR (Detected parity error) Preset bits: Bits 26..25 – DEVSEL_TIM (Set to 10) Other bits return LOW
Revision ID	08h	01h
Class Code	09h	118000h (Data acquisition + Other + None)
BAR0 – BAR2	10h	Bit 0 - MEM_IND (hardwired to LOW) Bit 2..1 - MEM_TYPE (hardwired to LOW) Bit 3 - PRE_FETCH (hardwired to LOW) Bits 15..4 - hardwired to LOW Bits 31..16 - r/w accessible
Subsystem Vendor ID	2Ch	201Ch
Subsystem ID	2Dh	0000h

Seen from the PCI bus, the entire user memory occupies an address range of 64 KB (16 address lines). As shown in **Table 6**, the block is divided into two regions: the first 64 bytes (16 double words) hold the Internal Registers and the rest of the space is used for the local bus accesses to the registers on the motherboard. Accesses to the Internal Registers and the local bus transfers may only be done as 32-bit double words via the Base Address Register 2 (BAR2) in the PCI Configuration Register.

Table 6: Assignment in the local memory of the device.

Offset	31 .. 24	23 .. 16	15 .. 8	7 .. 0
00h				
...	Internal Registers (r/w) (Table 5)			
03Fh				
040				
...	Local Bus (r/w)			
FFFFh				

The Internal Registers of the Glue Card controls three areas of activity: the JTAG and the I^2C busses and the General Purpose I/O lines. In addition the Internal Registers contains a Control and Status Register (CSR) which allows configuring several functions of the Glue card. A summary of all the Internal Registers is given in **Table 7**. All unused (not shadowed) registers return zero during read access.

In the current version of the Glue Card, the CSR (**Table 8**) allows enabling and disabling the JTAG and the I^2C bus and the General Purpose I/O lines. Disabling a bus puts the output pins in the high-impedance state. The CSR also allows selecting the width of the local bus and configuring the abort method when a time out occurs on the local bus. When a smaller local bus than 32 bits is used, it always starts from the least significant bit.

Table 7: Assignment of the Internal Registers.

Offset	31 .. 24	23 .. 16	15 .. 8	7 .. 0
00h				CSR (Table 6)
04h				JTAG_CON (Table 8)
08h				I2C_CON (Table 9)
0Ch				I2C_DAT
10h				PIO_PORT (Table 10)
14h				PIO_OE (Table 10)
18h				PIO_SSET (Table 10)
1Ch				PIO_SCLR (Table 10)
20h – 3Fh	Reserved			

Table 8: Bit assignment in the Control and Status Register (CSR). (Offset 00h in the Internal Registers).

Bits	Mnemonics	Description	Access
1...0	BSIZE [1..0]	Selecting the size of the local bus: 00 – 8 bits bus 01 – 16 bits bus 10 -- 24 bits bus 11 – 32 bits bus	r/w
2	TMO_ABORT	When set Target Abort is generated on a time out, this is missing slave response. When zero a dummy cycle is generated	r/w
3	BRST_ENA	When set enables burst operation on the local bus, otherwise only single data cycles are performed	r/w
4	JTAG_ENA	Enables/disables the JTAG bus	r/w
5	PIO_ENA	Enables/disables the General Purpose I/O port	r/w
6	I2C_ENA	Enables/disables the I2C bus	r/w
7	JTAG_TRST	JTAG reset line (active in low)	r/w

5.3.2 I²C Bus

Although serial buses don't have the throughput capability of parallel buses, they require less wiring and fewer IC connecting pins.

It must be remarked that a bus is not merely an interconnecting wire but it embodies all the format and procedures for communication within the system. Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss and blockage of information. The communication system must not be dependent on the devices connected to it, otherwise modifications or improvements would be impossible.

A procedure has also to be devised which device will be in control of the bus and when. If different devices with different clock speeds are connected to the bus, the bus clock source must be defined. I²C specifications take care of all requirements in detail.

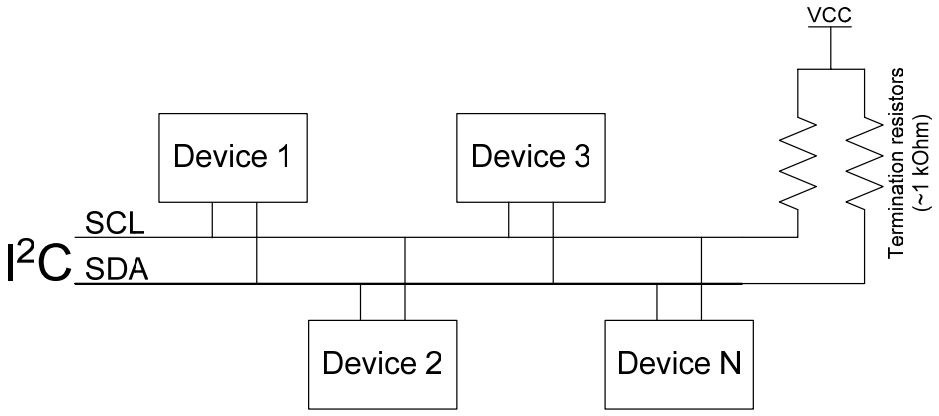


Figure 17: Generic I²C bus configuration.

I²C (Inter-Integrated Circuit) **[20]** is a simple bi-directional 2-wire bus developed by Philips in order to control efficiently integrated circuits interconnected with few signals on electronics boards. One line feeds the clock to the devices (*Serial Clock Line, SCL*) and the other carries data (*Serial Data Line, SDA*). Bi-directional data transfers can be made at up to 100kbit/s in Standard-mode, up to 400kbit/s in Fats-mode or up to 3.4Mbit/s in the High-speed mode but the last mode requires special routing techniques.

Each device is recognized by a unique peripheral address and can operate as either a transmitter or receiver, depending on the function of the device. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit the transfer; any device addressed is considered a slave.

The I²C bus is a multi-master bus: more than one device capable of controlling the bus can be connected to it; that means that more than one device master could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event, an arbitration procedure is implemented. This procedure relies on the wired-AND connection of all I²C interfaces to the I²C bus. A master can operate both as master-transmitter or master-receiver. Since on Freja the master of the I²C bus is always the FPGA on the Glue Card, bus arbitration is not discussed in more detail.

All I²C bus compatible devices incorporate an on-chip slow interface which allows them to be addressed directly between each other.

Bit structure

Bit transfer structure on I^2C is similar to many other serial buses. Unique situations are defined as START and STOP conditions:

- A HIGH to LOW transition on the SDA line while SCL is HIGH indicates a START.
- A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP.

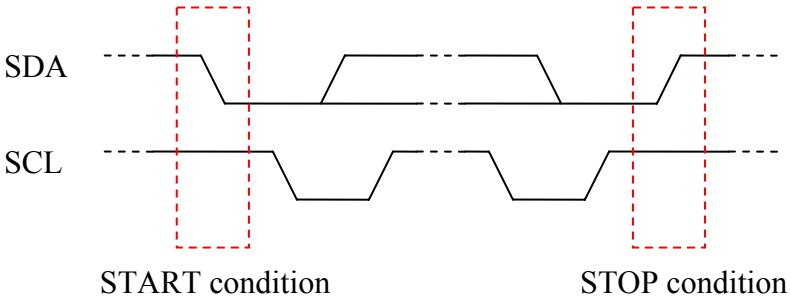


Figure 18: I²C START and STOP conditions.

START and STOP conditions are always generated by the master.

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW. Every data transaction put on the SDA line must be 8-bits long. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first. By holding the clock line LOW a slave can force a master into a wait state.

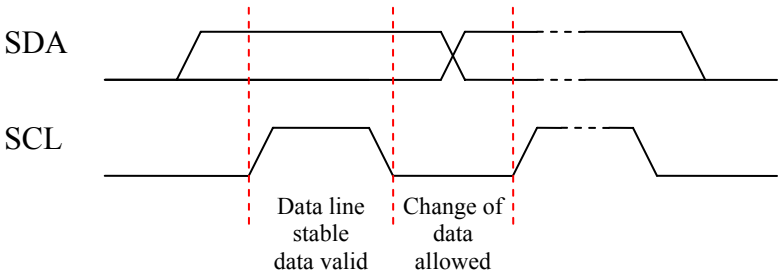


Figure 19: Valid Data line transaction according to Clock status.

Freja uses the format with 7-bit peripheral address (a 10-bit is available too) which is structured as follows:

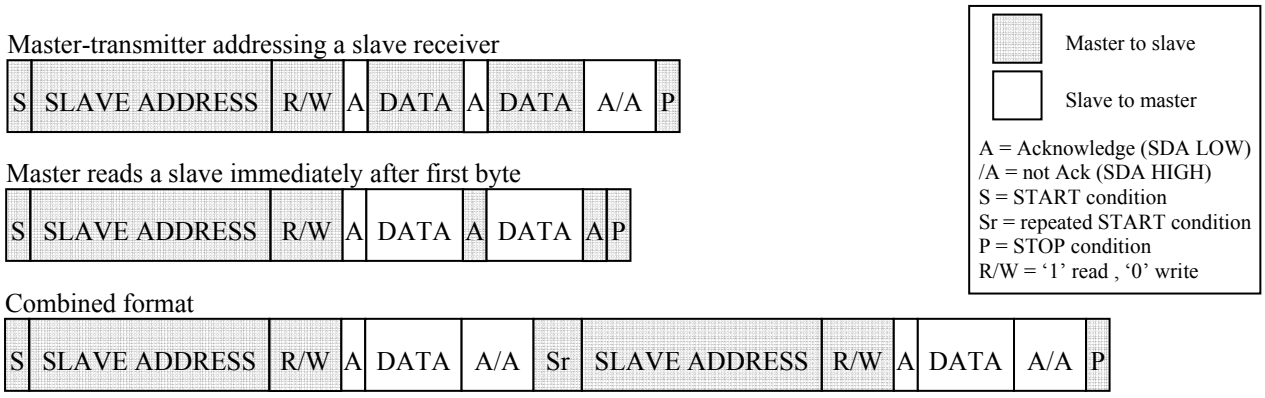


Figure 20: I2C data information structure for sequential memory access.

Many I²C devices allow to access more than one I²C register in a non sequential mode, in this case the structure is:

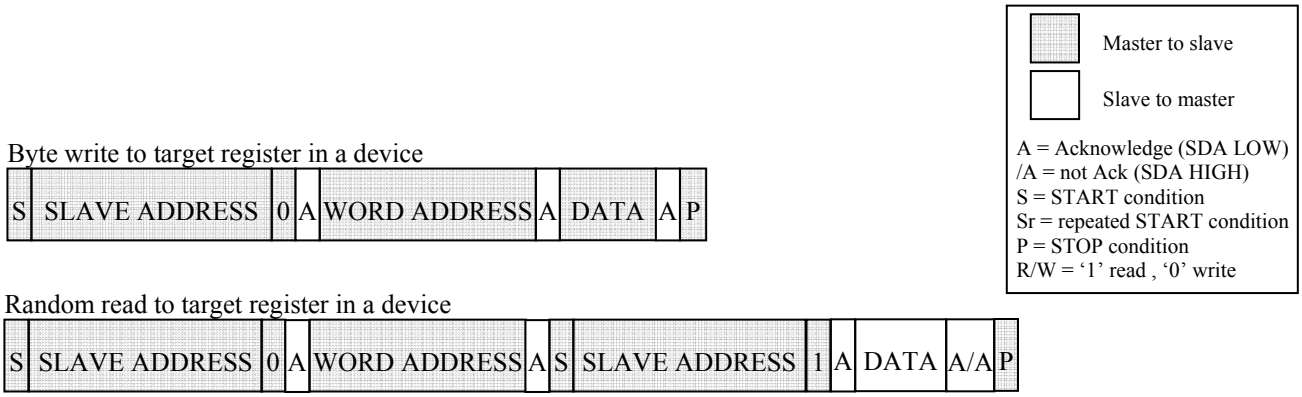


Figure 21: I²C data information structure for random memory access.

I²C on the glue card

The operation of the I²C bus on Freja is controlled via the PCI bus by accessing two 8-bit wide registers in the Internal Registers: I2C_CON and I2C_DAT. The I2C_CON register provides an interface to the I²C state machine implemented in the Glue Card FPGA. The procedure to operate the bus follows the standard specification described before and the bit assignment in the I2C_CON register is shown in **Table 9**.

Table 9: Bit assignment in the I²C control register I2C_CON (Offset 08h in the Internal Registers).

Bits	Mnemonic	Description	Access
0	NOACC	Error bit. Set after a bus request access failed (unsuccessfully writing START, STOP or data or reading data). Cleared after the first valid transfer. Note: Writing or reading the data register I2C_DAT when DONE is low will cause this error. Also reading data while in "sending" mode or writing data while in "receiving" mode will generate this error.	ro
1	ACK	The bit indicates that the slave acknowledged the address phase (for read or write). For write transfers ACK = '1' when byte is accepted.	ro
2	DONE	Set after completion of the start, the data or the stop phase . Cleared after setting the START or the STOP and reading or writing the register I2C_DAT.	ro
3	BUSY	The bit indicates that the I2C bus is being used. It is set after setting the START bit and it is cleared when the stop condition occurs.	ro
4	I2C_RESET	When I2C_RESET is set, a reset of the entire I2C controller is performed. The reset is a momentary action and extends only over two system clock cycles.	mom
5	LAST	Can be set only if DONE has been asserted. Forces no read acknowledge while sending data. Cleared after the stop condition occurs.	r/ws
6	STOP	Can only be set if DONE has been asserted. It generates a stop condition. It is cleared when the access has been completed.	r/ws
7	START	Can be set if either BUSY is LOW or DONE is HIGH. It generates the start condition. When BUSY is asserted, repeated start conditions may be generated. Cleared after the acknowledgment in the address phase	r/ws
Notes:		Bits 3 ... 0 are read only Bit 4 is a momentary write-only action bit and it doesn't retain its value. Bits 7... 5 are special action read/write bits. Writing a logic one to the bit sets the appropriate bit in the register. Writing a logic zero has no effect.	

Data are transferred to and from the I²C bus by writing and reading the I2C_DAT register (**Table 7**). The register holds the address byte during the address phase, and a byte of data to be transferred in write mode or a received data byte in read mode.

The least significant bit of the address byte contains the direction flag. A "0" means a write transfer and a '1' a read transfer.
The bus frequency is approximately 100 kHz by default for compatibility. It can be configured to run four times faster for devices that support the higher speed.

I²C on Freja

Three I²C devices are used on Freja (**Figure 22**):

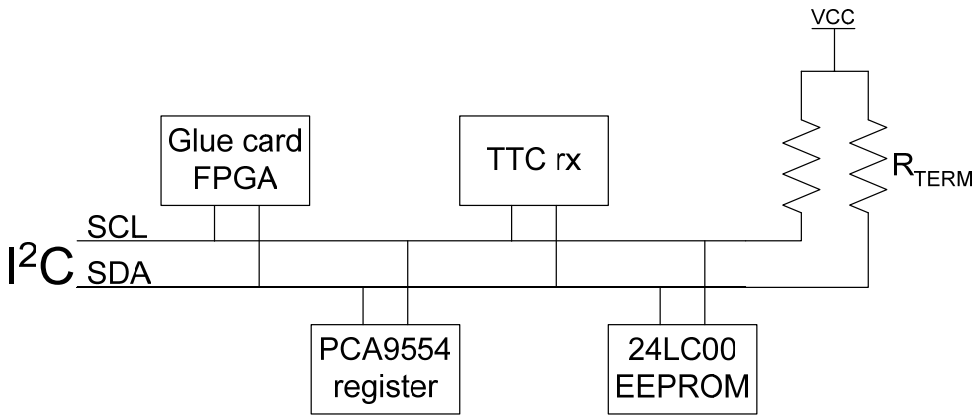


Figure 22: Block diagram of the devices connected to I²C bus on Freja.

a) **EEPROM 24LC00**: small non volatile memory that stores board identification information. This information is used by the control system to identify different types of boards (see **Section** Error! Reference source not found.). The EEPROM ha a storage capacity of 128 bits organized as 16 words of 8 bits. The memory structure is shown in **Table 10**.

Table 10: EEPROM physical address and internal structure.

Physical or peripheral address	**0xAX**	If X bit is: 0. Write memory 1. Read memory: whether random or sequential

Command byte	Description	
0x00	System ID	[TFC]
0x01	Board Type	[Freja, Odin, etc.]
0x02	Board Revision	[Ver 1, Ver 2]
0x03	Board Number	[0, 1, 2, 3, etc.]

b) **I²C REGISTER PCA9554**: device used to drive the control lines of simple components on the board. Each output line is associated and driven by an internal register (**Table 11**).

Table 11: I²C register physical address and control lines structure.

Physical or peripheral address	**0x4X**	If X bit is: 0. Write memory 1. Read memory: whether random or sequential

Command byte	Description	Value	Comments	
0x03	I/O configuration register	0x00	All I/O lines set to output	
0x01	Output port register		$\underbrace{X}_{FPGA\,reset\,[SRESn1]}$ $\underbrace{X}_{not\,used}$ $\underbrace{X}_{TTCrs\,reset\,[SRESn2]}$ $\underbrace{X}_{H\_EXT^*}$ $\underbrace{X}_{TTCrx\,reset}$ $\underbrace{X}_{SEL2}$ $\underbrace{X}_{SEL1}$ $\underbrace{X}_{not\,used}$ (SEL2 SEL1: Clock selection) * clock source on TTCrs	
			0x14 [10100]	Internal clock
			0x12 [10010]	External clock
			0x10 [10000]	TTCrm clock
			0x16 [10110]	TTCrs external clock
			0x06 [00110]	TTCrs internal clock

c) **TTCrx**: the configuration of the TTC receiver chip is done via I²C. The I²C bus of this device is based on a double register configuration: an address is written to an address register in order to target a memory location, while data are transferred through a data register. This access mode is thus slightly different from the one implemented with the other devices used on the TFC boards. More details can be found in **[8]**.

5.3.3 JTAG Bus

Boundary-Scan Testing (industry standard IEEE 1149.1 **[22]**) was developed in the mid-1980s by the *Joint Test Action Group (JTAG)* to detect physical faults on PCBs caused by increasingly crowded assemblies due to novel packaging technologies, such as today's BGA packages.

In space-constrained environments, or high-speed designs, it's not always possible to incorporate test pads that link the PCB to an *ATE (Automatic Test Equipment)* bed-of-nails interface. Boundary-scan embeds test circuitry at chip level to form a complete board-level test protocol. With boundary-scan is it possible to access even the most complex assemblies for finding and diagnosing hardware problems but also perform In System device Programming (ISP).

Figure 23 shows a block diagram of the JTAG components integrated inside boundary-scanable devices. The test logic consists of an instruction register and a controller and is accessed through a *Test Access Port (TAP)*. The boundary scan technique involves the inclusion of a shift-register stage (contained in a boundary-scan register cell) connected to each component pin so that signals at component boundaries can be driven and read using scan testing principles.

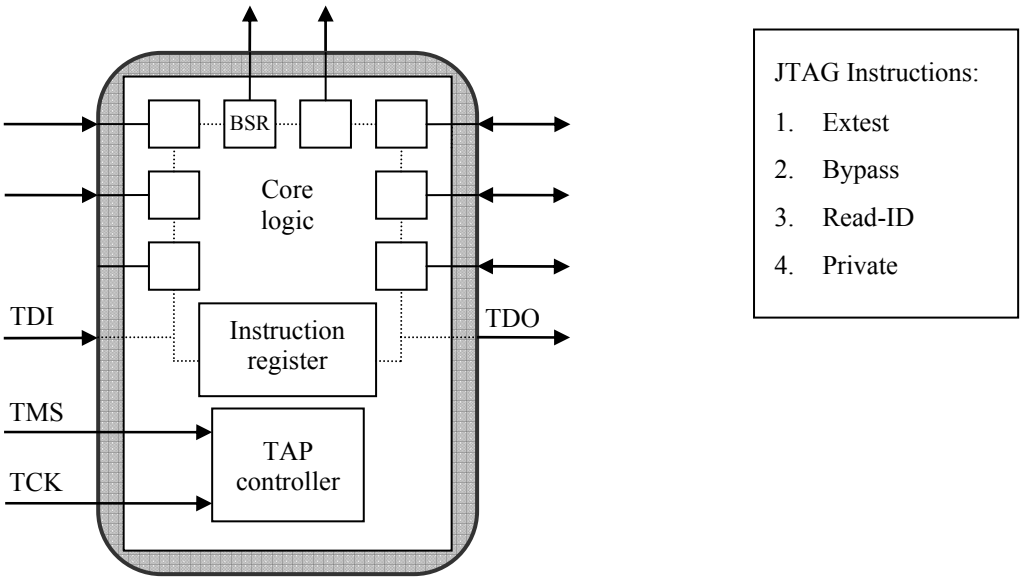


Figure 23: Internal structure of JTAG boundary-scanable IC.

An overview of JTAG building blocks follows:

1. **Test Access Port (TAP):** it is a general purpose port that provides access to all the test support functions integrated in a boundary-scanable device. Four signals are used to configure and communicate to and from a TAP:
   a) *Test Clock Input (TCK):* a clock separate from the system clock.
   b) *Test Data In (TDI):* data is shifted into the JTAG-compliant device via TDI.
   c) *Test Data Out (TDO):* data shifted out the JTAG-compliant device via TDO.
   d) *Test Mode Select (TMS):* TMS commands select test modes as defined in the JTAG specification.

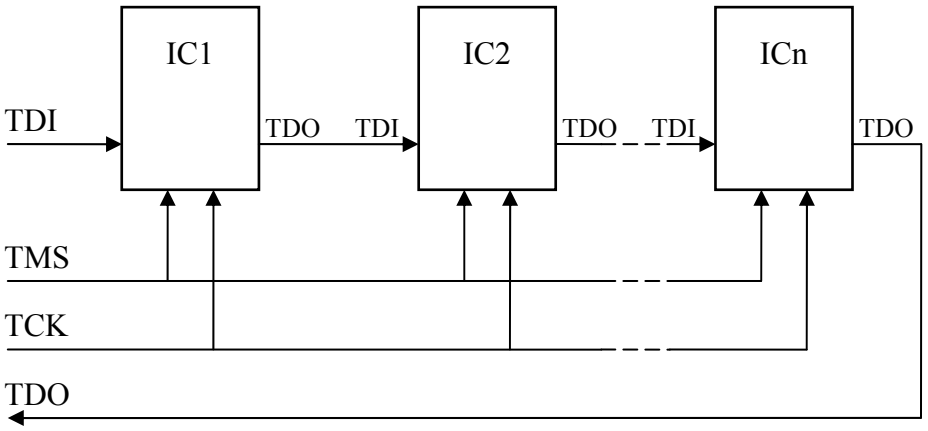


Figure 24: Configuration of JTAG devices with a serial connection.

2. **TAP controller:** the TAP controller is a state machine that provides access to the functions built into the JTAG-compliant device. The state machine controls all operations for the JTAG-compliant device. A user can sequence through the state machine functions via the TCK and TMS inputs.

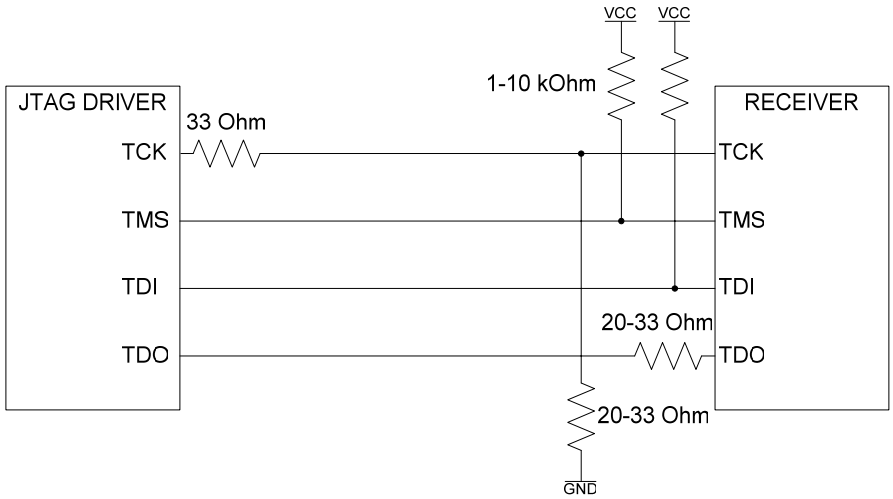


Figure 25: Configuration of the termination resistors of the JTAG line.

3. **Instruction register:** instructions used to select the test to be performed or the data register to be accessed are shifted into the design by the instruction register. A number of mandatory operations are defined by JTAG specification but design-specific instruction can be added as well to extend the functionalities of the test logic.
   The instruction register allows selection of the following compulsory registers and their relative functions:

   a. *Bypass:* reduces the BSC shift path through the device to a single bit register. This instruction is used when a direct operation into only some devices of the chain has to be done: bypass allows the user to bypass BSCs inside an IC to pass data directly into another chip.
   b. *Sample/preload:* instruction used either to sample the data currently contained in the BSCs, or to preload data into the BSCs.
   c. *Extest:* when extest is performed the BSCs attached to the JTAG device input pins act as sensors while the BSCs attached to output pins propagates data to interconnecting devices. The interconnecting devices may or may not be JTAG-compliant.

   Optional instructions which are very often implemented by chip designers are:

   d. *Usercode/Idcode:* instructions used to serially read a code from a target component. Use of the Idcode will provide information on the base component, while use of the Userdcode will provide information on the particular programming of an on-board component (e.g. used to program TFC FPGAs).

4. **Test data registers:** the test logic architecture contains a minimum of two test data registers. Selection of the register that forms the serial path at a given time is controlled from the instruction register (**Figure 26**).

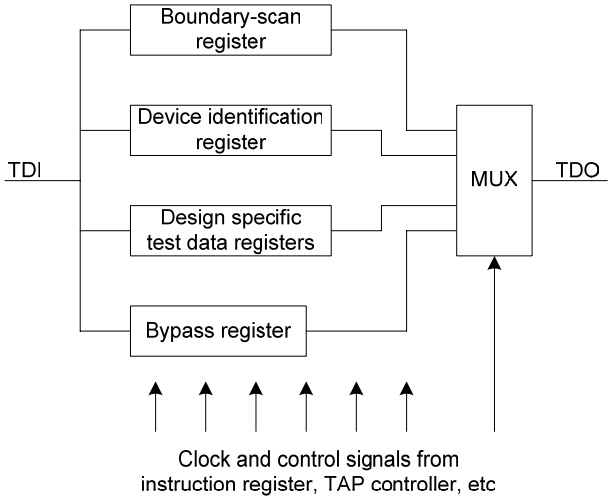


Figure 26: Test data registers.

a. *The bypass register:* provides a single-bit connection through the circuit when none of the other test data register is selected. This register can, for example, be used to allow test data flow through a device without affecting his BSCs.
b. *The boundary-scan register:* this allows testing of board interconnections, detecting typical production defect such as opens, shorts, etc. It also allows access to the inputs and outputs of components when testing their system logic or sampling of signals flowing through the system input and outputs.
c. *The device identification register:* optional test data register used determine for example device's manufacturer, part number, etc.
d. *The design-specific test data register:* optional registers may be provided to allow access to design-specific test support features. The manufactures of programmable devices use this register to implement the programming interface. This will be discussed in more details in **Section 5.5.5** and **5.5.6**.

5. **Boundary-Scan Register (BSR):** each IO pin of an Integrated Circuit (IC) is connected to three cells, each known as *Boundary-Scan Cell (BSC)*, of a shift-register (
6. Figure 27). The cells link the JTAG circuitry to the IC's I/O logic. All BSCs of a particular device constitute a *Boundary-Scan Register (BSR)***.** BSR logic becomes active when performing JTAG testing, otherwise under normal IC operation it remains passive.

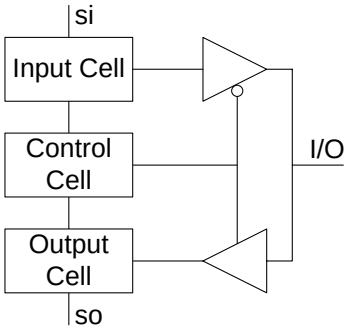


Figure 27: Structure of a Boundary-Scan Register.

JTAG on the glue card

The JTAG bus of the Glue Card is driven directly via the PCI bus by accessing the JTAG_CON register (**Table 12**) in the Internal Registers (**Table 7**).

In a read access the first four bits reflects the state of the JTAG bus lines. Transactions on the JTAG bus are performed by writing to the register.

The states of the TCK, the TDI and the TMS output lines are set by writing commands to the three two-bit registers that in a write access are associated with the three lines.

The set/clear operations are defined in **Table 12**. JTAG transactions are thus made by putting data on the TMS and the TDI lines and toggle the clock line TCK, and by reading the TDO line.

Table 12: Bit assignment in the JTAG control register JTAG_CON (Offset 04h in the Internal Registers).

Bits	Read Access	Write Access	
0	Read state of TCK line	00 – no action 01 – selective clear TCK bit 10 – selective set TCK bit 11 – no action	
1	Read state of TMS line		
2	Read state of TDI line	00 – no action 01 – selective clear TMS bit 10 – selective set TMS bit 11 – no action	
3	Read state of TDO line		
4	0	00 – no action 01 – selective clear TDI bit 10 – selective set TDI bit 11 – no action	
5	0		
6	0	**TMS**	Generate bus sequence (see Note 1)
7	0	**TDI**	
Note 1:	When during write access all bits in the field 5 ... 0 are set to "1", a four clock sequence is generated with the TMS and the TDI lines set according to bit 6 and 7, respectively.		

The Glue Card also allows making a full JTAG transaction for every PCI write access by generating a sequence of JTAG levels. The sequence is invoked when the six bits J2C_CON [5...0] are all HIGH. The sequence consists of four clock cycles. At the first cycle the level of the clock line TCK is de-asserted. At the second cycle the levels of the TMS and the TDI lines are set according to the state of bit 6 and 7, respectively, in the JTAG_CON register (**Table 12**). The TCK is asserted during the third and the fourth cycle, and is finally de-asserted at the end of the fourth cycle. The states of the TMS and the TDI lines remain until they are changed.

Holding the TMS line in the high state during more than four clock cycles resets the TAP controller of the remote JTAG interface. After power up the TMS line is set to HIGH by default.

5.3.4 Local Bus

The *Local Bus (L_BUS)* on the TFC boards provides a data path for configure, control and monitor data between the PCI Bus and non-PCI devices such as FPGAs, FIFO memories, etc. The implemented local bus functionality is a subset of the PLX 9030 Local Bus. The Glue Card provides a 32-bit multiplexed synchronous bus. At the back-end, the local bus width can be tailored to 8, 16, 24 or 32 bits, always occupying the least significant bits.

The Glue Card board acts as a Local Bus Master with a permanent bus ownership. The memory range between address 0040h and FFFFh in BAR2 is designated for the local bus transfers.

Table 13: List of the Local Bus signals.

Signal	Name	Type	Function
LCLK	Local Bus Clock	O	Local bus clock, up to 40 MHz
LRESETn	Local Bus Reset Out	O	Asserted when the CCPC is reset
LAD [31...0]	Address/Data Bus	I/O	During the address phase, the bus carries the address of the device on the [15...2] lines. During the data phase the bus carries 32-bit data words
LADSn	Address Strobe	O	Indicates a start of a new bus access and valid address. Asserted during the first clock cycle of the bus transaction
LW_Rn	Write/Read	O	LOW for read accesses and HIGH for write accesses
LWAITn	Wait Out	O	Asserted by the master to insert wait states
LRDYn	Local Ready Input	I	Indicates valid read data on the bus in a read access or that write data is accepted in a write access. The signal is not sampled until LWAITn is de-asserted
LLASTn	Burst Last Data	O	Asserted by the master to indicate the last data transfer in a bus access
LTERMn	Burst Terminate	I	Asserted by the slave to terminate a burst access

Four types of bus transactions can occur on the Local Bus:

- *Read* and *Write*
- *Read* and *Write Burst*

A Local Bus transaction is bounded by the assertion of the LADSn (address strobe) at the beginning and by the de-assertion of the LLASTn (last data transfer) or by the de-assertion of LTERMn (terminate) at the end. The access consists of an address cycle followed by one or more data transfers. During each clock cycle of the access, the Local Bus is in one of the four basic states as defined below. A clock cycle consists of one LCLK clock period.

Local Bus Clock

The Local Bus clock LCLK is generated externally to the Glue Card. All Local Bus signals except for the LRESETn are driven and sampled on the rising edge of the LCLK.

With respect to the LCLK, the following time constraints must be observed:

$\mathbf{T_{setup}}$ ≤ 7 ns - setup time
$\mathbf{T_{hold}}$ ≥ 1 ns - hold time

The current version runs at a frequency up to 40 MHz which allows easy attachment of relatively distant devices located on 9U VME motherboards.

Basic Bus States

The four basic states are idle, address, data/wait, and recovery. Upon request from the PCI target, the Local Bus Master starts a bus access by entering the address state while presenting a valid address on the address/data bus and asserting the LADSn for one clock period. Data is subsequently transferred in the data/wait state. The LRDYn is used by the slave to indicate that data is written or that data is available for reading. It may also be used to insert wait states by temporarily de-asserting the signal. The LWAITn may be asserted by the master to suspend the slave, which in this condition should not sample data during a write access and should not update data during a read access. The LLASTn is asserted by the master to indicate the last data transfer of the access. To terminate a burst access the slave may assert the LTERMn signal instead of only de-asserting the LRDYn.

The direction of the data transfer is determined by the LW_Rn signal, which is low for a read and high for a write access. The LW_Rn must be kept either asserted or de-asserted during the entire transaction.

After the data transfer, the bus enters the recovery state to allow the bus device to recover. The bus then enters the idle state and waits for another access.

Write data are accepted by the slave on the rising edge of the LCLK when the LW_Rn is high, the LWAITn is high, and either the LRDYn or the LTERMn is low. The master accepts the read data on the rising edge of the LCLK when the LW_Rn is low, and either the LRDYn or the LTERMn is low. This process repeats until the transaction is terminated either by the master or by the slave.

Acknowledgement by the Slave

In a bus transaction, the presence of the slave is signalled to the master by a low level on the LRDYn line or the LTERMn line. When multiple slaves are attached to the local bus, the device selection is based on internal decoding of the address lines by all slaves. Only one slave at a time may drive the LRDYn and the LTERMn. Consequently the slaves which are not selected must drive the LRDYn line in a tri-state. The same rule applies to the LTERMn. Internal on-board pull-up resistors have been implemented on these two lines on the Glue Card.

When no slave responds in a predefined period of time with either the LRDYn or the LTERMn asserted, the PCI interface completes the transaction with either a TARGET ABORT or with a valid dummy cycle. The method depends on the configuration of the TMO_ABORT bit in the CSR (see **Table 8**). In the case of a dummy cycle nothing happens in a write access and a read access returns all zeros.

Examples of Local Bus Timing

Figure 28 shows an example of two Local Bus transactions. The first transaction is a burst read access of four words with one wait state inserted by the master and one data phase delayed by the slave. The transaction is terminated by the master. The second transaction is a single write access which is also terminated by the master.

Figure 29 shows a burst write access of four words which is terminated by the slave. The second transaction is single word read access where the master attempts a burst read but the slave forces a single cycle by terminating the transaction.

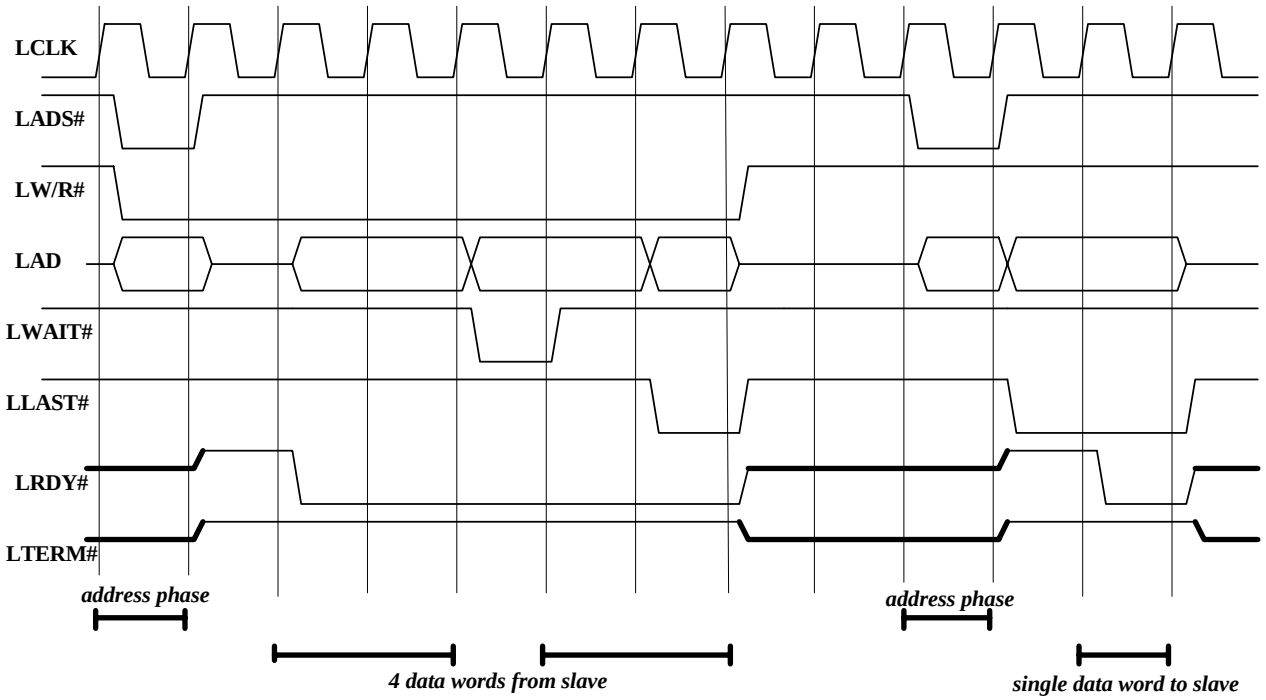


Figure 28: Timing diagram showing an example of a burst read and a single write transaction.

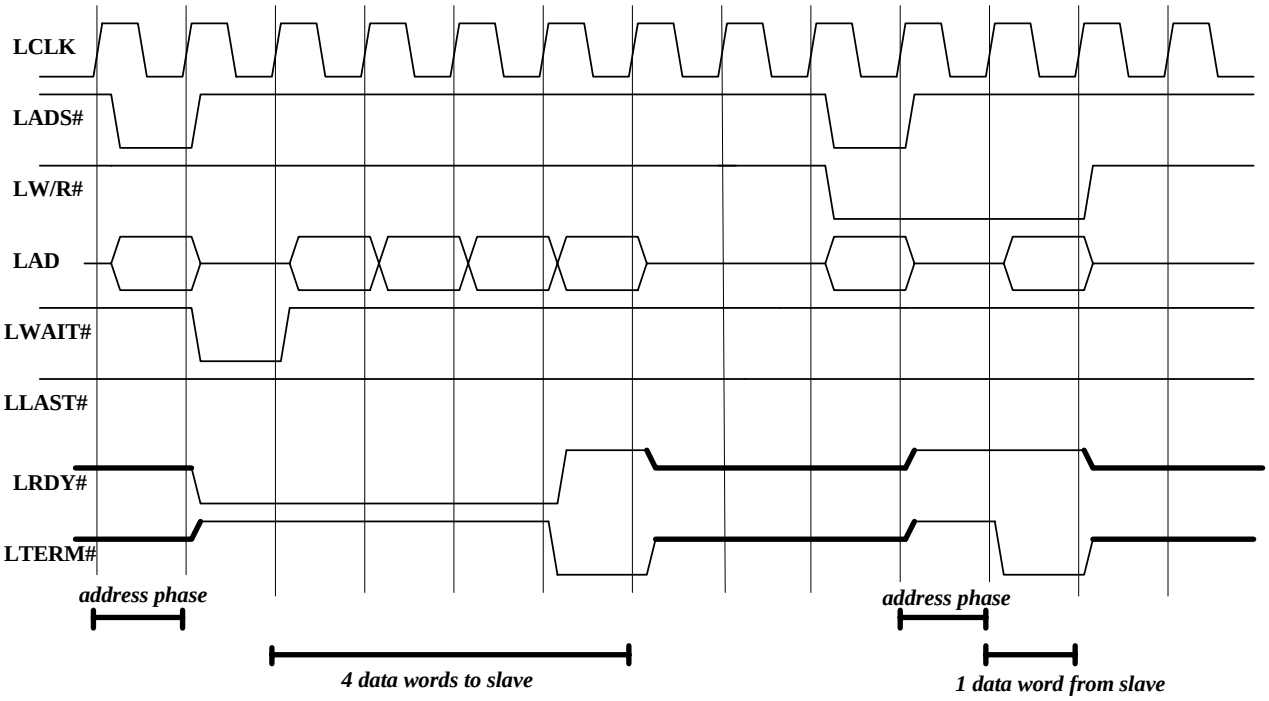


Figure 29: Timing diagram showing an example of a burst write and a single read.

5.4 General purpose Input/Outputs

5.4.1 Introduction

New signaling technologies for cable transmission and critical on board paths were introduced to avoid the bottle neck represented by the TTL and CMOS technology used in all IC devices capable of computing power. Current state-of-the art TTL and CMOS logic families have attained performance levels which require controlled impedance interconnect for even relatively short distances between source and load. As a result system designers who are using state-of-the-art TTL or CMOS logic are already forced to deal with special requirements of high speed logic. In addition the large output swings and relatively fast output slew rate of today's high performance CMOS/TTL devices increases the problems of cross talk and EMI radiation. This problem, along with common mode noise and signal amplitude losses, can be alleviated to a great degree with the use of differential signalling.

Even though conversion devices are required (which slightly increases the board costs for example) the gain in terms of throughput, signal quality at the receiver side and reachable distances is incomparable. Board design is simplified as well since termination techniques are easy to implement and well described by every standard definition.

With this idea in mind differential signaling technology was implemented on all the TFC boards to provide a safe communication path both on board and on cable connections.

5.4.2 LVDS technology

Low Voltage Differential Signaling (LVDS) is a communication technique introduced by National Semiconductor **[23]** with the aim to transmit data using a very low voltage swing (about 350mV) over a differential transmission line (both PCB traces and balanced cables). The advantage of the differential approach is that the noise is coupled onto the two wires as common mode (appears on both lines equally) and is thus rejected by the receivers which looks at only the difference between the two signals. The differential signals also tend to radiate less noise than single-ended signals due to the cancelling of magnetic fields. It is the fact that differential technologies, such as LVDS, have less noise which allows operating with lower signal voltage swing. The low swing nature of the driver means data can be switched very quickly (bit rates up to hundreds of Mbps).

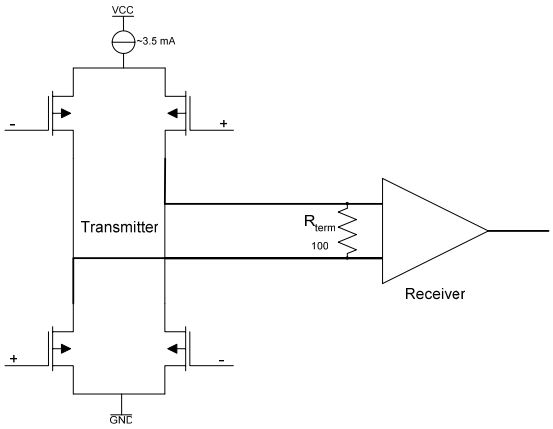


Figure 30: Simplified diagram of LVDS driver and receiver connection.

National's LVDS outputs consist of a current source (nominal 3.5 mA) which drives one of the differential pair lines. The receiver has high DC impedance (it doesn't source or sink DC current), so the majority of the driver current flows across the 100 Ohm termination resistor (that represents a very easy termination technique) generating about 350 mV across the receiver inputs.

Different topologies are available to configure LVDS devices. The TFC boards equipped with LVDS usually use point to point configurations. Special cases is represented by the LVDS I/O ports of the test board in which a bi-directional half-duplex configuration is used with LVDS transceiver chips in order to save I/O pins on the FPGA. **Figure 31**, **Figure 32** and **Figure 33** show the different configurations:

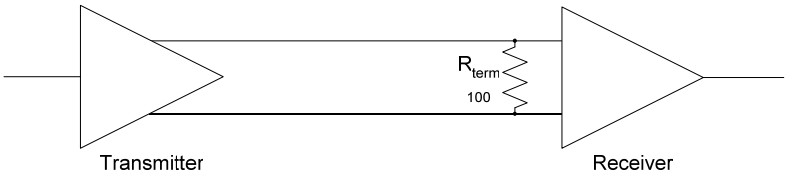


Figure 31: LVDS point-to-point configuration.

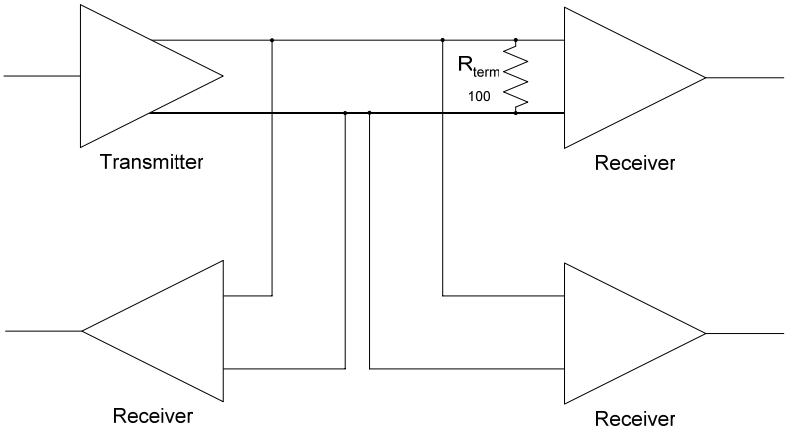


Figure 32: LVDS multidrop configuration.

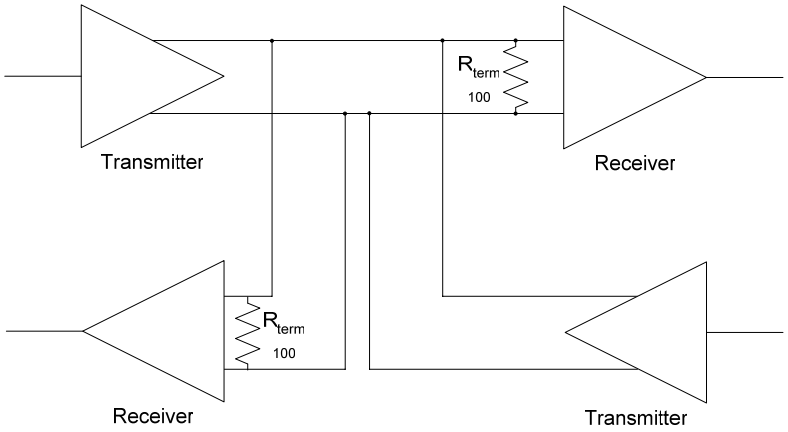


Figure 33: LVDS bi-directional half-duplex configuration.

LVDS represents the most suitable solution for the TFC high speed data transmission (40 MHz) over quite long distances (2 to 10 meters).

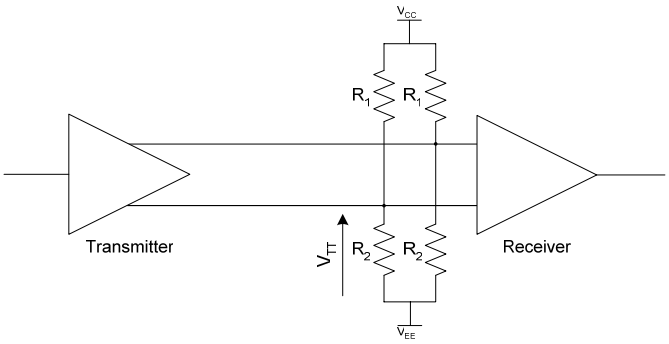
Due to its extreme simplicity, reliability and high-end performance in terms of power consumption and speed, LVDS is getting more and more a good candidate for all kinds of future applications.

5.4.3 ECL technology

Emitter Coupled Logic (ECL) in all its variations Positive (PECL), Negative (NECL) or Low Voltage (LVECL) is another largely used differential transmission technology **[24, AN1672/D]**.

While LVDS adopts a common solution for line termination, ECL offers the opportunity to choose between several different solutions. A standard ECL output driver typically uses a current switching differential with an emitter follower for level shifting the output.

For a proper static and dynamic emitter operation, the emitter follower must remain in the active region of operation, which means that an external resistive path should be provided from the output pin voltage, operating at a level more negative than the worst case V_{OL}, such as V_{EE}. The resistor terminations can be considered a DC termination of the Emitter Follower output structure. Terminations are used to match the characteristic line impedance to prevent reflections, jitter and skew over long lines.



$$V_{TT} = V_{CC} - 2.0V = V_{CC}\left(\frac{R_2}{R_1 + R_2}\right)$$

Figure 34: Thevenin termination for ECL lines and VTT computation formula.

For all the TFC differential and single ended lines driven by ECL logic the terminations implemented are of the Thevenin Equivalent (or parallel termination) type **[24, AND8020/D]**. Two reasons led to this choice: easy and precise calculation of the partition ratio to adjust the receiver input voltage, elegant design implementation by using resistor termination networks (specific designs for ECL logic are nowadays available on the market).

The V_{TT} supply is used to sustain the emitter follower output transistor in its active operating region under all operating conditions. A minimum continuous current occurs for the most negative V_{OL}, therefore the V_{TT} supply must remain more negative than the worst case V_{OLmin} and always sink current (**Figure 35**). While a Thevenin Parallel technique dissipates more termination power, it does not require an additional external V_{TT} supply. This additional power is consumed entirely in the external resistor divider network and thus will not change the current being sourced by the device, hence it does not alter the IC reliability of lifetime.

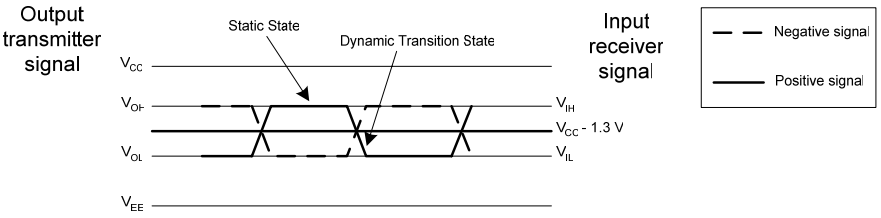


Figure 35: State levels of V_{OH} and V_{OL}.

The Thevenin equivalent of the two resistors needs to be equal to the characteristic impedance of the signal transmission line.

	PECL	LVPECL
$R_2 = Z_0\left(\frac{V_{CC}-V_{EE}}{V_{CC}-V_{TT}}\right)$	$R_2 = 50\left(\frac{5-0}{5-3}\right) = 125\Omega$	$R_2 = 50\left(\frac{3.3-0}{3.3-1.3}\right) = 82.5\Omega$
$R_1 = R_2\left(\frac{V_{CC}-V_{TT}}{V_{TT}-V_{EE}}\right)$	$R_1 = 125\left(\frac{5-3}{3-0}\right) = 83.3\Omega$	$R_1 = 82.5\left(\frac{3.3-1.3}{1.3-0}\right) = 126\Omega$

The theoretical values don't match any real value. As a sufficiently well approximated value R_1=81Ω and R_2=130Ω were chosen using BURNS 803 Series ECL Termination Networks.

To provide DC insulation to some receivers, capacitive coupling (**Figure 36**) has been also implemented and tested in different solutions **[24, AND8020/D]**. AC signal have the line DC blocked and requires then a DC restoration voltage which can be provided using the V_{BB} pin integrated in most of the actual ECL devices (**Figure 37**).

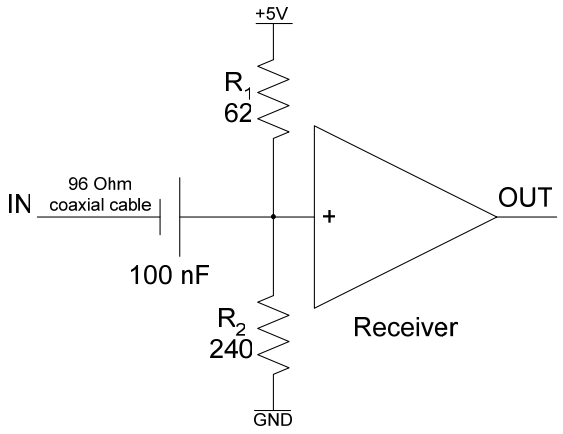


Figure 36: Single ended configuration with 96Ω cable matching and DC blocking capacitor.

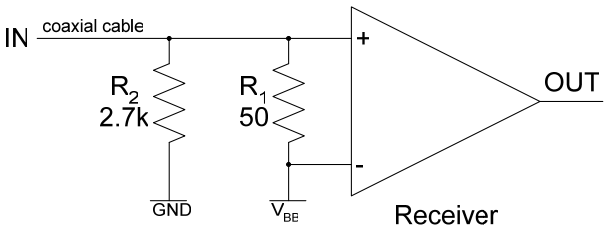


Figure 37: Single ended configuration with DC level recovery using V_{BB}.

A precise evaluation of the terminations is extremely important to ensure a good signal transmission (as experienced during tests, without termination transmitter devices will not even work!). None of the chips used in this project use on-chip integrated terminations; for that reason termination are always implemented on-board.

5.4.4 I/O interface technologies on the test board

ECL and LVDS technologies were chosen to allow full access to all the functionalities of the TFC boards during test routines. ECL is used to handle the TTC network and the TFC switch via the TTCrs. Six additional ECL lines, organized in three transmitters and receivers, are used for general tests. LVDS is instead used to connect Freja to both the L0 and L1 trigger path of the Readout Supervisor and to send and receive throttles from the throttle switch. In total 34 bi-directional LVDS lines have been implemented for this purpose.

5.5 Control logic: Altera APEX FPGA

5.5.1 Introduction

As introduced in the previous section (**Section 5.1**), the core part of the logic of the test board, is the FPGA. This is of great interest because of the importance gained by these devices in the last years.

The whole idea of the board was developed around the implementation of such a device: the use of an FPGA allows the users to build a tool which is a very good compromise between high performances in terms of speed and computing power (which best is achieved using specific hardware designs) and extreme flexibility.

It must be remarked that:

- All the electronics on the test board is organized around this chip.
- All the control and monitoring is done via the FPGA.
- Most of the configurations are dedicated to setup the functionalities of this device.

The advent of FPGAs caused a slow revolution in the way in which hardware is developed. In the past board designers were constantly struggling to solve complicated Boolean functions and trying to fit impossible numbers of chips on a board with few chances to repair and reconfigure the boards in case of bugs, malfunctions or design upgrades. Using FPGAs these days are over. Compactness and flexibility are the common denominators of all programmable devices.

The design starts from the choice of the programmable device itself: according to the number of I/Os needed, the size of the code to run in the chip, etc. Most of the planning phase can be done via software with compilers, synthesizers, simulators, etc; saving time and resources.

If money is not a crucial issue, an over sized chip ("fat and fast" compared to the code size and the speed requirement predicted) will ensure a good safety margin even for future improvements. If, on the other hand, money represents a design limitation (as is often the case) code software simulation during the planning phase will assure a precise choice of the chip to invest in.

5.5.2 The evolution of FPGAs

Before the advent of programmable logic, custom logic circuits were built at the board level using standard components, or at the gate level in expensive *Application-Specific Integrated Circuits (ASIC)*.

A *Field Programmable Gate Array (FPGA)* is an integrated circuit that contains many identical logic cells that can be viewed as standard components. Each logic cell can independently take single customized functions using a small set of primitives (building blocks) such as Look-Up Tables (LUT), logic gates (AND, OR, etc.), flip-flops, etc.

The logic cell architecture varies between different device families. Generally speaking, each logic cell combines a few binary inputs (typically between 3 and 10) to one or two outputs according to a Boolean logic function specified in the user program. In most families, the user also has the option of registering the combinatorial output of the cell, so that clocked logic can be easily implemented. The cell's combinatorial logic may be physically implemented as a small look-up table memory (LUT) or as a set of multiplexers and gates. LUT devices tend to be a bit more flexible and provide more inputs per cell than multiplexer cells at the expense of propagation delay.

The individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks.

It's immediately clear how one of the points of strength of an FPGA is the fact that it allows users to implement complex functions in an integrated circuit by simply programming it directly on the field after manufacture. This concept is summarized in the expression "Field Programmable". Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up, or it can be part of an In System Programming sequence.

FPGA capacities have been growing year after year and modern devices can provide something like two million gates in a single chip. This is about the level of complexity of the first Intel Pentium microprocessors! When originally introduced FPGAs only had a handful of gates and designs were specified as logic equations connecting them up. Nowadays they are programmed in high level Hardware Description Languages (HDLs) such as VHDL, which superficially resembles high level programming languages such as C.

On the other edge of modern programmable devices *CPLDs (Complex Programmable Logic Devices)* can be found, with their PAL-derived, easy-to-understand AND-OR structure, offering single-chip solutions with fast pin-to-pin delays, even for wide input functions. Once programmed, the design is stored in non volatile memories and thus made secure. The limited complexity (<500 flip-flops) makes CPLDs the best choice, for example, for "glue logic" functions.

In older families, the high static (idle) power consumption prohibits their use in battery-operated equipment. Modern CPLD chips such as CoolRunner **[26]** devices are nowadays a notable exception, as they offer the lowest static power consumption (less than 50 μA) of any programmable device.

FPGAs offer much higher complexity and their idle power consumption is reasonably low, although it is sharply increasing in the newest families. Since the configuration bit stream must be reloaded every time power is re-applied, design security is an issue, but the advantages and opportunities of dynamic reconfiguration, even in the end-user system, are an important advantage.

FPGAs offer more logic flexibility and more sophisticated system features than CPLDs: clock management, on-chip RAM, DSP functions, (multipliers), and even on-chip microprocessors and Multi-Gigabit Transceivers. Briefly summarizing: CPLDs has still a good future employment, especially with technologies like CoolRunner devices, for small designs, where "instant-on", fast and wide decoding, ultra-low idle power consumption, and design security are important (e.g., in battery-operated equipment); while the use of FPGAs will grow in larger and more complex designs.

5.5.3 Choice of the FPGA for Freja

The TFC project was started in 2000 by a small group of people. At that time the idea of using FPGAs as the base line device of the main board (the Readout Supervisor) still required feasibility tests. In the past this type of system was typically implemented using discrete logic due to the tight timing constraints.

A few pre-studies were done on the Readout Supervisor on using older families of programmable devices like Altera's MAX and Flex chips. The tests showed sufficient performance but since these chips are small in terms of number of gates, it was decided to go to a new family to reduce the number of chips. The final choice fell on the recently added Apex 20K family, which had many more gates and speeds comparable to the MAX and Flex chips.

Thus, since this family was already used in the TFC project, the choice of the chip for the test board was then quite natural.

5.5.4 APEX 20K Family

The APEX 20K devices **[30]** incorporate in one chip innovative features such LUT-based logic and integrated memory. Signal interconnections within APEX 20K devices (as well as to and from pins) are proved by the FrastTrack Interconnect: a series of fast, continuous row and column channels that runs the entire length and width of the device (**Figure 38**).

Each I/O pins is fed by an I/O Element (IOE) located at the end of each row and column of the FastTrack Interconnect; each IOE contains a bi-directional I/O buffer and a register that can be used as either an input or output register to feed input, output, or bi-directional signals. IOE provide a variety of features such as JTAG boundary-scan support and tri-state buffers.

APEX 20KE series (used in the design of the test board) offer enhanced I/O support including LVCMOS, LVTTL, LVPECL and LVDS **[31]**.

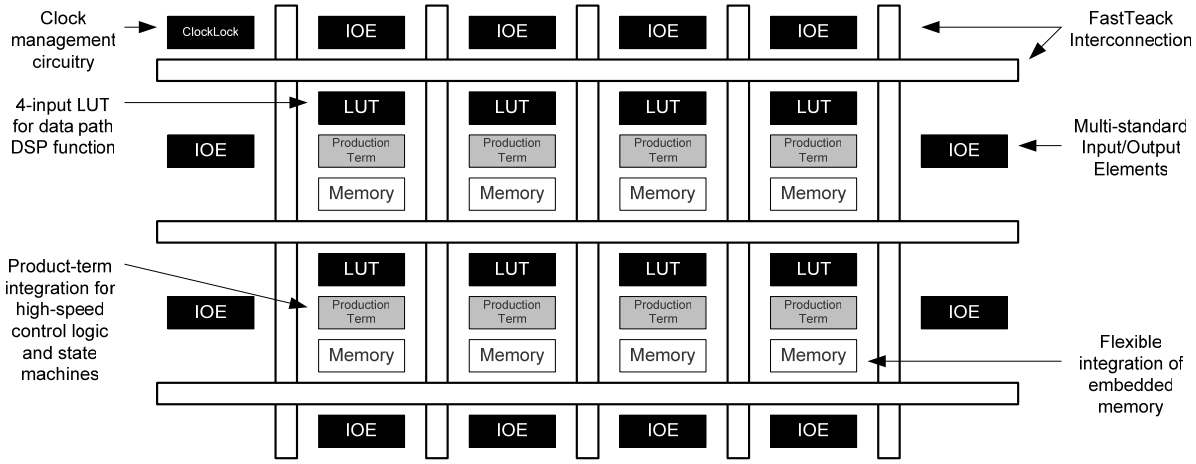


Figure 38: APEX 20K device block diagram.

A variety of memory functions can be implemented in the chip: CAM, RAM, dual-port RAM, ROM and FIFO. Embedding the memory directly into the die improves performance in terms of read and write access speed.

APEX 20K devices provide two dedicated clock pins and four dedicated input outputs pins that drive control inputs. These signals ensure efficient distribution of high-speed, low-skew control signals. These signals use dedicated routing channels to provide short delays and low skews. Four of the dedicated inputs drive four global signals; these signals can also be driven by internal logic. The dedicated clock pins featured on the APEX 20K devices can also feed logic. "X"-devices also integrate two *Phase Locked Loops (PLL)* for operations on clock signals. APEX 20KE devices provide two additional dedicated clock pins.

5.5.5 FPGA programming

Several different programming modes are implemented:

1. **Configuration EEPROM**: an EPC2 stores a basic configuration code to program the FPGA in order to make the power up sequence faster.
2. **JTAG standard programming**: a JTAG bus from the glue card to the FPGA allows its configuration directly via the ECS interface.

3. **GPIO programming lines**: JTAG over GPIO is used to reprogram the configuration EEPROM via the ECS.

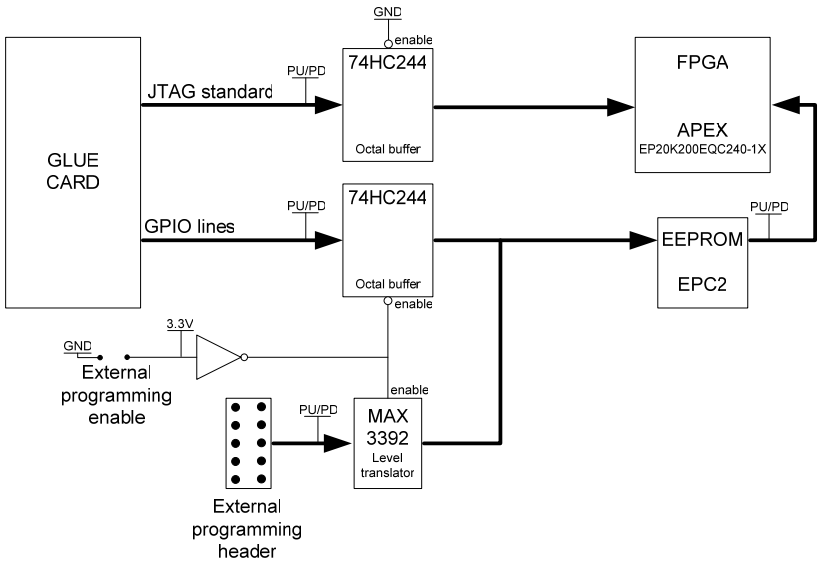


Figure 39: Diagram of the different programming options.

4. **Programming header**: a connector on board allows users to reprogram the EPC2 directly via PC using a Byte Blaster Cable provided by Altera. In normal condition this method is not used (the header can't be accessed comfortably if the board sits in the crate) it is so far disabled by default to prevent also wrong behave on the bus. If intent to use, this option has to be enabled manually via a jumper.

5.5.6 STAPL language for In System Programming

The *Standard Test And Programming Language (STAPL)* is a JEDEC standard **[32]** designed to support the programming of programmable devices and testing of electronic systems via JATG.

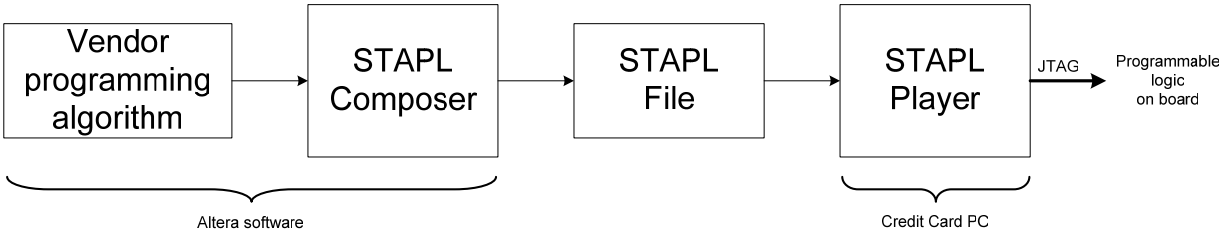


Figure 40: Flow of STAPL Composer and Player.

The use of this standard is highly recommended in *In System Programming (ISP)* designs **[33]**, such as in the TFC system, where embedded programming devices are used to program other logic on board. STAPL support the programming of any JTAG compliant programmable device.

STAPL is based on a STAPL file which is a sequence of program statements. The STAPL file contains the programming data and the programming algorithm. A single STAPL file may perform several different functions, such as programming, verifying and erasing programmable devices. STAPL file makes these different high-level functions available through ACTION statements that correspond to user controls in an interactive system.

The STAPL Composer and the STAPL Player are software programs that write and interpret STAPL files (**Figure 40**).

Using STAPL for Altera devices, a STAPL Composer (writer) is integrated in the manufacturer software (Quartus II 4.0 described in **Section 7.1.2**) replacing the original Jam Language. The STAPL composer generates the STAPL files in accordance to its standard specifications.

The STAPL Player is the interpreter that reads and executes STAPL files. STAPL may be implemented as either an interpreted or a compiled language. In an interpreted implementation, the STAPL file statements are executed directly without first compiling these statements into binary executable code. In a compiled implementation, the STAPL file statements are first pre-processed and then executed. In the TFC board a STAPL player is integrated in the CCPC which executes binary code STAPL files compiled with Quartus II 4.0.

When a STAPL player executes a STAPL file, only one ACTION will be executed. Because a STAPL file may contain multiple ACTION statements, execution of a STAPL file begins by searching the STAPL file for the ACTION statement whose name matches the one specified by the user. Execution continues with calls to the list of PROCEDURE blocks listed in the ACTION statement. Execution terminates either when the end of the ACTION statement is reached or when an EXIT statement is processed.

The flow of execution within each of the called PROCEDURE block is controlled using three methods:

- *Branches:* a GOTO statement is used to jump into the file. Using IF statements in combination with GOTO to create conditional branches. The branching technique improves the programming times.
- *Calls:* the CALL statement causes the execution to jump to a PROCEDURE and the location of the CALL statement is saved in a stack. An IF statement can be used with the CALL to call a subroutine conditionally.
- *FOR loops:* the FOR statement is used for iteration. Combined with compressed byte code this feature ensures very small files, which suit the requirement of programming memory constrained designs.

6 PCB DESIGN

6.1 Introduction

PCB (Printed Circuit Board) design is a very delicate part of the process to realize a board.

Years of developments in all fields of production (board design, manufacturing, mounting and testing) led the technology to extreme edges: boards manufactured at CERN can reach 16 layers, 100µm nets and clearance, 300µm vias and other impressive specification and tolerances.

9U VME standard boards **[34]** are used for all the TFC sub-parts almost exclusively SMD (Surface Mount Devices) ICs are implemented in designs using the latest SMD packages (like TQFP, SSOP, TSSOP, etc) avoiding the use of more delicate packages such as Ball Grid Arrays (BGA). The fact that the I/O needs on the TFC boards permitted the use of Quad Flat Packs (QFP) also render routing and hardware debugging easier. Still, even SMD components mounting needs to follow a precise and specific procedure, special tools (automatic placing machines, reflow ovens) to guarantee proper contact and fixation with high precision between pads and pins.

Some sub-parts are implemented as mezzanines, like for example the Glue Card or the TTC modules. The main advantage of using this kind of solution is that a mezzanine can be easily replaced by new corrected or improved versions. The disadvantage is that signals pass through connectors which can impair high speed signals and induce noise.

All the TFC board are designed using Protel 99SE and its updated version DXP. Protel is a complete CAD tool that allows the user to fully manage the design from the schematic drawings to the gerber files, passing trough board routing (both manual and automatic), creation of libraries for custom footprints, schematics components, etc.

6.2 Board schematics

In **APPENDIX A - Board Schematics** the board schematics are shown. As a general aim the same type of components are used on the different TFC boards to the maximum extent possible. The goal using similar or common solutions is to reduce the probability of mistake during the design phase. Using well known devices eases the implementation, allows the use of common design "tips and tricks" as well as common hardware debugging techniques. All the components used in the TFC project are kept in a specific TFC Protel library.

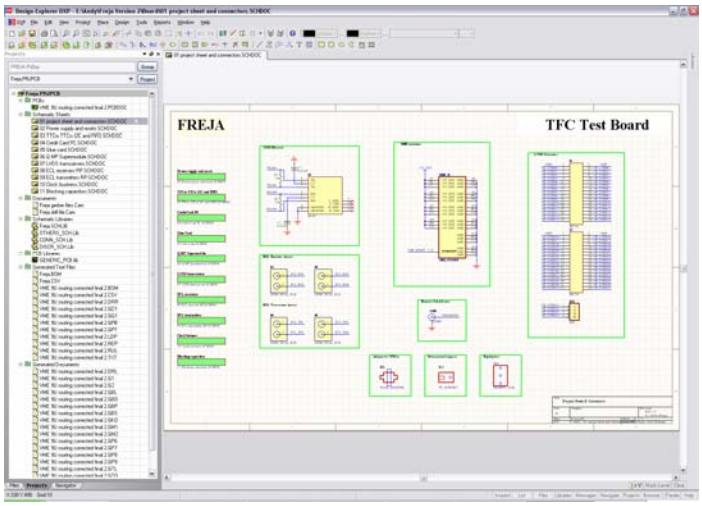
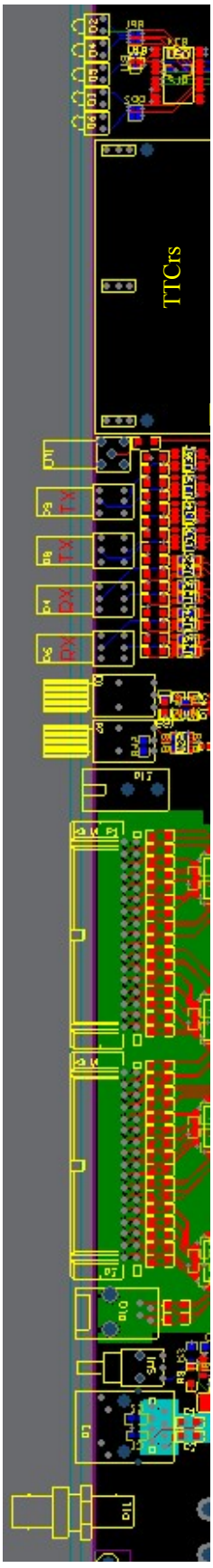


Figure 41: Screen shot of the working enviroment of Protel DXP.

6.3 Front panel view



Monitoring LEDs (top to bottom):

1. Power: GREEN when board is on
   Power monitoring: blinking RED when voltage is out of range (4.7÷5.3 V)
2. TTCrx clock is present and PLL is locked goes GREEN
3. General purpose RED LED from FPGA
4. General purpose GREEN LED form FPGA
5. Configuration GEEN LED when FPGA is programmed

TTCrs mezzanine:

1. Bunch clock input
2. Bunch clock output 1
3. Bunch clock output 2
4. TTC channel A+B output
5. TTC channel A output
6. TTC channel B output

Board external clock input

ECL transmitter* (dual lemo connector):
TX3 - TX4
TX1 - TX2 *seen from the front, lower value is on the left

ECL receiver* (dual lemo connector):
RX1 - RX2
RX3 - RX4 *seen from the front, lower value is on the left

Optical TRANSMITTER

Optical RECEIVER

Front panel support

LVDS transceivers*:

1. Connector 1 (3M 17 pair connector):
   Pair 16: Ground
   Pairs 0 to 15: Signals
2. Connector 2 (3M 17 pair connector):
   Pair 16: Ground
   Pairs 0 to 15: Signals
3. Connector 3 (RJ9):
   Pairs 0 to 15: Signals

The LVDS lines are driven by 4 transceiver chips (DS92LV090A) with 9 channels each.

*seen from the front, negative pin is on the left

Credit Card PC reset push button

Credit Card PC Ethernet RJ45 connector

TTCrx optical input bulk adapter

6.4 Board layer configuration

A carefully studied layer configuration prevents noise injection into signals due to power planes and provides EMI shielding, ground coupling and even a better mechanical rigidity.

The first studies on the board structure were done with 6 layers. However, in order to separate routing and power planes to avoid noise injection on the signal nets due to regulators and dc/dc converters, it was decided to use an 8 layers board with two uniform ground plains separating 3.3V and 5V planes from the others (**Figure 42**).

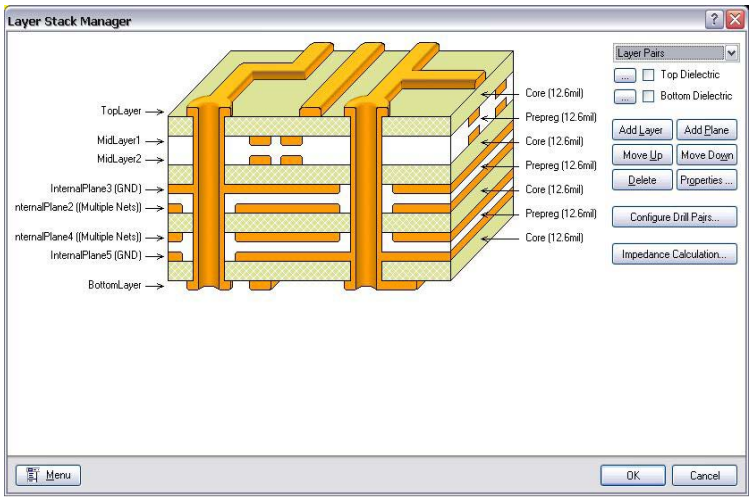


Figure 42: Layer stack manager (Protel DXP screen shot).

The 5V supply is provided directly from the crate back plane which means that there is no particular attention to pay for what concerns the 5V current consumption. On the other hand 3.3V is supplied by voltage regulators on board. Thus it's important to compute carefully the 3.3V overall consumption to avoid overloading and overheating the regulators. The FPGA with its PLLs requires a very clean and stable voltage. It is therefore important to use separate regulators for this device. Two supply planes, 3.3V for IO and 1.8V core logic, have been implemented through cut-outs on the power planes (**Figure 43** and **Figure 44**).

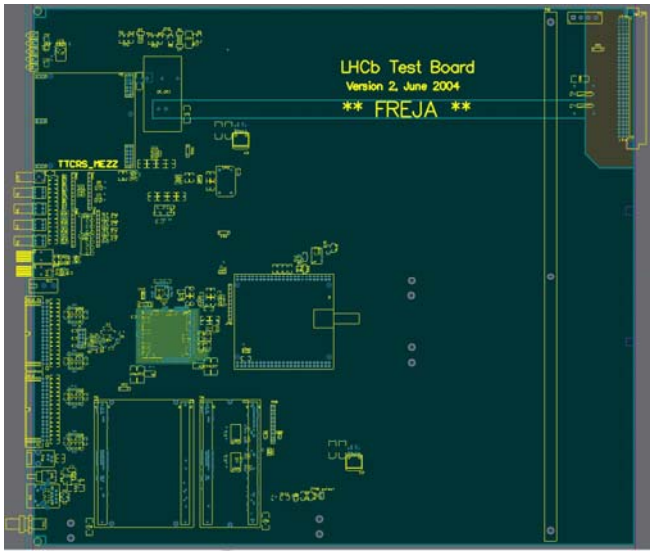


Figure 43: Internal 5 V power plane (Protel DXP screen shot).

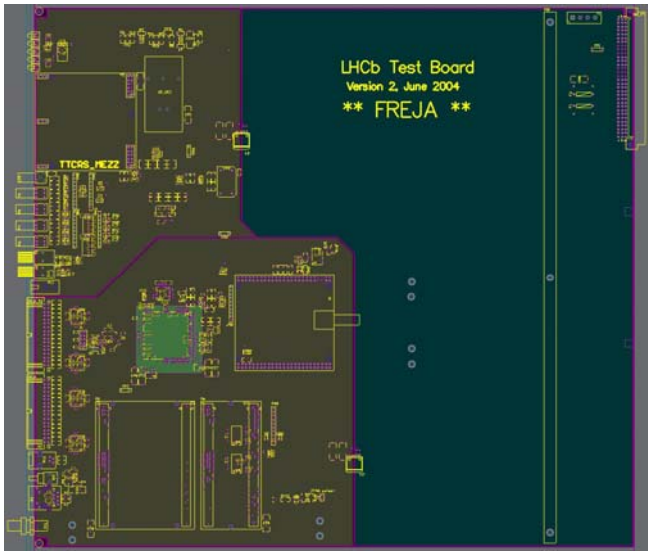


Figure 44: Internal 3.3 V power plane (Protel DXP screen shot).

High speed differential lines, like LVDS, need particular attention during the routing stage to avoid crosstalk and to optimize ground coupling. The design rules as defined by the chip manufacturer include the distance between the nets and the ground plane and the type of dielectric in between (**Figure 45**).

- S < 2W
- S < h
- x ≥ 2S (distance between pairs)
- x ≥ 2W
- x >> 2S (between different voltage supplied pairs)

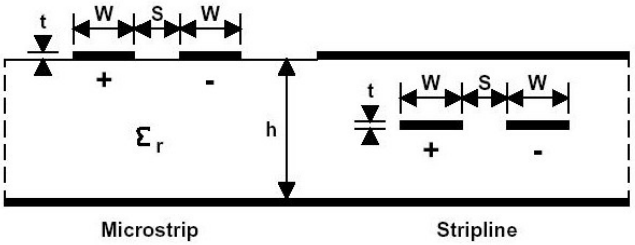
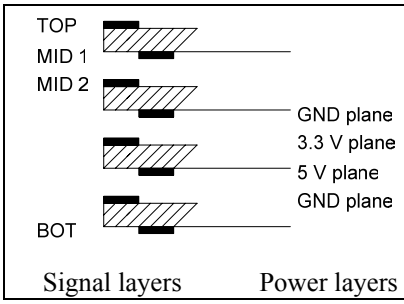


Figure 45: Rules for the calculation of line thickness and clearance.

The Microstrip solution is adopted for all the differential lines since they are routed on the top layer of the board. A polygon ground plane has been added underneath to adjust the ground coupling. The dielectric constant is specified (**Table 14**) by the CERN manufacture atelier (ε_r=4.4 ÷ 4.8). The tolerance is due to the fact that the thickness of the glue between the two layers can't be controlled to a precision higher than 185±85µm.

Table 14: board specifications given by the CERN PCB work shop.



Layer	Thickness
core	200 µm
glue	185 ± 85 µm
core	500 µm
glue	2 x 185 µm
core	500 µm
glue	185 ± 85 µm
core	200 µm

As seen before, the terminations on the ECL differential lines are essential for them to work. The use of SMD resistor such as the 1206 packages for that purpose can sometimes represent a problem. Even the smallest packages occupy a considerable amount of space considering that the differential line distance must be kept constant and that the line length should be short. Vias also introduce noise and length differences between the pairs so it's not recommended either to put termination resistors on the other face of the board. For those reasons Single In Package (SIP) Thevenin termination networks were chosen and implemented in the design.

6.5 Routing tips and techniques

Once the layer stack is defined, the next step is the effective placing and routing of the components. First thing to decide is the dimension of the board which is directly related to the front panel size. Since the test board has many IO ports, a 9U was needed. This was also the natural choice since all TFC boards are based on 9U.

As mentioned before the FPGA is the central device. Thus it is important to keep it in a central position on the board and surround it with all the other devices paying attention to routing direction and the most delicate nets (like clock lines). Placement of components on the bottom layer is limited by the tight height limit (less than 1 mm). It's recommended just to place very thin components on the bottom (0805, 1206, 0603 or SOT only).

The routing was done completely manually using the DRC to check clearance and other design specifications and rules.

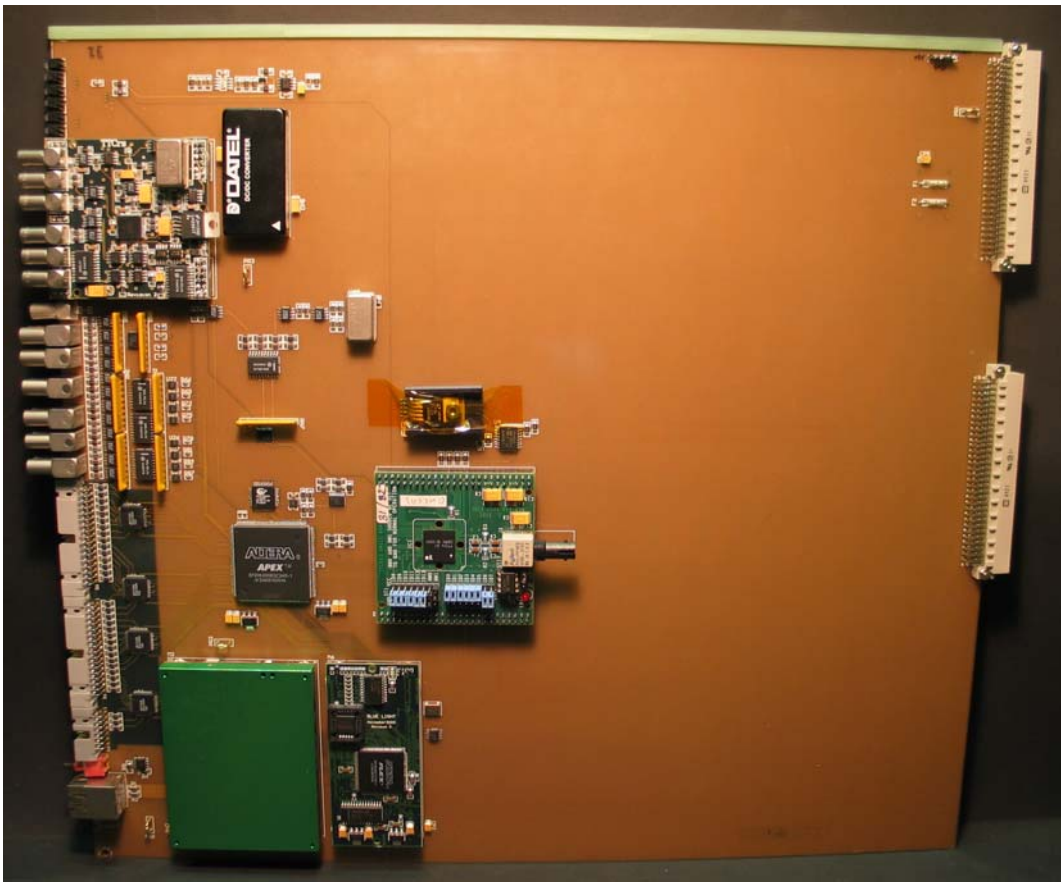


Figure 46: Picture of Freja Version 1.

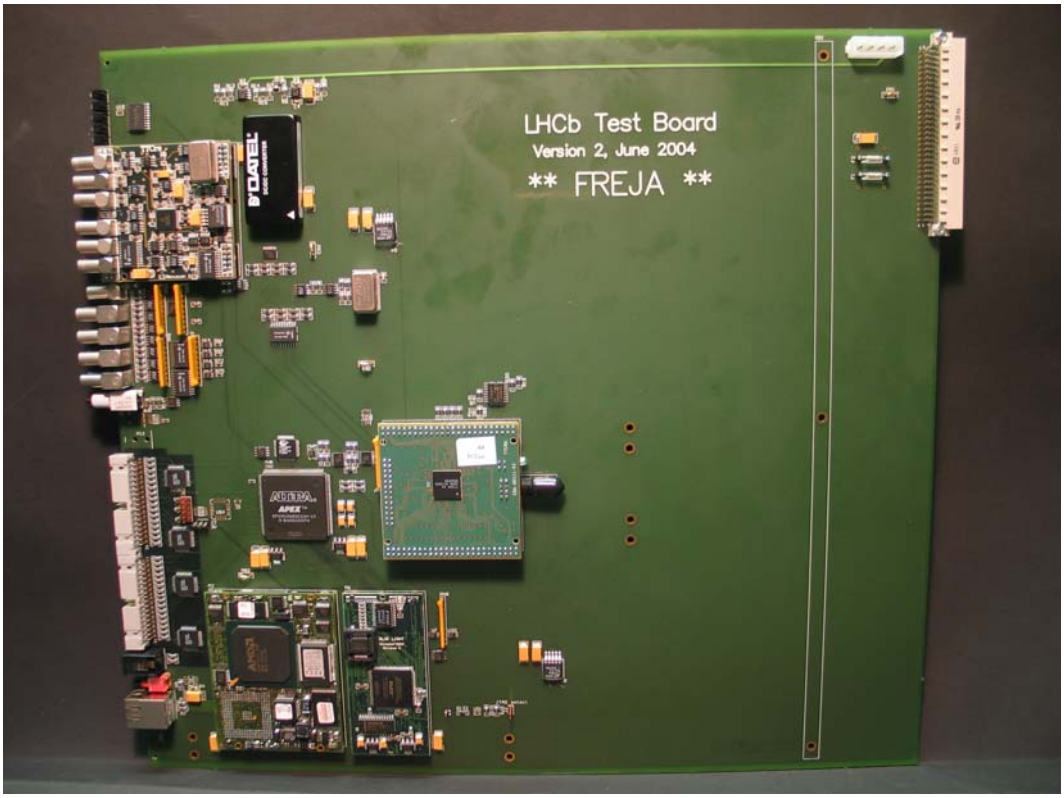


Figure 47: Picture of Freja Version 2.

7 VHDL PROGRAMMING

7.1 Introduction to VHDL

VHDL stands for *VHSIC (Very High Speed Integrated Circuit) Hardware Description Language*. Beside the complicated acronym VHDL is a language to describe the structure and behaviour of digital electronic in hardware design (**Figure 48**).

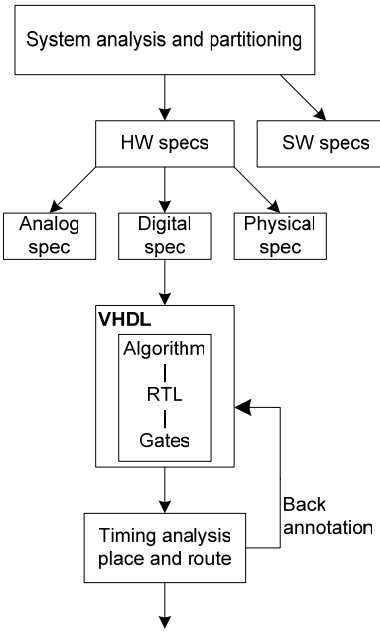


Figure 48: VHDL integration into hardware design flow.

VHDL for analog electronics is not very well developed, although, because of the language flexibility it has been stretched to handle analog and switch level simulation in limited cases.

VHDL can be used to describe digital electronics hardware at many different levels of abstraction (**Figure 49**). When considering the application of VHDL to ASIC or FPGA design it's important to understand the three levels of abstraction listed below:

1. **Algorithm:** a pure algorithm consists of a set of instructions that are executed in sequence to perform some task. A pure algorithm has neither a clock nor detailed delays. Some aspects of timing can be extracted from partial ordering of operations within the algorithm.
2. **RTL (Register Transfer Level):** this description has an explicit clock. All operations have been scheduled to occur in specific clock cycles, but there are no detailed delays below the cycle level.
3. **Gate:** a gate level description consists of a network of gates and registers instanced from a technology library, which contains technology-specific delay information for each gate.

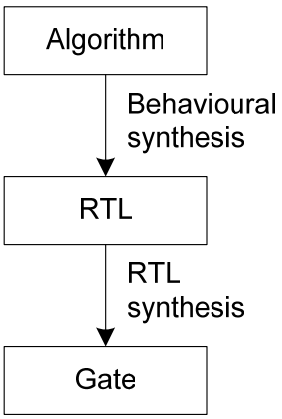


Figure 49: VHDL level of abstraction schema.

VHDL is a high level programming language completely specified in two Language Reference Manuals (IEEE 1076 **[36]** and IEEE 1164). The first defines the language while the second specifies standard data types in order to describe logical components and other crucial aspects. Being a standard set apart VHDL from other hardware description languages, which are to some extent defined in an ad hoc way by the specificity of the tools that use them (like for example the AHDL language by Altera).

7.1.1 Language basic elements

Like any high level programming language, VHDL allows to split an algorithm in several small pieces to make the problem easier.

The outer most block is the *entity* which represents a block of hardware with well defined inputs and outputs and a well defined function. The main entity in the case of the TFC test board is the whole FPGA but it can also be a PCB, an ASIC, a microprocessor and so on. A design entity is split in two parts:

- The *entity declaration* which represents the external interface of the design entity.
- The *architecture body* which represents the internal description of the design entity (behaviour, structure or both).

Inside the architecture body, signals for both internal and external use, are used to connect components: a component (self designed or picked up from libraries) can be "plugged" in a design by making three operations:

- *Declaration:* is the operation that declares a component, whose function is defined elsewhere, inside an architecture.
- *Instantiation:* is a local copy of the corresponding design entity to make use of a component; in other terms making an instantiation is like plugging a chip in a board socket.
- *Port mapping:* links the ports of the component to the signals of the logic in which it is instantiated. In other words, a port map makes an electrical connection between "pieces of wire" (signals) in an architecture and pins on a component (ports). The same signal may be associated with several ports.

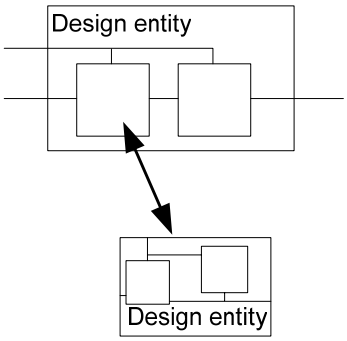


Figure 50: VHDL allows splitting a problem in several sub-problems in a nested manner.

7.1.2 RTL-to-gate process

Once the VHDL code is written, the next sequence of steps transform the algorithm in an optimized way into an implementation which is based on the primitive gates of the FPGA which has been selected.

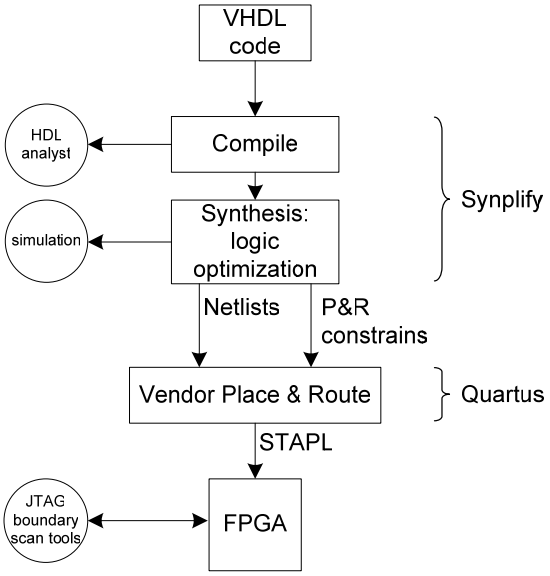


Figure 51: VHDL code synthesis flow.

The first step is to compile the VHDL source code with a logic synthesis tool that makes the optimization of the implemented functions (state machines, memories, etc) at a high-level, such as Synplify (**Figure 51**).

In a project all the piece of code written are linked. By selecting a clock signal the program will make a first attempt to satisfy the timing constrains. Selecting the FPGA used a rough approximation of the inside chip occupation will be done.

Synplify for example uses a proprietary Behaviour Extracting Synthesis Technology (B.E.S.T.) which converts the HDL into small, high-performance, design netlists that are optimized for popular technology vendors.

With the new Altera FPGAs, *VQM (Verilog Quartus Mapping)* as output files should be used to transfer the RTL file to the placer and router: this standard was developed by Synplicity together with Altera in order to make the first steps of compilation more efficient. For the latest families like Apex, Cyclone and Stratix, Altera suggests to only use this format to produce synthesis files. Optionally, the software can write post-synthesis VHDL and Verilog netlists that can be used to verify functionalities through simulation.

Once the synthesis is done the phase of placing and routing starts: since the internal gate structure is known just by the manufacturer, placing and routing has to be done using proprietary software. Altera Quartus II 4.0 is used for all TFC boards (**Figure 53**).

Since the FPGA dimension (in terms of gates) were well defined during the design phase and the timing delays after a few test were considered fine, there was no need to go into the details of the placing. It is important to keep in mind that with Quartus it's possible to manually place and route the logic inside the chip.

Once the code is placed and routed the last part of the job consists in creating the file that will be loaded in the chip. Quartus can produce several types of output files. *STAPL (Standard Test and Programming Language)* the high level language described in **Section 5.5.5** is used in the TFC system for In System Programming (ISP).

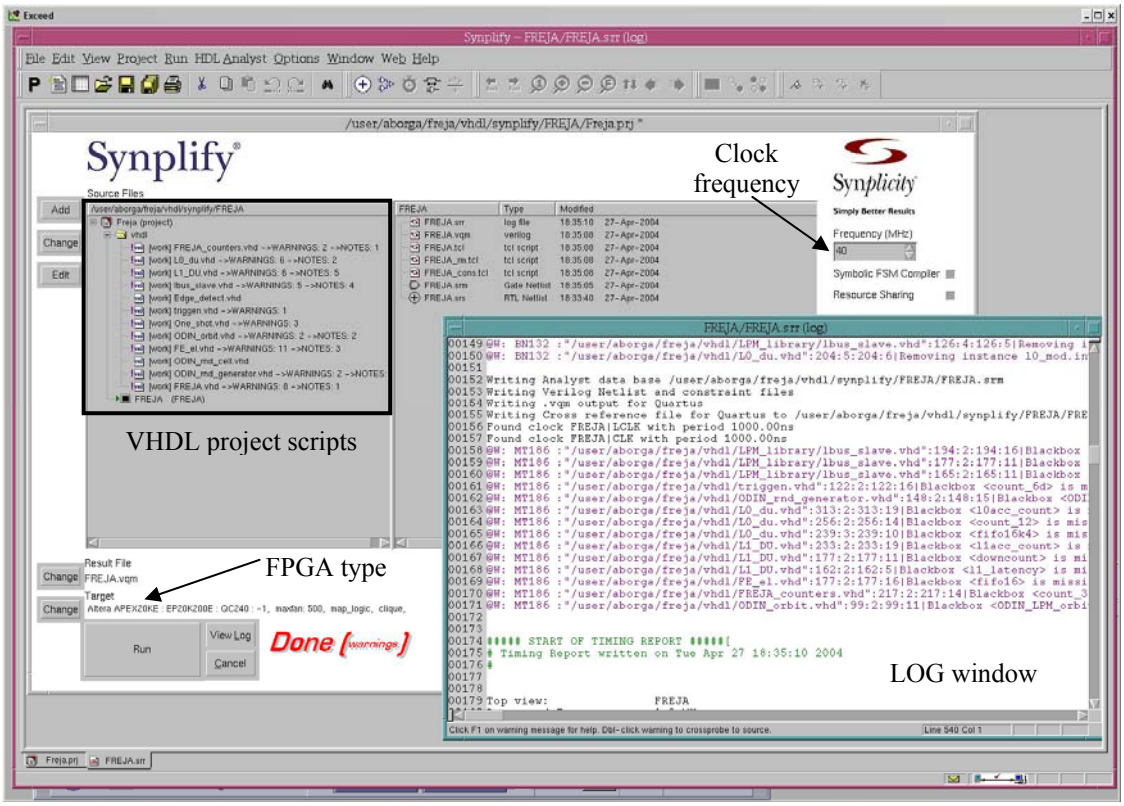


Figure 52: A screen shot of the Synthesis tool, Synplify.

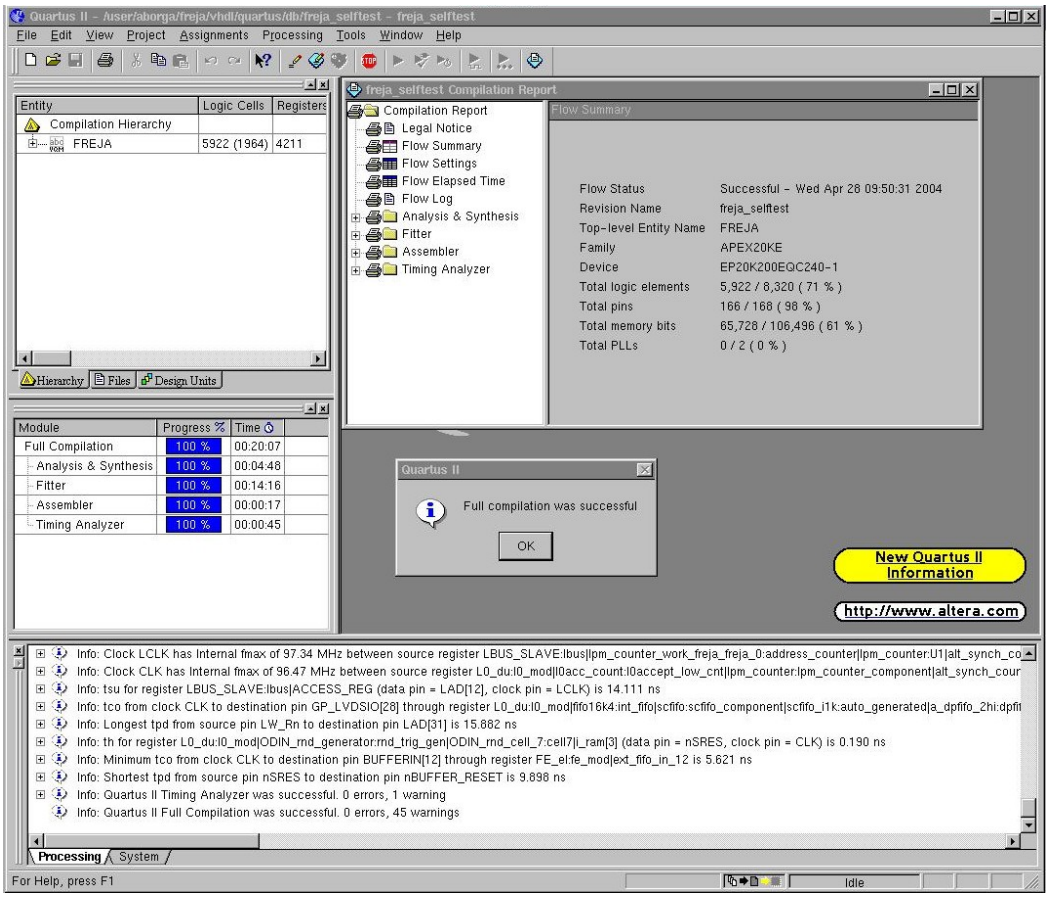


Figure 53: A screen shot of the placer and router, Quartus II 4.0.

7.2 VHDL code for TFC test board

In the next sections the code written for the FPGA in the test board is described in details. In summary three applications were developed:

1. **Self-test code [Section 7.2.1]:** small test routines to debug the hardware of the test board.
2. **Feasibility test code [Section 7.2.2]:** to test the capabilities of modern test FPGAs some feasibility test code has been written.
3. **TFC system testing code [Section 7.2.3]:** code written to accomplish the main task of the board. For the time being the code is sufficiently developed. Future improvement on the TFC board might require code expansions.

7.2.1 Board debug: SELF-TESTS

In order to discover and fix bugs efficiently it's important to approach the board with a good debugging scheme. Simple code was written to self-test all the different hardware functions of the board. Hereafter is listed the strategy used for Freja:

1. ECS interface test: starting from the access to the credit card PC via Ethernet up to the Glue Card via PCI.
2. Peripherals connected directly to the ECS interface: debug of all the buses (I^2C, JTAG, Local BUS) using low level C routines to read and write hardware registers.
3. Clock distribution to on board devices. Measurements of signal quality and propagation with oscilloscope.
4. Input/output interfaces (ECL and LVDS) debugging both with lab instruments and driving them from the FPGA using dedicated code.

7.2.2 Alternative TFC switch implementation

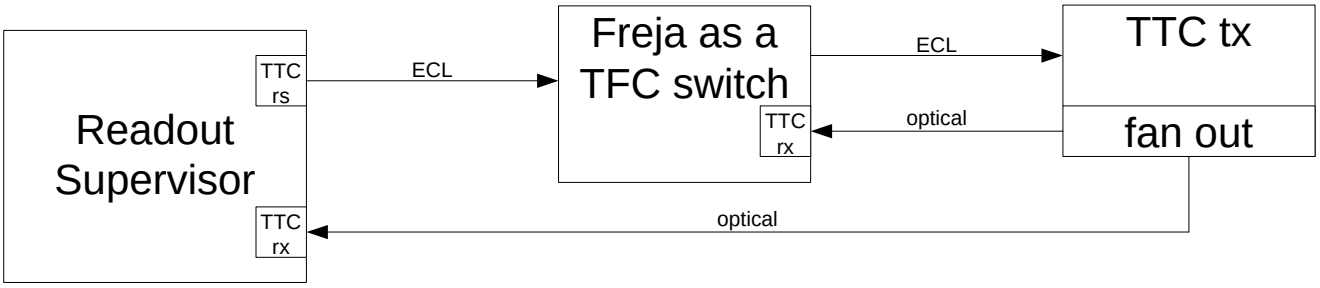


Figure 54: A diagram of the configuration for the TFC switch emulation test.

As widely underlined in the previous chapters the use of FPGA in electronics boards is increasing rapidly. New families of devices are more and more fast and reliable from the point of view of timing (propagation delay, skew, jitters, etc).

The TFC Switch is currently implemented in fast discrete logic based on ECL in order to satisfy the strict requirements on the timing characteristics of the switch. The application requires that the skew between any combinations of inputs-outputs is no more than 50 ps. In addition, the TFC Switch must not introduce more jitter than 50ps. However, for several reasons it would be advantageous to implement the TFC Switch function in an FPGA. It would allow building easily a bigger switch than 16x16, it would simplify the implementation and reduce significantly the power consumption.

A feasibility test was done using the FPGA on the test board (**Figure 54**): a small piece of code was written in order to implement the functionalities of the TFC switch. Four 4:1 multiplexers provide the full combinational logic of a 4x4 switch to make the test.

The results of the measurements revealed rather astonishingly that with the actual technology available it is possible to build an equivalent of the TFC switch using programmable devices only. However, it would probably demand some manual placing and routing of the logic in the FPGA to minimize the differences in propagation delay.

The information gathered by this experience will be kept in mind for a future upgrade of the board, in particular if the partitioning has to be improved such that the switch must support 32 destinations.

7.2.3 TFC full system testing

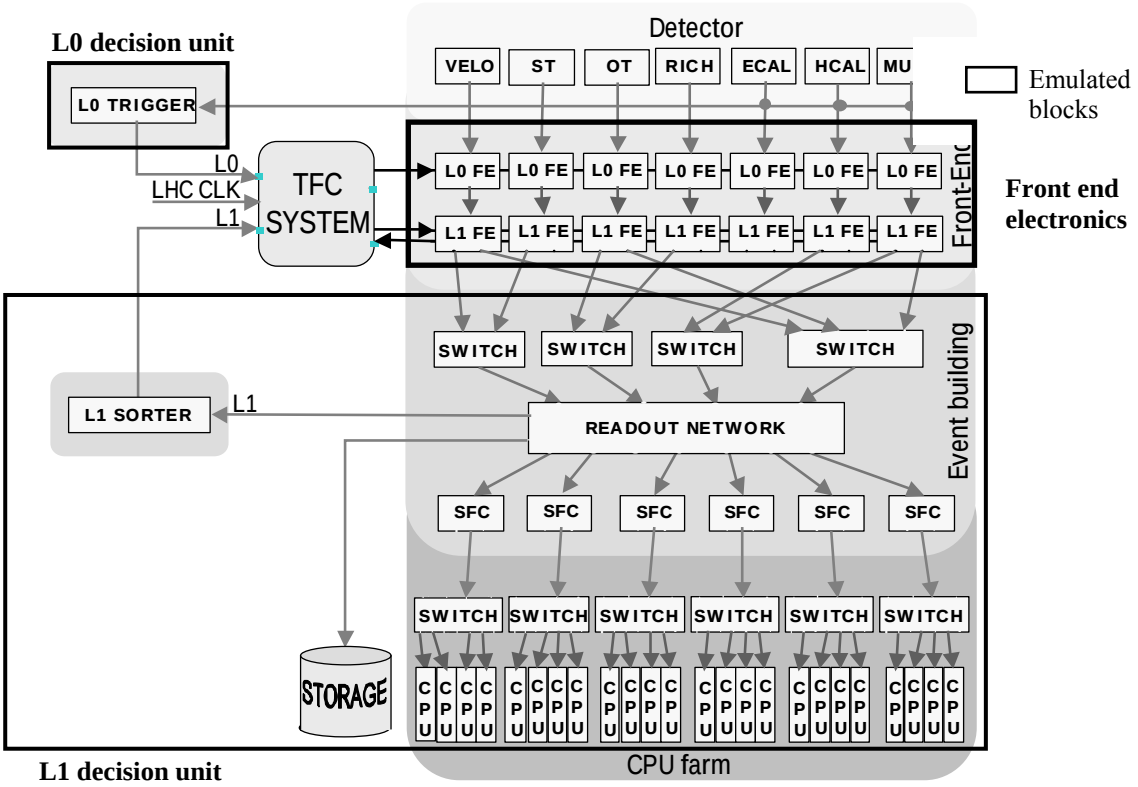


Figure 55: Emulated modules of the LHCb readout system.

In **Section 3.1** an overview of the LHCb readout system has been given. In order to test the full TFC system the test board needs to emulate three different LHCb sub-parts (**Figure 55**):

1. **L0 Decision Unit**: in order to test the L0 trigger path, the test board should generate L0 triggers by emulating the L0 Decision Unit. The L0 Decision Unit emulator in Freja includes a random trigger and a periodic trigger generator. It also has the Bunch ID counter which produces the 12 bit trigger ID used to check the synchronization of the trigger in the Readout Supervisor. Freja transmits the trigger words to the Readout Supervisor via LVDS lines (L0 trigger path).

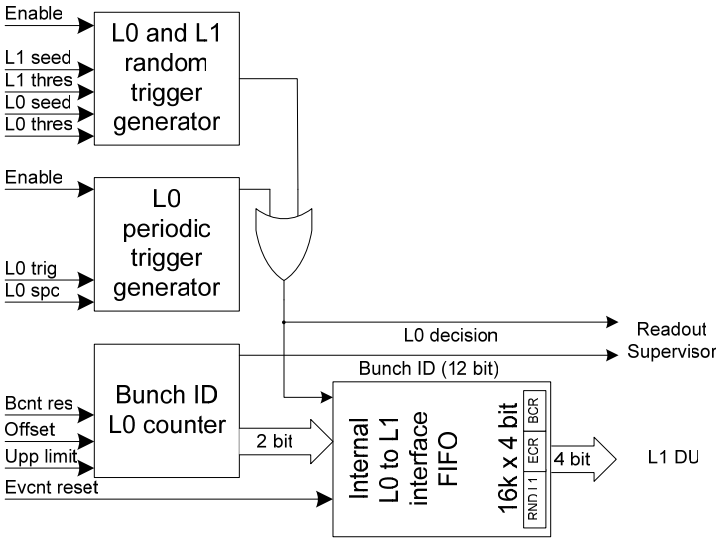


Figure 56: Block diagram of emulated L0 DU.

2. **L1 Decision Unit**: in order to test the L1 path in the TFC system, Freja should also emulate the function of the part of the computer farm responsible for L1 decision taking. The "L1 Decision Unit" is directly interfaced with the L0 Decision Unit emulation in order to produce L1 decisions for every L0 triggers accept. The unit incorporates a random and a periodic trigger generator. The L1 Decision Unit also has an Event ID counter (12 bits) in order to transmit the L1 trigger ID which allows the Readout Supervisor to check the trigger synchronization. In order to emulate properly the timing and the operations, L1 Decision Unit is controlled by a state machine. Two bits of L0 bunch ID are received for each L0 trigger accept from the L0 Decision Unit. The decision words are transmitted to the Readout Supervisor via the optional LVDS interface on the Readout Supervisor (L1 trigger path).

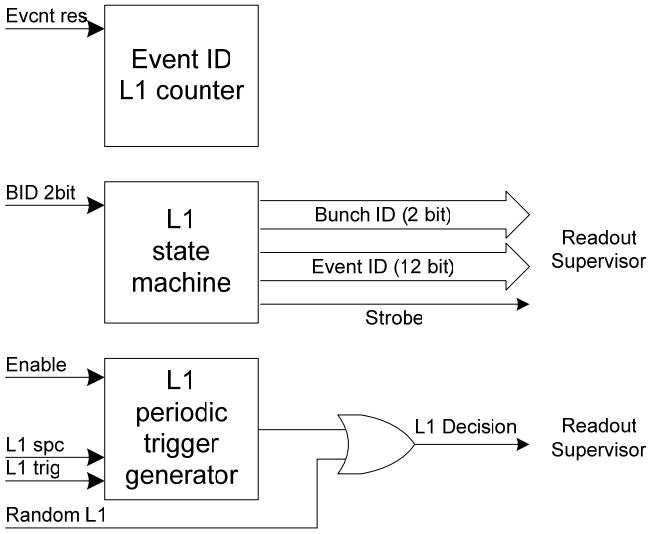


Figure 57: Block diagram of emulated L1 DU.

3. **L0 and L1 Front End electronics**: In order to check the operation of the full TFC system, Freja should also emulate the L0 and the L1 Front End electronics. The Front End electronics store the event data temporarily while the decisions are taken, and reject event data according to the trigger decisions. Thus the contents of the Front End buffers at different stages are a function of the trigger decisions and the operation of the TFC system. By emulating the Front End electronics in addition to the trigger decision units, Freja can verify the correct functioning of the TFC system. The front end emulation generates Event IDs and Bunch IDs in the place of event data. The IDs are derandomized, buffered and selected by the trigger decisions exactly like in the standard front end up to the L1 buffer.

The IDs are readout from the L1 buffer at every L1 decision sent from the Readout Supervisor and compared with the 2 bit of L1 Event ID transmitted with the L1 decision for error checking. In total the Front End emulator consists of an Event ID (2 bits) and a Bunch ID (12 bits) counter, a 160 deep pipeline, a 16 deep derandomizer and output links to the external FIFO acting as a L1 buffer.

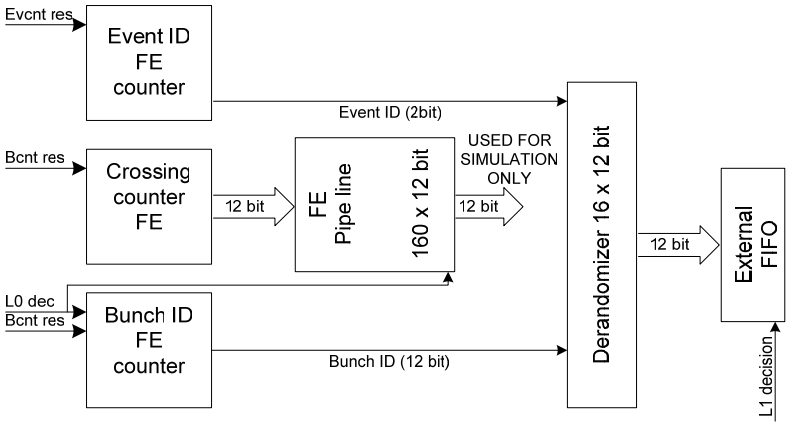


Figure 58: Block diagram of emulated FE EL.

Each sub-module has fine tuning settings and delay lines for time alignment to ensure a precise synchronization and a correct emulation of the real system.

The code is organized in several files to increase readability and modularity. A scheme of the structure follows (**Figure 59**):

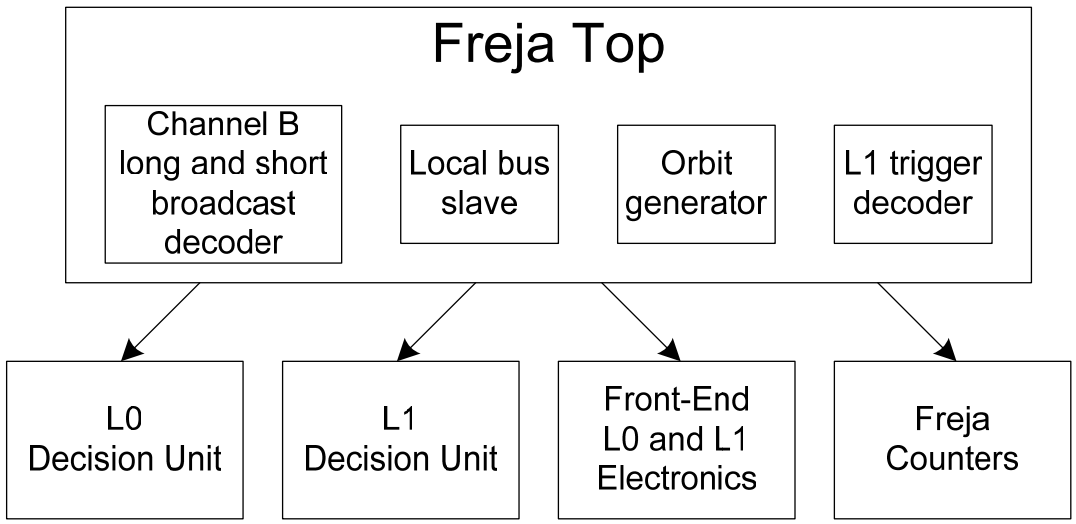


Figure 59: VHDL code structure of Freja.

A top architecture (FREJA) incorporates the module which is responsible for handling the bus between the FPGA and the ECS interface, the local bus slave (lbus_slave). The top architecture also instantiate the different sub-parts (L0_DU, L1_DU and FE_EL) and a counter module (FREJA_counters) in which most of the system's general counters are located. A few special counters are placed directly inside the modules for convenience.

Two more modules are implemented for testing purposes: a channel B long and short broadcast decoder and an orbit signal generator (ODIN_orbit). The purpose of these modules is the monitoring of the TTC system.

During the first testing phase it was discovered that some of the optical devices were affected by jitter and skew under heavy load conditions. These test sequences provides the tools to monitor permanently the optical system.

Due to the extreme modularity of the VHDL language the structure can be easily modified, reorganized, upgraded or stripped for future application. Single purpose modules (like the orbit generator for example) can be cut and paste in and from another codes with sufficient simplicity.

The list below gives an idea of the structure from a different point of view:

FREJA	[FREJA.vhd]
1. Local bus	[lbus_slave.vhd]
2. Orbit generator	[ODIN_orbit.vhd]
• One shot	[One_shot.vhd]
• Edge detect	[Edge_detect.vhd]
3. L0 Decision Unit	[L0_du.vhd]
a. L0 periodic trigger generator	[triggen.vhd]
• Edge detect	[Edge_detect.vhd]
b. Bunch ID counter	[count_12.vhd]
c. L0 accept counter	[l0acc_count.vhd]
d. L0 and L1 random generator	[ODIN_rnd_generator.vhd]
• Random cell	[ODIN_rnd_cell.vhd]
e. Internal interface FIFO	[fifo16k4.vhd]
4. L1 Decision Unit	[L1_DU.vhd]
a. Latency counter	[L1_LATENCY.vhd]
b. Read out counter	[downcount.vhd]
c. L1 periodic trigger generator	[l1_yn_cnt.vhd]
d. Event ID counter	[count_12basic.vhd]
e. L1 accept counter	[l1acc_count.vhd]
5. Front End Electronics	[FE_el.vhd]
a. Bunch ID counter	[count_12basic.vhd]
b. Crossing ID counter	[count_12basic.vhd]
c. Event ID counter	[count_12basic.vhd]
d. Synchronization counter	[count_32.vhd]
e. Derandomizer	[fifo16.vhd]
f. Read out counter	[downcount.vhd]
6. CH B long and short broadcast decoder	
7. L1 trigger decoder	
8. Counters	[FREJA_counters.vhd]
• Bcntres	[count_32.vhd]
• Evcntres	[count_32.vhd]
• All commands	[count_32.vhd]
• All triggers	[count_32.vhd]
• L1 destinations	[count_32.vhd]
• L1 MEP flush counter	[count_32.vhd]
• HLT destinations	[count_32.vhd]
• HLT MEP flush counter	[count_32.vhd]

7.3 Register list

All the functions on board are accessed via readings and writings in registers inside the FPGA. The registers can both be accessed "manually" using low level C routines via a remote connection (usually using telnet) on the CCPC; or via the user interface (DIM plus PVSS) described in **Section 0**. The list of registers of the test board with all the information and comments is reported in **APPENDIX B - Registers list**.

7.4 Code simulation

When the code grows bigger, finding bugs via hardware measurements and intuition might become a nightmare… for that reason software simulation frameworks were developed achieving nowadays impressive results.

The TFC full system emulation has been implemented at clock level in Visual Elite 3.5.1 and then simulated (**Figure 60** and **Figure 61**).

Simulation eases life of designers during the phase of time alignment and synchronization of signals: by simply plotting a set of signals over a certain time period it's possible to find the exact delay (in term of clock cycles) between them and find out which is the best delay compensation setting to implement (**Figure 62**). It's also easer to find out hiccups in the functions following the signal flow (source to destination or vice versa).

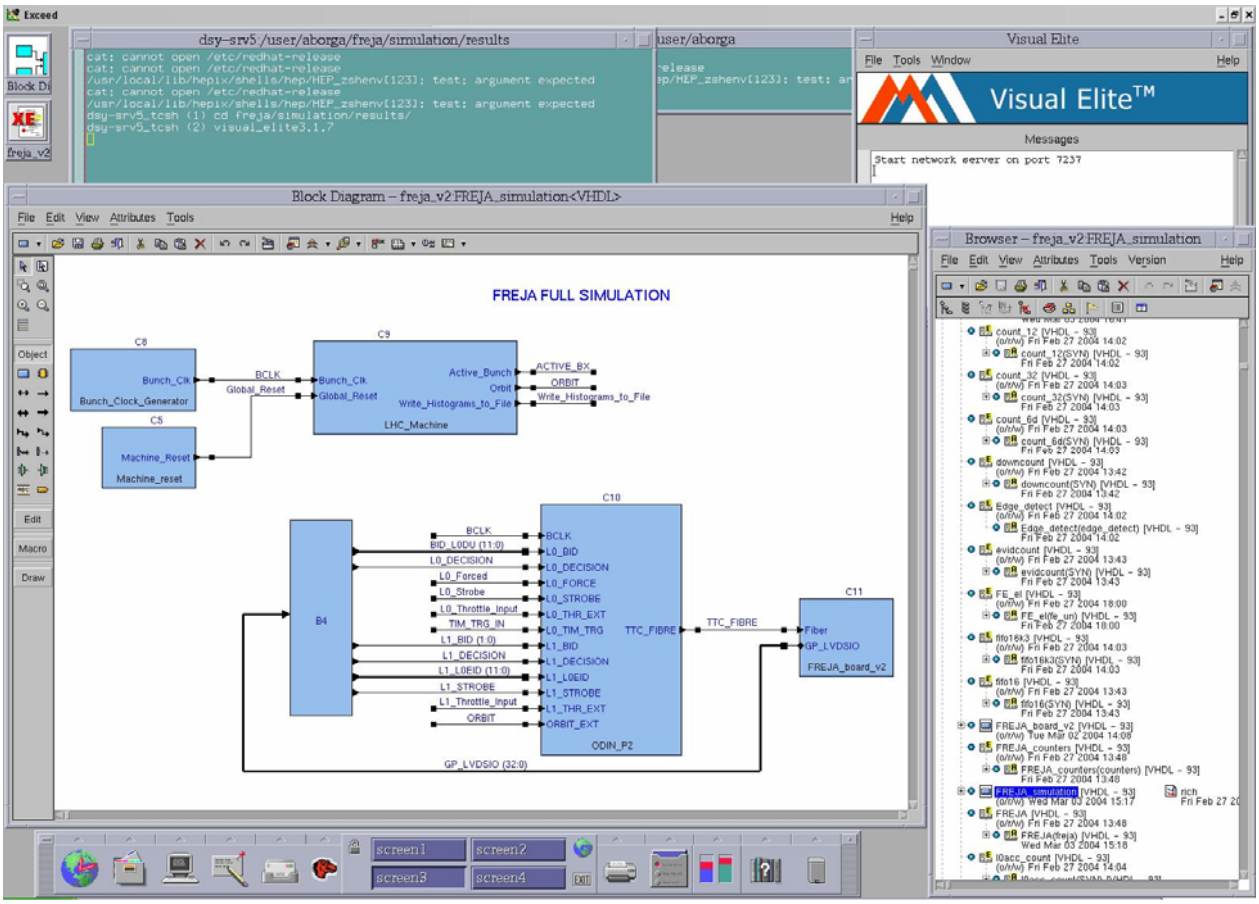


Figure 60: A screen shot of the working environment in Visual Elite 3.5.1.

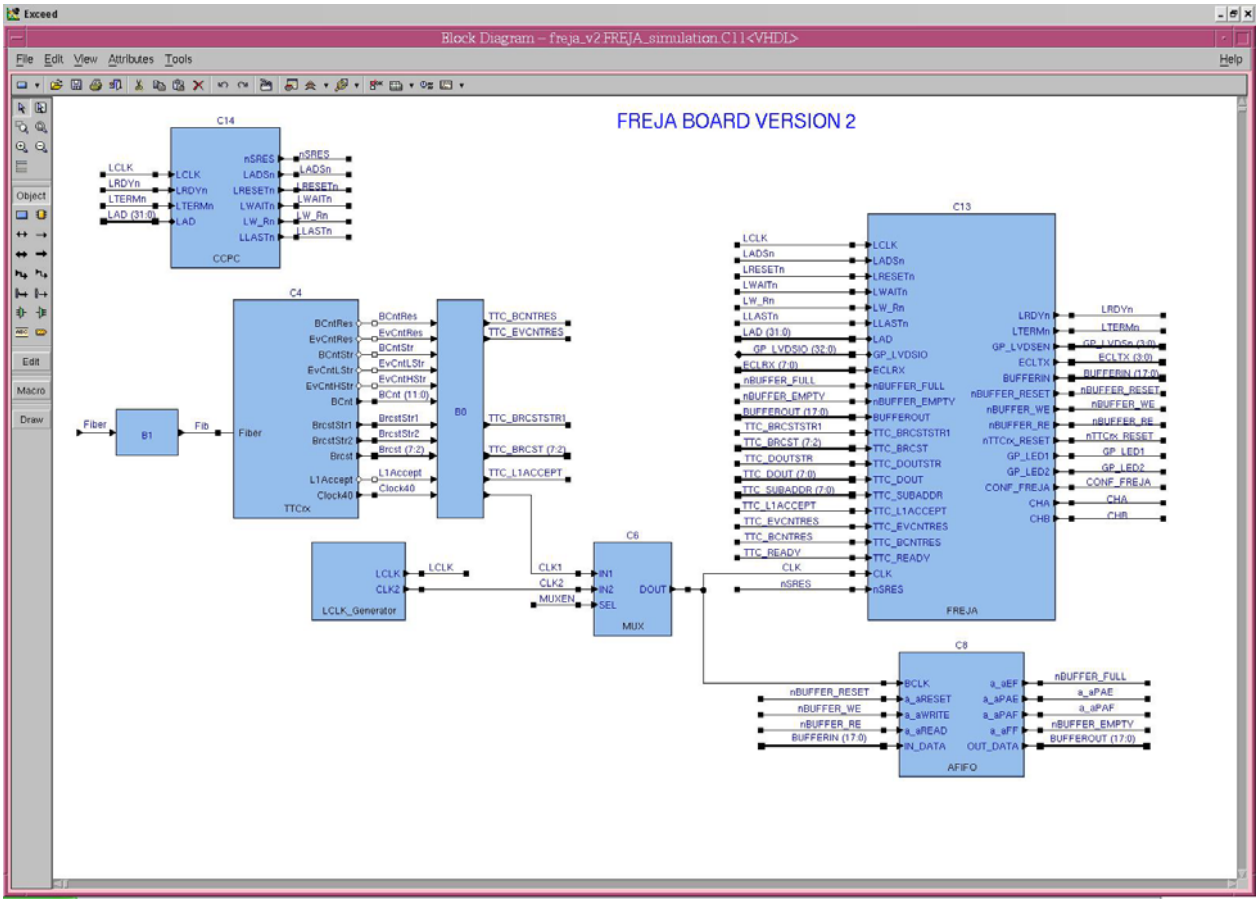


Figure 61: A screen shot of the construction of a simulated board in Visual Elite 3.5.1.

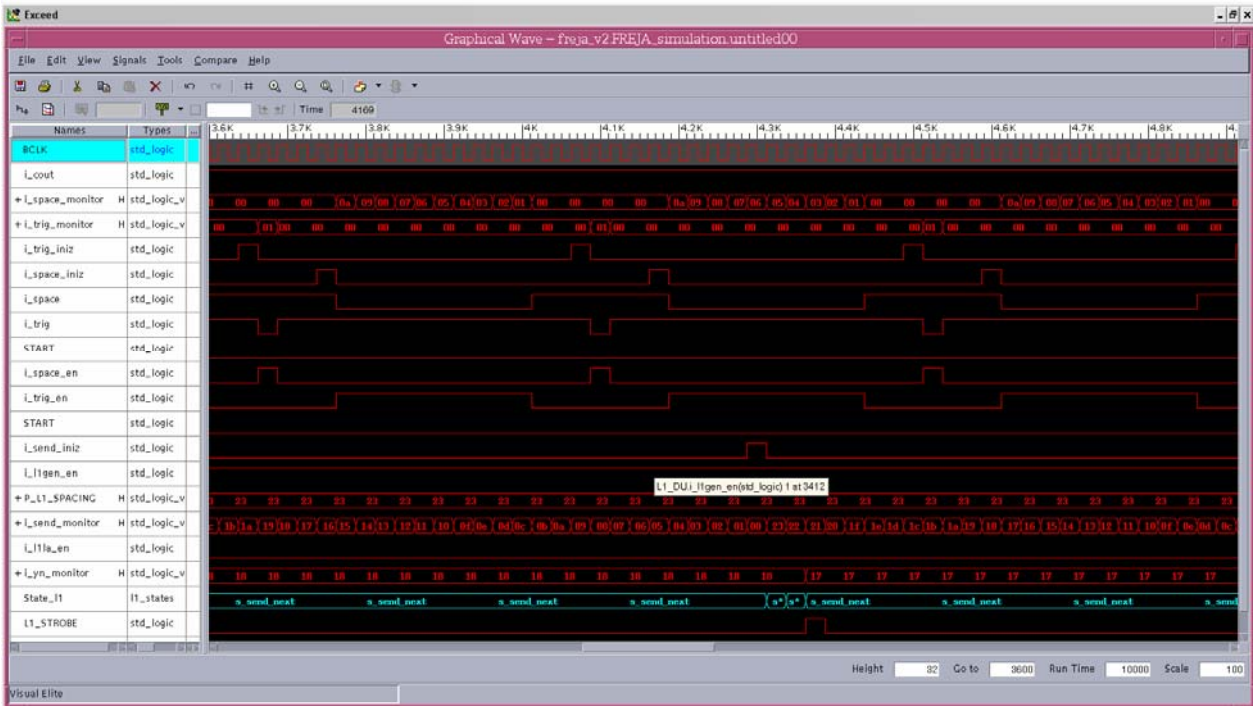


Figure 62: A screen shot of the timing plot in Visual Elite 3.5.1.

8 Board control

8.1 Experiment Control System

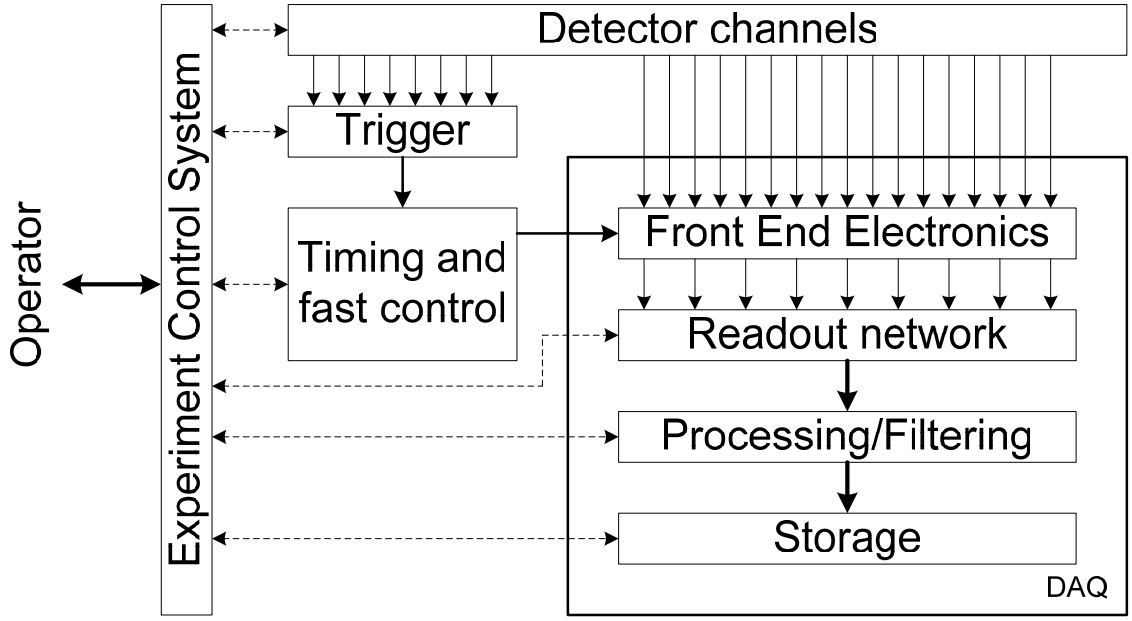


Figure 63: Scope of the Experiment Control System.

The *Experiment Control System* (ECS) **[12]** is responsible for the handling of the configuration, monitoring and operation of all experimental equipment involved in the different activities of the experiment:

- Data acquisition system:
  timing, trigger, front-end electronics, readout network, Event Filter Farm, storage etc.
- Detector operations (DCS):
  Gases, high voltages, low voltages, temperatures, etc.
- Experimental infrastructure:
  Magnet, cooling, ventilation, electricity distribution, detector safety, etc.
- Interaction with the outside world:
  Accelerator, CERN safety system, CERN technical services, etc.

The ECS provides a unique interface between the users and all experimental equipment (**Figure 63**). The ECS tasks for the detector's hardware equipment management are mainly accomplished by sending commands and settings, and reading back information. The control system can take decisions on its own (like recovering from errors) and let the operator interact with the system by presenting the information and accepting commands. The ECS is interfaced with two databases: a *Configuration database* that stores all the information regarding the equipment (geographical location, access addresses, default settings for different running modes, etc.) and a *Conditions database* which gather a subset of data needed for the offline analysis of the recorded data.

From the hardware point of view (**Figure 64**), the control system consists of a small number of PCs (high end servers) on the surface connected to large disk servers. These supervise other PCs that are installed in the counting rooms and provide the interface to the experimental equipment (see **Chapter 5.2.1**).

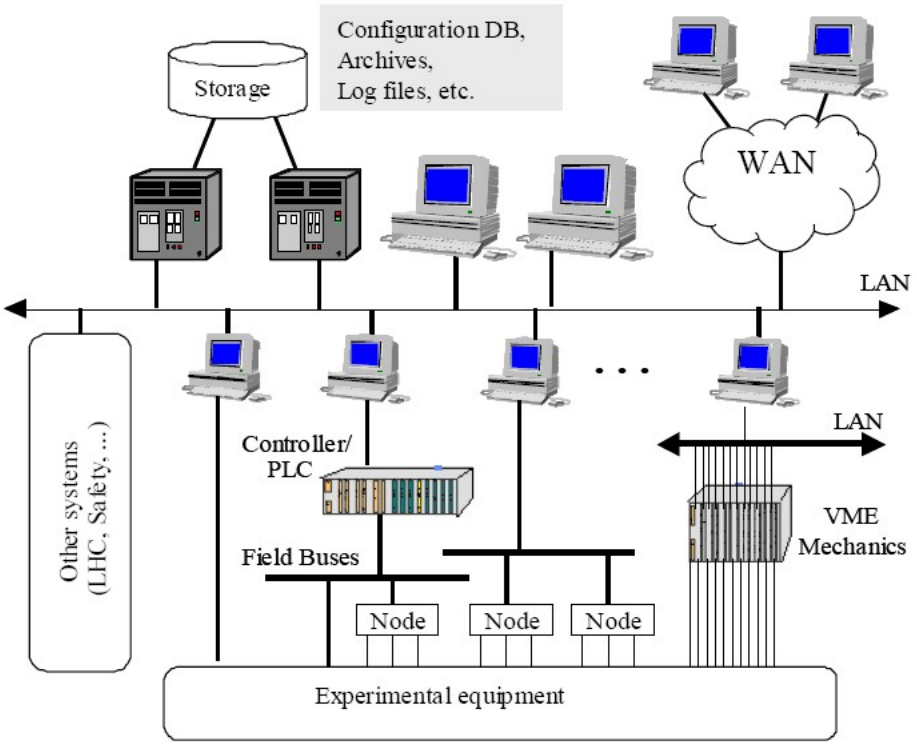


Figure 64: ECS hardware architecture.

From the software point of view the ECS has a hierarchical, tree-like structure to represent sub-detectors, sub-systems and hardware components. Two types of nodes compose the tree: *Device Units* which are capable of driving the equipment to which they are associated and *Control Units* which can monitor and control the sub-tree below (**Figure 65**).

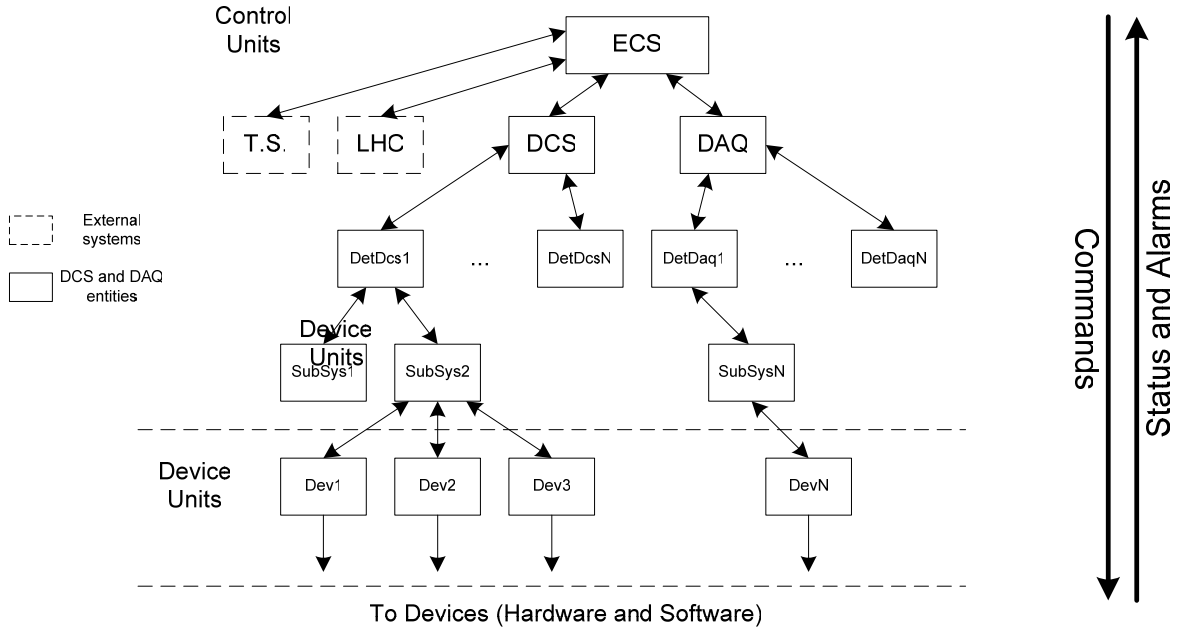


Figure 65: ECS Software architecture.

8.2 Control software framework

The requirements of an ECS system can be summarized with the definition of framework given in the LHCb on line system TDR **[4]**:

"An integrated set of guidelines and software tools used by detector developers to realize their specific control system application. The framework should include, as far as possible, standard elements and functions required to archive a homogeneous control system and to reduce the development effort as much as possible for the developers".

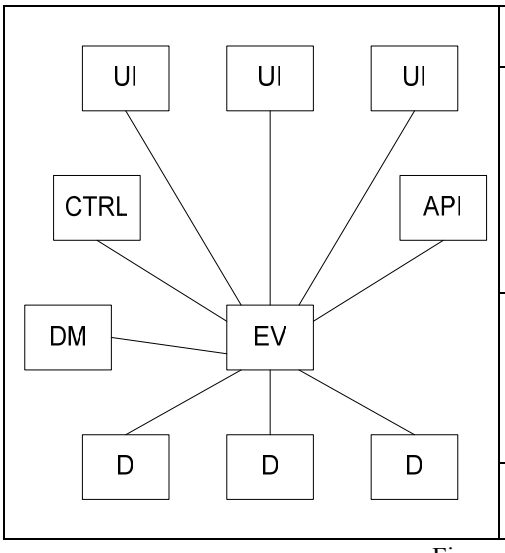
The framework tools which are specified by the LHCb on-line system group for hardware control are briefly described in the next chapters.

8.2.1 PVSS II

The *Supervisory Control And Data Acquisition (SCADA)* is a standard system largely used in industry to monitor wide scale systems. As the name indicates, it is not a full control system, but rather focuses on the supervisory level.

PVSS II (ProzessVisualisierungs-und Steuerungs System) **[37]** is a SCADA product used for the development of control systems at CERN **[38]**. PVSS II is a software tool for visualization, control and monitor based on device-oriented modelling. Device orientation is a high-level abstraction that allows the description of complex equipments in simple terms. The device description contains all the data and the high level commands that are needed in order to operate the equipment, even though the equipment could be composed of many channels.

PVSS II is structured in a modular manner. The individual tasks are handled by special program modules called managers, and communication is based on the client-server principle.



User Interface layer
User Interface Manager (UI): visualizes process states and handles user inputs.

Processing layer
Control (CTRL): an event-driven, multi-tasking language specific to PVSS II, capable of processing several function block at once. Its many applications include creating complex control functions, or performing testing and simulation tasks.
Application Program Interface (API): for interfacing external programs. Existing software packages can be integrated using class libraries, or dedicated managers developed for implementing industry-specific packages.

Communication and storage layer
Event Manager (EV): this forms the centre of the system, receiving messages, evaluating and distributing them.

Database Manager (DM): saves process changes in a high-speed database, archives data and generates analyses and reports.

Driver layer
Drivers (D): the interface between PVSS II and peripheral devices.

Figure 66: Schematic structure of PVSS II.

PVSS II is based on the principle of *Data Point Types* which are similar to structures in C. The data point type describes the structure and the format of its data point variables. The data point structure reflects the real-life situations: logically associated variables describing for instance a hardware board are grouped into hierarchically structured data points. This means, for instance, that a

TFC test board is managed internally in PVSS as a single data point containing all of the hardware registers in a structured manner according to the implementation in the hardware and the functionality to which they belong.

8.2.2 DIM

PVSS is connected to the hardware via the *Distributed Information Management (DIM)* **[40]**, a tool based on the Client-Server paradigm across TCP/IP.

The basic concept in the DIM approach is the concept of *Service* which is a set of data (of any type or size) and recognized by a name (named services). Servers provide services to clients.

Services are normally requested by the client only once (at start-up) and they are subsequently automatically updated by the server either at regular time intervals or whenever the conditions change (according to the type of service requested by the client).

The client updating mechanism can be of two types, either by executing a call-back routine or by updating a client buffer with the new set of data, or both. The last type works as if the clients maintain a copy of the server's data in cache, the cache coherence being assured by the server.

A special type of service, called *Command* allows a client to send a command to a server. Upon receiving a request from the client, the server executes a call-back routine in which actions can be taken based on the data which is sent by the client along with the command.

In order to allow transparency (a client does not need to know where a server is running) as well as to allow easy recovery from crashes and migration of servers, a *Name Server* was introduced. Servers publish their services by registering them with the name server (normally once, at start-up). Clients subscribe to services by asking the name server which server provides the service and then contacting the server directly, providing the type of service and the type of update as parameters. The name server keeps an up-to-date directory of all the servers and services available in the system.

PVSS can behave as a DIM Client (i.e. receive information from or send commands to DIM servers) or as a DIM Server (i.e. send information to or receive commands from DIM clients).

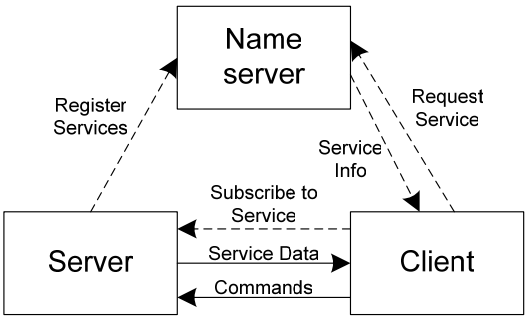


Figure 67: DIM components interaction.

8.2.3 SMI++

Control units are typically implemented using *Finite State Machines (FSM)*, which is a technique for modelling the behaviour of a component using the states that it can occupy and the transitions that can take place between those states. PVSS II does not provide for FSM modelling and therefore another tool SMI++ **[42]** has been integrated with PVSS for this purpose. SMI++ provides for rules-based automation and error-recovery.

8.3 The TFC local control system

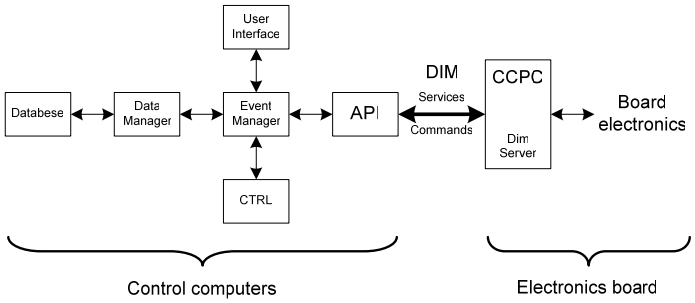


Figure 68: Schematic view of the TFC local control hierarchy.

Freja has been integrated in the TFC local control system. **Figure 68** shows a schematic view of the local control hierarchy which follows the same structure as the global ECS described earlier.

The Credit-Card PC accesses the board electronics via its PCI bus as described in **Section 5.2**. In order to receive commands, such as read and write registers, from user interfaces in PVSS and to return values of readings, a Credit-Card PC runs a server which publishes a set of DIM services and commands:

- *ReadWriteRegister* (command)
- *ReadRegister* (service)
- *SubscribeCounters* (command)
- *ReadCounters* (service)

The *ReadWriteRegister* command has five arguments: access method (I^2C, JTAG, Local bus), register address, data, a mask in order to effect only selected bit in a register, and a read/write bit whether the command is requesting a write or only a read. A write action always consists of a read of the register, followed by a write of the new data and last another read in order to send back the data that was actually written. In this way, the control system can verify that the write action was properly performed. After a read the data is returned by updating the *ReadRegister* service, which consists of three arguments: method, address and data.

The *SubscibeCounters* command allows a client to ask for a set of registers, typically counters, to be sent to the client regularly. The command has three arguments: access method, the address of the register and the interval at which the server should update the value. The server will thus send the values of the counters at the requested intervals by updating the *ReadCounters* service. The *ReadCounters* service consists of a dynamic array of two arguments: the address of the register that is updated and the value.

The Credit-card PC of each hardware board publishes its own set of these services and commands. The service and the commands are distinguished by having the name of the board in the service and the command name. The PVSS system can send commands to and receive data from the boards by subscribing to these services and commands.

As mentioned in the previous section, the PVSS database contains a data point list which reflects the entire register structure of the board. **Table 15** shows the Freja data point list as an example. The structure is common to all TFC boards.

In order to verify the writings automatically in PVSS, the registers exist in two copies: **Register Settings** and **Register Readings**. In addition, since there may not always be a one-to-one relation between physical registers (always 32-bits) and functional parameters such as "enable random generator", "L0 readout time", the data point also stores the values of the functional parameters, again as both **Parameter Settings** and **Parameter Readings**.

Table 15: Data Point List file for Freja.

FREJA_P2.FREJA_P2	1. Name of the data point list corresponding to *one* single board.
State Q_MODs_loaded Locked Ready Running Paused Error Monitored	2. **States stores** the status of the board
Register Readings Q1 R000 R004 R008 R00C ... I2C QA0 R00 R01 R02 R03 Q40 R00 R01 R02 R03 QFF R00 R01 R02 R03	3. **Register** contains the contents of the whole 32 bit words stored inside the corresponding hardware registers. **Reading** and **Settings** are handled separately in order to cross check settings.
Settings Q1 R000 R004 R008 R00C ... I2C QA0 ... Q40 ... QFF ...	← Identifies a **module** (FPGAs or I^2C devices) ← Identifies a **register** ← Identifies an **I^2C device** according to its physical address (EEPROM, I^2C register or TTCrx)
Parameter Readings L0DU R_L0PER_ENB R_L0RND_ENB ... L1DU P_L1_LATENCY P_L1_SPACE ... FEEL R_FE_ROTIME FIFOMON R_L0FIFO_MON R_L0FIFO_FULL	4. Since a single register can contain several different pieces of configuration data, **Parameters** are used to identify each one inside the 32 bit register. Parameters are grouped according to mnemonic structures specified during the User Interface design phase. Again **Reading** and **Settings** are treated separately.

... R_FEFIFO_MON ...	
Settings L0DU R_L0PER_ENB ... L1DU P_L1_LATENCY ... FEEL R_FE_ROTIME FIFOMON R_L0FIFO_MON ... R_FEFIFO_MON ... Apply	← Identifies the **set of parameters** following ← Identifies a **parameter**
Action ReadReg Data ReadCnt Data ReadWriteReg Method Address RegValue Mask Write SubscribeCnt Address Interval	5. **Actions** are data points used to exchange data between the board and PVSS.

The data point of each board also contains a set of **Data Point Items** which are associated with the DIM services and commands in order to transmit and receive data:

- **ReadReg:** associated to the *ReadRegister* service to receive data.
- **ReadCnt:** associated to the *ReadCounters* service to receive the counter values.
- **ReadWriteReg:** associated to the *ReadWriteRegister* command to request writing or reading of registers.
- **SubscribeCnt:** associated to the *SubscribeCounters* command to request counter values at regular intervals.

PVSS allows associating call-back routines with changes on data points. Whenever the content of a Data Point changes a function in a script file, structured like a C program, is called to perform a set of actions associated with the change. This is the means by which data is received via the DIM services. For instance, when the server running in the CCPC updates the *ReadRegister* service upon a read request, the data is written into the **ReadReg** data point item. A call-back routine associated with the data point item decodes the address and writes the data to the proper register in the data point. The script is also handling the translations between parameters and registers during write requests and read requests. **Figure 69** shows the sequence of actions during a write request. A user requests a change of the value of a parameter in the user interface. Upon applying the value, the value will be written to the appropriate parameter under **Parameter Settings** in the data point. A call-back routine in the script associated with the action will change the value of the parameter in the register in which it is located and write the new value of the register to the register under Register Settings and also write the new value, access method, address and mask to the data point item **ReadWriteReg** which is associated with the *ReadWriteRegister* DIM command.

The server will react by reading the register, modifying the value of the parameter and writing the new register value. The value is then read again and returned via the *ReadRegister* DIM service. As mentioned above the data is received by the **ReadReg** data point item and written to the appropriate register according to the received address under **Register Readings**. In addition to updating the parameters of the register under **Parameter Readings**, the script will also verify that the return value of the register matches the requested value. Since the user interfaces can subscribe to the parameters, the display is automatically updated with the latest settings.

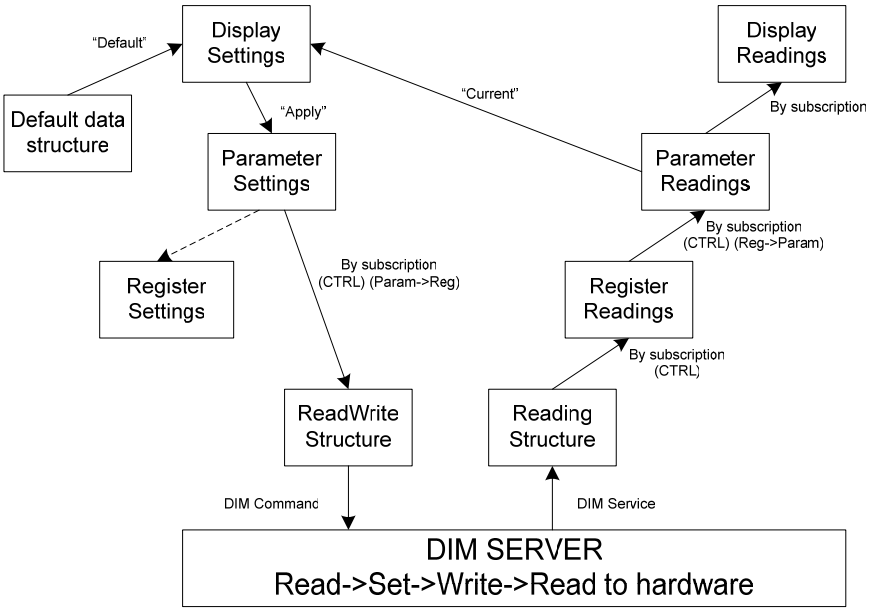


Figure 69: Flow of the Reading and Writing from and to TFC boards.

8.3.1 Freja control panels

The graphical interface has been designed using the Graphical Editor of PVSS II. A main panel Freja_P1.pnl and Freja_P2.pnl is used to control all board functions on both versions. An Expert panel is used to access all the board registers by inserting directly Hexadecimal addresses and values. In Version 2 the control panels have been implemented to allow users to access the features in a more intuitive and user friendly manner.

The panel of the first prototype includes the following features:

- A General monitor displays the contents of the principal test registers and parameters.
- A Channel B decoder which displays the contents of the short TTC broadcasts.
- A Channel B long TTC broadcast monitor which shows the contents of the long TTC broadcasts. Both L1 Trigger and High Level Triggers and the number of times the buffers relative to each type of triggers inside the Readout Supervisor is emptied (Multi Event Packet, MEP flush).
- A number of buttons allows performing basic operations on the board like resets, subscription and update of registers.
- An Expert panel allows users to access directly registers on board using Hexadecimal addresses and data. Both Local bus and I^2C functions are supported, by default the expert panel acts on Local bus, by selecting the proper button operations via I^2C are enabled.

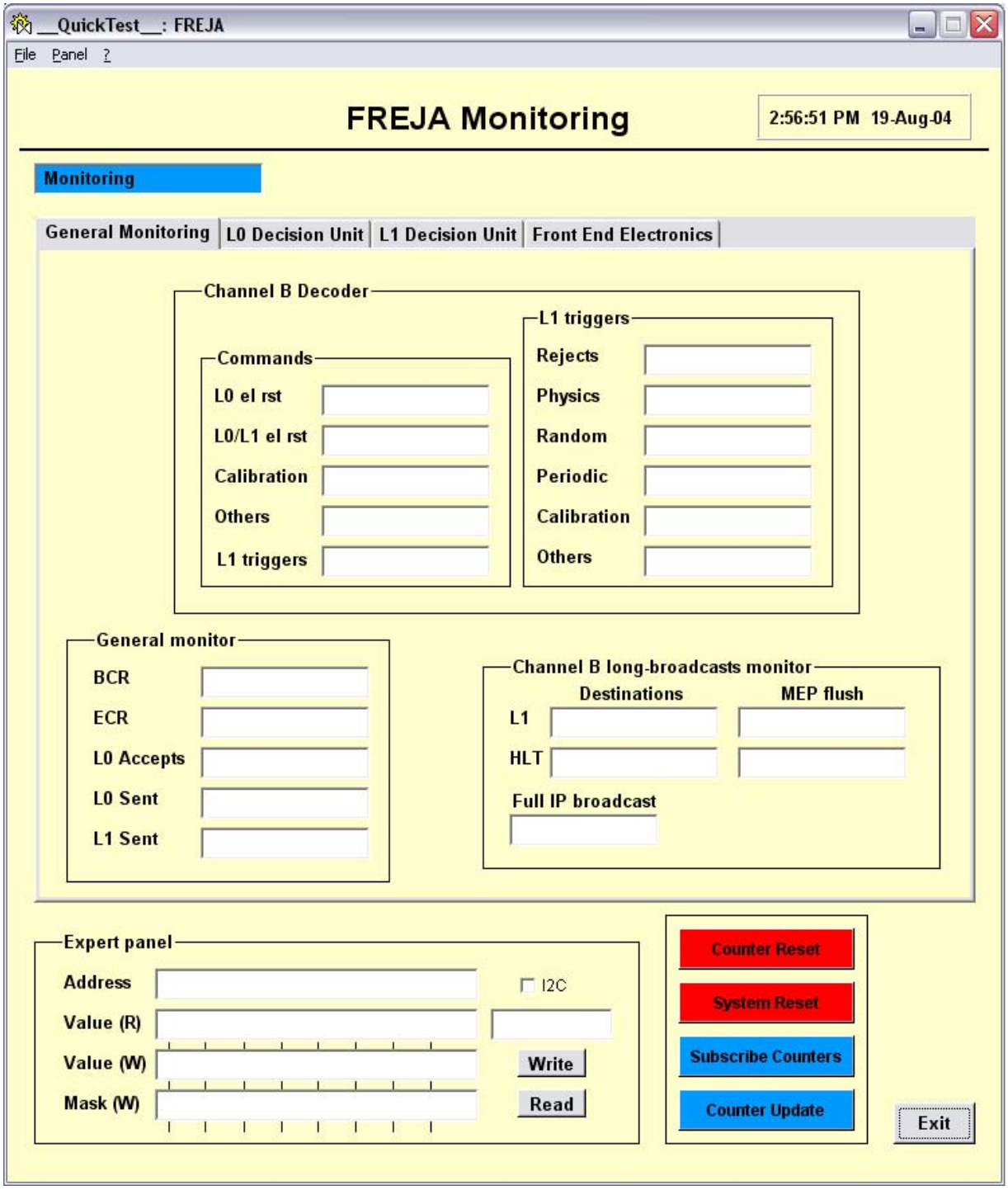


Figure 70: Screen shot of Freja V2 panel.

The main panel of the final prototype is not different from the one of the previous version described above. Configuration menus have been added to allow the setup of the different emulated functions of the test board.

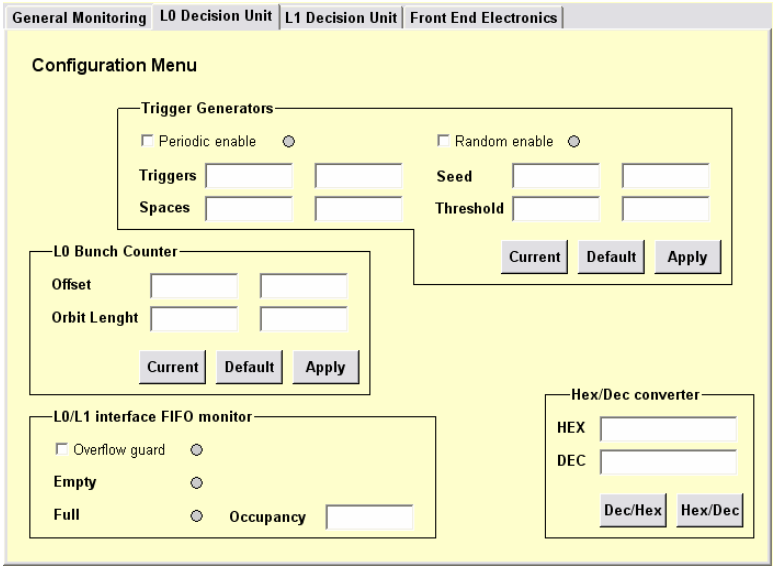


Figure 71: L0 DU configuration and monitor menu.

The enabling and disabling of functions is visualized by "LEDs". Each parameter can be set to any acceptable value. A Current button retrieves information of the actual content of a register. A Default button retrieves the default settings for each parameter. An Apply button sends the settings to the board.

Boxes to display Current values and insert settings are kept separate for readability. The enabling and disabling of functions is immediate as soon as the tick box is checked. To send the configuration parameters to the board the clicking of the Apply button is necessary.

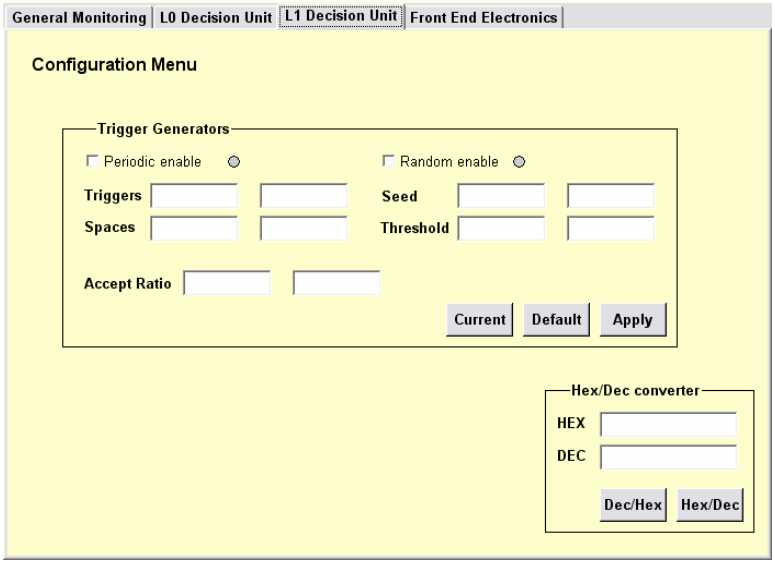


Figure 72: L1 DU configuration menu.

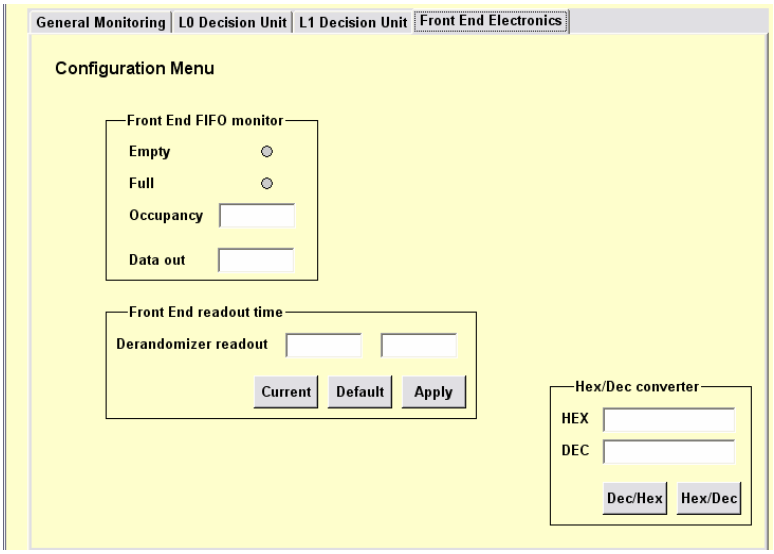


Figure 73: FE EL configuration and monitor menu.

In the expert panel all the registers must be set in either 32-bit or hexadecimal format. To simplify the calculation and the visualization of results a small converter has been implemented in the panel.

9 Ringraziamenti

E così… sembra che tu stia crescendo...
Piccola mia...

Ringraziamenti... Ovvero pensieri a ruota libera!

Scritti in italiano, dopo un prolisso parlare in lingua straniera, senza un motivo specifico... o forse perchè pensieri a ruota libera si esprimo nella lingua che più si avvicina a quella del cuore...

Primi ringraziamenti ai miei due supervisori, per avermi supportato e soprattutto sopportato durante tutta la mia permanenza al CERN.

A *Richard Jacobsson*: non ci sono parole per esprimere la mia gratitudine nei confronti di una persona che non solo si è rivelata un "capo" gentile, ma anche un collega con cui ridere in ufficio, anche nei momenti meno felici, ed un amico col quale trascorrere e condividere piacevoli momenti al di fuori dell'orario lavorativo... cenazze, birrette e tutto il resto!

A *Daniele Trinchero* la cui presenza "nell'ombra degli uffici del poli" mi ha permesso di arrivare alla fine del mio stage senza alcun intoppo.

A tutto lo staff dell'LHCb, in particolare ai colleghi del gruppo che si occupa della parte on-line dell'esperimento, per tutto quello che ho potuto vedere, conoscere, imparare ed apprezzare grazie a loro. Cito e ringrazio in particolare i colleghi con cui sono stato più a stretto contatto.

Ramy Abdel-Rahman per le lunghissime "nargila nights" e tutti i bei discorsi accompagnati dal gorgoglio della pipa che mi hanno portato all'incontro con una faccia della cultura medio orientale che non conoscevo.

A *Zbigniew Guzik* (Zbig) la cui conoscienza nel campo dell'elettronica è superata solo dalla sua passione per la stessa. Alla sua Vodka Polacca, le sue celebrazioni per qualsiasi motivo e tutte le sue tecniche di pasteggio alcolico accompagnate da sane risate...

A *Grzegorz Kasprowicz* (Gregory) per l'ispirazione nel creare apparecchiature elettroniche con mezzi di fortuna (cfr. lo sviluppo di PCB con l'uso di riviste per donne!). Al suo incredibilie robot ed alla sua ancora più incredibile macchina foto digitale per fotografia astronomica.

Ad *Artur Barczyk* e *Benjamin Gaidioz* inguaribili "computer boys"... per tutte le discussioni su sistemi operativi, editor di testo e quant'altro... probabilmente non imparerò mai!

A *Jean-Pierre Dufey*, un signore in camicia e barba bianca, il cui sguardo dice spesso più di mille parole. Gli insegnamenti di vita, di fronte a caffè e brioche, mi hanno aiutato ad osservare con più criticità ciò che mi circonda. Cercherò di tenere a mente tutto ciò che è stato detto...

Alla mia famiglia, *mamma* e *papà* per primi, per l'amore trasmesso anche a distanza, la pazienza nel sopportare i miei sbalzi d'umore e nel prendersi cura della mia salute in qualsisi circostanza. Non ci sono parole per esprime la gratitudine per avermi dato la possibilità di venire qui senza difficoltà.

Ad *Isa* e *Yann* poi, per tutto quello che hanno fatto per me durante quest'anno via da casa... A partire dai sorrisi sinceri e al sostegno morale per finire con i bei momenti trascorsi insieme e tutto il resto. Non basteranno un paio di inviti a cena al Thai dietro casa per sdebitarmi!

Ai miei amici e compagni d'università per i piacevoli momenti trascorsi insieme negli ultimi anni. Un grazie particolare a *Giulio Coluccia* "il capo" le cui ripetizioni nel tempo libero (unite alla passione, la pazienza e la meticolosità trasmessa durante gli incontri) mi hanno spesso permesso di superare ostacoli che sembravano insormontabili. Arriverò alla laurea anche grazie a te!

Ai miei amici, bambini cattivi, con i capelli lunghi e gli orecchini... siete parte di me.

Mi prostro infine di fronte alla potenza di *Google*... lo zerbino sulla porta di un mondo virtuale. Se non fosse stato inventato, a quest'ora, il mio lavoro sarebbe ancora a metà!

10 References

INTRODUCTION:

[1] www.cern.ch — CERN public home page
[2] user.web.cern.ch — CERN users home page
[3] http://lhcb-public.web.cern.ch/lhcb-public — Introduction to LHCb

SYSTEM OVERVIEW:

[4] http://lhcb.web.cern.ch/lhcb/TDR/TDR.htm — LHCb TDRs
[5] Readout Supervisor specifications — In the CD
[6] L1 trigger path note — In the CD

TTC system:

[7] http://ttc.web.cern.ch/TTC — TTC system homepage
[8] TTCrx users manual — In the CD

TEST:

[9] http://www.issco.unige.ch/ewg95/node80.html
[10] http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C13/glass.htm
[11] http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C13/black.html

CCPC:

[12] http://lhcb-online.web.cern.ch/lhcb-online/ecs/ccpc/default.htm — CERN CCPC/ECS site
[13] http://lhcb-online.web.cern.ch/lhcb-online/ecs/ccpc/docs/SM520PCX.pdf — CCPC user manual
[14] PLX 9030 data sheet — In the CD

PCI:

[15] http://www.pcguide.com/ref/mbsys/buses/types/pci.htm — PCI small introduction
[16] http://www.techfest.com/hardware/bus/pci.htm — PCI standard specifications
[17] MindShare inc, "PCI System Architecture revision 2.2", Addison Wesley
[18] MindShare inc, "PCI-X System Architecture", Addison Wesley

GLUE CARD:

[19] The Glue *light* user manual — In the CD

I^2C:

[20] Philips I^2C standard specification — In the CD

JTAG:

[21] http://www.jtag.com — JTAG official website
[22] IEEE 1149.1 standard specification — In the CD

LVDS:

[23] http://www.national.com/appinfo/lvds — LVDS home page

ECL:

[24] http://www.onsemi.com/ — Material on ECL technology in application notes
[25] http://www.lemo.com/techlibrary/index.jsp — LEMO company technical libraries

FPGA:

[26] http://www.xilinx.com — Xilinx FPGA vendor
[27] http://www.xenatera.com/bunnie/xi/rec.html — Link on MIT web site
[28] http://www.synplicity.com — Synplify home page
[29] http://www.altera.com — Altera FPGA vendor
[30] http://www.altera.com/products/devices/apex/apx-index.html — Altera APEX 20k family
[31] http://www.altera.com/literature/ds/apex.pdf — APEX family data sheet

STAPL:

[32] www.jedec.org/download/search/jesd71.pdf — JEDEC standard specification
[33] www.actel.com/documents/ISP_STAPL.pdf — ISP and STAPL

VME:
[34] IEEE 1101 Standard specification for VME boards mechanic — In the CD

VHDL:
[35] http://vhdl.org/vi — VHDL home page
[36] IEEE 1076 VHDL Standard specifications — In the CD

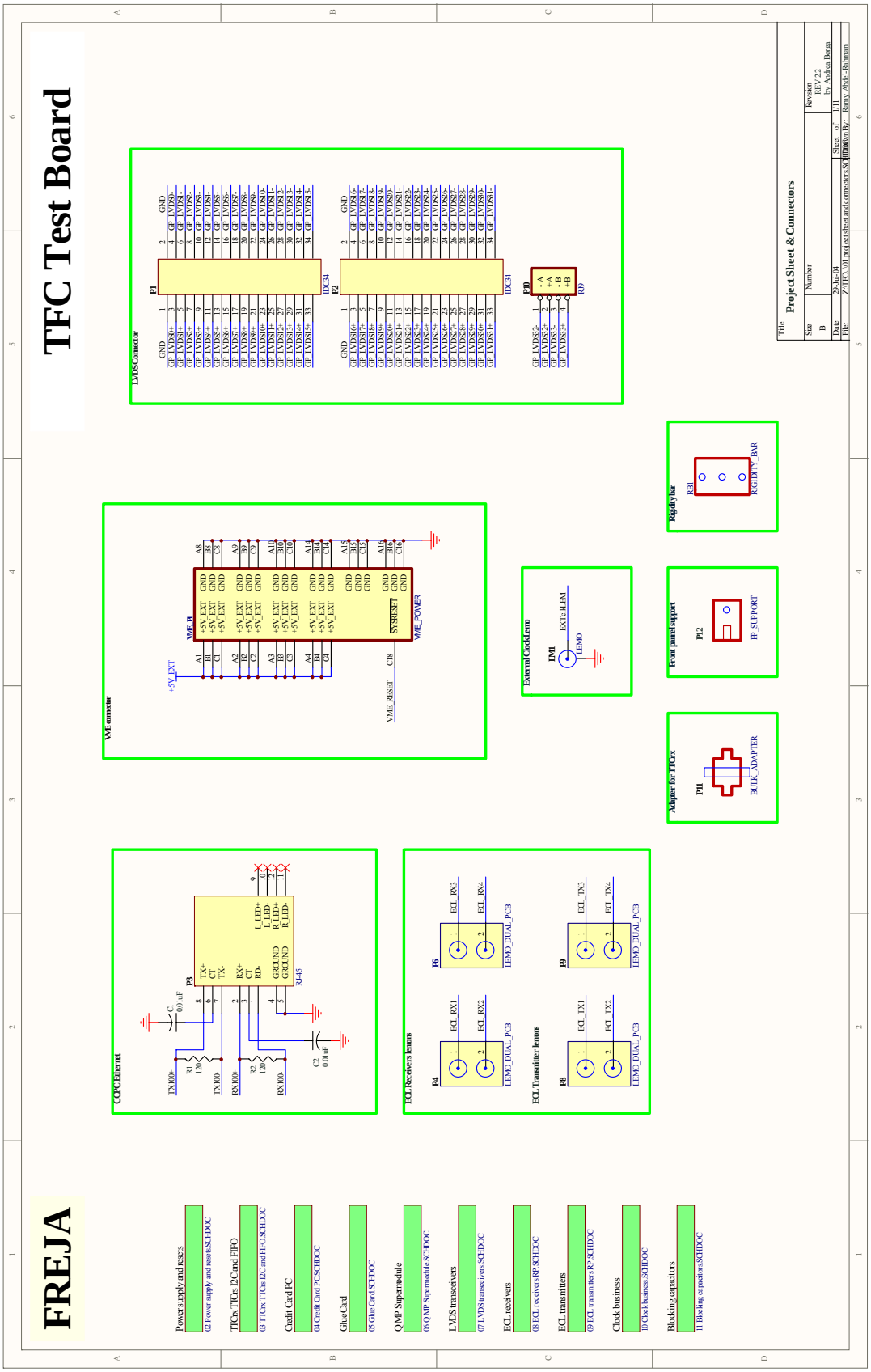
PVSS:
[37] http://www.pvss.com/english — PVSS official website CERN
[38] http://itcobe.web.cern.ch/itcobe/Services/Pvss/Documents/welcome.html — PVSS web page CERN PVSS
[39] http://itcobe.web.cern.ch/itcobe/Services/Pvss/Documents/PvssIntro.pdf — Introduction course to PVSS

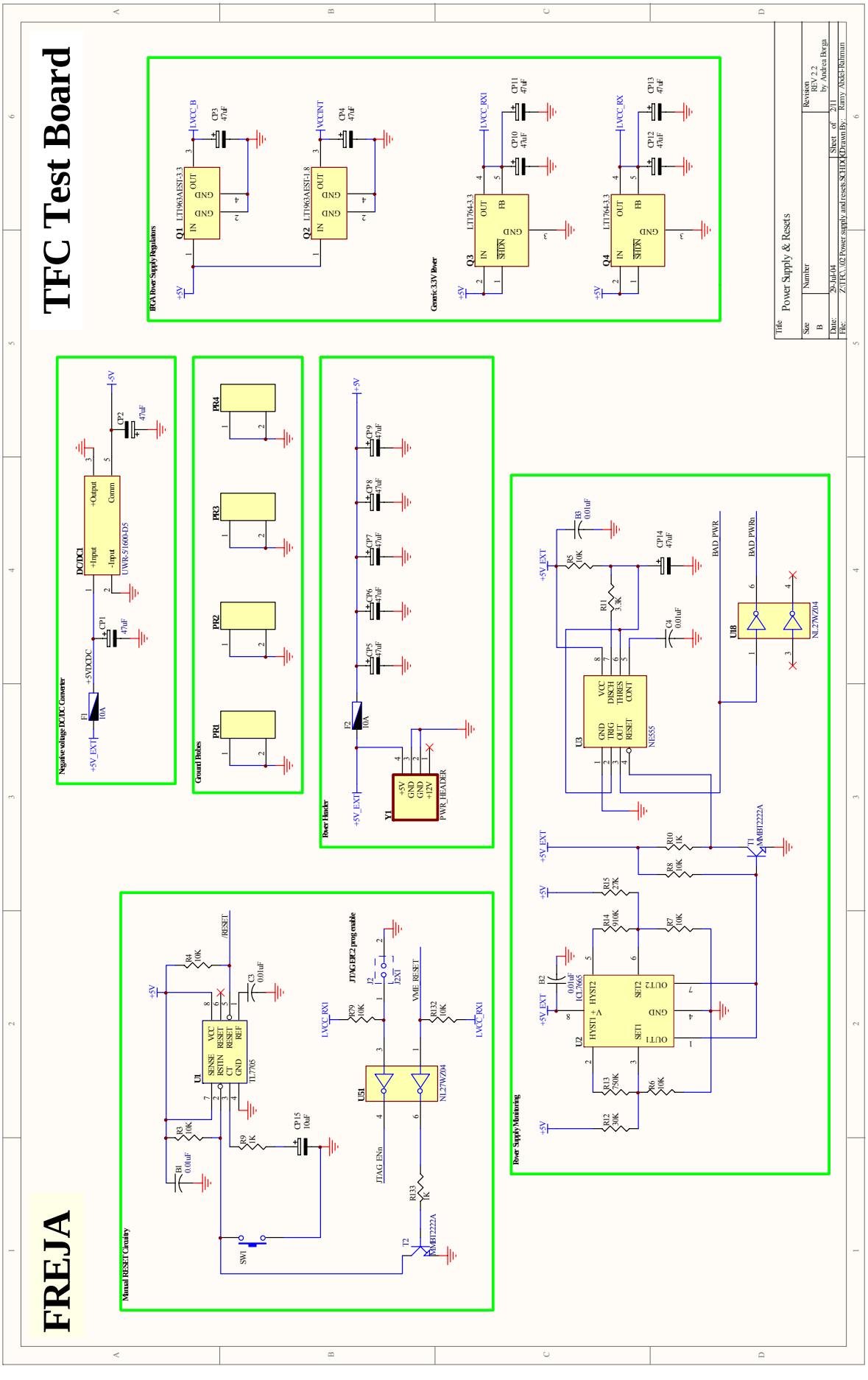
DIM:
[40] http://dim.web.cern.ch/dim — DIM web site
[41] http://clara.home.cern.ch/clara/fw — PVSS interface to DIM

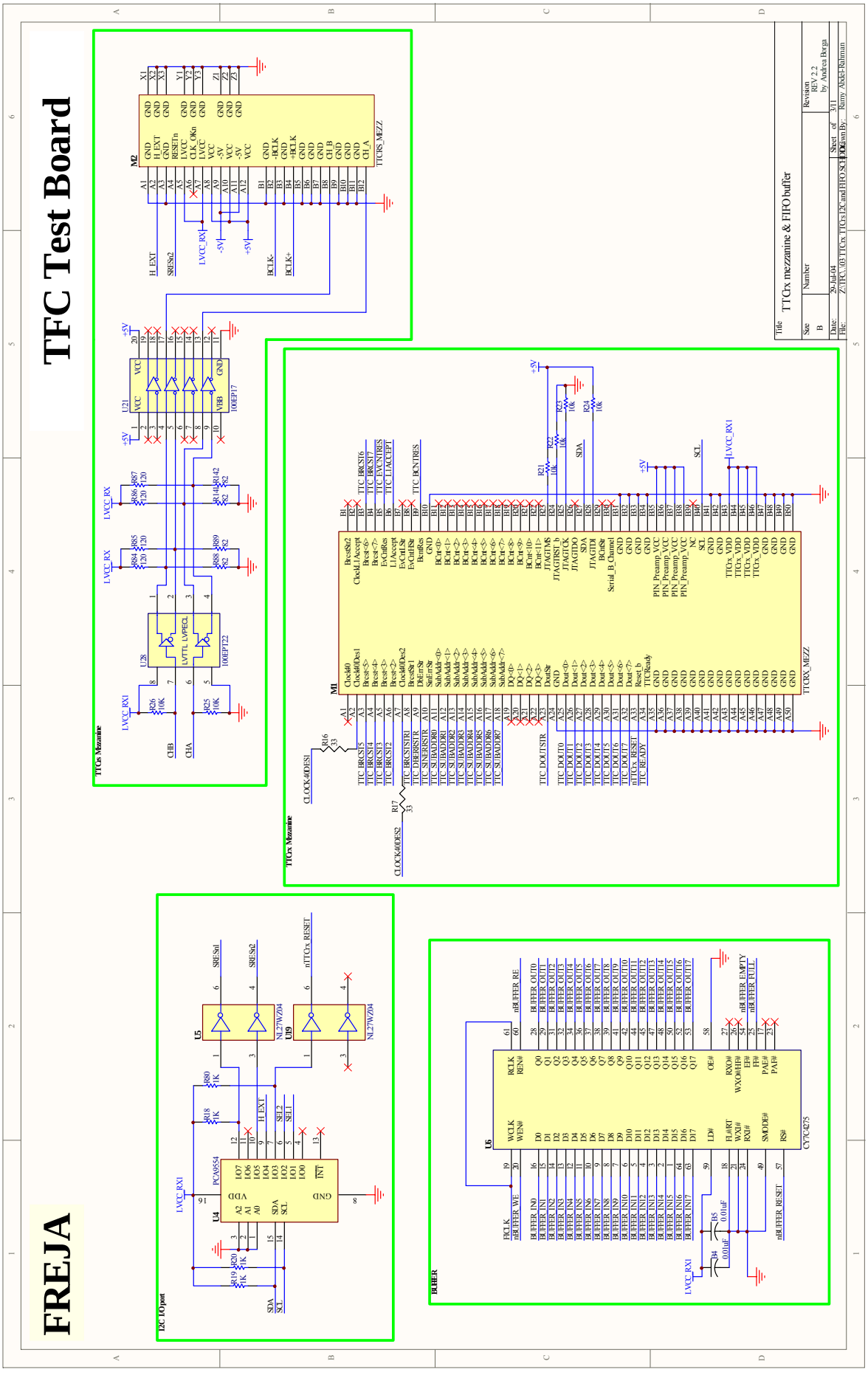
SMI++:
[42] B. Franek and C. Gaspar, "SMI++ - An Object Oriented Framework for Designing Distributed Control Systems" — CERN Technical note about SMI++

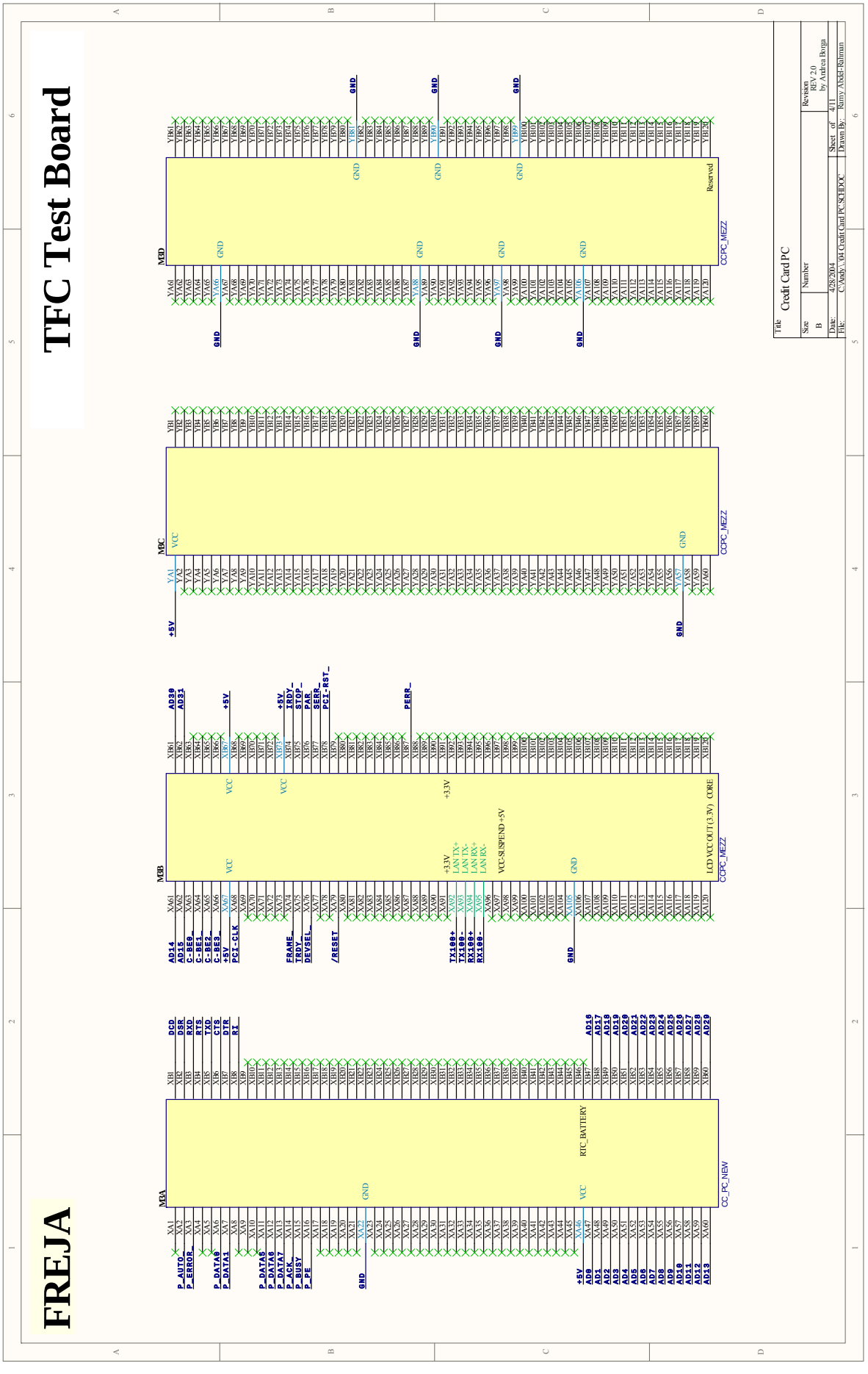
11 APPENDIX A - Board Schematics

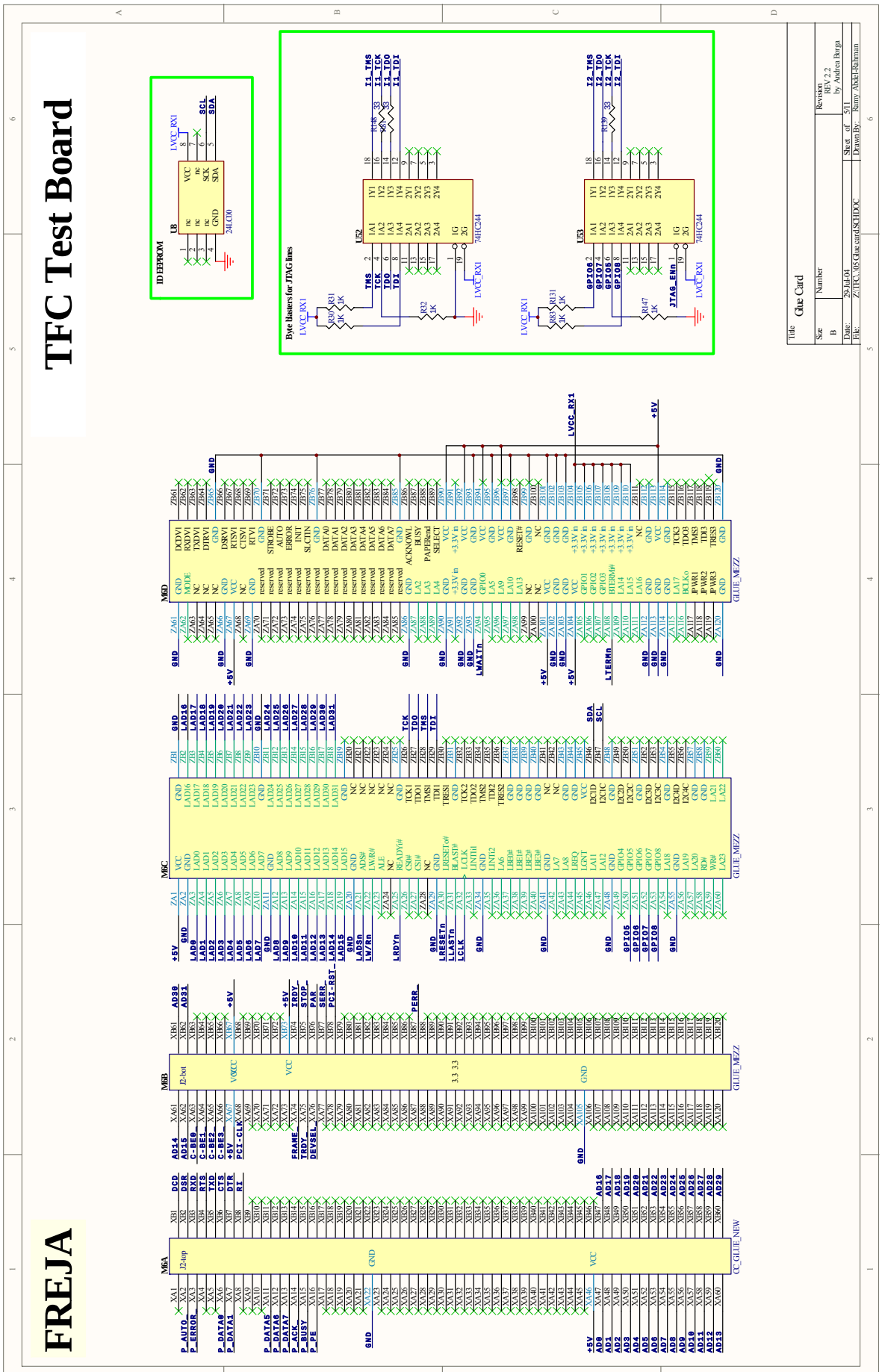
As mentioned in **Section 6.2**, in the next pages the board schematics are shown. All used components are TFC standard and kept in a specific Protel library.

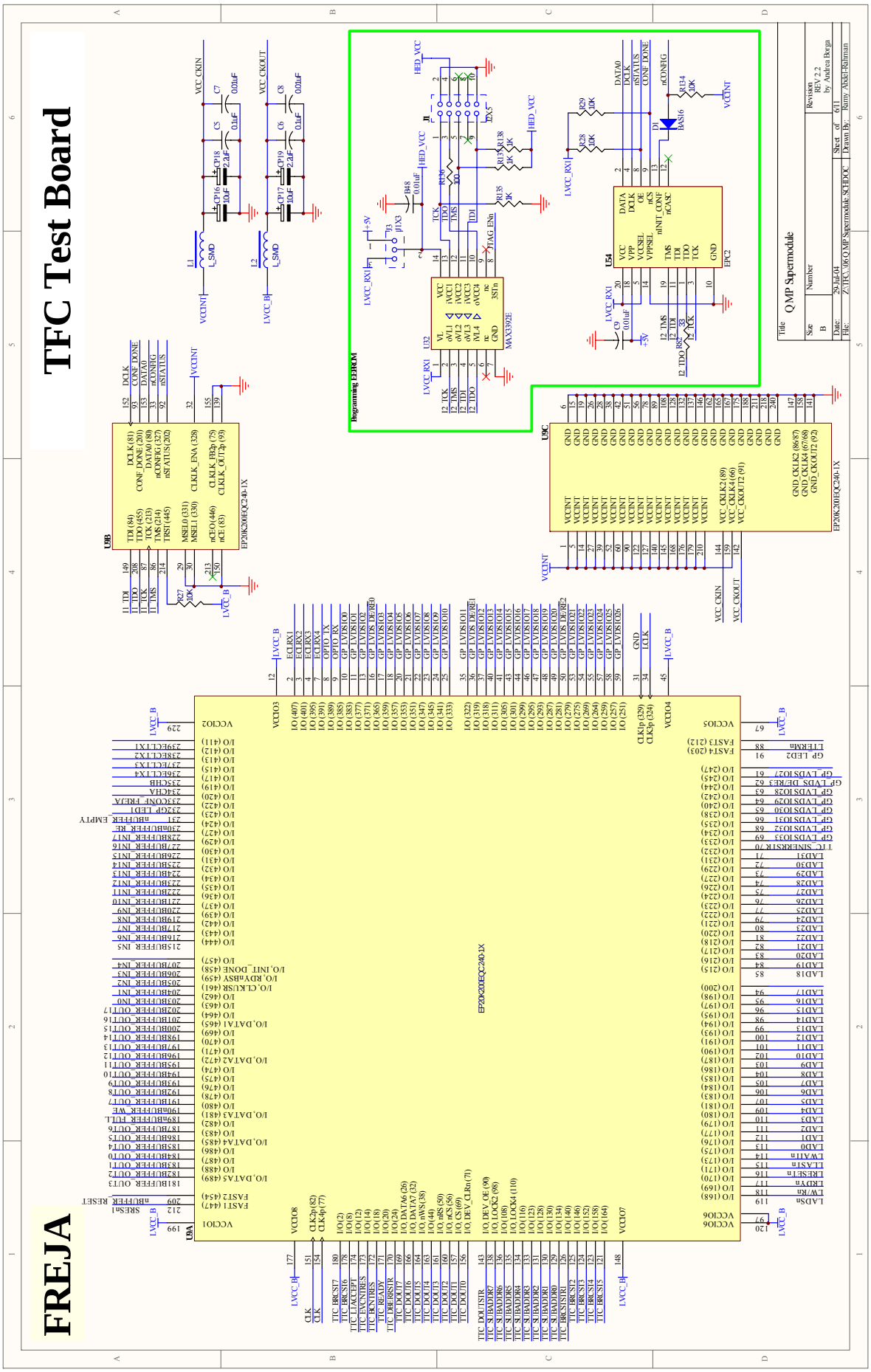


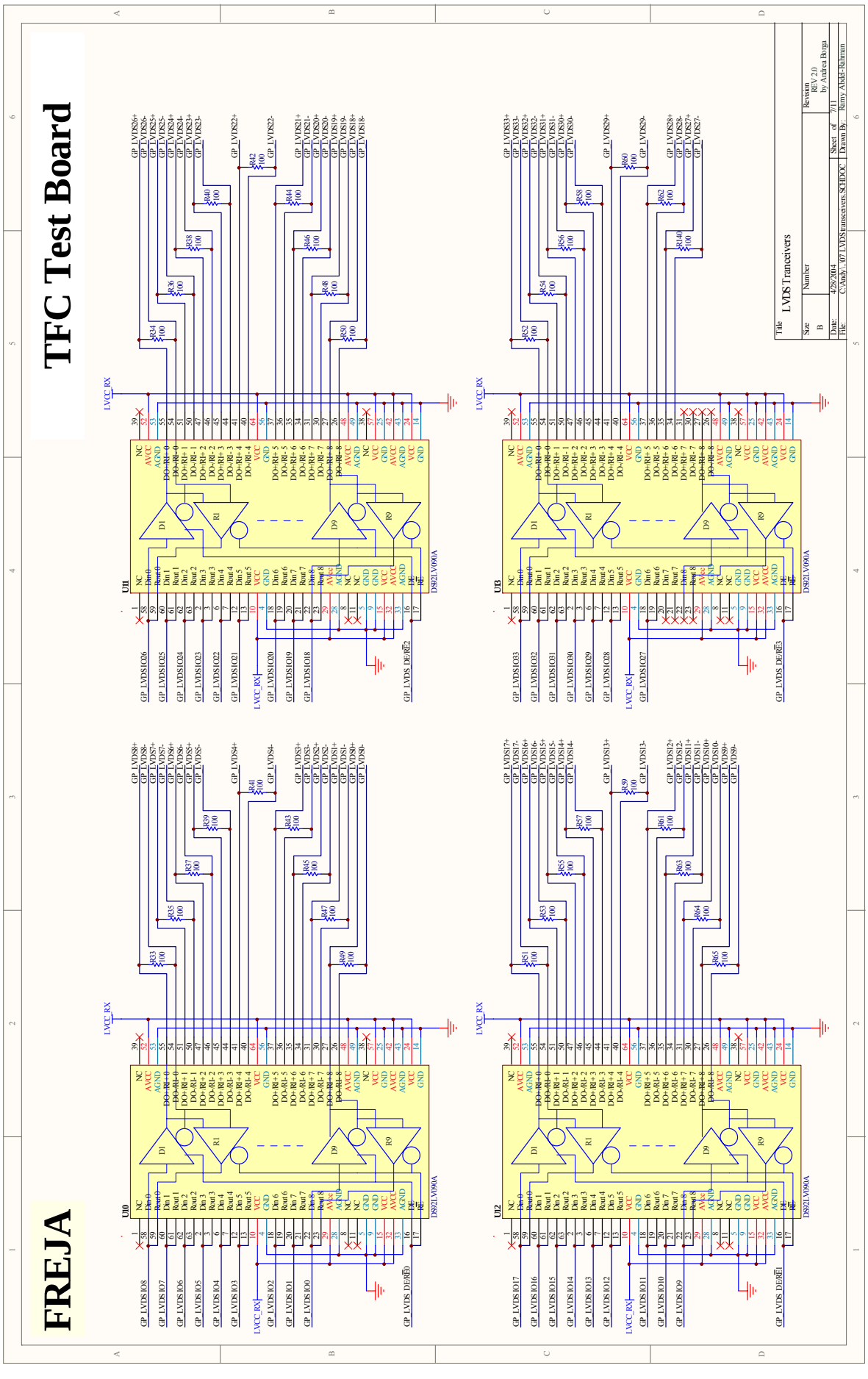

TFC Test Board
FREJA
Power Supply & Resets
Title
Size
B
Number
Revision
REV 2.2
by Andrea Borga
Date:
29-Jul-04
Sheet of 2/11
File:
Z:\TFC_02\Power supply and resets.SCHDOC
Drawn By:
Ramy Abdel-Rahman
IFCA Power Supply Regulators
Q1 LT1963AEST-3.3
Q2 LT1963AEST-1.8
Generic 3.3V Power
Q3 LT1764-3.3
Q4 LT1764-3.3
Negative voltage DC/DC Converter
DCDC1
UWR-5/1600-D5
Ground Holes
PR1
PR2
PR3
PR4
Power Header
Y1
PWR_HEADER
Power Supply Monitoring
U2
ICL7665
U3
NE555
U8
NL27WZ04
T1
MMBT2222A
BAD_PWR
Manual RESET Circuitry
U1
TL7705
U51
NL27WZ04
T2
MMBT2222A
SW1
JTAG EN
JTAG EDC2 prog enable
VME RESET

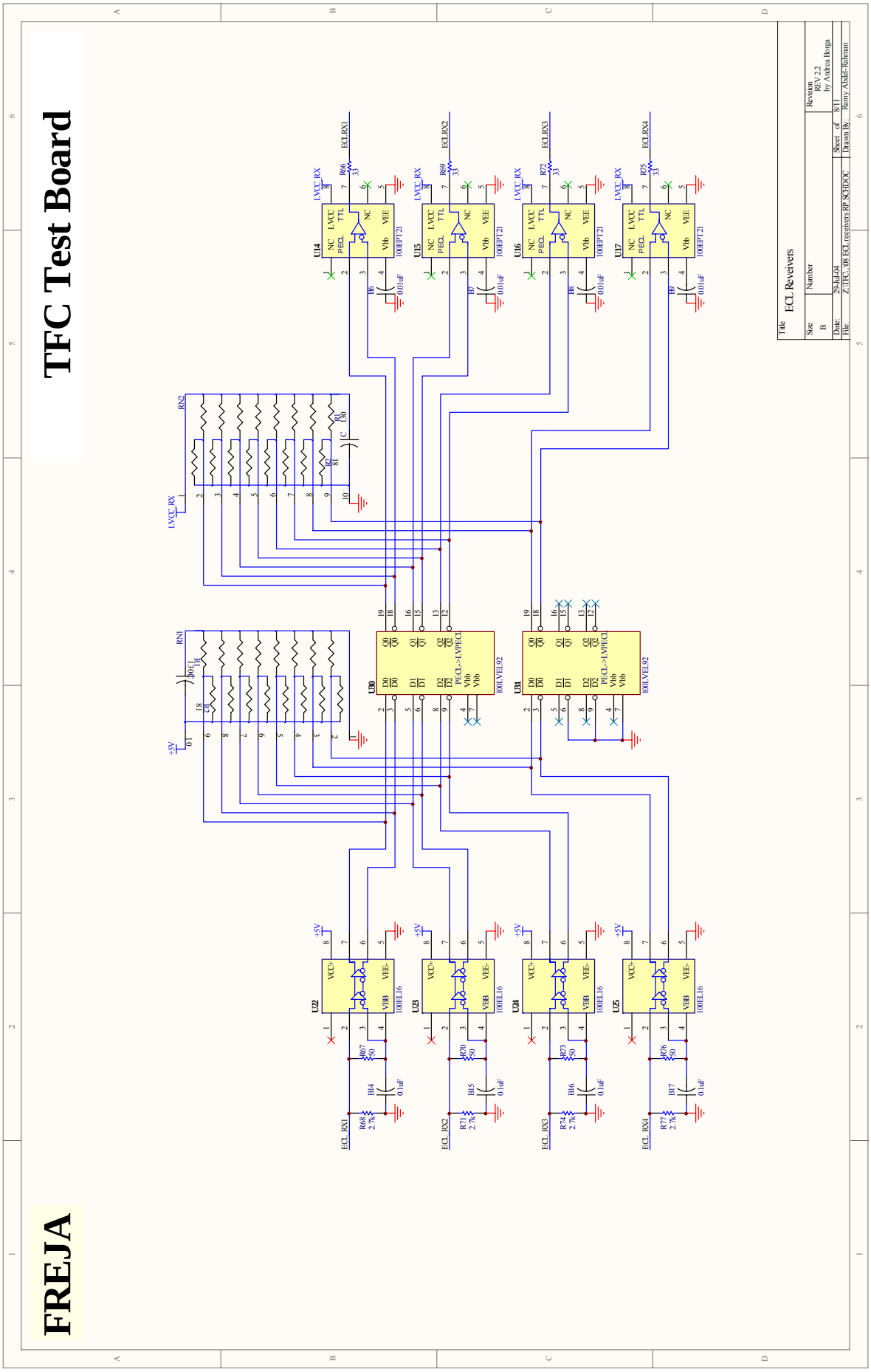

TFC Test Board
FREJA
TTCx mezzanine & FIFO buffer
Revision: REV 2.2
by Andrea Borga
Sheet of 3/11
Drawn By: Ramy Abdel-Rahman
TTCx Mezzanine
I2C I/O port
BUFFER
M1
M2
U4
U5
U6
U9
U21
U28
TTCRX_MEZZ
TTCRS_MEZZ
CY7C4275
PCA9554
NL27WZ04
100EP17
100EPT22
LVCC_RX
LVCC_RX1
CLOCK40DES1
CLOCK40DES2
H_EXT
SRESn1
SRESn2
nTTCrx_RESET
mBUFFER_RESET
mBUFFER_WE
mBUFFER_RE
mBUFFER_EMPTY
mBUFFER_FULL
SDA
SCL
BCLK-
BCLK+

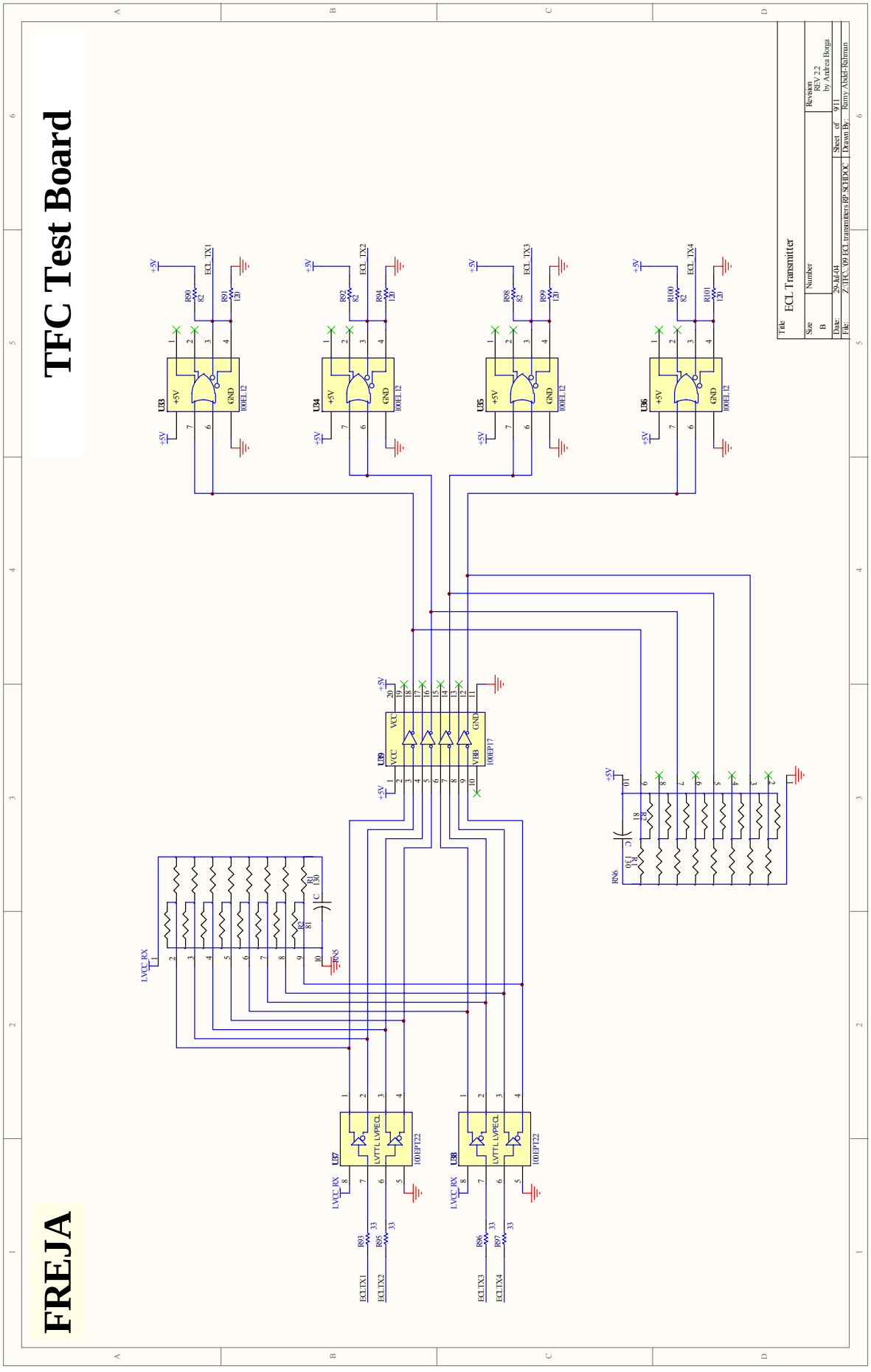

FREJA
TFC Test Board
Title
Credit Card PC
Size
B
Number
Revision
REV 2.0
by Andreas Borga
Date:
4/28/2004
Sheet of
4/11
File:
C:\Andy\...\04 Credit Card PC_SCHDOC
Drawn By:
Ramy Abdel-Rahman
CC_PC_NEW
CCPC_MEZZ
RTC_BATTERY
Reserved

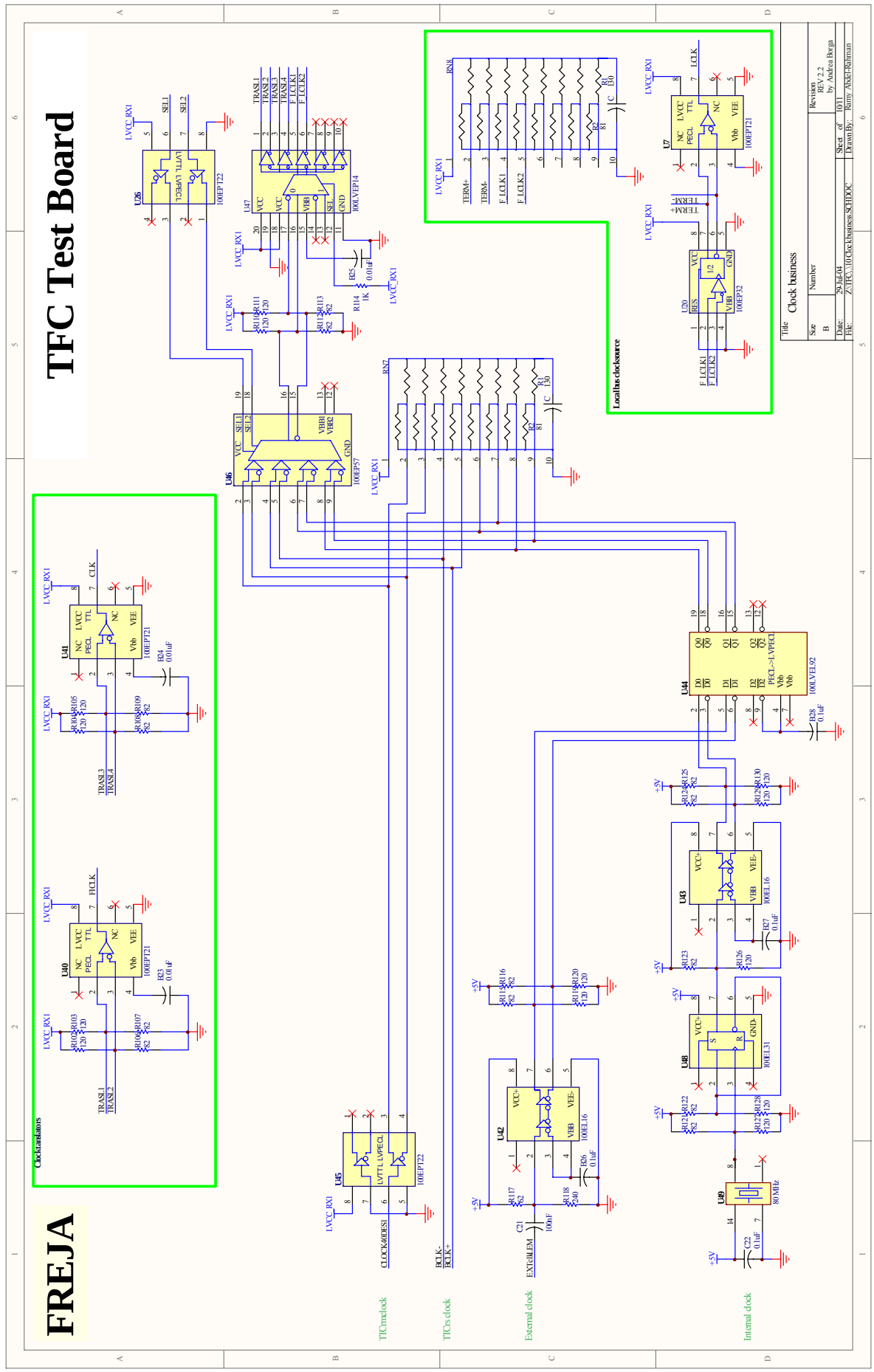

FREJA
TFC Test Board
ID EEPROM
U8
24LC00
Byte Blasters for JTAG fines
U52
U53
74HC244
Title
Glue Card
Size
B
Number
Revision
REV 2.2
by Andrea Borga
Date:
29-Jul-04
Sheet of
5/11
File:
Z:\TFC_05 Glue card\SCHDOC
Drawn By:
Remy Abdel-Rahman
M6A
M6B
M6C
M6D
CC_GLUE_NEW
GLUE_MEZZ
J2-top
J2-bot

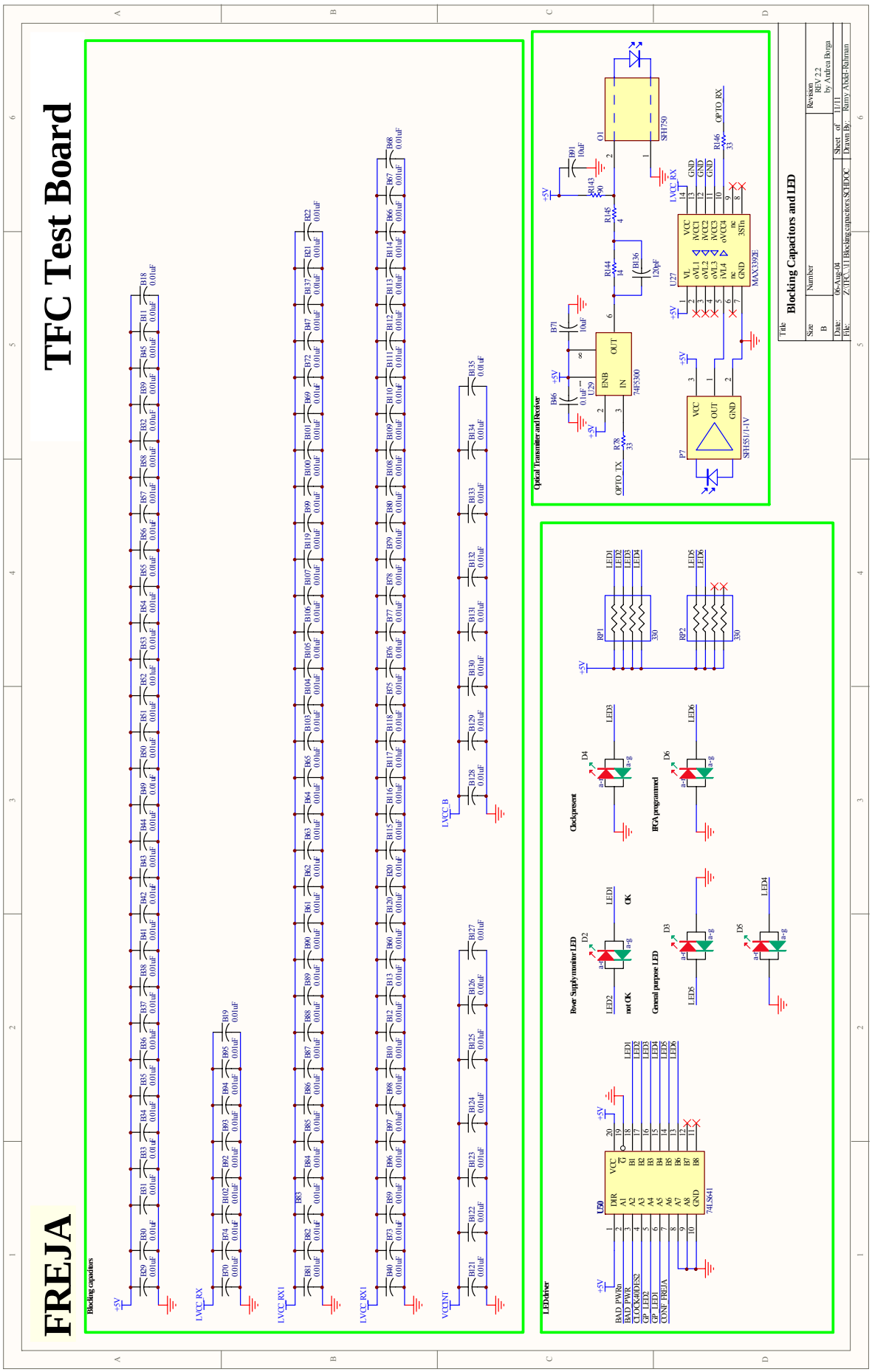

TFC Test Board
FREJA
QMP Supermodule
Programming EEPROM
EP20K200EQC240-1X
MAX392E
EPC2
BAS16
Title
Size
Number
Revision
REV 2.2
by Andreas Borga
Date:
29-Jul-04
Sheet of
6/11
File:
Z:\TFC_06\Q.MP.Supermodule.SCHDOC
Drawn By:
Rémy Abdel-Rahman


FREJA
TFC Test Board
LVDS Tranceivers
Title
Size
B
Number
Revision
REV 2.0
by Andrea Borga
Date:
4/28/2004
Sheet of
7/11
File:
C:\Andy\..\07 LVDStransceivers.SCHDOC
Drawn By:
Remy Abdel-Rahman
U10
U11
U12
U13
DS92LV090A
LVCC_RX
R33
R35
R37
R39
R41
R43
R45
R47
R49
R51
R53
R55
R57
R59
R61
R63
R65
R34
R36
R38
R40
R42
R44
R46
R48
R50
R52
R54
R56
R58
R60
R62
R64
R140


TFC Test Board
FREJA
ECL Reveivers
Size B
Number
Revision REV 2.2 by Andrea Borga
Date: 29-Jul-04
Sheet of 8/11
File: Z:\TFC_08 ECL receivers.SCHDOC
Drawn By: Remy Abdel-Rehman
U30
U31
100LVEL92
PECL->LVPECL
U2
U3
U4
U5
100EL16
U14
U15
U16
U17
100EPT21
RN1
RN2
LVCC_RX
ECL_RX1
ECL_RX2
ECL_RX3
ECL_RX4
+5V


TFC Test Board
FREJA
ECL Transmitter
Revision
REV 2.2
by Andreas Borga
Sheet of 9/11
Drawn By: Ramy Abdel-Rahman
Date: 29-Jul-04
File: Z:\TFC_99\ECL transmitters RP.SCHDOC
Size: B
Number
Title
U83
U84
U85
U86
U87
U88
U89
100EL12
100EP722
100EP17
ECL_TX1
ECL_TX2
ECL_TX3
ECL_TX4
LVCC_RX
LVTTL LVPECL
VCC
VBB
GND
+5V
ECLTX1
ECLTX2
ECLTX3
ECLTX4
RN5
RN6


TFC Test Board
FREJA
Clocktranslations
Localbus clocksource
Title: Clock business
Revision: REV 2.2 by Andrea Borga
Size: B
Number
Date: 29-Jul-04
Sheet of 10/11
File: Z:\TFC\10\Clockbusiness.SCHDOC
Drawn By: Rémy Abdel-Rahman
TTCrmclock
TTCrs clock
External clock
Internal clock
CLOCK40DES1
BCLK-
BCLK+
EXTclkEM
LVCC_RX1
+5V
TRASL1
TRASL2
TRASL3
TRASL4
F_LCLK1
F_LCLK2
TERM+
TERM-
SEL1
SEL2
U26
U47
U46
U7
U20
U41
U40
U45
U42
U43
U48
U49
U44
RN7
RN8
100EP722
100LVEP14
100EP57
100EP721
100EP32
100LVEL92
100EL16
100EL31
80 MHz


FREJA
TFC Test Board
Blocking capacitors
Optical Transmitter and Receiver
LEDdriver
Power Supply monitor LED
Clockpresent
IFGA programmed
General purpose LED
Title
Blocking Capacitors and LED
Size
B
Number
Revision
REV 2.2
by Andrea Borga
Date:
06-Aug-04
Sheet of
1/11
File:
Drawn By:
Remy Abdel-Rahman

12 APPENDIX B - Registers list

FREJA registers (Base address 0x1000)

Address	Bits	Register	Read/Write	Explanation	Remark	Default
000	0	i_update		Update registers	Updates in both read and write (not sensible to write)	0
004	0	ALL_COUNTER_RESET		Clear all counters in freja's modules		
	1	TTCRX_RESET		TTC rx reset line		
	2	EXT_BUFFER_RESET				
	3	L0FIFO_RESET				
	11 ... 4	COUNTRES_REG			bit 0: l0accept bit 1: bcntres bit 2: evcntres bit 3: l0trg bit 4: cmd bit 5: trg bit 6: l1trg bit 7: Lbrcst	
008	0	R_L0PER_ENB		L0 *periodic* trigger generator		
	1	R_L0RND_ENB		L0 *random* trigger generator		
	2	R_L1PER_ENB		L1 *periodic* trigger generator		
	3	R_L1RND_ENB		L1 *random* trigger generator		
00C	3 ... 0	ECL_W_ENABLE		ECL transimetters enable		0
010	3 ... 0	LVDS_RW_ENABLE		LVDS transceivers enable	1' transmitters '0' receivers	15 [F]
014	31 ... 0	TEST_REG		Local bus 32 bit test register		
020	12 ... 0	P_BID_ERR_INJ		Error injector for bunch ID check		
024	0	R_L0FIFO_GUARD_EN		Overflow L0/L1 interface buffer control	Can be switched off when running L0s only, to force overflows	
028	5 ... 0	P_TRNUM		Number of L0 triggers (L0 DU)	Check plus or minus bits	01 [01]
	13 ... 8	P_TRSPC		Number of L0 spaces (L0 DU)	Check plus or minus bits	34 [22]
02C	15 ... 0	P_L0RND_SEED		L0 random generator starting seed		[CAFE]
030	15 ... 0	P_L0RND_THRESH		L0 random generator threshold		[F998]
034	11 ... 0	P_BID_L0DU_OFFSET		Programmable offset for L0 DU bunch counter	Depending on the cable: 10 mt [25]; short [23]	
038	11 ... 0	P_ORBIT_LEN		Adjustable orbit length for L0 DU bunch counter and orbit generator		[DEB]
050	23 ... 0	P_L1_LATENCY		L1 periodic trigger generator latency (Latency timer)		20 [14]
054	5 ... 0	P_L1_SPACE		L1 periodic trigger generator spaces (RO timer)		35 [23]
058	5 ... 0	P_L1_ACCEPT_RATIO		L1 periodic trigger generator threshold		25 [19]
05C	15 ... 0	P_L1RND_SEED		L1 random generator starting seed		[CAFE]
060	15 ... 0	P_L1RND_THRESH		L1 random generator threshold		[F5C1]
080	5 ... 0	P_FE_ROTIME		Derandomizer read out timer (FE)		35 [23]
084	11 ... 0	R_SYNCH_CHECK	RO	Crossing & Bunch ID synchronization check		0
100	0	R_L0FIFO_EMPTY		L0 to L1 interface fifo flags		
	1	R_L0FIFO_FULL				
104	13 ... 0	R_L0FIFO_OCC				
108	1 ... 0	R_FE_FIFO_MON		Front End INTERNAL fifo (derandomizer) monitor	bit 0: fifo empty bit 1: fifo full (both active LOW)	
10C	3 ... 0	R_FE_FIFO_OCC				
110	11 ... 0	R_FE_FIFO_DATA				
114	1 ... 0	R_EXT_FIFO_MON		Front End EXTERNAL fifo monitor	bit 0: fifo empty bit 1: fifo full (both active LOW)	
200	31 ... 0	C_BCNTRES	RO	Bunch counter reset counter		
204	31 ... 0	C_EVCNTRES	RO	Event counter reset counter		
208	31 ... 0	C_L0ACCEPT_LOW	RO	Total number of L0 trigger accepts		
20C	31 ... 0	C_L0ACCEPT_HIGH				
210	31 ... 0	C_L0TRIGGER_LOW	RO	L0 triggers SENT from Freja		
214	31 ... 0	C_L0TRIGGER_HIGH				
218	31 ... 0	C_L1TRIGGER_LOW	RO	L1 triggers SENT from Freja		
21C	31 ... 0	C_L1TRIGGER_HIGH				
220	11 ... 0	C_BID	RO	Bunch IDs generated by Freja		
224	31 ... 0	C_CMD_L0ERST	RO	CHB decoder	L0 front end electronics reset	
228	31 ... 0	C_CMD_L01ERST			L0 + L1 front end electronics reset	
22C	31 ... 0	C_CMD_L1TRG			L1 trigger broadcast	
230	31 ... 0	C_CMD_CALIB			Calibration command	
234	31 ... 0	C_CMD_OTHERS			Other type of trigger	
238	31 ... 0	C_L1TRG_REJECT	RO	CHB decoder		
23C	31 ... 0	C_L1TRG_PHYSICS				
240	31 ... 0	C_L1TRG_RANDOM				
244	31 ... 0	C_L1TRG_PERIODIC				
248	31 ... 0	C_L1TRIG_CALIB				
24C	31 ... 0	C_L1TRG_OTHERS				
250	31 ... 0	C_L1_DEST_MO	RO	CHB long broadcast decoder	L1 destination assignmennt counter	
254	31 ... 0	C_L1_DEST_FLUSH_MO			L1 MEP flush counter	
258	31 ... 0	C_HLT_DEST_MO			HLT destination assignment counter	
25C	31 ... 0	C_HLT_DEST_FLUSH_MO			HLT MEP flush counter	
260	15 ... 0	C_LONG_BROADCAST	RO	Full long broadcast		
2FC	15 ... 0	C_VERSION	RO	Release version control register		

13 APPENDIX C - Backup CD

The CD contains the whole software material developed for the project:

- PCB project: both board version complete from schematics to gerber files
- VHDL source code of the TFC full system simulation
- C++ test scripts
- PVSS control interface project
- Excel table for the register configuration
- Electronic version of this note

It also contains most of the documents, application notes and manuals mentioned in the note grouped by topics in different folders.

You will also find additional information about me and my hobbies in a CV and in the “personal interests” folder.

14 APPENDIX D – Norse Mythology

A bit of Scandinavian mythology to give a better understanding of all the fancy names given to the TFC boards.

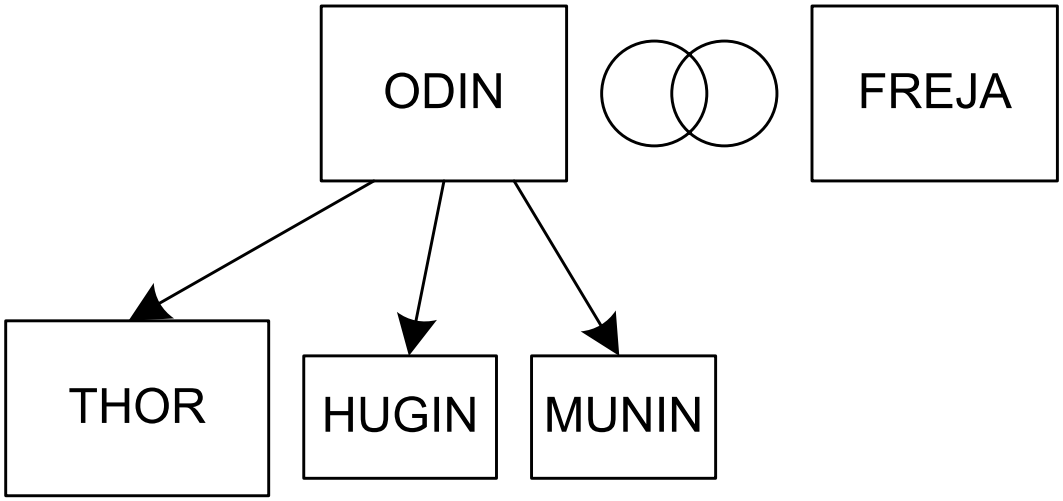


Figure 74: Swedish gods' family tree.

Scandinavian deities live in the garden of gods named Asgård.

Odin is the supreme god: his wisdom came when his drank from the source of knowledge.

Freja is his mistress, deity of beauty and fertility. But not without a bit of trickiness to test Odin's patience!

Thor is his son riding in the sky. With his enchanted chariot he creates lightings, and with his legendary hammer he makes thunders.

Odin has two ravens, Munin and Hugin, which he sends all over the world to collect information.

Figure 75: Odin the god of all gods.

For more information about the topic refer to http://en.wikipedia.org/wiki/Main_Page .