UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Proyecto Fin de Carrera

A Framework for the Development of FPGA-Based Ethernet Monitoring Systems

Entorno para el desarrollo de sistemas de monitorización Ethernet basadas en FPGA's

Fernando Alonso Marcos

SEPTIEMBRE 2008

ENTORNO PARA EL DESARROLLO DE SISTEMAS DE MONITORIZACIÓN ETHERNET BASADOS EN FPGA'S

AUTOR: D. Fernando Alonso-Marcos
TUTOR: D. Sergio Lopez-Buedo

Networking Research Group
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2008

PROYECTO FIN DE CARRERA

Título: *Entorno para el desarrollo de sistemas monitorización de Ethernet basados en FPGA's*

Autor: D. Fernando Alonso Marcos

Tutor: D. Sergio López Buedo

Fecha de lectura:

Calificación:

Resumen:

Este proyecto de fin de carrera trata de crear un entorno FPGA, para desarrollar sistemas de monitorización Ethernet. El objetivo principal es alcanzar una precisión temporal de decenas de nanosegundos en las marcas de tiempo generadas. Se ha desarrollado un algoritmo para compensar las derivas temporales del oscilador local de cuarzo, usando un GPS como fuente de sincronismo. Los errores en la sincronización son menores de 100 nanosegundos. La arquitectura del sistema se basa en relojes asíncronos, permitiendo una resolución en las marcas temporales de 10ns con independencia de la frecuencia de la red o del sistema. Esta arquitectura, basada en diferentes dominios de tiempo, permitirá mejorar la resolución hasta 5ns en la FPGA Virtex II Pro seleccionada. No obstante, esta idea ha provocado la necesidad de desarrollar un modulo específico con el fin de evitar problemas potenciales como la metaestabilidad.
El sistema es capaz de filtrar paquetes haciendo una comparación directa con las reglas, basadas en quíntuplas (direcciones, puertos y protocolo) definidas por el usuario. La estructura trata de establecer un primer paso para implementar un árbol de filtrado PAM. Se almacenarán, en la memoria de la FPGA, los 50 primeros bytes de cada paquete que supere el filtro y posteriormente estos serán mandados, junto a su marca de tiempos, a un PC. Este ordenador estará ejecutando un programa en JAVA para mostrar los resultados.
El sistema de monitorización ha sido implementado en una placa de desarrollo que no esta basada en una conexión PCI, lo cual otorga independencia al sistema. Al final de la memoria, se publican los resultados de las pruebas las que se ha sometido este proyecto y que muestran el éxito del mismo.

Palabras clave:
Ethernet, FPGA, monitorización de red, marcas de tiempos, alta precisión, sistemas embebidos.

A Framework for the Development of FPGA-Based Ethernet Monitoring Systems

AUTHOR: Fernando Alonso-Marcos
ADVISOR: Ph.D. Sergio Lopez-Buedo

Networking Research Group
Escuela Politécnica Superior
Universidad Autónoma de Madrid
September 2008

PROYECTO FIN DE CARRERA

Title: *A Framework for the Development of FPGA-Based Ethernet Monitoring Systems.*

Author: Fernando Alonso Marcos

Advisor: Ph.D. Sergio López Buedo

Abstract:

This master thesis aims at creating a framework for developing a FPGA-based Ethernet monitoring system. Its key objective is to achieve a time accuracy of tens of nanoseconds in the generated time-stamps. An algorithm for compensating the time drift of the local quartz oscillator has been developed, using a GPS signal as synchronization source. Errors in time synchronization have been measured to be less than +-100 ns. The system architecture is based in asynchronous clocks, thus allowing a time-stamping resolution of 10ns independently of the system or network clock frequency. This architecture, based on different time domains, will even permit to improve the resolution until 5 ns on the selected Virtex-II Pro FPGA. However, this approach has required the development of a specific module for handling the communication between time domains, to avoid potential problems such as metastability.
The system is able to filter packets making a direct comparison with the rules, based on quintuples (addresses, ports and protocol) defined by the user. This structure tries to establish the first step for developing a PAM filtering tree. The 50 first bytes of the packets that pass the filter are stored on the FPGA memory and then sent by an Ethernet connection to a PC. This computer is running a JAVA application for showing the results.
The monitoring system has been implemented on a FPGA development board which does not require a PC hosting for working. Finally, a complete test platform has been implemented in order to validate this project.

Key words:
Ethernet, FPGA, Network Monitoring, high precision time stamping, embedded systems.

Agradecimientos

En ocasiones los barcos necesitan de un faro para poder llegar a puerto. En este proyecto ese faro ha sido mi tutor, el profesor Sergio Lopez-Buedo, que ha sido la referencia sin la cual, este proyecto no hubiese llegado a buen destino. Es para él, mi más profunda gratitud. También quiero mostrar mi agradecimiento, a todos mis compañeros del laboratorio 209, que me han prestado siempre, incondicionalmente, su ayuda, apoyo y afecto, así como al resto de miembros del grupo Network Researching Group.

Me resulta imposible no acordarme en estos momentos de mis padres, que pacientemente han sabido guiarme a lo largo de todos estos años, y de mis amigos que siempre han estado cuando más los necesitaba. A todos ellos les dedico este proyecto.

Fernando Alonso Marcos
19 de Septiembre 2008

INDEX

FIGURE INDEX

TABLE INDEX

1 Introducción

1.1 Motivación

Hoy en día hay muchas actividades comerciales y de investigación () que requieren conocer, con buena precisión, el estado de la red en diferentes puntos y en tiempo real. En concreto estos análisis requieren medidas de gran precisión sobre los tiempos de latencia.

El gran incremento en el número de nodos, la evolución de los requisitos de los servicios sobre la red, loas altos volúmenes de datos y la heterogeneidad de las diferentes tecnologías han creado una necesidad de comprender en gran detalle las interacciones del tráfico.

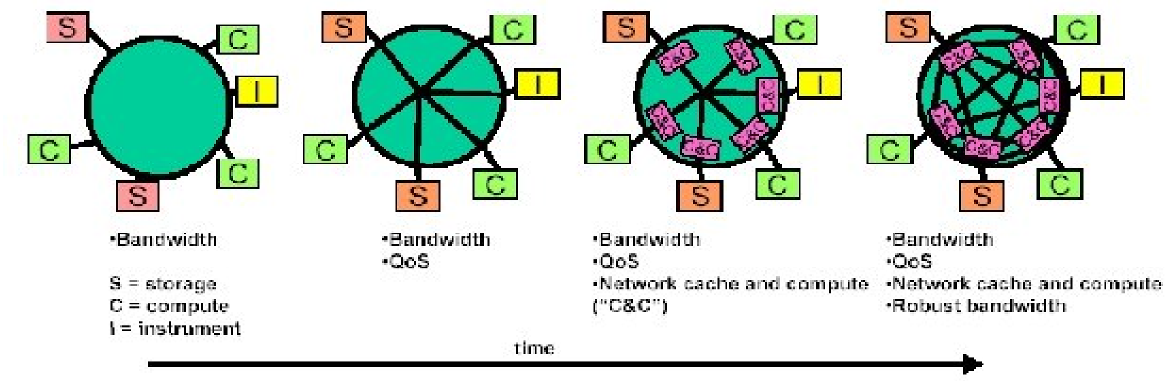


Figura 1-1: Evolución de los requisitos de los servicios de red sobre el tiempo [14].

Como se explica en [14], hay un gran espacio, “GAP”. Este es definido como “el espacio que existe entre lo que medimos/entendemos y lo que verdaderamente esta ocurriendo”. “Ya es grande y es muy probable que siga creciendo”. Debajo se puede apreciar este fenómeno.

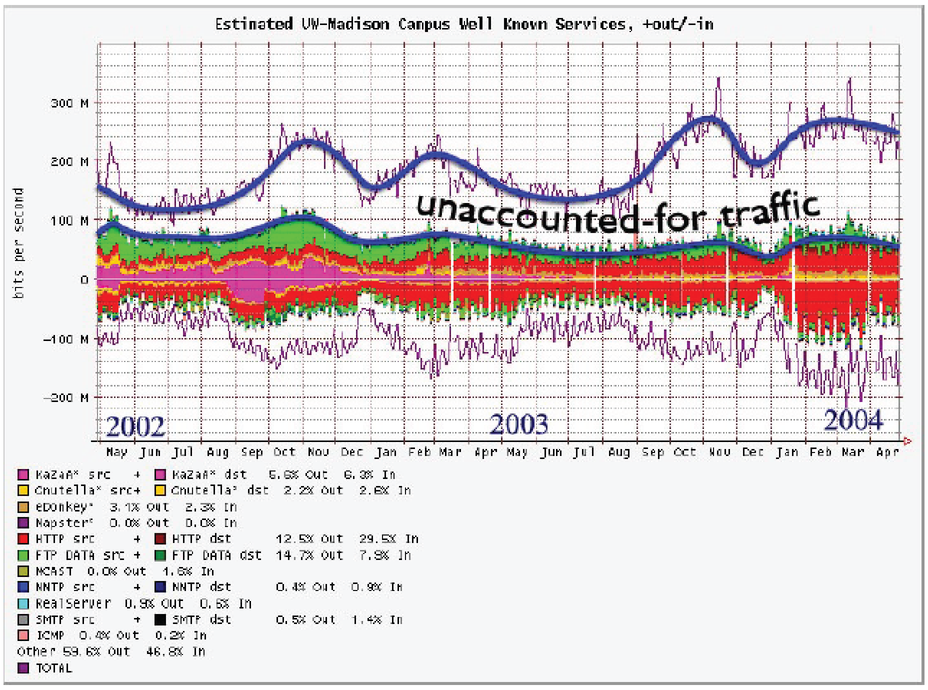


Figure 1-2: “The Gap” [14].

Este proyecto de fin de carrera trata de resolver estos problemas en las redes de Ethernet. Una primera aproximación podría ser una solución basada exclusivamente en software que pudiera estar equipado con un sistema operativo en tiempo real. Hay varias razones en contra de esta sencilla solución. Aquí se puede apreciar un breve listado de los inconvenientes que esta solución puede generar [1][2].

- La gestión de interrupciones generadas por la tarjeta Ethernet. Incluso estando bajo un sistema en tiempo real, la incertidumbre introducida por la propia tarjeta dificulta bastante la consecución de una precisión por debajo de los microsegundos, que por otra parte son requeridos por la mayor parte de los sistemas de monitorización.
- Reloj del PC. En su gran mayoría están basados en cristales de cuarzo económicos, los cuales ofrecen un error de 100ppmo lo que es lo mismo un par de segundos al día.
- El sincronismo entre distintos puntos de la red. Incluso si se emplean osciladores de alta precisión, pequeñas derivas temporales pueden echar por tierra la precisión requerida. Por lo tanto, se hace necesario el empleo de un reloj común para todos los puntos de medida.

Estos problemas hacen necesario el desarrollo de una solución Hardware. Esta implementada en una FPGA, debido a que estos dispositivos ofrecen la mejor solución teniendo en cuenta la versatilidad y rendimiento en volúmenes medios y pequeños de producción [3]. Hay dos opciones para la implementación. La primera opción sería crear una tarjeta PC-PCI , DAG Endace Cards son un buen ejemplo. La segunda sería usar un placa de desarrollo conectada a un dispositivo de mayor inteligencia, empleando un enlace Ethernet, Esta solución ha sido la elegida para este proyecto, ya que permite el ahorro de costes al no requerir de un PC en el punto de medida.
Los siguientes puntos detallan la solución propuesta.

1.1 Objetivos

La principal meta de este proyecto es desarrollar una plataforma de monitorización Ethernet no intrusiva. Deberá proveer un motor para la captura de paquetes y sus correspondientes marcas de tiempos, así como la generación de histogramas. Como ha sido mencionado en el punto anterior, todas estas operaciones serán implementadas en hardware. Debido a esto, este proyecto será desarrollado principalmente en VHDL.

El resultado final del proyecto servirá de referencia como una plataforma de monitorización basada en un sistema embebido en una FPGA. En lo referente al hardware el sistema contara con una FPGA y dos interfaces Ethernet, un banco de memoria y dos puertos de comunicación serie. El sistema interceptará una conexión. Los dos extremos de la conexión estarán conectados a los puertos Ethernet del sistema. En el interior de la FPGA, el tráfico entrante será dirigido hacia la otra interfaz de red, minimizando su retardo. A la vez, este tráfico será reenviado a los dispositivos de captura de paquetes y de marcas de tiempos

La información recopilada será almacenada en memoria y enviada. Desarrollos futuros sustituirán este Puerto serie por una tercera interfaz de Ethernet.

El sincronismo será conseguido mediante el uso de la tecnología GPS, que proveerá al proyecto de una referencia UTC próxima a la obtenida con un reloj atómico. Para ello, se ha pensado en una solución mixta hardware y software. Los GPS comerciales proveen dos señales complementarias NMEA, fácilmente tratado por un procesador, y PPS que requiere un uso hardware para lograr la máxima eficiencia

Hay que destacar, finalmente, que el desarrollo de un sistema de monitorización es un enorme proyecto que no pede ser tratado en un solo proyecto de fin de carrera. Es por tanto el fin de este proyecto, el establecimiento de un punto de partida sobre el que añadir nuevos componentes y características al sistema que permitirán el refinamiento de esta.

1.2 Retos

Por todo lo expuesto anteriormente este proyecto presenta multitud de problemas que han de ser resueltos. A continuación, se presenta una breve descripción de los mismos, con el fin de mejorar la comprensión sobre las decisiones y diseños del proyecto.

- **Frontera temporal:** Se define dominio temporal como aquellos elementos que reciben la misma señal de reloj. Hay varios motivos por los cuales la comunicación entre distintos dominios de reloj presentan varios problemas [34].
    - Los fallos no siempre se repiten.
    - Los problemas varían entre tecnologías e incluso entre dispositivos de la misma familia.
    - Las herramientas EDA no suelen detector este problema.

  El típico error producido en el intercambio de información entre diferentes dominios es la metaestabilidad [34].

  Hay algunas soluciones que pueden tratar de solventar este problema.

    - <u>Control de fase</u>: un PLL interno o incluso un DLL puede ser utilizado para permitir una comunicación libre de riesgos. No obstante, no siempre se puede utilizar ya que requiere que un reloj sea múltiplo del otro.

    - <u>Double flopping</u>: Consiste en dos Flip-Flops alimentados por el mismo reloj. La información llega de un tercer Flip-flop, pero que esta sincronizado con el otro reloj.

    - <u>FIFO structure</u>: Una fifo asíncrona puede resolver el problema. En la entrada los datos serán almacenados empleando un reloj. Por contra serán leídos bajo el otro dominio de reloj.

  Para una explicación más detallada de estos problemas y sus soluciones se aconseja fehacientemente la lectura de la siguiente referencia [34].

- **Precisión de las marcas de tiempos:**

Muchos proyectos relacionados con la monitorización de redes han sido desarrollados recientemente. No obstante, la precisión temporal aún le queda campo para mejorar [2][35]. El error de estas marcas decrementa el comportamiento de los sistemas. Por lo que la mejora de las marcas temporales es una obligación para este proyecto. El propósito inicial es lograr un error +/- 100 ns comparado con la señal PPS, proveniente de la fuente temporal. Esto significa un error de +/-150 ns sobre el tiempo UTC.

Dos ideas surgen a la hora de intentar esa mejora.

- Capa física: Si el proceso de marcado se realiza tan cercano como sea posible a la capa física, la precisión aumentará. Desarrollar este proyecto usando una FPGA permitirá aplicar esta solución.

- Deriva temporal: ningún reloj de cuarzo se ve exento de padecer la tan temida deriva temporal. [2]. Al usar el reloj de cuarzo para calcular las marcas temporales es fundamental tratar de lograr la máxima sincronización entre este reloj local y su referente global. Con ese fín, se propone el empleo de un algoritmo de corrección de la deriva temporal.

- **Escalabilidad:**

El diseño de un sistema de monitorización para Ethernet con una velocidad de 10/100 se antoja una tarea bastante complicada. No obstante, resultaría un esfuerzo fútil si el diseño no fuese escalable. En este proyecto, la arquitectura se ha pensando de tal manera que permita soportar velocidades de hasta 1 Gbps. Incluso, lo ideal sería que la velocidad pudiera ser configurada a las necesidades puntuales del usuario.
Pero en cualquier caso, en esta etapa inicial, los resultados serán publicados para una velocidad de 10/100

- **Eficiencia de costes:** Sin perder de vista la escalabilidad es igualmente importante que el proyecta cumpla con un cierto limite en sus costes. Debido a ello y a pesar de que será implementado bajo una FPGA de la familia Virtex II Pro, los requisitos de diseño para serán tenidos en cuenta implementarla en una familia de más bajo como Spartan 3.

1.4 Organización de la memoria

Esta memoria ha sido organizada en varios capítulos.

- **Introduction:** esta sección debería de responder a la pregunta. ¿Por qué hacer este proyecto?

- **State of Art**: Pretende dar un conocimiento general sobre ciertos de los aspectos tecnológicos que atañen a este proyecto.

- **Design**: Presenta un estudio de las diferentes soluciones para resolver los distintos retos y objetivos del proyecto.

- **Implementation**: Ofrece un explicación de las soluciones elegidas y de su funcionamiento.

- **Tests and results**: Muestra los experimentos y análisis llevados a cabo para estudiar el rendimiento y las características del proyecto.

- **Conclusiones y trabajo futuro**: Un resumen de los resultados del proyecto y de su deposable desarrollo futuro.

1 Introduction

1.1 Motivation

Nowadays there are many different commercial and research activities (i.e. Q.O.S. measurement, SLA verification and traffic engineering [1][2]) that require to establish, with really good accuracy, the real-time status of the network in different points.
The huge increases in the number of nodes, the evolution of network services requirements, the higher data rates and the mix between the different kinds of technologies have also created a clear need for the comprehension, in great detail, of the traffic interactions. In particular, these analysis demand high precision measurements of the network latency.

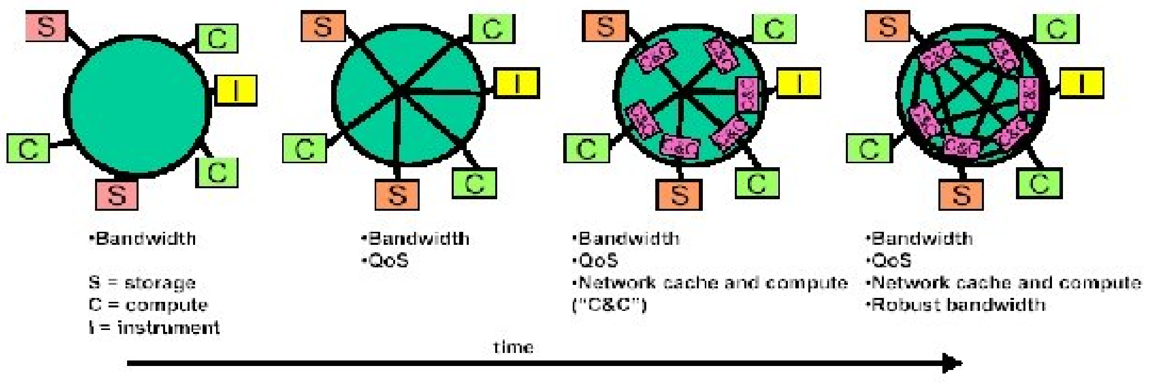


Figure 1-1: Evolution of Network Services Requirements over Time [9].

As [14] says, there is a big gap defined as "what we measure/understand, and what is really going on out there". "It is already large and is probably getting larger".
Below it can be seen the increase of the unaccounted-for traffic (the gap) through 2 years.

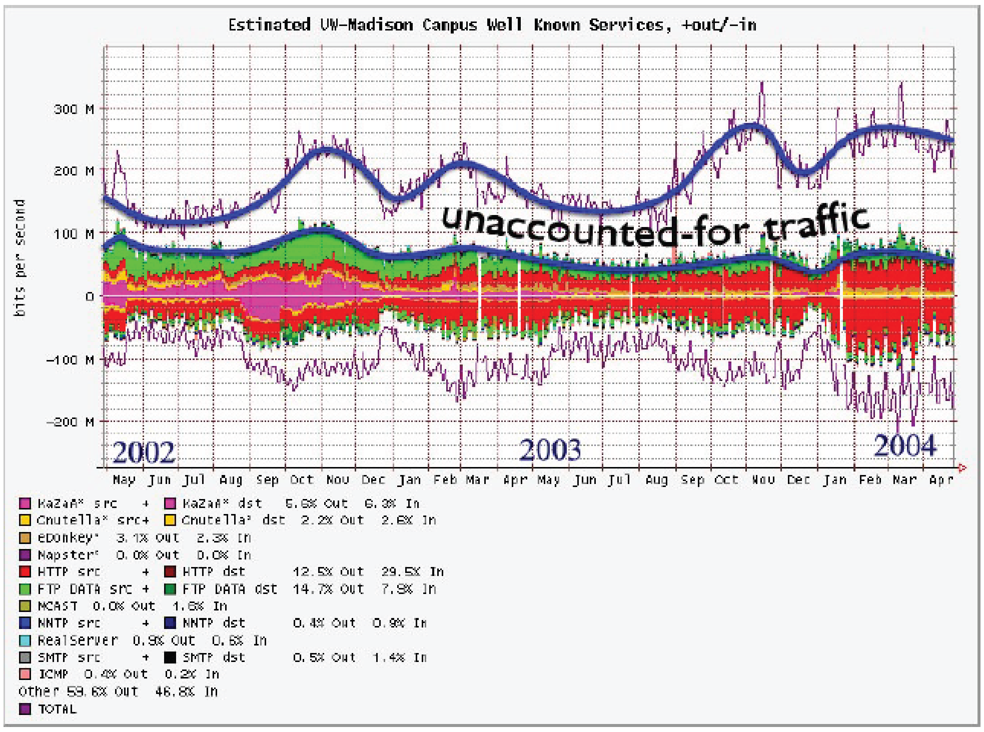


Figure 1-2: The Gap [14]

The following paragraph explains quite well the motivation for researching in monitoring systems. It has been taken from the objectives webpage [12] project.

"To make an analogy from the physical sciences, network traffic monitoring systems are to the networking scientists as necessary as telescopes are to astronomers. Indeed, in the same way that a telescope monitors the outer space and records all interesting astronomic events, a traffic monitoring sensor monitors the cyberspace and records all interesting networking events. It is difficult to imagine where astronomy would be today if Galileo hadn't turned his telescope up towards the distant skies. On the contrary, it is easy to realize how this single monitoring instrument, i.e. the telescope, enhanced our understanding of the outer space, and eventually improved our life on earth. It is our belief that advanced network traffic monitoring sensors can have a similar impact to our understanding of the cyberspace, and can result in a significant improvement of the quality of our everyday lives".

This master thesis aims at solving this engineering problem over Ethernet networks. A first approach to this project could be an all-software solution running over a standard PC, maybe equipped with a real-time operative system. However, there are many reasons that prevent us from taking this simple approach. Here is a brief summary of the drawbacks that this solution might have [1][2]:

- Delay when processing the interrupts generated by the Ethernet card. Even if a real-time operating system is being used, the uncertainties introduced by the card FIFO would make very difficult to achieve the microsecond precision required by most analysis.
- Clock of the PC. It is commonly based on an inexpensive quartz crystal, which has an error of about 100 ppm (that is, a few seconds per day).
- Clock synchronization between different points of the network. Even if highly stable clock oscillators are used, minimal drifts could jeopardize the microsecond accuracy required for latency measurements. It is thus necessary to have a common clock in all measurement points.

These problems make necessary to develop a hardware solution. It will be implemented on a FPGA, because programmable logic devices provide the best tradeoff between versatility, cost and performance for small to medium volumes of production [3]. There are two options for implementing an FPGA based solution. The first option would be to create a PC-PCI card, DAG Endace Cards[1] are a pretty good example. The second solution is to use a development board connected to an upper intelligent device by an Ethernet connection. This option is the chosen one for this project, because it permits saving money due to avoiding the requirements of a PC in the measurement point.
The following points detail the proposed solution.

1.2 Objectives

The main goal of this thesis is to develop a framework for non-intrusively monitoring Ethernet networks. It will provide a basic engine for packet capturing and times-tamping, as well as histogram generation. As it was mentioned in the previous point, all these operations need to be implemented in HW, so this project will mainly use VHDL.

The final outcome of the thesis will be a reference implementation of the monitoring framework on a simple FPGA-based embedded system. At the HW level, this system will be composed by an FPGA with two Ethernet interfaces, some external memory and two serial communication ports. Basically, the system will be placed in-between an existing Ethernet connection that will be cut in half. The two endpoints of the cut-through connection will be connected to the two Ethernet ports of the system. Inside the FPGA, the data coming from one interface will be redirected to the other one with the minimal delay and, at the same time, it will be forwarded to the packet capturing and time-stamping engine.
The information gathered by the engine (captured and time stamped packets, as well as traffic histograms) will be stored in a memory and sent via a serial port to a host computer. Future developments will substitute this serial port for a third Ethernet interface.

Time synchronization will be achieved using GPS that will provide a common UTC clock reference with near-atomic accuracy. In order to do so, a mixed HW/SW solution is required. Commercial GPS receivers provide two complementary signals: NMEA, which is more easily managed with a conventional processor, and PPS, that should be handled in HW to achieve the highest precision.

Finally, it should be remarked that a complete Ethernet monitoring system is a huge project that could not be accomplished in just one master thesis. The main purpose of this project is just to provide the starting point for other thesis, which will add more components to the system and will also refine the monitoring framework developed here.

1.3 Challenges

In this project there are many problems that must be solved. A brief descriptive list is given in order to try to improve the general understanding about both designs and decisions of the project.

- **Time Border**: [34] A clock domain is where all the elements in a design receive the same clock signal. There are several reasons that makes clock domain communication a major problem:
    - Failures are not always repeatable.
    - Problems vary from technology to technology and even between devices.
    - Such problems are difficult to detect in current Electronic Design Tools.

  The typical error committed as a time border cross is the metastability. This term refers to a "timing violation, when the data input to a Flip-Flop transitions within a window around the active clock edge as defined by the setup and hold times"[34].

  There are some solutions that can be employed in order to solve this issue:

    - Phase control: An internal PLL(Phase locked loop) or a DLL(Delay locked loop) are employed in order to permit the data transition without any risk. However, this is not always possible due to adjusting the phase of a clock may not be possible. In other words, one clock must be multiple of the other.

- Double flopping: It consists on two Flip-Flops that are feed by a clock. The information comes from a third Flip-Flop, this time synchronized by a different one.

- FIFO structure: An asynchronous FIFO can be implemented or included in the design. In the input side of the FIFO, the data will be stored using one clock domain and at the output one, the data will be read synchronize with the other clock domain. Of course, with this solution it should be employed a handshake protocol and it would also be advisable to prior know the transmission rate, with the intention of define the maximum queue size.

For a further explanation about this problem and its solutions, it is strongly recommended to read the reference provided [34].

- **Time-Stamp accuracy:**

Many projects concerning a monitorization device have been developed these days. However, the time accuracy is still not enough accurate or at least it could improve [2][35]. Hence, the weight of the timing error is still quite big, so it decreases the general performance of the monitoring system.

Improving the time-stamp accuracy is a must in this project. The initial aim is to achieve a +/- 100 ns error from the Pulse per Second signal, provided by the time source device, what it means about +/-150 ns from the UTC time.

Two different approaches are combined in this project for targeting the fulfillments of the initial constraints.

- Physical layer: If the time-stamp procedure takes place as close as the physical layer as possible, the accuracy improves. Developing the project on a FPGA development board will help in this duty.

- Temporal drift: Any quartz clock device suffers from time drift [2]. It is a must to try to have the time-stamp clock source as synchronized as possible to an external, global time source. The application of a drift correction algorithm is the proposed method.

- **Scalability:**

Designing a 10/100 accurate monitorization device could seem rather complicated. However, it would almost be a completely waste of time if it is not easy scalable. In this project the general architecture must support 1 Gbps, with a few changes. Furthermore, ideally, the network interface speed should be customizable by the user.
Nevertheless, the first results will be published under a 10/100 structure, relaying the rest of the improvements as future work.

- **Cost efficiency:**

At the same time that is important to keep an eye on scalability, it can not be forgiven the needs of the product for meeting some cost limits. Therefore, the project even though is going to be developed in a Virtex II pro FPGA, the constraints are fulfilled for working on a cheaper Spartan 3.

1.4 Thesis Organization

This thesis has been organized in several chapters:

- **Introduction:** This section should give the answer to the following question. Why this project?
- **State of Art**: It aims to provide a general knowledge about different project aspects, other related projects and technologies that are likely to be employed in the project.
- **Design**: It presents a study of the different solutions for solving the challenges, problems and objectives of the project.
- **Implementation**: It gives an explanation of the chosen solution and how it works.
- **Tests and results**: It shows experiments and studies of the implemented device and its characteristics.
- **Conclusions and future work**: It illustrates an analysis of the outcomes of the project and new challenges & ideas for continuing it.

2 State of Art

2.1 Network Monitoring Systems

2.1.1 Definition and characteristics

The term **network monitoring** describes the use of a system that monitors a computer network targeting several objectives that can be used in the network management. These aims can be from knowing the traffic source to attacks detection.

The network monitoring systems can be classified in two:

- Active measurement:[9] It means injecting traffic into the network and observing its response. It requires cooperation between the sender and the receiver participating in the measurement. Usually, these methods send testing flows from one source to multiple receivers. The results are then processed in order to know some characteristics of intermediate links like delay, or loss rate.

- Passive measurement: [2] It consists on observing packets on a data link or network media without generating any additional traffic which may disturb the existing network performance. Its operation can be explained as a process where the following steps are given. The measuring host applies a time-stamp to each incoming packet. The entire packet, or a part of it such as headers or quintuple, is then optionally passed through a filter that selects packets by already defined criteria, attending to the packet information stored.
  The chosen packets are then available for further analysis or storage as a packet trace file for later use. These traces can be used for a wide number of applications, from network diagnostic to usage patterns.

2.1.2 Related work

There is a clear need for attempting to reduce the gap described in 2.1.1. Besides, network monitoring systems must improve also for detecting attacks as fast as possible, in order to increase the broadband security. Many projects have already attempted to increase both speed and reliability in monitoring, and nowadays there are some others trying to do it too.

Some of the past and present projects can be checked in the following list where a brief resume of each one is given.

- **SCAMPI**: it was a two-and-a-half-year European project to develop a scalable monitoring platform for the Internet[11]. It aimed to promote the use of monitoring tools for improving services and technology, developing a network adapter, initially at 10 Gbps speed.. Other goal was to investigate the technical challenges of developing systems for 100 Gbps speeds and beyond.

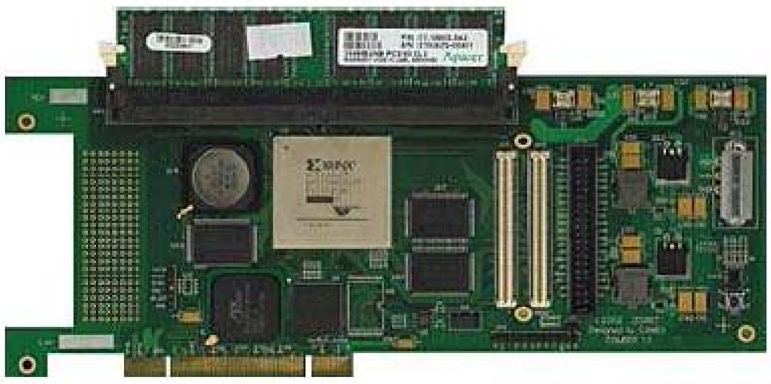


Figure 2-1: The COMBO6 motherboard [14]

The COMBO6 motherboard is one of the deliverables from SCAMPI project. It is based on a XILINX Virtex II and in a PCI port.

The project started on April 2002 and ended on September 2004.

URL: http://www.ist-scampi.org

- **LOBSTER**: it was a pilot European infrastructure for accurate Internet traffic monitoring, based on passive monitoring at speeds starting from 2.5 Gbps, and possibly up to 10 Gbps.[12]

The devices that have been employed are the ones developed in SCAMPI project which allow monitoring at 10 Gbps, combining new hardware and software components.

Instead of collecting flow-level traffic summaries or actively probing the network, passive network monitoring records all IP packets (both headers and payloads) that flow through the monitored link more accurately than methods based on flow-level statistics or active monitoring.

The passive monitoring applications running on the systems have been developed based on MAPI (Monitoring Application Programming Interface), which has been developed in both SCAMPI and LOBSTER projects. MAPI allows the implementation of the monitoring needs. Therefore, selecting the relevant information, and balancing the monitoring overhead with the amount of the received information.

LOBSTER project has developed and also improved the performance of security monitoring applications which for example helps in the detection of the spread of zero-day worms. It has also executed the appropriate data anonymization tools for prohibiting unauthorised access to the original traffic data, it has created and European Internet measurement system and among other objectives an accurate traffic characterization for programs using dynamic ports.

The LOBSTER project concluded on 30 June 2007, but the sensor network remains operational, and the sensor software continues to be maintained.

URL: http://www.ist-lobster.org

- **MOMENT**: Monitoring and Measurement in the Next Generation Technologies[9]

In previous projects like LOBSTER, large efforts have been taken to successfully develop and deploy measurement infrastructures to gain insight into the operational network, without being intrusive from the perspective of commercial ISPs. Repositories like MOME were dedicated to keep an overview on available measurement data, tools, infrastructures and projects.

The Project MOMENT continue this process by research and development towards the following objectives

  - To create added value from single monitoring and measurement infrastructures by the integration of the results
  - To define a mediator architecture providing a unified interface using a Service-oriented Architecture for Future Internet applications

Moment partners have developed several measurement infrastructures within the European Sixth Framework Programme. The developments of Moment will facilitate network applications access to measurement data collected from different infrastructures, offering integrated results. For achieving these results a mediation engine will be implemented as the unified interface to handle application requirements and delivering appropriate results. This mediator architecture should be an open and flexible platform which permits not only easy integration of partners developed architectures but also new ones.

The communication between mediator, applications and measurement tools is wrapped in XML format.

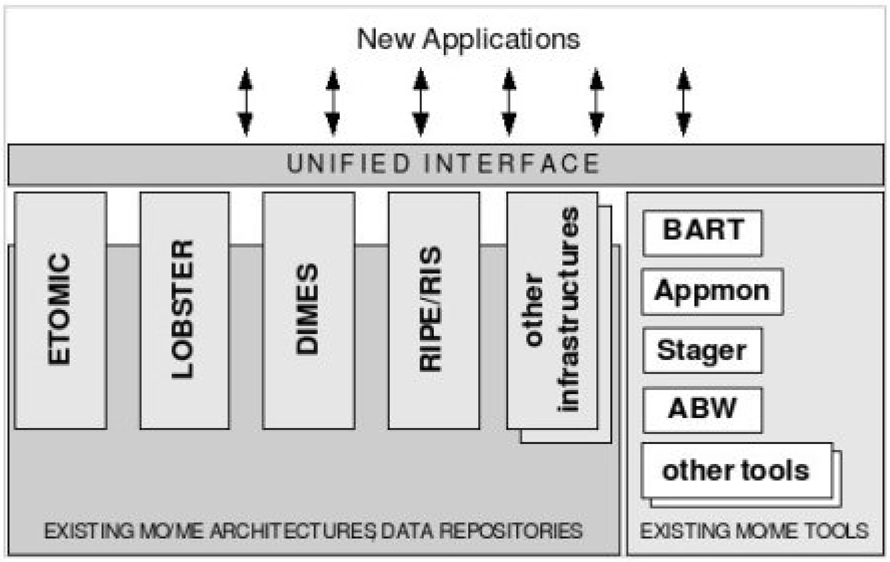


Figure 2-2: Integration (mediation layers) [9].

URL: http://www.fp7-moment.eu/

- **EVERGROW**: it is a project that aims to generate a “virtual laboratory” with the purpose of providing a test environment where new services can be developed and evaluated. “The key objective is the integration, in a single project, of the advances in knowledge of data networks and the content they carry” [24]. This virtual laboratory will have “equipment” like archived measurements, and tools for simulate future conditions.

The laboratory will be based on grid computing, that according to Sun Microsystems [22] it is a computing infrastructure that provides dependable, consistent, pervasive and inexpensive access to computational capabilities. By pooling federated assets into a virtual system, a grid provides a single point of access to powerful distributed resources. It delivers on the potential in the growth and abundance of network connected systems and bandwidth: computation, collaboration and communication over the Advanced Web.

URL: http://www.evergrow.org/news.php

- o DIMES is a subproject of the EVERGROW. It is a distributed scientific research project, aimed to study the structure and topology of the Internet, with the help of a volunteer community [23].

  URL: http://www.netDimes.org

- o ETOMIC is a subproject of the EVERGROW. The European Traffic Observatory Measurement InfrastruCture[15].

It was created in 2004-05. It aims to grant a public testbed (platform for experimentation for large development projects) for active measurement methods in Europe. The nodes are re-configurable, GPS synchronized and therefore quite accurate (ns).

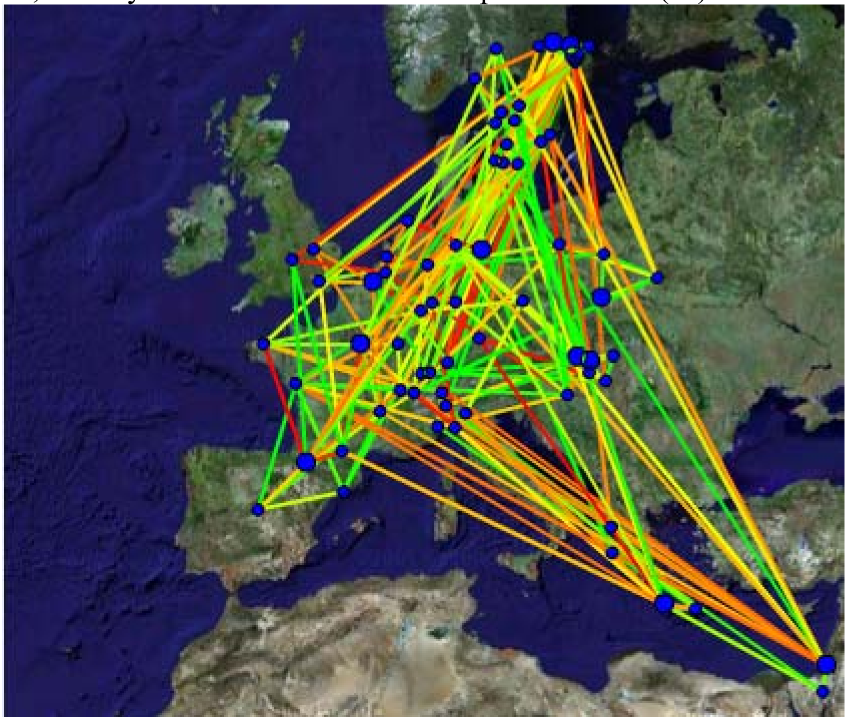


Figure 2-3: ETOMIC Network [15].

The ETOMIC infrastructure was awarded by the Best Testbed Award on the TridentCom 2005 Conference, and it was opened and introduced to the community on the INFOCOM 2006 Conference.

URL : http://www.etomic.org/

- **RIPE** Test traffic Monitoring (RIPE TTM) [25][26]

RIPE NCC offers Test Traffic Measurements as a service, to its users. It measures key parameters of the connectivity between client site and other points on the Internet. TTM allows to comprehensively and continuously monitor the connectivity of your network to other parts of the Internet, recording one-way delay and packet-loss measurements as well as router-level paths ("traceroutes") between a large set of probe machines ("test boxes"). Every test box includes a precise time source - typically a GPS receiver. The TTM project has provided extremely high-quality and stable delay and loss measurements for a large set of network paths (IPv4 and IPv6) throughout Europe covering at the same time an interesting mix of research and commercial networks.
Currently offers daily plots with the following information:

  - Delay measurements
  - Packet loss rates
  - Delay variations (jitter).
  - Bandwidth

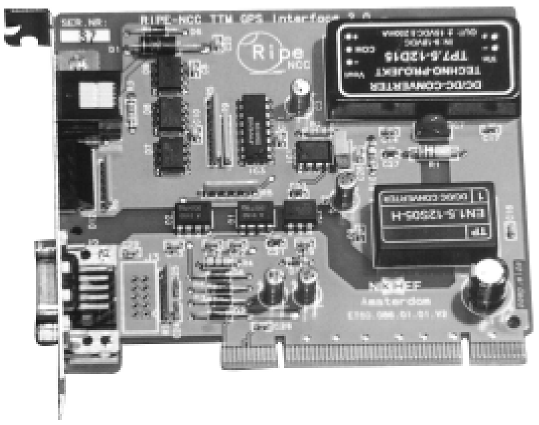


Figure 2-4: RIPE NCC Test-Box [26]

URL : http://www.ripe.net/ttm/

2.1.3 Available services

An excellent repository of monitoring services is included in this deliverable from the MOMENT project [9].

http://www.fp7-moment.eu/images/FP7-MOMENT-WP2-D2.1.pdf

2.2 Alternatives for the implementation of embedded systems

An embedded system is a special-purpose digital system designed for the fulfilment of one or several functions.

Engineers can usually find a solution for implementing a function on several technologies. Any of them may resolve mostly of the problems, and it is an engineer work to decide which one could be more suitable for his purpose. This decision will be influenced by the conscripts of the project (e.g. budget, timing, development cost).

Some advantages and disadvantages, according to this master thesis, will be included in the brief description of these devices. The different solutions are:

2.2.1 Microcontrollers

Microcontrollers are usually the cheapest & simplest solution for solving none demanding high speed or versatility problems. Basically, they consist on a CPU with some peripherals as timers, counters, PWM, external memory, RS-232 and so on. The usually offer a huge integration, and a low power consumption.
As [31] says… *"Microcontrollers, the little 35-cent parts in ancient technology that runs toasters?"*
However, in the same blog entry it is later explained that the trend for microcontrollers is changing. CAP MCU new devices, they are basically an ARM-based system-on-chip with a fixed selection of peripherals and a Metal Programmable (MP) block, are aiming to the DSP-FPGA market. They offer customizable logic at minimum power consumption [31] [32][33]. The general impact of these new devices, in the embedded system market, is still unknown.

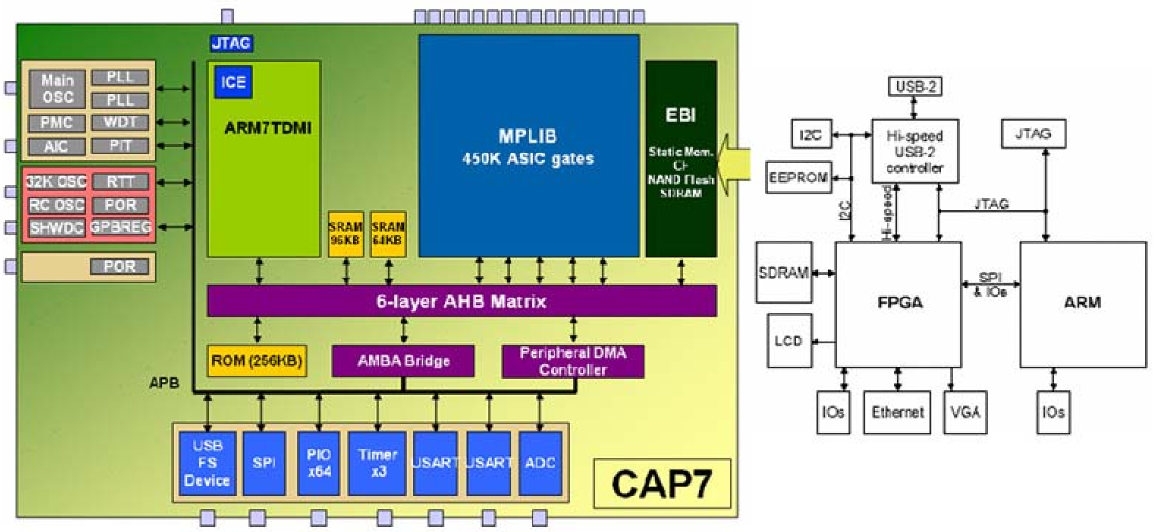


Figure 2-5: Comparing the architectures of CAP7 and the Xylo-L board. The MPLIB in CAP7 takes the place of an FPGA[33].

The development time is the fastest of all the embedded technologies but unfortunately they can not be the solution for this project. Mainly because it is required to work close to the physical level in order to fulfil the accuracy requirements.

2.2.2 ASIC

It is an integrated circuit customized for a particular use, rather than intended for general-purpose use. Most designers use ASIC only for non field programmable devices.
Application Specific Integrated Circuits (ASICs) allow full customizability and utilize low-power standard cells, achieving much better performance than possible with a similarly configured FPGA. However, ASIC design and production can be prohibitively expensive for smaller companies or those producing only a limited quantity of product [33].

The main disadvantage for using it in this project is the high cost that carries both developing and supporting.

2.2.3 FPGA

The “Field Programmable Gate Array” technology born in 1985. Its parent is the CPLD, complex programmable logic. Its main objective was to introduce two key points in implementation; these were flexibility and user-programmable.

The FPGA’s have been improved through these years adding new features which have boost up its capabilities. Some of these are the partial re-configuration, the embedded microprocessor and the easy add on of peripherals.

In many aspects, like development cost, speed, are the middle solution between microcontrollers and ASIC. However, they are power hungry and can quickly overwhelm the power budget of an embedded system [33].

For this project is quite clear that a FPGA is the perfect solution. Several reasons support this decision. An FPGA allows customizing a physical level and to fulfill the bandwidth requirements. This is so important in order to achieve better timing results. Besides, re-configuration adds a superb capability for modifying the filtering options in real time. And even more, EDK tools permit to rocket the developing time building a System On Chip in a simpler way than designing and ASIC.

2.2.3.1 FPGA-based Embedded Microprocessors

Nowadays, there are several embedded microprocessors solutions. They can be classified in several groups according to [20]. These are hard, firm and soft cores.

- **Hard cores** are optimized for a specific silicon technology and can not be modified by the system designer. These cores have a pre-defined layout and floorplan that is included in the architecture of the design. They have the advantage of fixed timing and can be treated as library elements during the design cycle. The disadvantage of hard cores is that it lacks customization capability which can generate problems for meeting some requirements.

- **Firm cores** are delivered as a mix of source code and technology- dependent netlist. In these cores, the source code is visible to the designer and specific parts of the core can be customized by the designer. However, a netlist is technology-specific and the user can not easily switch chip vendors or use a different technology with the same vendor.

- **HDL-based soft cores** offer full technology independence and flexibility. The design can be readily modified or resynthesized to multiple technologies in order to switch vendors or target a new process. The disadvantage with soft cores is that critical timing is not guaranteed and the core must be synthesized, floorplanned, and placed-and-routed for each use.

	Hard Cores	Firm cores	Soft Cores
Hardness	Pre-defined layout	Mix of source code and technology-dependent netlist	Delivered as behavioral source code
Modeling	Modeled as a library element	Mix of fixed and synthetizable blocks Can share resources with other cores	Synthesized with other logic
Flexibility	Can not be modified by designer Several hard cores on one chip can result in inefficient system implementation	Techonology dependant Specific functions customizable	Design can be modified Technology indenpendent
Predictability	Timing guaranteed	Critical path timing fixed	Timing not guaranteed Can not be fully verified before released to the user

Table 2-1 Embedded cores [20]

Now it is time to look to the market for seeing the different options provided by the vendors.

Category	**Nios II**	**MicroBlaze**	**LEON 3**
Maximum MHz	200 MHz	200 MHz	125 MHz
Reported DMIPS	150 DMIPs	166 DMIPs	85 DMIPs
ISA	32 bit RISC	32 bit RISC	32 or 64 bit RISC
Cache Memory	Up to 64kB	Up to 64kB	Up to 256kB
Floating Point Unit	IEEE 754	IEEE 754	IEEE 754
Pipeline	6 stages	3 stages	7 stages

Custom instructions	Up to 256 Instructions	None	None
Register File Size	32	32	2 to 32
Implementation	FPGA	FPGA	FPGA
Area	700-1800 Les	1269 LUTs	N/A

Table 2-2 FPGA based embedded cores [21]

2.3 Synchronization methods and Time Sources

In order to be synchronized, an event occurring simultaneously at all measurement points should receive the same time-stamp from all of the measurement systems. Therefore, synchronized clocks are "compulsory" when measuring a network at a single point, in order to maintain packet arrival order information.

The clock signal used for generating time-stamps can come from a variety of sources like for example, a host computer, an interface bus clock, a locally hardware, or an external clock.

If the clock is host generated or local, it is most likely based on a crystal oscillator. A crystal oscillator is typically a quartz crystal that has been cut for oscillating at a desired frequency. Unfortunately, the frequency of a crystal oscillator will vary with age, supply voltage, and temperature [46]. Table 2-3 shows the typical frequency error of different types of crystal oscillators in parts per million [I].

Drift rates are often expressed in units of parts per million (ppm), which is the number of microseconds a clock drifts from UTC every second.
A 100MHz crystal oscillator with an error of 100ppm could reach, after 24 hours, an accumulated error of 8.64 seconds, due to its frequency 10 kHz frequency error.

Type	Error(ppm)
Normal	15-100
Precision	5
Temperature Compensated	0.3-5
Oven Controlled	0.2

Table 2-3 Typical Error in Crystal Oscillator Frequency [2].

Inexpensive crystals with frequency errors in the range 50–100ppm are commonly used to provide CPU and bus clocks in PCs, as the exact frequency in these systems is unimportant, as a result they are a poor choice for time-stamp clocks. Oven controlled crystal oscillators have the best stability. They work by thermally insulating the crystal, and heating it to a constant temperature above the maximum ambient temperature under thermostatic control. This makes them large and expensive, and requires a warm-up period before the oscillator is within its specifications. The heating elements may also consume a lot of power, especially during the warm-up period. Even with the best available crystal oscillators, drift is significant, and errors of the magnitude of a minimum sized packet time will accumulate quickly [2].

There are many sources for an accurate global time-source. Relatively inexpensive receivers for the US Global Positioning System (GPS) can provide a pulse every second, with the rising edge accurate to within plus or minus 100ns to UTC. However, GPS receivers require a relative clear view to the orbiting GPS satellites, in order to receive enough signal, usually requiring mounting on the roof of the building. This solution will be explained in more detail in the section 2.3.3.

Digital cellular telephone systems such as CDMA maintain a global network time to coordinate multiplexing. These signals are often strong enough to penetrate buildings, and CDMA receivers emit time pulses as GPS receivers. Nevertheless, attention must be paid to the distance from the cellular base station because the propagation time is unknown, which can lead to an unknown offset from UTC. For solving this issue, it may be necessary to survey each site where CDMA is employed with more accurate timing equipment in order to determinate this offset.

After choosing a right time source it is time to find a synchronization solution. In the following tables there is a comparison between different technologies [6].

	1588	**NTP**	**GPS**	**SERCOS**
Spatial extent	A few subnets	Wide area	Wide area	Local bus
Communi-cations	Network	Internet	Satellite	Bus
Target Accuracy	Sub-microsecond	Sub-microsecond	Sub-microsecond	Sub-Microsecond
Style	Master/slave	Peer ensemble	Client/server	Master/slave
Resources	Small network message and computation	Moderation network and computation	Computation	Moderate
Latency correction	Yes	Yes	Yes	No
Protocol specifies security	No	Yes	No	No

Administration	Self organizing	configured	N/A	Configured
Hardware?	For highest accuracy	No	RF receiver	Yes
Update interval	~2 seconds	Varies, nominally seconds	~1 second	Every TDMA cycle, ~ms

Table 2-4 Comparison between different synchronization solutions [6]

With the proper time source and synchronization solution, the measurement system can time-stamp each signal as it would a packet arrival. These synchronization time-stamps can then be stored and used in post processing to correct the packet time-stamps, or they can be used in real-time just associating them to each incoming packet.

Synchronization is key for the proper function of the nowadays networks and service providers want a reliable way to extend and distribute synchronization in next generation networks.

2.3.1 Precision Time Protocol (PTP)

The Precision Time Protocol (PTP) is defined in the IEEE1588 – 2002 standard. It enables precise synchronization of clocks in multicast capable networks like Ethernet [6].

[7] "IEEE 1588 is designed to fill a niche not well served by either of the two dominant protocols, NTP and GPS. IEEE 1588 is designed for local systems requiring very high accuracies beyond those attainable using NTP. It is also designed for applications that cannot bear the cost of a GPS receiver at each node, or for which GPS signals are inaccessible."

PTP is aimed for relatively localized systems typical of industrial automation, test and measurement. It tries to achieve sub-microsecond synchronization and it can be used on any Local Area Network technology that supports multicast communication, therefore is not limited to Ethernet.
Besides, it has also other objectives like to support heterogeneous clock systems which can have different precision, resolution and stability. At the same time, it should consume minimal resources on networks and hosts.

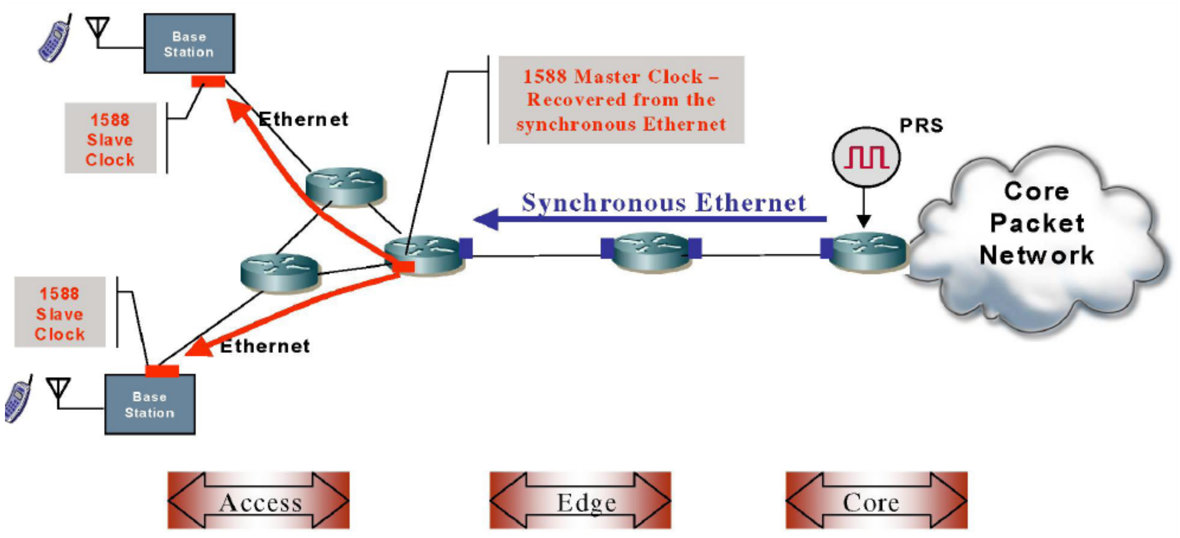


Figure 2-6: IEEE-1588 and Synchronous Ethernet implementation [8]

The key for understanding this protocol is to see the need of a set of slave devices for discovering the offset between time measurements on their clocks and time measurements on a master device [6]. Let the variable t represent physical time. For a given slave device, the offset o(t) at time t is defined by:

$$o(t) = s(t) - m(t)$$

Where s(t) represents the time measured on the slave device's clock at physical time t, and m(t) represents the time measured on the master device's clock at physical time t.
The synchronization is realized in the following order:

Step 1:

- Sync message sent from master to slaves.
- All Sync message detectors (SMD) note the time on the local clock this message appears at the SMD.

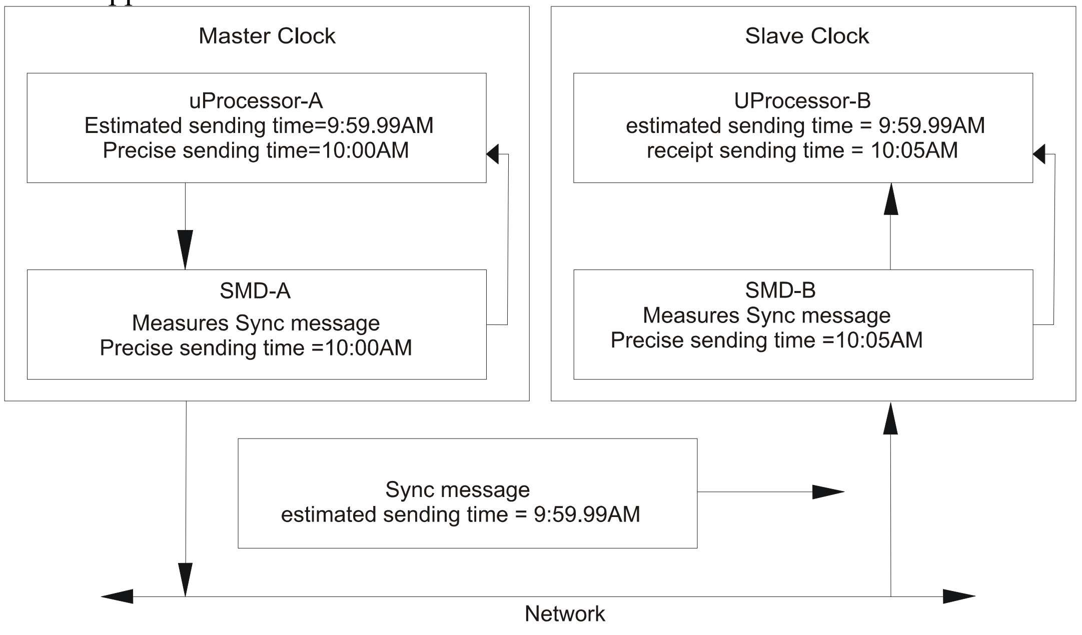

Master Clock
uProcessor-A
Estimated sending time=9:59.99AM
Precise sending time=10:00AM
SMD-A
Measures Sync message
Precise sending time =10:00AM
Slave Clock
UProcessor-B
estimated sending time = 9:59.99AM
receipt sending time = 10:05AM
SMD-B
Measures Sync message
Precise sending time =10:05AM
Sync message
estimated sending time = 9:59.99AM
Network

Figure 2-7:IEEE 1588 synchronization Step 1 [6]

The master time stamps the time T1 measured on its clock when it sends the sync message. A slave receiving this message takes note of the time T2 measured on its clock when it receives this message. If d is the transit time of this message, and $\tilde{o}$is the constant offset during this transaction, then:

$$T_2 - T_1 = \tilde{o} + d$$

Step 2:

- Follow-Up message containing precise sending time sent from master to slaves • All slaves compute offset and correct their clock

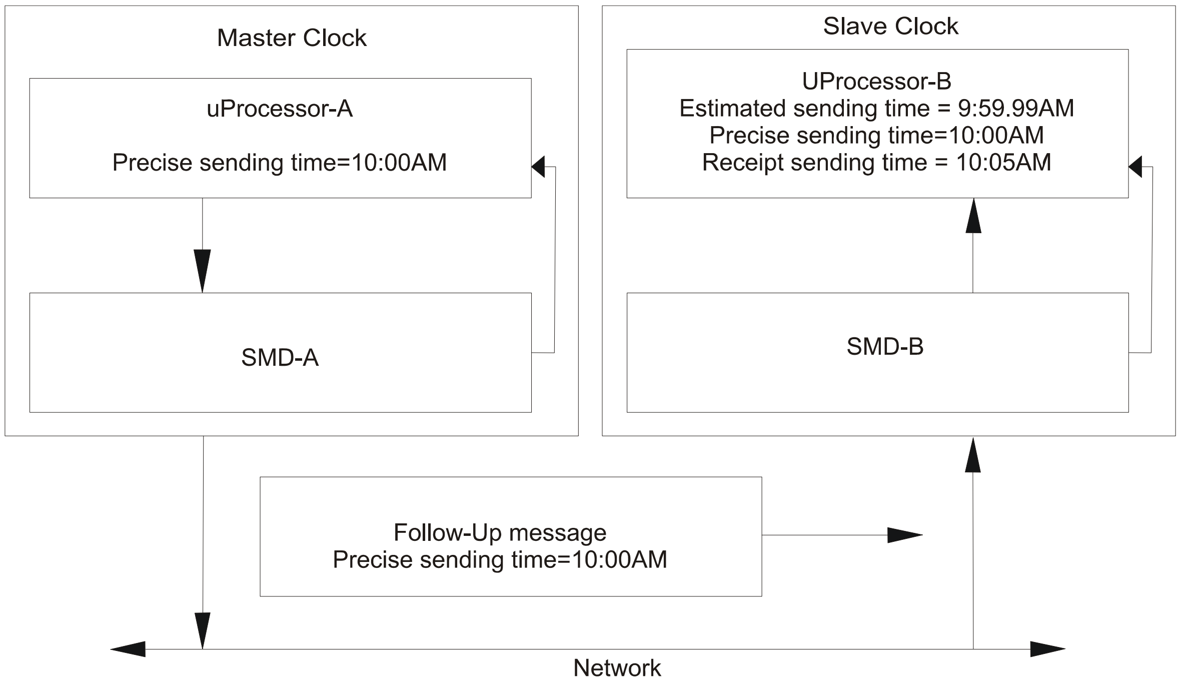
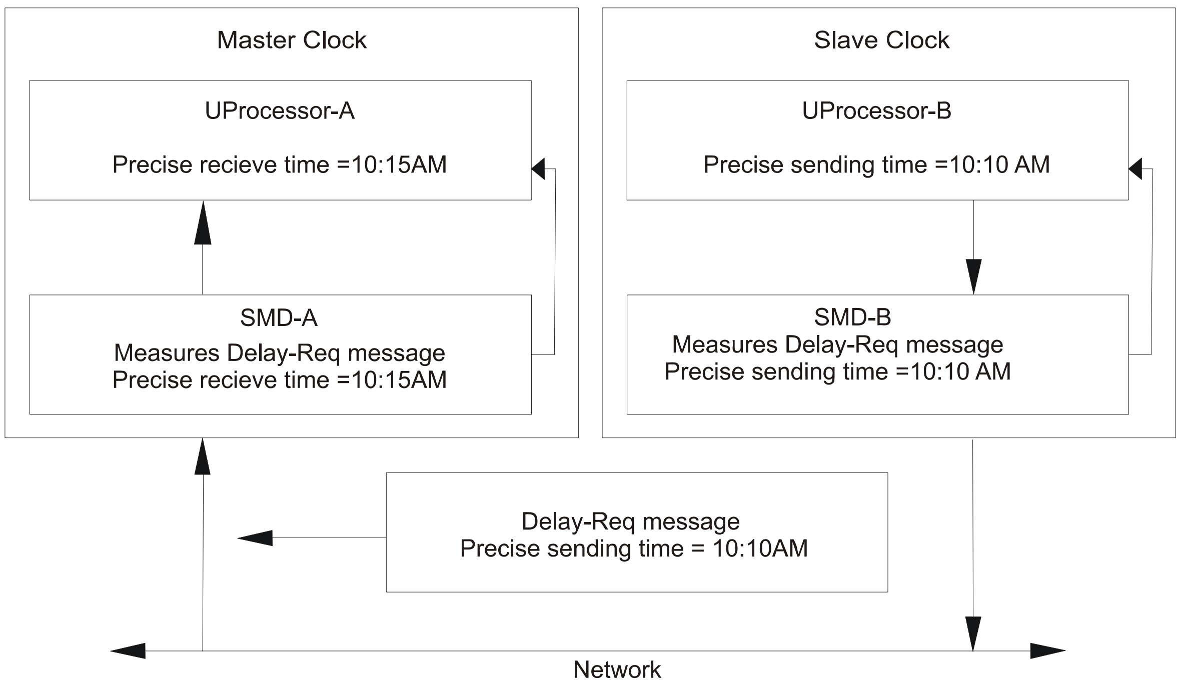

Master Clock
uProcessor-A
Precise sending time=10:00AM
SMD-A
Slave Clock
UProcessor-B
Estimated sending time = 9:59.99AM
Precise sending time=10:00AM
Receipt sending time = 10:05AM
SMD-B
Follow-Up message
Precise sending time=10:00AM
Network

Figure 2-8 IEEE 1588 synchronization Step 2

The master next sends a follow up TIME T1 message to notify the slaves of which time it measured when it send the SYNC message. Each slave now knows T1 and T2.

Step 3

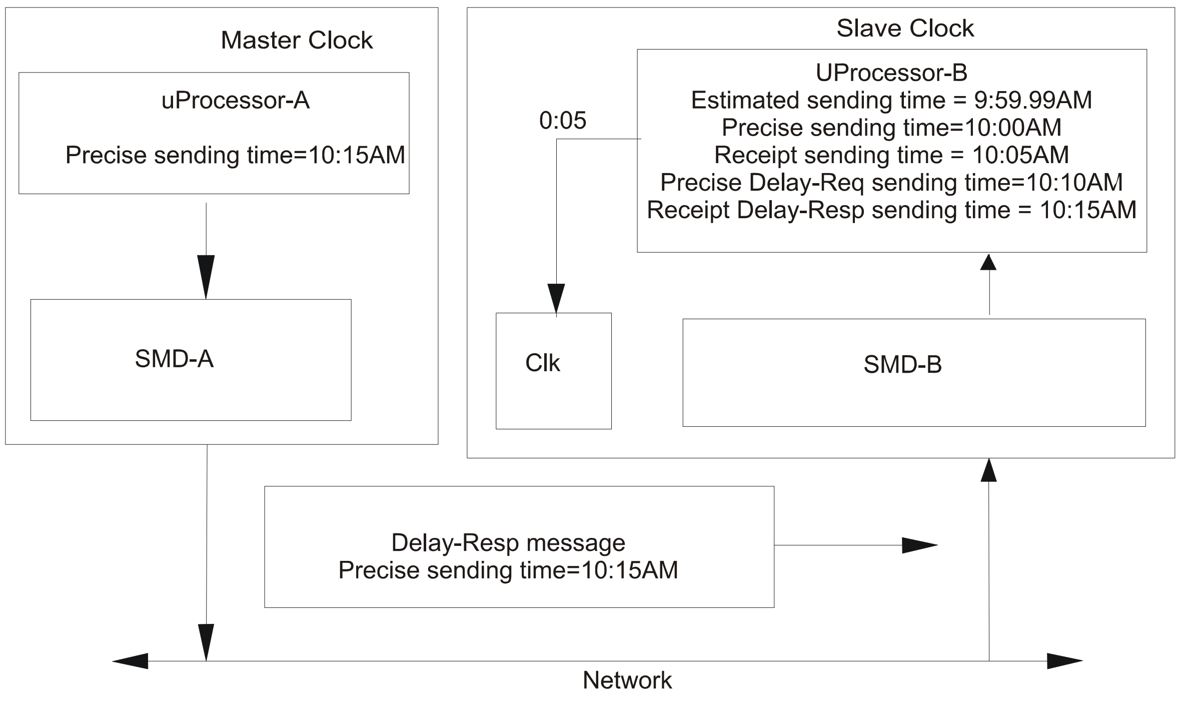


Master Clock
UProcessor-A
Precise recieve time =10:15AM
SMD-A
Measures Delay-Req message
Precise recieve time =10:15AM
Slave Clock
UProcessor-B
Precise sending time =10:10 AM
SMD-B
Measures Delay-Req message
Precise sending time =10:10 AM
Delay-Req message
Precise sending time = 10:10AM
Network

Figure 2-9 IEEE 1588 synchronization Step 3

Each slave now sends a Delay-Req message back to the master.. The slave measures the time T3 that it sends this message, and the master measures the time T4 that it receives this message.

Step 4:



Master Clock
uProcessor-A
Precise sending time=10:15AM
SMD-A
Slave Clock
UProcessor-B
Estimated sending time = 9:59.99AM
Precise sending time=10:00AM
Receipt sending time = 10:05AM
Precise Delay-Req sending time=10:10AM
Receipt Delay-Resp sending time = 10:15AM
0:05
Clk
SMD-B
Delay-Resp message
Precise sending time=10:15AM
Network

Figure 2-10 IEEE 1588 synchronization Step 4

The master then sends a directed multicast TIME T4 message back to the slave to notify the slave what time it received the RESPONSE message. Note that

$$T_4 - T_3 = -\tilde{o} + d$$

The slave now knows times T1, T2, T3, and T4. Combining the above two equations, we get that

$$\tilde{o} = (T_2 + T_3 - T_1 - T_4)/2$$

The slave now knows the offset $\tilde{o}$during this transaction. While this offset will drift with time, it will be corrected the next time this exchange of transactions is launched.

2.3.2 Network Time Protocol

The Network Time Protocol (NTP) is widely employed to synchronize a computer to Internet time servers or other sources. It provides accuracies typically less than a millisecond on LANs and up to a few milliseconds on WANs[27].

Typical NTP configurations utilize multiple redundant servers and diverse network paths in order to achieve high accuracy and reliability.

The NTP works with a hierarchy of levels, where each level is assigned a number called the stratum.

Stratum 0
These are devices such as atomic (cesium, rubidium) clocks, GPS clocks or other radio clocks. Stratum-0 devices are not attached to the network; instead they are locally connected to computers (e.g. via an RS-232 connection using a Pulse per second signal).

Stratum 1 (primary)
Servers at the lowest level are directly synchronized to national time services.

Stratum 2 (secondary)
Servers at the next higher level are synchronize to stratum 1 servers and so on. These are computers that send NTP requests to Stratum 1 servers. Normally a Stratum 2 computer will reference a number of Stratum 1 servers and use the NTP algorithm to gather the best data sample

Stratum 3
These computers employ exactly the same NTP functions of peering and data sampling as Stratum 2, and can themselves act as servers for lower strata, potentially up to 16 levels. NTP (depending on what version of NTP protocol in use) supports up to 256 strata.

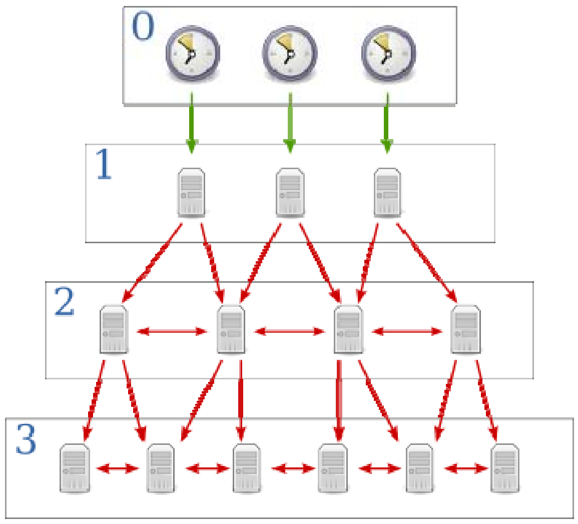


Figure 2-11 NTP synchronization

Green arrows indicate a direct connection; red arrows indicate a network connection.
NTP uses a hierarchical system of "clock strata". The stratum levels define the distance from the reference clock and exist to prevent cycles in the hierarchy.

The 64-bit time-stamps used by NTP consist of a 32-bit seconds part and a 32-bit fractional second part, giving NTP a time scale of 232 seconds (136 years) and a theoretical resolution of 2−32 seconds (0.233 nanoseconds).

2.3.3 Global Positioning System

2.3.3.1 Function as a time source

GPS satellites transmit signals to equipment on the ground. GPS receivers need a clear view of the sky, so current technology is used mainly outside and does not work well in mountainous areas or near forests or tall buildings [28].
Each GPS satellite transmits data that indicate its location and the current time. All GPS satellites synchronize operations so these repeating signals are transmitted at the same instant. Ground stations precisely track each satellite's orbit.
GPS satellites transmit signals on two main carrier frequencies L1 and L2. The signals arrive at a GPS receiver at different times because some satellites are further than others.

The GPS constellation consists of at least 24 orbiting satellites, which are not geostationary, so satellites in view are constantly changing. Each GPS satellite contains four atomic clocks, which are continuously monitored and corrected by the GPS control segment. Consequently, the GPS constellation can be considered a device of 24 orbiting "clocks" with worldwide coverage.

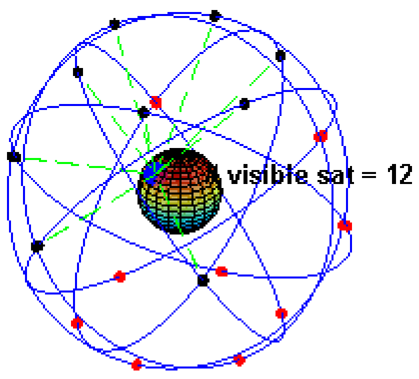


Figure 2-12 GPS constellation satellites & visible ones

In addition to serving as highly-accurate time sources GPS timing receivers are used to synchronize distant clocks in communication or data networks. This is possible because all GPS satellites are corrected to a common master clock. Therefore, the relative clock error is the same, regardless of which satellites are used.

2.3.3.2 NMEA 0183

The NMEA 0183 Interface Standard defines electrical signal requirements, data transmission protocol and time, and specific sentence formats for a 4800-baud serial data bus. Each bus may have only one talker but many listeners [29][5].
It is a simple, yet comprehensive ASCII protocol which defines both the communication interface and the data format. It was originally established to allow marine navigation equipment to share information. Since it is a well established industry standard, NMEA 0183 has also gained popularity for use in applications other than marine electronics.
For those applications requiring output only from the GPS receiver, NMEA 0183 is a popular choice since, in many cases, a software application code already exists.

The table below lists the standard characteristics of the NMEA 0183 data transmissions.

Signal	NMEA Standard
Baud Rate	4800
Data Bits	8
Parity	None
Stop Bits	1

Table 2-5 NMEA 0183 RS232 characteristics [5]

Message	Description
GGA	GPS fix data
GLL	Geographic position - Latitude/Longitude
GSA	GPS DOP and active satellites
GSV	GPS satellites in view
RMC	Recommended minimum specific GPS/Transit data
VTG	Track made good and ground speed
ZDA	Time and date

Table 2-6 NMEA Messages [5]

In this system, ZDA message is the only one that is really needed, for that reason a briefly introduction about the format of this message has been included below.

<u>ZDA Time & Date</u>

The ZDA message contains UTC time, the day, the month, the year and the local time zone.
$GPZDA,hhmmss.ss,xx,xx,xxxx,,*hh<CR><LF>

Field #	Description
1	UTC
2	Day (01 to 31)
3	Month (01 to 12)
4	Year
5	Unused
6	Unused
hh	Checksum

Table 2-7 ZDA Message

More information about NMEA messages can be found in http://www.nmea.org/pub/0183/

2.3.3.3 Other global positioning solutions

Countries including Russia, the European Union, Japan and China are developing their own international satellite navigation systems [30][28].

- GLONASS (for Global Navigation Satellite System), is a Russian radio satellite navigation system whose satellites began entering service in 1983. The system, operated for the Russian government by the Russian Space Forces, was completed in 1995.
  Like GPS, the GLONASS constellation consists of 24 satellites. Because of economic condition in Russia, only about 14 satellites are now operating, according to media reports.
  The Russians developed an advanced GLONASS satellite with an operational life of seven years and launched it, on December 26, 2004. An even more improved GLONASS satellite, with reduced weight and an operational life of 10 to 12 years, is due to enter service in 2008.
  In a 2005, the Indian government agreed to share development costs of the improved GLONASS satellites and launch two of them from India. With this support, GLONASS will be operational again by 2008 with 18 satellites, and by 2010 with 24 satellites.

- Galileo is the European Union alternative to GPS and GLONASS. The proposed Galileo positioning system will be composed by 30 satellites and it should be operational by 2010.
  Galileo is intended to give users access to greater precision than is now available. Only one of Galileo's planned four navigation services will be available at no charge to users.
  Since 2003, several countries have joined the project, China, Israel, Ukraine, India, Morocco, Saudi Arabia and South Korea.

- Quasi-Zenith Satellite System (QZSS) is a Japanese three-satellite positioning system, which will supplement and be interoperable with GPS. The first satellite launch is scheduled for 2008, the second and third in 2009.QZSS could improve regional service for positioning, timing and navigation users in Japan and surrounding areas, where mountainous terrain and population density sometimes make GPS unavailable.

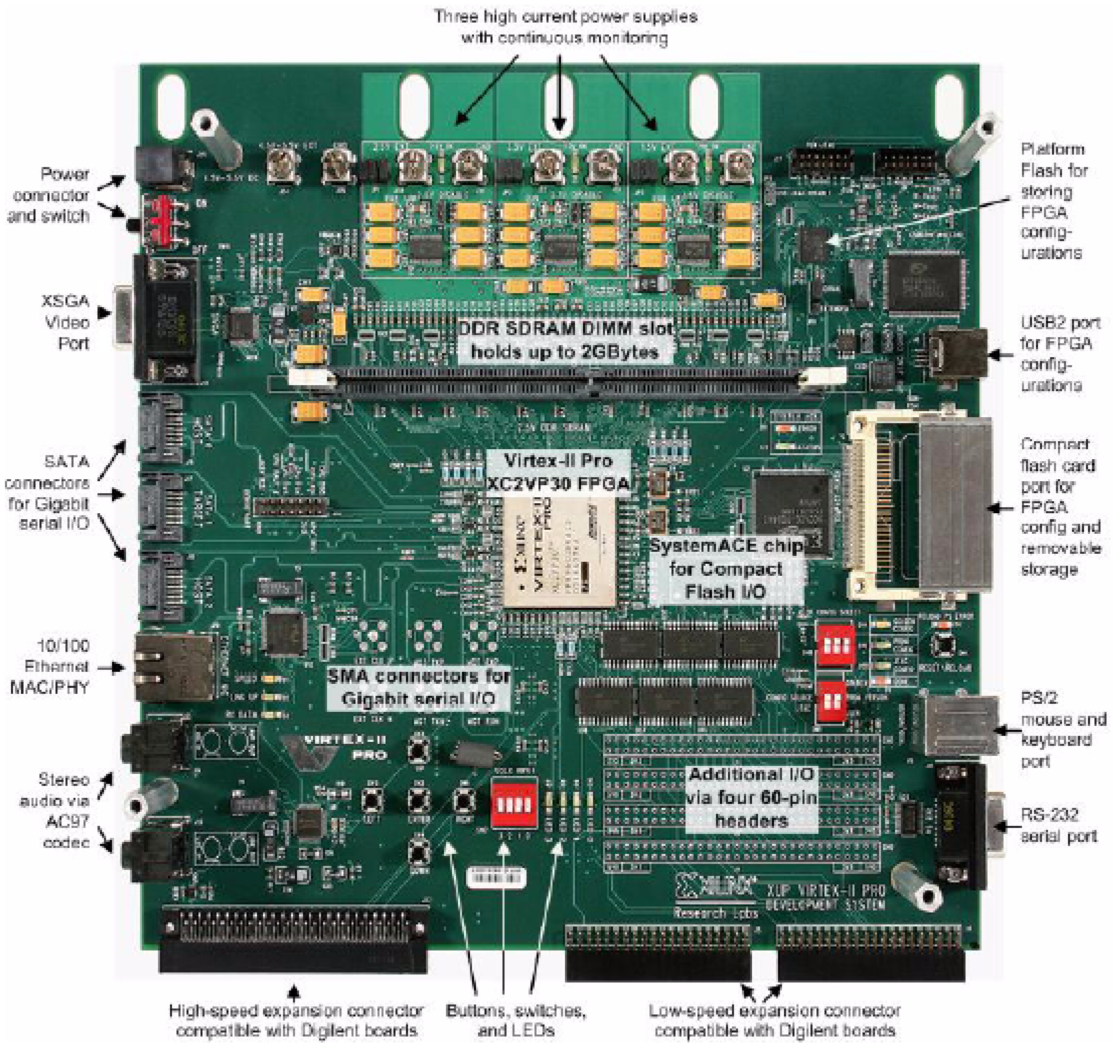
- China is also developing an independent navigation satellite system. The Twinstar Rapid Positioning System, or Beidou Navigation System, consists of two satellites in geosynchronous orbits.

3 Design

3.1 Development tools

3.1.1 Development Board

The hardware is going to be design in a XUP Virtex II Pro development board. Nevertheless, it is going to be implemented in a way that could be resource compatible with the Xilinx low-cost family SPARTAN 3.



Three high current power supplies with continuous monitoring
Power connector and switch
XSGA Video Port
SATA connectors for Gigabit serial I/O
10/100 Ethernet MAC/PHY
Stereo audio via AC97 codec
Platform Flash for storing FPGA config-urations
USB2 port for FPGA config-urations
Compact flash card port for FPGA config and removable storage
PS/2 mouse and keyboard port
RS-232 serial port
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
High-speed expansion connector compatible with Digilent boards
Buttons, switches, and LEDs
Low-speed expansion connector compatible with Digilent boards

Figure 3-1: XUP Virtex-II Pro Development System Board [36].

This board brings several advantages for this project. As it can be seen in the bottom part of the figure there is a “High-Speed expansion”, which must be employed for the Ethernet interface board for avoiding additional delay.
It also have an integrated 10/100 Ethernet physical/MAC, in the left side. This is a must in this project due to the need for sending the filtered data to a host.

The XUP Virtex II is associated to a university program reducing the acquisition cost. Moreover it has achieved a great distribution generating lot of information, papers, technical notes and forums supporting this board. With all these characteristics it would not be difficult to find a better development board for this thesis.
Apart from the development board there will be two more boards:

- Ethernet interface: this board, a physical/Mac one holding two Ethernet connectors, has been already implemented at the start of this project. Basically, it includes the elements that allow a physical/MAC layer. The main chip on this board is the "DP83849CVS" that offers a 10/100 Mbps Ethernet Physical Layer Transceiver.

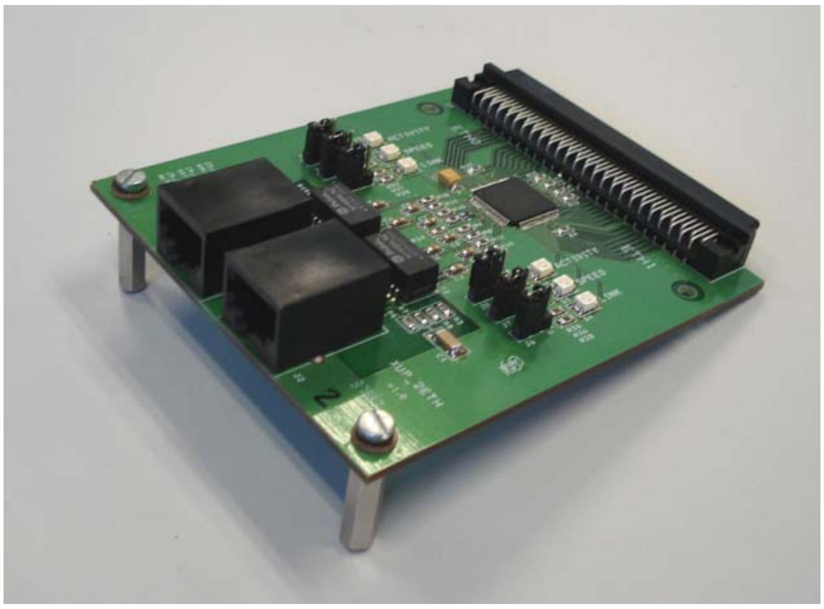


Figure 3-2: Ethernet interface board.

- GPS interface: this board consists on two parts. One is the Resolution T receiver 3.4.1.1, the other one is a simple framework for allowing an easy and safe connection to the low speed expansion connects of the XUP board.

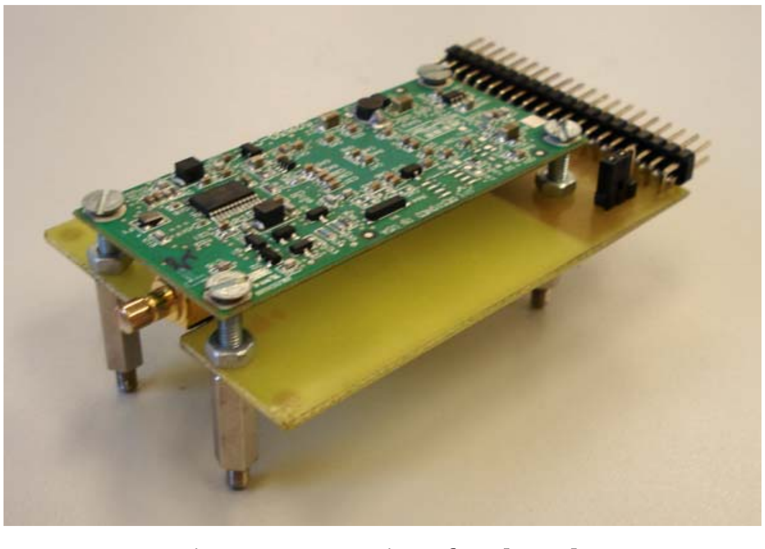


Figure 3-3: GPS interface board.

And finally a photo of the whole system:

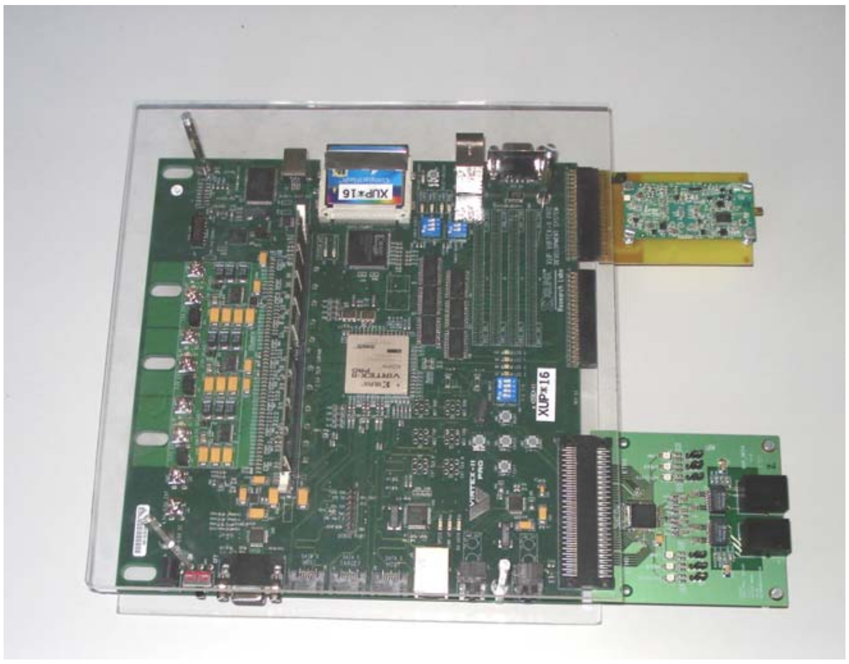


Figure 3-4: The system

3.1.2 EDA tools

- Xilinx ISE Foundation:
  It is a software that integrates everything needed in a complete logic design environment for all Xilinx FPGA.

- Xilinx Embedded Development Kit:
  This tool is composed by several applications which aim to generate an easier embedded implementation. Apart from the configuration of the system processor/s, it permits adding peripheral IP cores and it also supports part of the debugging procedure. However, it is not a fully independent tool due to still need the help of the ISE Xilinx for realizing the place and rout routines.[43]

- Xilinx Chipscope:
  It inserts logic analyzer, bus analyzer, and virtual I/O low-profile software cores directly in the design, allowing to view any internal signal or node, including embedded hard or soft processors. Signals are captured at or near operating system speed and brought out through the programming interface. Captured signals can then be analyzed through the included ChipScope Pro Logic Analyzer [37].

3.2 Design considerations for the embedded platform

3.2.1 MicroProcessors

Two embedded microprocessor are supported by the XUP development board 3.1.1:

3.2.1.1 PowerPC

The PowerPC 405 processor is a 32-bit implementation of the PowerPC embedded environment architecture that is derived from the PowerPC architecture.
It is a “Hard core” which means that it can not be customize by the design requirements, 2.2.3.1. Nevertheless, its hard-core condition also brings the normal advantages of these devices; that is the higher speed, it executes instructions at sustained speeds approaching one cycle per instruction, which improve the general performance of the microprocessor. It also saves FPGA resources [38].

Furthermore, the PowerPC architecture provides a software model that ensures compatibility between implementations of the PowerPC family of microprocessors.

3.2.1.2 MicroBlaze

The MicroBlaze embedded soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx FPGAs. It has been developed to support a high degree of user configurability [39].
This allows customization of the processor to meet specific cost/performance requirements. Some of the optional parts can be appreciated in the following picture.

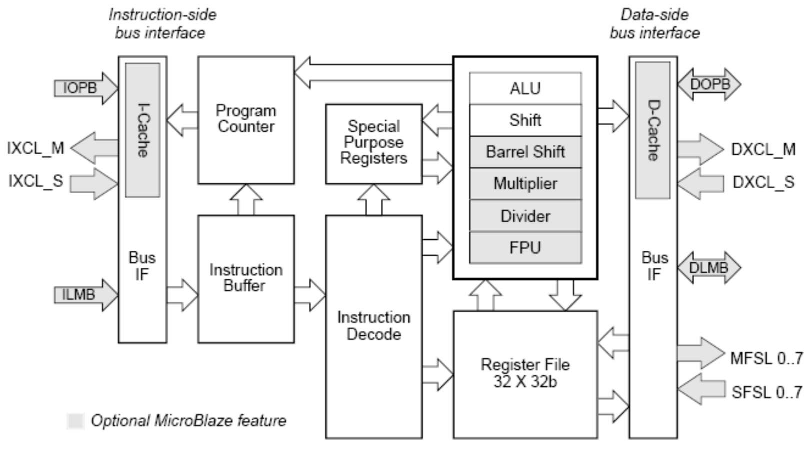


Figure 3-5: MicroBlaze architecture [39]

Despite of the high customization, it keeps a strict low-area consume allowing the possibility for using it in smaller and cheaper FPGA, like the Spartan 3.

The major arguments for choosing this core in the implementation of the system have been the combination of both customization and support for Spartan 3.

3.2.2 Peripherals

EDK(3.1.2) permits the simple addition of several peripherals to the system. Of course it also depends of the development board.
Some possibilities can be complained in the next picture.

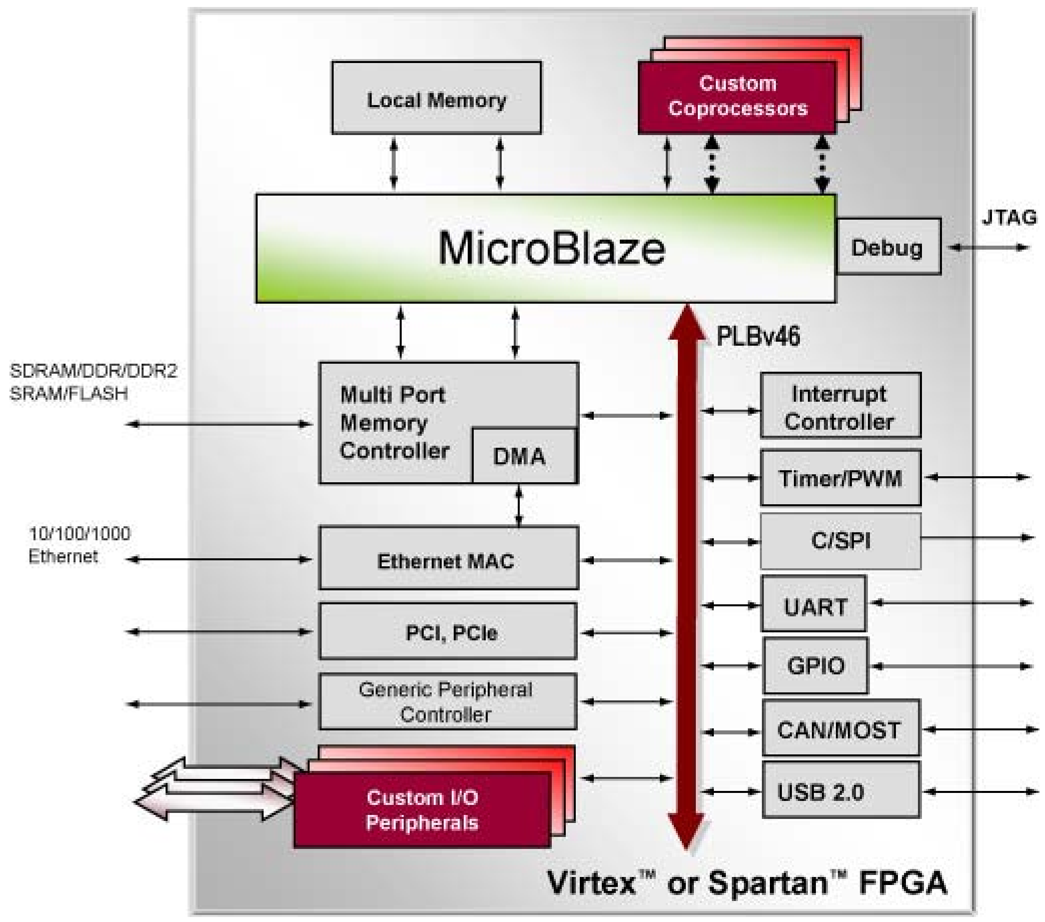


Figure 3-6: MicroBlaze and its EDK add on Peripherals [40].

Some of these peripherals have been chosen for designing the system. A list of them and the reasons supporting their election is given.

- UartLite: two IP cores are needed for this project. One of them will handle the communication with the GPS receiver. The other one will be connected to a computer for serving as a console.

- Ethernet: A Ethernet 10/100 Mbps device is included in the development board. This device will work for putting the collected filtered data into the network.

- DDR RAM: The use of a Operational System as Xilkernel and the great number of libraries that must be employed exceeds the limits of the faster BRAM memory, which is maximum MicroBlaze customization value is 64 Kbytes. This makes compulsory to include a bigger, but slower, memory in the design.

The available memory block capacity is a 256 MBytes.
EDK peripheral is an IP that simplifies the writing and reading operations from the Memory. So the communication protocol becomes transparent to the designer.

- Timer: A timer IP is just needed due to the requirements of XilKernel OS and also for the LwIP library.

3.2.3 Software

3.2.3.1 XilKernel OS

Xilkernel is a small, robust, and modular kernel. It is highly integrated with the Platform Studio framework and is a free software library that you get with the Xilinx Embedded Development Kit (EDK). It allows a very high degree of customization, letting users tailor the kernel to an optimal level both in terms of size and functionality. It supports the core features required in a lightweight embedded kernel like the execution of threads [41].

3.2.3.2 LwIP

LwIP is an open source implementation of the TCP/IP protocol suite.
The LibXil, is a modified library based on LwIP, that provides block access to files and devices through the LibXil File module. This module provides standard routines such as open, close, read, and write to access file systems and devices.

It supports two interfaces:

- Raw: It is the lowest level interface of the stack for achieving a better performance.

- Berkeley Socket: It is the "de facto" standard API for communication which allows an easier implementation. Because in this project the performance of the Ethernet interface is not critical and also due to a better compatibility BSD(Berkeley Sockets) are going to be implemented.

The following protocols are supported IP, ICMP, TCP and UDP.
It is important to note that LibXil requires a timer, for calculating round trips, retransmissions and other TCP facilities [42]. It only works under the XilKernel OS system.

3.2.3.3 GPS communication

The GPS communication it has to be through a RS232 interface. The configuration of this port will change depending on which GPS device is selected.The Resolution T receiver is the chosen GPS receiver (3.4.1.1) and operates using one of two interface protocols: Trimble Standard Interface Protocol (TSIP) or NMEA 0183.

- TSIP (Trimble Standard Interface Protocol): It is a powerful binary packet protocol that allows control over the GPS receiver for optimum performance in timing applications. TSIP supports multiple commands and their associated response packets for use in configuring the Resolution T receiver to meet user requirements.

- NMEA 0183 (National Marine Electronics Association): It is an industry standard protocol common to marine applications. NMEA provides direct compatibility with other NMEA-capable devices such as chart plotters, radar, etc. The Resolution T receiver supports the ZDA NMEA message for GPS timing. Other NMEA messages and output rates can be user selected as required.

The factory default setting for the I/O port is TSIP bi-directional.

3.3 System Architecture

The objectives section (1.2) gives an idea about how the device should be or at least which parts are the key ones.

After considering many kinds of architectures the final outline looked like this one below.

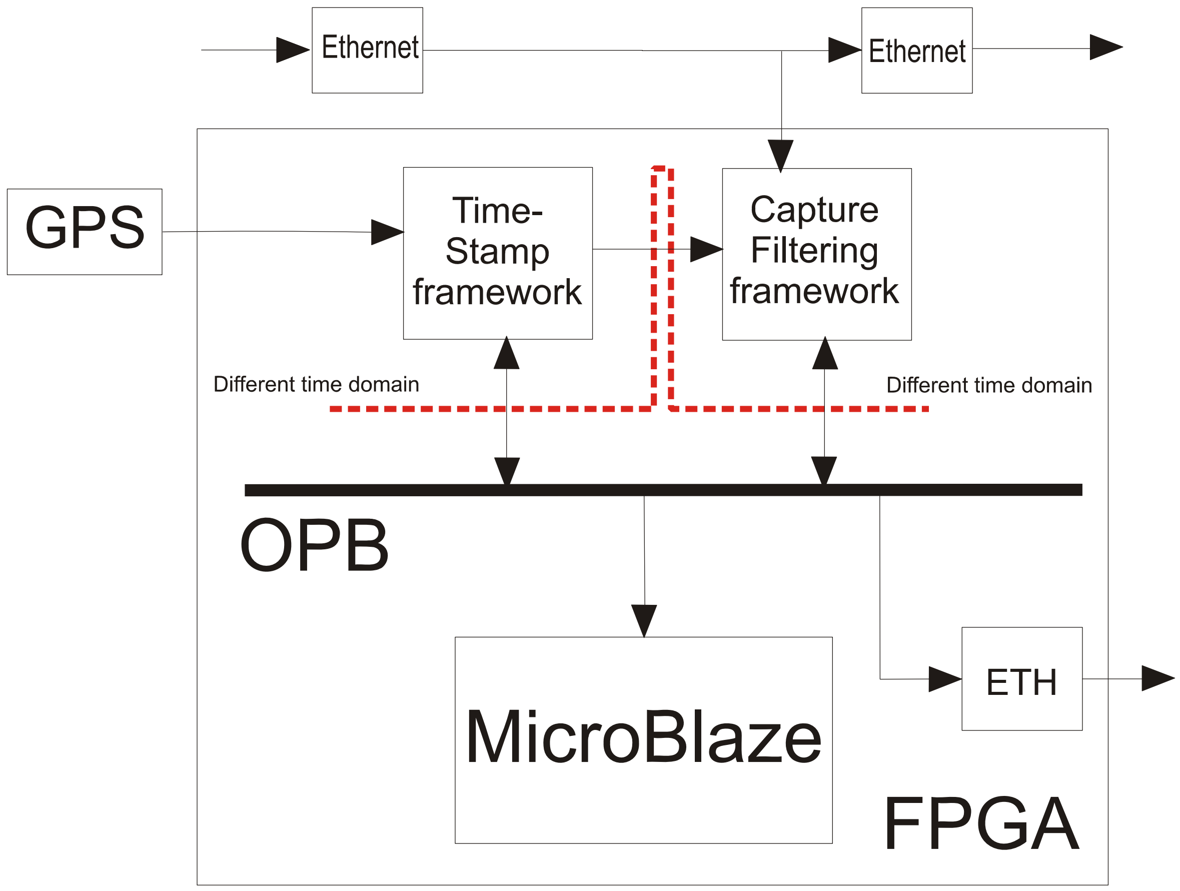


Figure 3-7: General design

About the picture just to notice that the "OPB" and the "time domain" concepts will be explained further in this section. Its reference numbers are in corresponding order 3.3.1 and 3.3.2.

Some of the critical points in the project architecture can be pointed as:

- Overcoming the asynchronous problem without delaying to much the bandwidth.
- Achieve a minimum delay in time stamping incoming packets.
- Resolve the filtering of data and its sending without missing information.

The different time domains are produced due to the employ of different time sources.
A scheme shows the clock structure in the following picture.

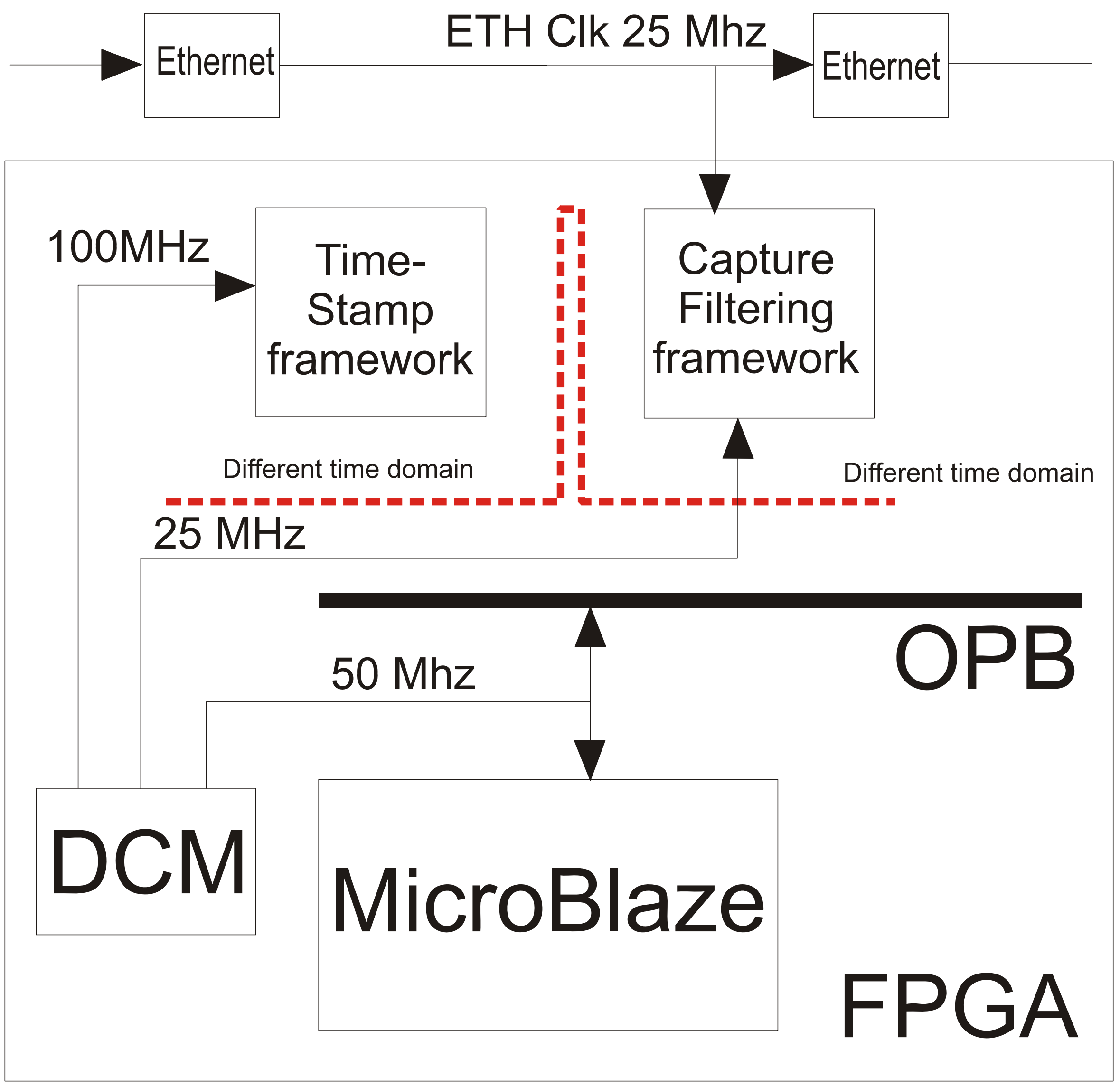


Figure 3-8: Clock design

The "backbone" of the system, that is the processor and the OPB bus, will works at 50 MHz, in order to implement the design in a Spartan 3 device.

The incoming data from the Ethernet it arrives which its own clock signal that should be close to25 MHz in order to meet the 100 Mbps constraint. Therefore, it will have two clock sources from both Ethernet directions and also an exact 25MHz input from the DCM that will be employed by the "DP83849 CVS" that implements the Ethernet physical layer.

The time stamp generation process will work with an independent clock that in this project is expected to achieve 100 MHz. The idea of this autonomous clock is clearly to be able to increase this frequency in order to accomplish a better time resolution.

3.3.1 On-chip Peripheral Bus (OPB)

OPB it synchronous bus employed for connecting random access peripherals. An OPB system is composed of masters, slaves, a bus interconnect and an arbiter, as shown in the next picture [44].

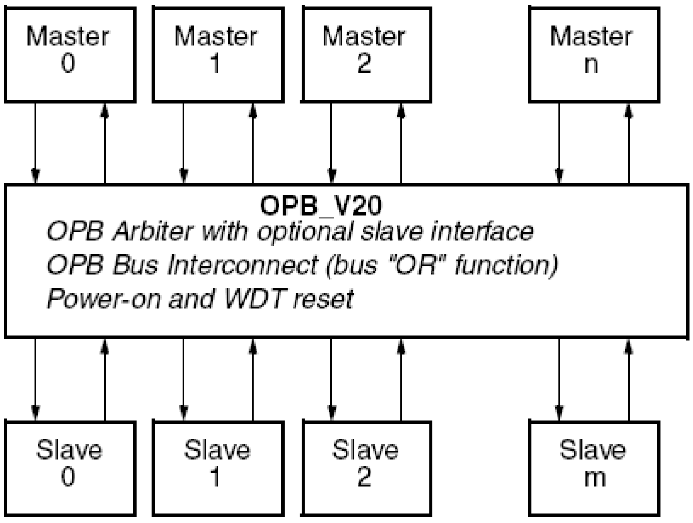


Figure 3-9: OPB Structure [44]

OPB IPIF: It is an IP core that provides a bidirectional interface between a user IP core and the OPB 32-bit bus standard. The Xilinx OPB IPIF is available for use with various FPGA device families that support embedded hard or soft processors.

This part is vital for understanding some modules like the 'time-domain border' (4.1) or even the 'TAP core' (4.3). For this reason it is hardly recommend reading this reference [45]

OPB represents the slowest but simplest way for adding additional peripherals to the system. Taking in consideration the bus has few data transfer this solution becomes the more suitable for the project. Other solutions could have been FSL or PLB.

3.3.2 Time domain border

As it has been explained in this section, the system uses several time domains, and unfortunately there is a need of communication between them.

Communication through different time domains brings different kind of problems:
The main one occurs when in one side it is prepared for reading and in the other one it is not prepared to be read, that means it is still changing the value of some bits.

This project will evaluate these two solutions:

- Handshake protocol: When the logic wants to read the time-stamp, it sets one control signal high in order to freeze the time-stamp counter. This is acknowledged by a transition on another control signal. The advantage of this straightforward protocol is that the clock boundary concerns are limited to the two control signals, because the counter is stopped when it is being read.

- Gray counter: When transferring big numbers through a clock boundary, it might happen that some bits are updated while others remain with the old value. Using Gray code for the time-stamp counter will prevent this kind of errors, because in this code two successive values differ in only one digit.

Now, it is time to analyze how this problem can be solved.
A Gray counter takes a lot of area of a FPGA, and requires lot of work to be implemented. Moreover, it has another disadvantage. It can not be used with other information different than a counter. As it has been showed, different information than time-stamp value has to be transferred, and certainly there is no way for transferring a random 8-bit word through the time boundary.

The Handshake design proposes a more versatile solution that will allow the communication of any information. Due to all this advantages, this project has been implemented using the following handshake protocol.

3.4 Time-stamping framework

Accurate time-stamping is quite important in any measurement system, in order to be sure that each measured event has its own time-stamp, and that this one is unique for that input in the whole system.

Nowadays, there are many options for trying to do a time-stamp process and it is really easy to figurate out that as precise is the clock of the time-stamping generator as precise will the process will be done. For achieving enough precise time-stamps, it is necessary to have a nice resolution and also a good synchronization between the other time-stamping devices.

It is time to analyze then the different solutions. A brief description was introduced in the 2.3, so now, only an explanation of the advantages or disadvantages that the different technologies generate for this project.

- Network Time Protocol offers a good solution for LAN synchronization but it lacks the capability for reaching it in the long distances.

- Precision Time Protocol it lacks accuracy, so it is also dismissed.

- GPS: lacks resolution but offers a global accuracy clock.

Hence, it can be seen than any of these technologies can perform as the clock for generating the time-stamps. As a consequence, a different clock source must be used. This source is going to be the quartz clock from the FPGA. The main problem from using a FPGA's clock is the time drift of the quartz oscillation. GPS it is not as accuracy as it is required but it offers an excellent way for synchronize all the devices no matters where they are.

As a corollary of this analysis, the time source is going to be a quartz clock but the time drift and the lack of synchronization that suffers is going to be corrected with the Pulse per Second signal arriving from the GPS receiver.
NMEA ZDA (time packet) could be employed for the initialization of the quartz counter, which is going to work as a time source.

Below it can be seen, the outline of Time Stamp Generator.

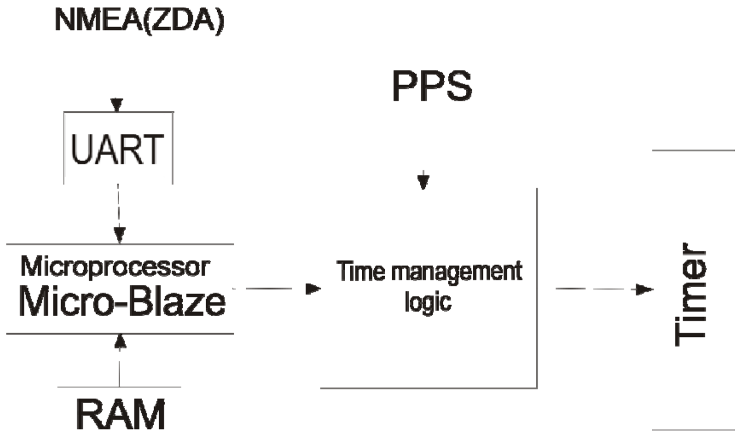


Figure 3-10: Time Stamp generator design

3.4.1.1 Trimble Resolution T

The Trimble Resolution T embedded timing board is a GPS receiver. The receiver is designed to operate on the L1 (1575.42 MHz) frequency, providing standard position service (SPS). It features a low cost software programmable general purpose Digital Signal Processor (DSP) instead of a custom GPS ASIC.

Timing applications are assumed to be static. The software configures the unit into an automatic self survey mode at start up. The receiver will average position fixes for a specified time (one per second) and at the end of this period will save this reference location. At this time the receiver will go into an Over-determined Clock mode and no

longer solve for position but only for clock error and clock bias using all of the available satellites. This provides an accuracy of less than 15ns (1 Sigma) for the 1PPS output.
User settings such as port parameters and NMEA settings can be stored in the receiver's non-volatile (Flash) memory. These settings are retained without main power or battery back-up power applied. The Resolution T receiver has a single configurable serial I/O communication port.

3.4.1.1.1 Pulse Per Second (PPS)

The Resolution T GPS timing receiver provides a one millisecond wide, CMOS compatible Pulse-Per-Second (PPS). The PPS is a positive pulse available on pin 6 of the power and I/O connector. The rising edge of the PPS pulse is synchronized with UTC. The timing accuracy is within 15 nanoseconds to UTC when it works in Over-determined Mode. The rising edge of the pulse is typically less than 20 nanoseconds. The distributed impedance of the attached signal line and input circuit can affect the pulse shape and rise time.

PPS signal suffers from an error a quantization error that it is explained in the datasheet. However, a quick look of this kind of error can be appreciated in the figure below these lines.

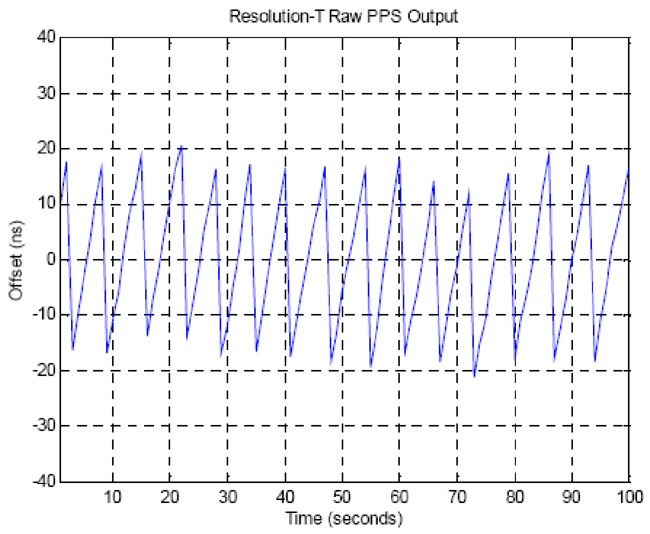


Figure 3-11: Resolution-T Raw PPS Output[5]

For more information about this device consult the datasheet reference [5].

3.4.1.2 Time drift correction

Nowadays, most computers and networking equipment use inexpensive piezo-electric quartz oscillators. An as it has been explained in 2.3, they suffer from time drift.

It is time now for analyzing the several algorithm and methods for trying to correct it.

Each correction algorithm must have its own methods to determine the values of the initial offset between the clocks (the master and the slave) and the relative drift rate between them.

A lot of time has been employed reading and understanding some algorithms and therefore, there are many methods that can be implemented for trying to fix this problem. Nevertheless, as the analyzed algorithms are very complex for being explained in this master thesis, the following references, [46][47][48][49], should cover the lack of explanations about them.

For selecting or generating a good algorithm, first it is compulsory to know the nature of the errors that affects our time source.

- There is an error from our GPS transceiver. Obviously, if this is our reference clock this is going to influence when it is tried to correct the quartz drift error. Furthermore, this quantification error can not be removed and therefore, introduces +/- 20 ns random error.

- Apart from this error, there is the already explained "drift-error" that changes from one clock to another. This error behaves as a constant.

All these errors are accumulative in the poor counter. It means that if an error occurs, like a +30 ns per second, it will be there for the next second. So, if in the next second the counter will drift +15ns the real error will be +45ns.

The complexity of these algorithms, and the nature of these errors, has forced the creation of a simpler algorithm that will be explained in detail in the implementation part 4.4.1.

Basically this algorithm must attend to compensate both errors using an average error, and a small tuning correction for the small jumps of the PPS output. This tuning will also compensate the accumulative error.

The solution must be implemented by software. Using software for drift correction could sound inaccurate. However, it has some advantages. For example, it is easy to point out that the algorithm for controlling the time drift can be easily improved and changed.

The main disadvantage of SW is that is less accurate, but taking in consideration that the system uses the Pulse Per Second signal for synchronizing, it is not a bad idea that when the signal is employed an interruption is generated. Interrupting the microprocessor once per second should not be such a problem in the global system performance.

A Hardware implementation of the solution will be high-time consuming. If any change has to be done it will be harder and quite more complicated in hardware. On the other hand, it will offer even more accuracy, so perhaps after evaluating an algorithm in software could be the right time for its implementation in hardware.

3.5 Capture and filtering framework

The objective of the system is to be transparent to the traffic which means that the copy has to be done as fast as possible. In order to accomplish this task, the logiest procedure would be to make a copy of all the data that comes from the outside, and to link that signals directly with the outputs. And that is exactly what the system does.

As soon as the data is copied, it is also time-stamped, after these procedures two strategies can be employed. Filtering information before store it or vice versa.
The first option allows to safe memory resources due to the fact that the valid flow is stored. However, it will increase the area because it requires processing the information as soon as it arrives.
On the other hand, firstly storing and secondly filtering will suffer the opposite effects.

Choosing between both should be determinate by the board capability or design constraints such as memory capacity, area or real-time analysis.
According to the nature of this framework project, it is more logic to filter first and then to store the selected information. Another option will have sense if the device is attacked by very heavy traffic, collapsing the filtering system and as a result, creating the need for a buffering input.

One of the key parts in this project is the design of a good filter that allows generating a flow without errors.
About filtering strategies and its architectures, it is easy to discover, when trying to identify flows, that they can share some of its values. As an example, there could be a communication where all transmitted packets have the same parameters but ports number, therefore they match all the filters except the port parameter, which allows sharing hardware resources.
As a result of this idea, the flow identification can be classified in different deep orders, depending on how many characteristics they share in the filtering.
This brings the idea of developing a tree. One good example is explained in [17] where a detailed PAM-tree algorithm can be found and even other solutions like CSPF, BPF or MPF.
Certainly, if the order of this tree is zero, it would mean that a group of dedicated filters are used for identifying independent flows, which not share any characteristic between them.
Developing a straightforward 'zero-tree' method will let decreasing the implementation time. As a disadvantage the huge amount of area required for recognizing many flows. However, it is the previous step for developing filters based in deeper trees and therefore is going to be implemented in this system.
This will target to the optimums solution that it would be a pipelined PAM-tree filtering, which combines the speed of the pipeline and the low-use area of the PAM tree solution.

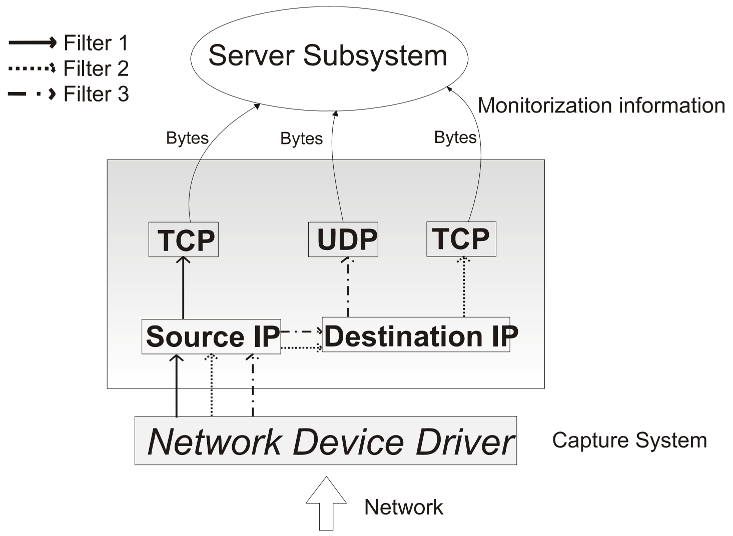


Figure 3-12: PAM tree architecture example [17]

For further information about this field is strongly recommended to read [17].

3.5.1 CRC checker

The Cyclic Redundancy Check (CRC) is an error detection technique that is widely utilized in digital data communication and others such as data storage, data compression… There are many CRC algorithms. They have each own predetermined generator polynomial G(x) that is utilized to generate the CRC code. For example, in TCP/IP protocol, the most frequently utilized CRC algorithm is the CRC-32 algorithm employed by Ethernet [50].

CRC checking must be performed in our system and it is designed for being implementer in the capture and filtering core.

There are different ways for a Hardware implementation of CRC-32. They can be classified in two, serial or parallel [51].

3.5.1.1 Serial Hardware Implementation of CRC-32

The single-bit data input (serial) calculation of CRC-32 is implemented with a linear feedback shift register (LFSR). The CRC-32 LFSR is illustrated in Figure3-12 (register bits "3" through "25" are left out of the figure to simplify the drawing).

Presetting the flip-flops to 0xFFFFFFFF is equivalent to complementing the first 32-bits of the data stream. For the first 32 cycles, the right-most XOR gate in the figure is an inverter. The XOR of any data with a binary "1" results in the complement of the original data. The serial implementation is not optimal as most IEEE 802.3 transceivers communicate the data stream to the Media Access Controller (MAC) over a 4-bit bus; therefore, a parallel implementation is necessary.

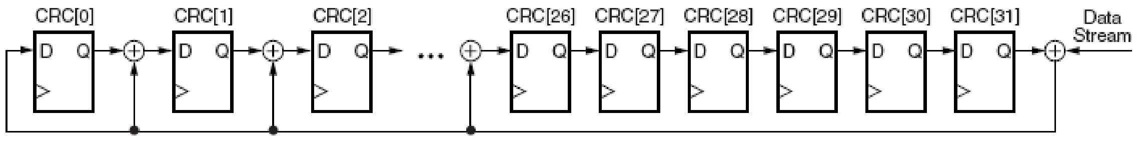


Figure 3-13: CRC-32 Serial implementation [51]

3.5.1.2 Parallel Data Input Hardware Implementation of CRC-32

A parallel implementation operates on multiple bits of the data stream per clock cycle. An algorithm for generating the next-state equations for parallel implementation of CRC-32 is discussed in "A Symbol Based Algorithm for Hardware Implementation of Cyclic Redundancy Check (CRC)"[52]. The algorithm involves looping to simulate the shifting, and concatenating strings to build the equations after "n" shifts. Other implementation technique based on the same arguments can be observed in [50].

Here it can be seen the scheme of the Xilinx parallel implementation solution that is going to be implemented in this system.

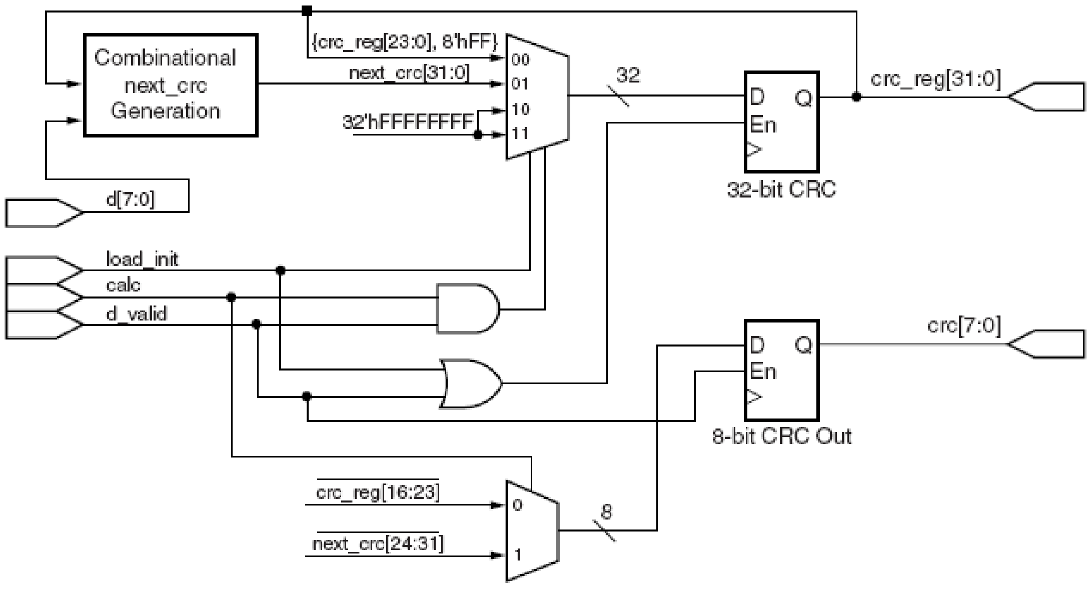


Figure 3-14: CRC-32 Parallel implementation [51]

4 Implementation

4.1 Time-domain border

In previous Sections 1.2, 3.3.2, problems like transmitting data between different time domains has been exposed. Also, it was decided to implement a handshake protocol device.

Now is time to show the implementation.
Remember that this device is going to be in the time domain border, acting as a transparent mediator in the communication procedure.

First of all, the time border scheme.

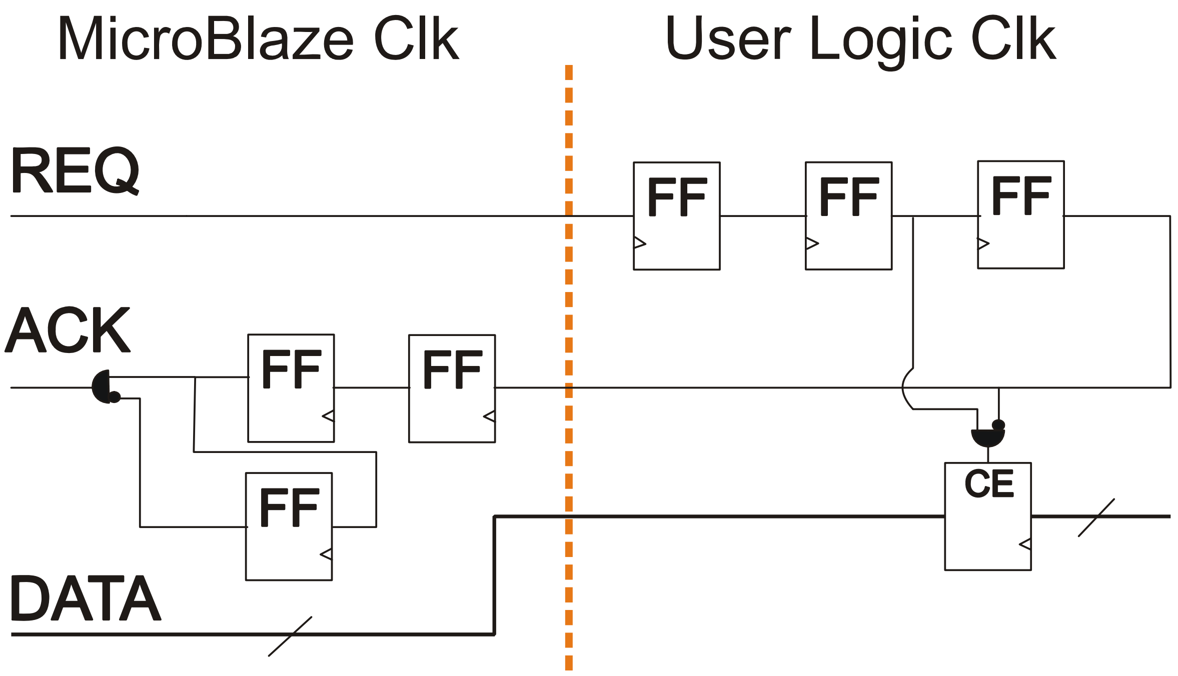


Figure 4-1: Time domain border scheme

Notice the double flopping for protecting the information from the metastability problems [34]. Also observe the rising edge flank detector which allows only activating the Data FF one cycle per each request. The same occurs with the ACK signal.

This device can work in both communication directions and now it is time for explaining how.

- Reading from User-Logic(target):

In this process uBlaze raises the ‘Req’ signal, which goes through the two Flip-Flops. At this moment, the CE of the ‘Data’ flip-flop will be activated allowing the data bus to be refreshed. This bus will keep the value until the next CE activation. As soon as the uBlaze receive the ‘1’ value of the ACK signal will read the data bus and cleared the REQ signal...

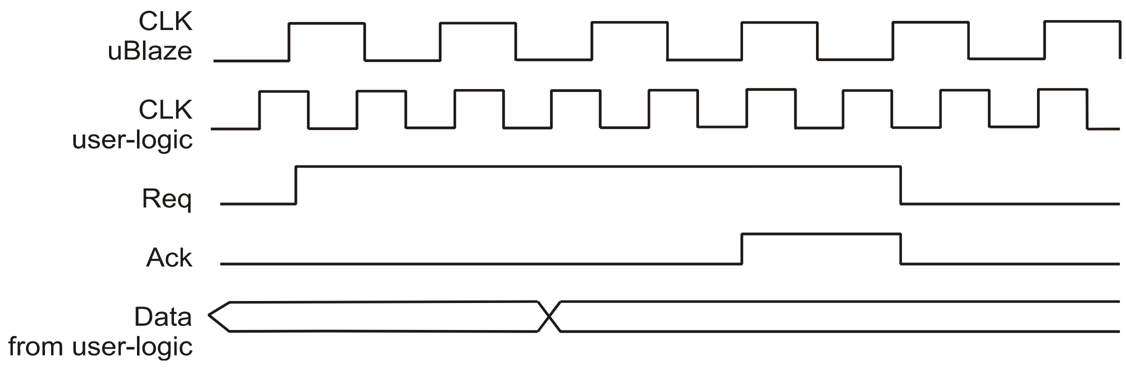


Figure 4-2: Reading from user logic

- Writing in User-Logic(target):

In this situation, uBlaze modifies at the same cycle the 'Req' and data signals. And the data will keep the value until 'Ack' = '1';

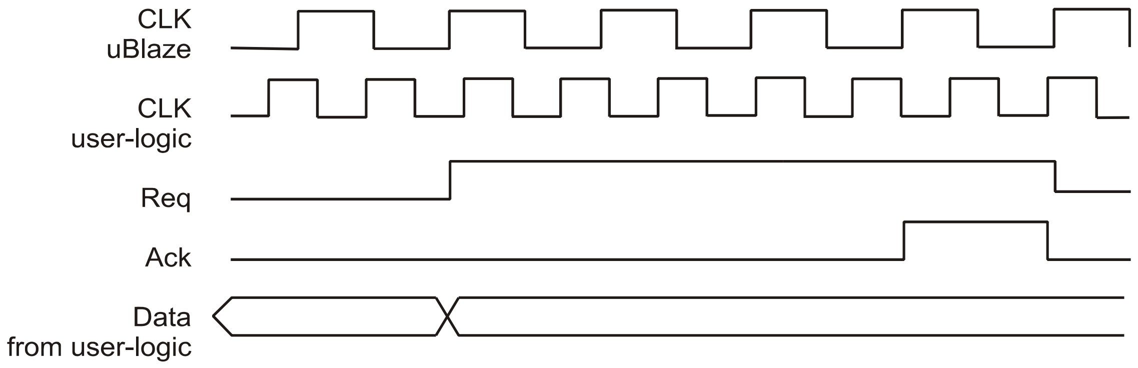


Figure 4-3: Writing in user logic

So as it can be seen in the figures above, the microprocessor or upper intelligent, uBlaze, is the chosen one for controlling the protocol.

4.2 Time-stamp generator core

After choosing a quartz clock as a time source for the time-stamp in 3.4, it is the right moment for explaining how to generate time-stamp labels with it.
The time-stamp can be created just counting, with a 32bits counter device, the rising edges of the FPGA's clock, and then correcting it in real time. So, in this way the time drift can be compensated just jumping or increasing the counter count, instead of correcting the time-stamp labels, as many algorithms says.
The only problem is how to figurate out how often should it jumps or increases and how to compare it with another source, in order to know the error committed.

It is at this point when Resolution T brings the solution. Its called Pulse Per Second, which certainly means that each time a second starts it will rise a signal to one during 1 ms.
This pulse has a huge accuracy, in the order of ns. The PPS will help us synchronizing the clock and correcting the time drift.

Next points tell about how the core and its parts work. Later it will be explained how this framework works but first, please take a look on the scheme.

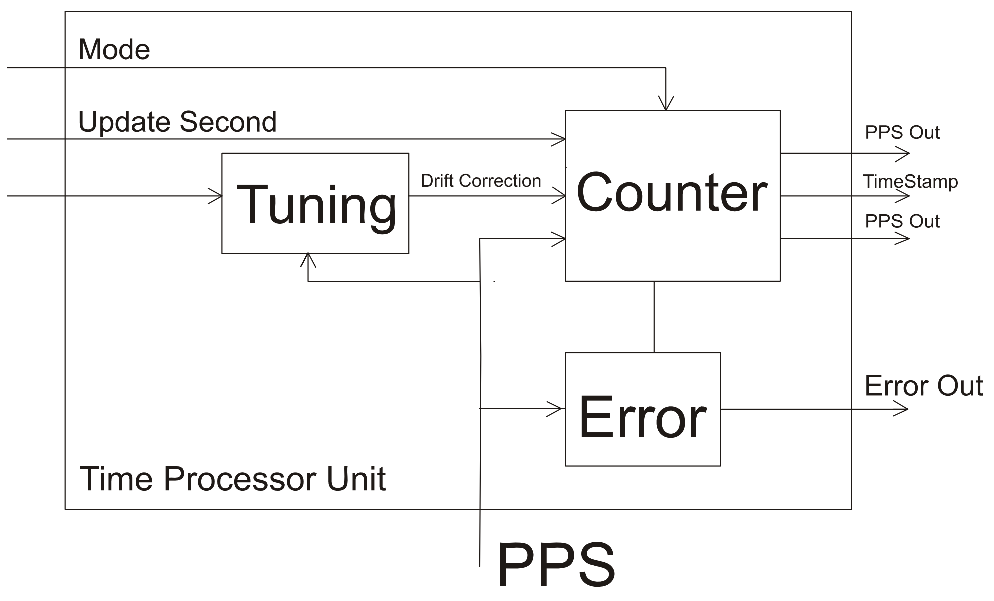


Figure 4-4: Time Stamp Generator

4.2.1 Counter

This module indeed contains two 32 bits counters. One it is used for nanoseconds and the other one for seconds. Therefore, the produced time-stamp gives a 32 bit resolution for both seconds and nanoseconds. This decision is just for making easier the upgrading work. If for example, the resolution of the counter is increased, then the 32 bit field still will cover the increase in accuracy.

32 bit for Seconds	32 bits for –Nanoseconds

The nanoseconds field is indeed a 10 ns field. This is just due to the 100MHz clock that it has a period of 10ns.

For the better understanding of the “Counter” a list with the important inputs and outputs is given:

Inputs:

- Mode: when its value is ‘1’, starts the counter or keeps it running. When ‘0’ it switches off the counter.

- Drift Correction: Certainly this signal it is the one that helps in the drift correction. Normally, the nanosecond counter will increase its count by one per cycle, however if the drift correction signal value has a different value than “11” then the following actions could be taken.

  “00”: New ns value <= Old ns value
  “01” New ns value <= Old ns value +2
  “10” New ns value <= Old ns value +3

- Update second: It carries the current second signal from the GPS.

- PPS_input: Each time that a new seconds happens this signal will be activated just one cycle. This occurs because the raw signal from the GPS is processed in a rising edge flank detector.

Outputs:

- Time-Stamp: The time-stamp that corresponds for each operation cycle. This signal goes to the error module in this generator core and also outside.

- PPS_output: A PPS created by the own counter.

Operation:

The count starts when both mode and the PPS_input signal value is '1'.

Then, the nanoseconds counter increases the value by one all the cycles but some of them. This is determined by the drift correction signal. When the counter reaches the value of a second, the PPS output goes to one for 1 ms and the seconds counter increase one its value.

When the PPS_input comes, the counter sends an error signal with the current nanoseconds value, which of course represents the error committed calculating the last second.

4.2.2 Tuning

It has a counter inside that also counts nanoseconds. When this counter reaches an absolute value, established previously by the processor, will send "00", "01", "10" depending on what is needed for correcting the main counter.
The counter will be reset when the PPS_input signal reach '1'.

4.2.3 Error

Each time a "PPS_input" signal arrives the Error device makes a direct comparison between the nanoseconds value arrived form the counter and the constant value that it should have. The result of the subtraction goes directly to the processor, which will determinate by software the value for the "tuning" device; the software procedure is explained in 4.4.1.

The Time-Stamp generator core works in the following way.

First is initialized, that is Mode = '1' and PPS_input = '1'.
The next step is to calculate the average error for the drift correction algorithm 4.4.1.
After, it is requested to the Resolution T, GPS receiver, the seconds of the week field. This procedure is explained in 4.4.2.

Then it is the time for switching of the clock. Because, due to the time drift correction calculation a lot of error is accumulated. Therefore, Mode = '0' and the counter is stopped.

Again it is started, but now the clock-counter already has the basic information, UTC seconds and average drift, for running in a smooth way.

Then, always that the PPS signal arrives, MicroBlaze asks for the Error. It processes it, using the already famous drift algorithm 4.4.1, and returns the tuning value that will help for correcting the counter-clock during the next second.

4.3 TAP core

The idea behind this project is to create a transparent TAP and its final implementation is one of the key parts of this project. For that reason, it is going to be explained in more detail than any other.

Its architecture is showed in the figure below.

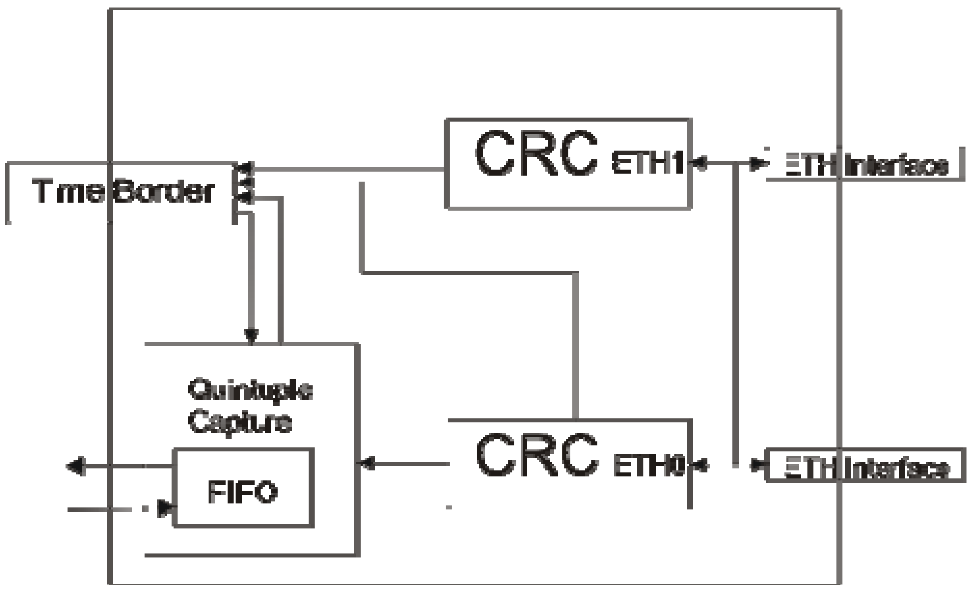


Figure 4-5: TAP architecture: Data acquisition & filtering

Some explanations about its function must be given.

Firstly, the data is processed in the CRC modules, one for each communication direction. The CRC (Cyclic Redundancy Check 4.3.1) module implements more functions than it was supposed to be commissioned. So, each CRC gives 'CRC_OK', 'CRC_NOK' information, plus 'End_frame', 'New_frame'. Furthermore, it creates 8-bit big-endian words. This is a compulsory requirement for the proper function of both the CRC analyzer and the system.

Secondly, as it can be seen in the figure, the information arrives to the Quintuple Capture module or modules where the filtering and storing takes place. It does not wait until the CRC is checked, because this would have delayed all the process. Moreover, perhaps someone is interested in knowing how many CRC_NOK are produced during a communication. As a consequence, when the data in the CRC module reaches one byte then it is carried to the Quintuple module.

The last step in the process consists on storing all the packets that have fulfilled the filter conditions. They are introduced in a FIFO, waiting for the reading request of the microprocessor.

For transferring the information to the microprocessor, there are several clock boundaries that must be saved, once more, using the time domain border module 4.1. However, some of the communication in the Quintuple capture module across the Ethernet clock domain and the MicroBlaze one are handled by some FIFOs included in the Quintuple module. This process will be described in detail in the Section 4.3.2.

4.3.1 CRC checker

This solution is based on [51] Xilinx note. This note shows a CRC parallel checker based on 8-bit parallel execution. The device is already done in Verilog. This characteristic plus others like the need for additional processing of the raw data, pointed out to create an upper instance with the aim to prepare all the signals in order to set-up the signals for the filtering process.

This upper CRC device identifies the beginning of an IP packet, then prepares and sends the bytes to the Xilinx unit, and finally checks that the CRC solution is right at the end, which will raise CRC_Ok or on the other hand will reach CRC_Nok.
Other signals are also generated. Some of them will be also used in other parts of the system, like for example in the filtering core 4.3.2 . They will be explained at the end of this section.

Here there is a figure of the upper CRC, called CRC checker.

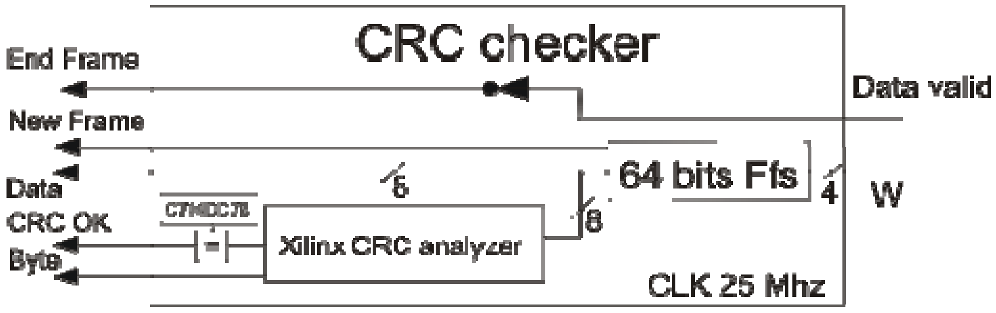


Figure 4-6: CRC checker

It is interesting to see the behaviour of the Xilinx CRC analyzer before explaining the rest of the functions of CRC checker. As a resul,t some waveforms are included.

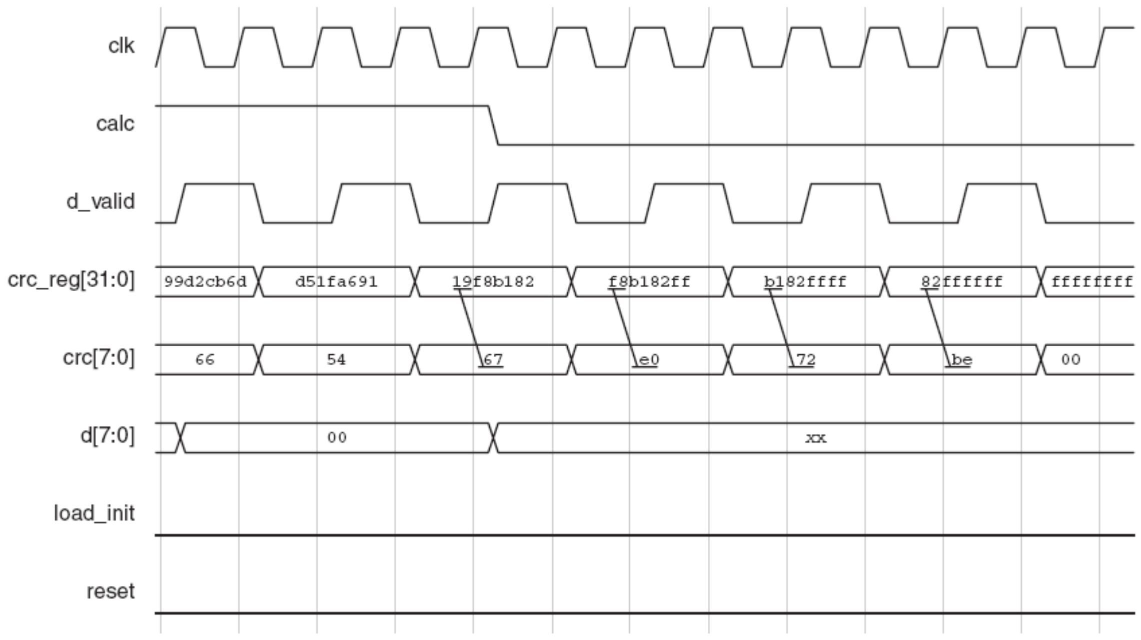


Figure 4-7: CRC waveform of the end of a packet

-

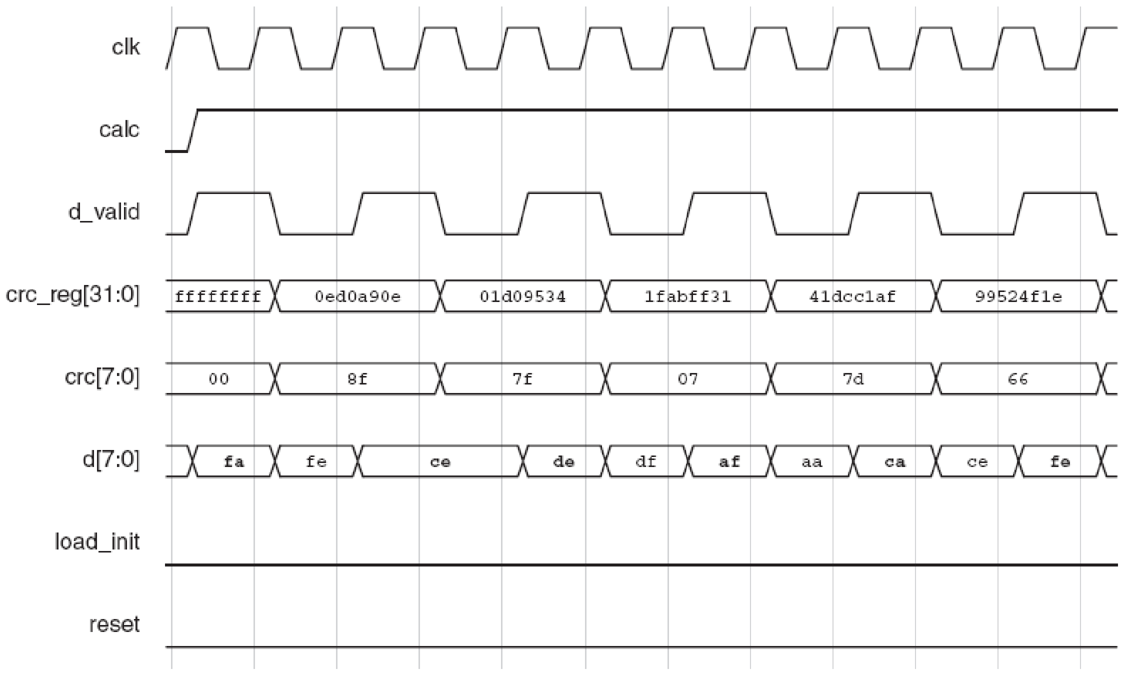


Figure 4-8: CRC waveform of the begin of a packet

Notice that when the packet ends “crc_reg” it is compared with the offset **0x4704DD7B**. If it is true, then CRC_OK = ‘1’, if not CRC_NOK = ‘1’.

The CRC checker also performs quite important roles for the right function of the whole system. These are some of the most relevant signals:

- New frame: there are two fields called ‘Preamble’ and ’Start-of-Frame-Delimiter’ as a part of an Ethernet frame arrival. These are the first ones to arrive and of course activate the procedure for storing the frame data.
- End frame: it is calculated based on a direct not function over the ‘calc’ signal of the above figures. It is employed in the data acquisition & filtering. It points out the right moment for starting the filtering of the just arrived frame.
- Word creation: the frame arrives in 4-bit word burst, besides its format is little-endian. The Xilinx’s CRC requires 8-bit big-endian words. Also the MicroBlaze processor works under big-endian and therefore a conversion is needed. This process is realized in the CRC checker module.

4.3.2 Filtering core (Quintuple capture)

In Section 3.5, an explanation about the importance of both data acquisition and filtering was given. Also some details were given about its different strategies.

The optimum solution explained in that section will be postponed and presented as a future work in 6.2. However, the implemented architecture is a previous step for reaching it.

The entire Quintuple Capture module can be seen in the next figure.

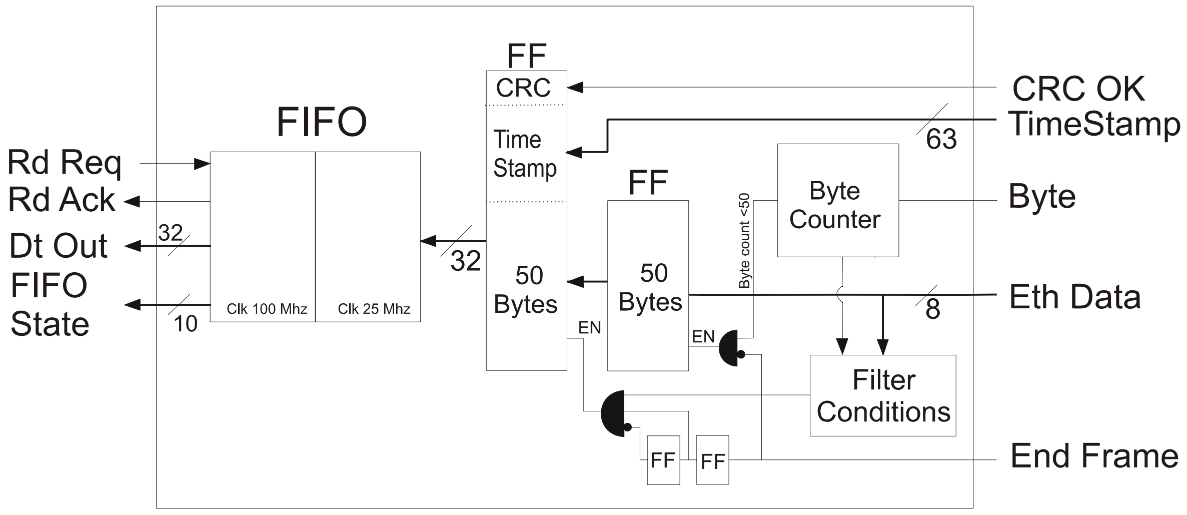


Figure 4-9: Quintuple Capture & Filtering Detail

Even though this figure gives a really good detail about how this module works a few considerations must be done before explaining the general behaviour of the module.

The main one is to clarify the data structure that arrives to the FIFO and later to the microprocessor.

1bit	31 bit	32bits	50 first packet bytes
CRC	Seconds	Nanoseconds	Data

As it can be notice, one bit from the second's field has been substituted by the CRC one. The 2^{32} field gives more 100 years count. Therefore nothing happens if the field is reduced by one.

Another quite good detail is the two Flip-Flops that delay the End Frame signal. For understanding why are there, it is compulsory to keep in mind that the CRC checker shows the valid CRC result one cycle after the arrival of the last word of the packet. So, delaying the decision, for introducing the data to the Flip-Flop next to the FIFO by one cycle, allows the synchronization with the rest of the information.
This waiting time is far for being a problem because is key to remember that there is a minimum inter-arrival time between packets, and at 100 Mbps it is bigger than 20 ns (two 100 MHz clock cycles).

The function of the device starts when a new packet arrives. A signal called 'New Frame,' which is not included in the figure, performs all the necessary actions for processing it in the Quintuple Capture module. In somehow it works like a reset.

Each new byte is stored in the 50 bytes Flip Flop. At the same time the "filter conditions" module select the quintuple characteristics from its packet. These are the source IP address, destination IP address, source port, destination port, and protocol.

Finally, when the End_Frame signal rises to '1' and if the filter conditions are fulfilled the EN signal goes to '1' allowing to pass the information from the 50 bytes FF, time-stamp and CRC to FIFO FlipFlop. This Flip Flop will send the information to the FIFO in 32 bits words.
The time-stamp has been requested to the time-stamp generator core as soon as the Carrier Sense signal has been activated. This signal is activated before the packet arrives, so it covers the delays that the packet may suffer until been time-stamped.

The FIFO handles the communication with the MicroBlaze using a handshake protocol.
MicroBlaze can ask the status of the FIFO, which means how many 32 bits words can be read. And of course, It can also request a 32 bit word.

Last but not least, a Quintuple Capture module is required for each filtering conditions and for each ETH interface. For example, if it is required an analysis of the traffic that attacks port 80, in both Ethernet senses, and then two Quintuple modules are needed. Otherwise, in just one sense, it would be only one. Even though this seems redundant, indeed it adds versatility to the design allowing to separate the analysis in different senses of traffic and therefore reducing the FPGA area needs in that situation.

4.4 System Integration

The system employs several interrupts which are described, in order from lower to higher priority, as follows:

- **MAC**: It is compulsory for the using of the LwIP library.

- **PPS**: It comes from processing the PPS signal from the GPS using a rising flack detector. This interrupt it is quite important for correcting the time-stamp source drift errors.
- **Timer**: The most important one. It is a must for XilKernel OS and LwIP.

The flow of the main program is the following one.

- Initialization of the time-module 4.2.
- Initialization of the XilKernel. Main thread launched.
- PPS interrupt handling configuration.
- Re-starting counter 4.2.
- Initialization of the LwIP.
- Launching of socket thread 4.4.3. Endless loop at this point.

In order to facilitate the software support and further develop, the program uses the following data structure.

```
typedef struct
{
  Xuint32 BaseAddress  ;       /* Base address of registers */

  Xuint32* Mode_Status ; /*Pointers for reading& writing in time module*/
  Xuint32* Error       ;
  Xuint32* Tuning      ;
  Xuint32* nSec        ;
  Xuint32* Sec         ;
  Xuint32* UpdSec      ;

  Xint32 nSec_error[window_size]; // For the windows drift correction system
  Xint32* p_error;

  Xint32 Register_nSec ;//Characteristics of a time_module
  Xint32 Register_Mode ;
  Xint32 Register_Error;
  Xint32 RegisterSec   ;
  Xint32 Tuning_average;
  Xint32 Tuning_step ;
  Xuint32 update_second;
} Time_module;
```

4.4.1 Time Drift Correction

The final implemented version of the algorithm works in the following way.

Firstly the counter-clock runs 30 seconds without correcting the time drift. At the end of these 30 seconds, it is calculated the average error and then fixes a constant value that will correct the counter each second.

After this process, the counter is reset and re-started.

As it has been explained in many sections, the error committed per second is requested from the 4.2.3, by the Microblaze once per second (PPS).
This error value must be processed in order to get a right tuning value.
This tuning value is defined as the number of times that the counter-clock must be delayed or accelerated in order to correct the error from the last second.

Two actions take place for achieving a nice result:

- Mean error: after a first estimation of the mean or average error the clock runs periodically thirty seconds for attempting to correct it. This means that a change in the mean error value is produced if the mean error in 30 seconds.
  Specifically, each second the error is added to the result of the sum of the 29 past errors. Once per thirty seconds the total accumulative value is dived by 30. If the quotient is more than 1 or less than -1 then the mean error is increased or decreased respectively.

- Drift error: It is defined by Step = Step/2 + Error where Error is the raw input from the time-stamp generator core and Step is a variable which accumulates the error. This recursive function aims to mitigate the effects of the peaks of the PPS quantification error.

Now, it is time to justify why this method works and rules.
All this method has been tested and developed under Matlab simulations before been implemented in the system. This simulation is based on a simple model of a time source. As it has been explained in some references in Section 3.4.1.2, the error between two time sources can be characterize as: $E(t) = A + B(t)$, where A is the average error and B represents the drift-value.
Thus, the simulation uses a normal distribution, where its mean is the mean error value observed with the oscilloscope and the variance is defined also in an empiric method.

In the first figure, there is a clear view of the raw error accumulation, without any correction, during 10000 seconds through simulation. As it was described in the Design Section 3.4.1.2, it follows an almost linear function.

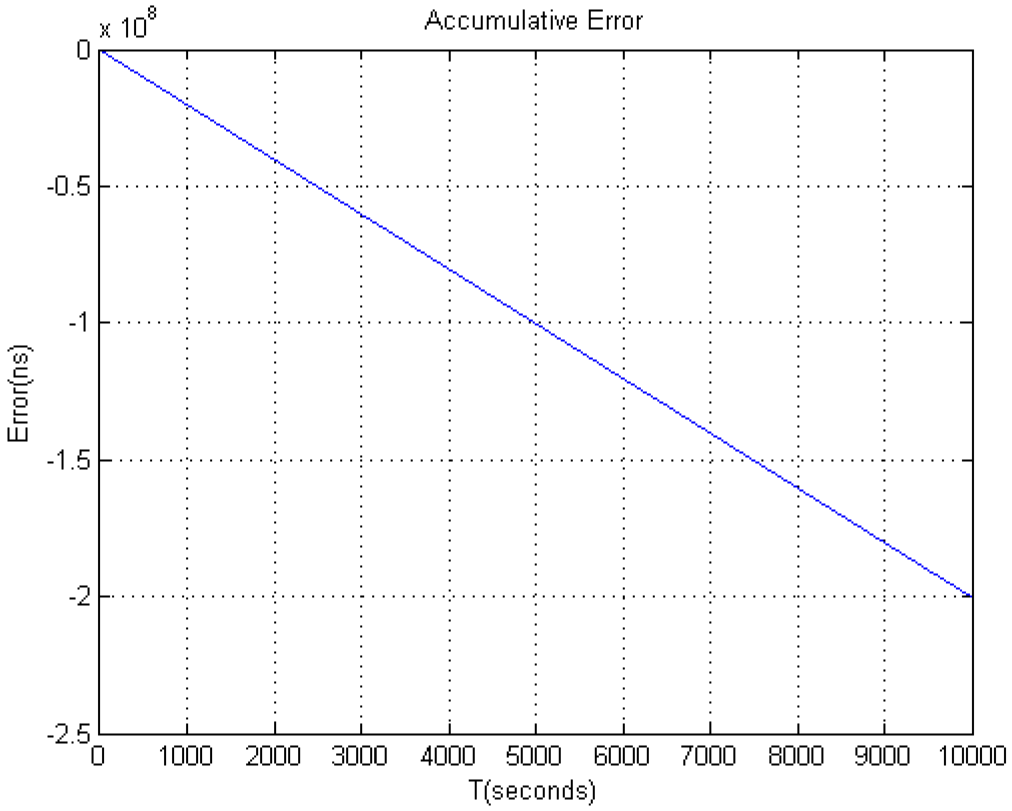


Figure 4-10: Accumulative Error in HW generated clock vs. GPS

It is impossible to know before hand the drift error, because this depends on many random factors as for example temperature. This means that there is no way to calculate a valid offset that offers warranty of reliability in the long term (few hours).

This is showed in the next figure, where only a 30 seconds evaluation of the mean error takes place.
GPS time is represented in red and the accumulative error of the FPGA clock in blue.

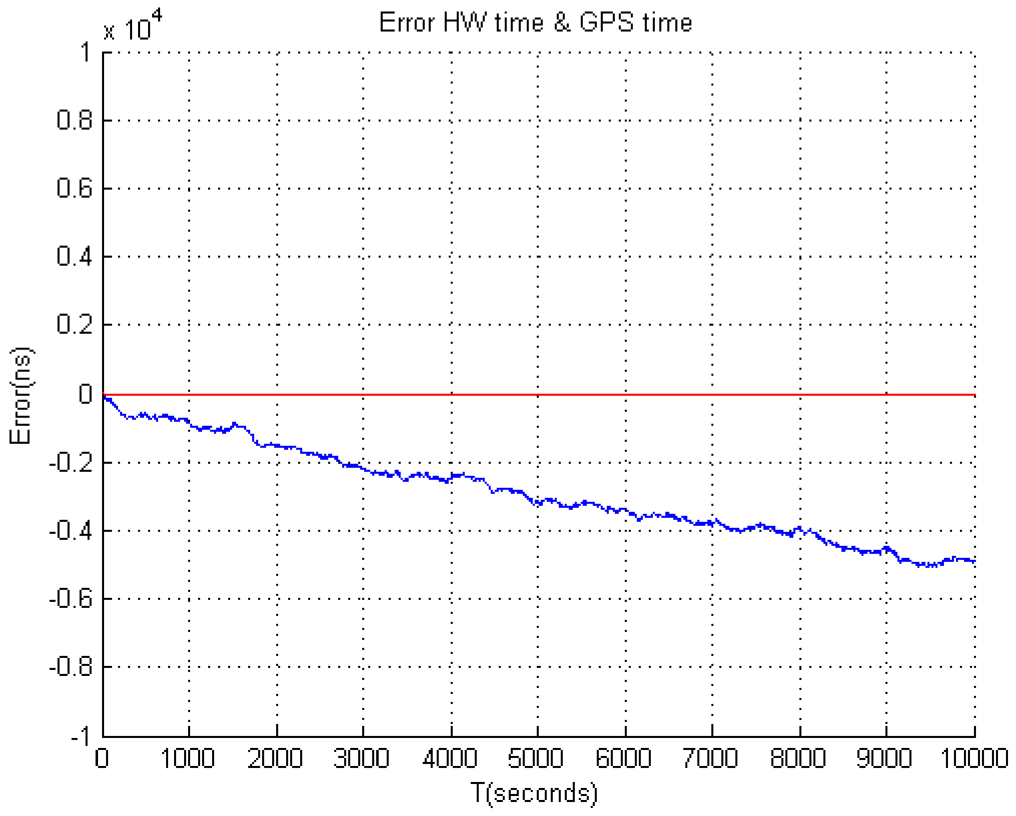


Figure 4-11: Accumulative Error 30 seconds average correction vs. GPS

The result has an easy explanation. No matters how well the mean of the error is calculated, because this mean can change over the time. Furthermore, as much error is committed trying to find this mean value as much error the system will have.

A clever idea would be to correct the mean average through all the time, which due to memory limitations means to modify the average each 30 seconds by adding 1 or -1 to the original value. This +1 or -1 corrective value is calculated from the last 30 seconds errors. If the accumulative error divided by 30 exceeds a positive threshold then average is correspondingly modified and vice versa.
So, in this way a bad luck 30 seconds estimation would not ruined our results.

The result of applying a 30 window average can be seen in the next figure.

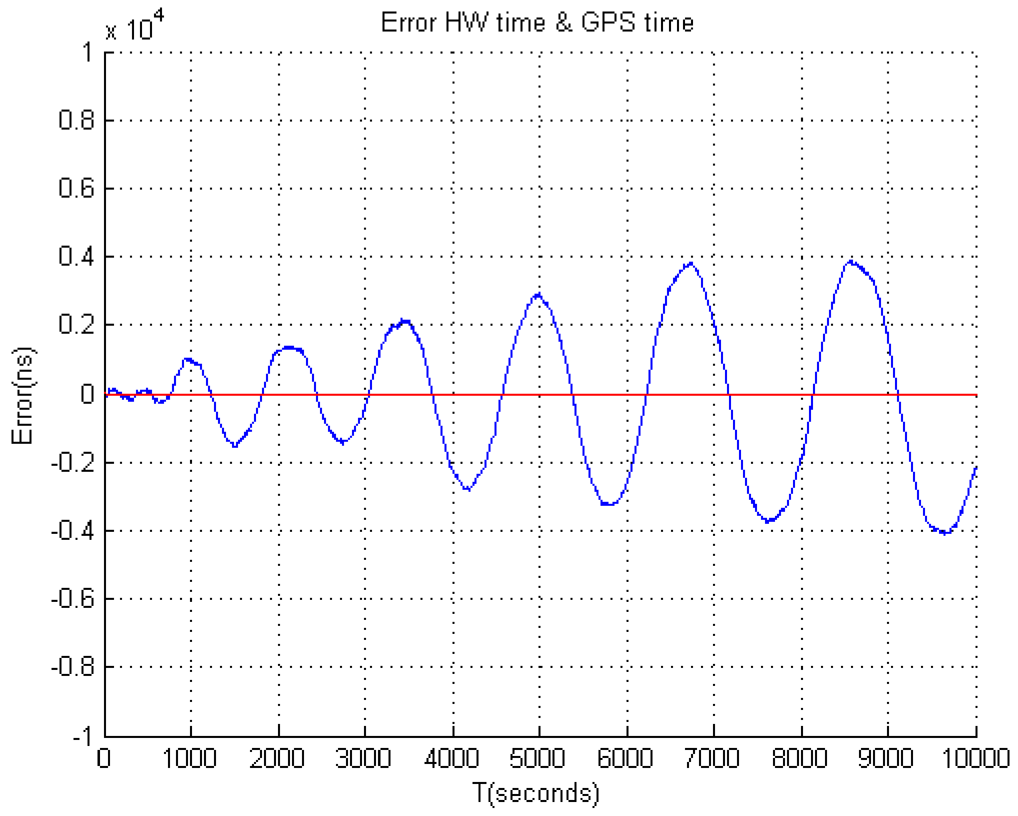


Figure 4-12: Accumulative Error 30 seconds window average correction vs. GPS

What had happened?

Of course, it is better than the previous model, but in the long term explodes. The problem is that no matter how well calculated is the average, because there is always going to be a bad luck 30 seconds window that it is going to influence in a bad way our first estimation.

This bad influence generates a non-stoppable accumulate error that generate worse average error calculation as the time increases.

Therefore, it has been showed the need of B(t), drift-value function, that will allow to correct the mean value in real time, as it can be seen in the next figure the result of the simulation of this formula. Again blue colour shows the error committed by the FPGA clock versus GPS clock in red.

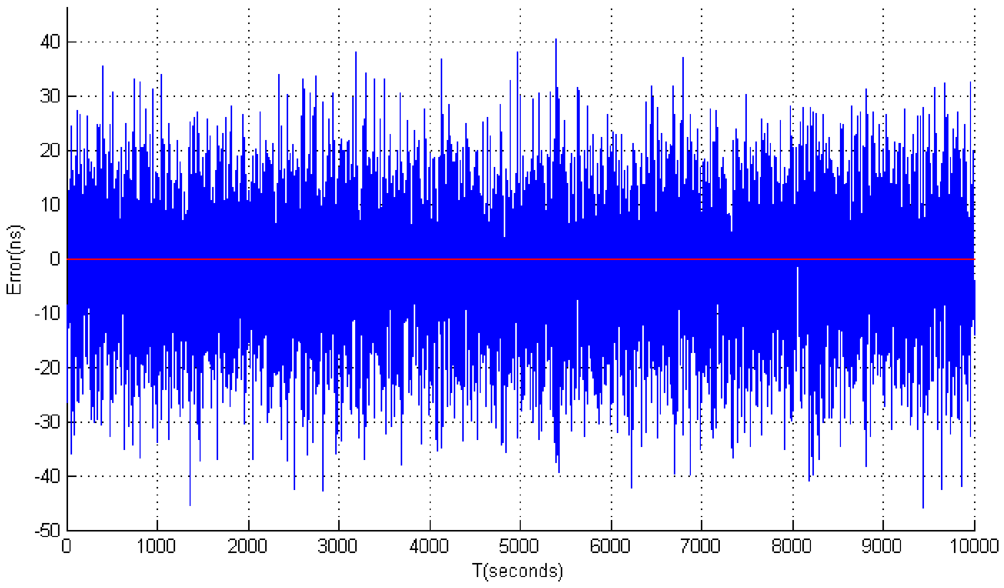


Figure 4-13: Accumulative Error: well corrected clock vs. GPS

The system is able to be stable between +/- 50 n, no matters how well the average was calculated, in other words, it is able to adapt to any change or bad luck error.

Just in case someone is thinking something like, Hey! What will happen if we just employ the B(t) drift correction?
In the next figure this question is answered.

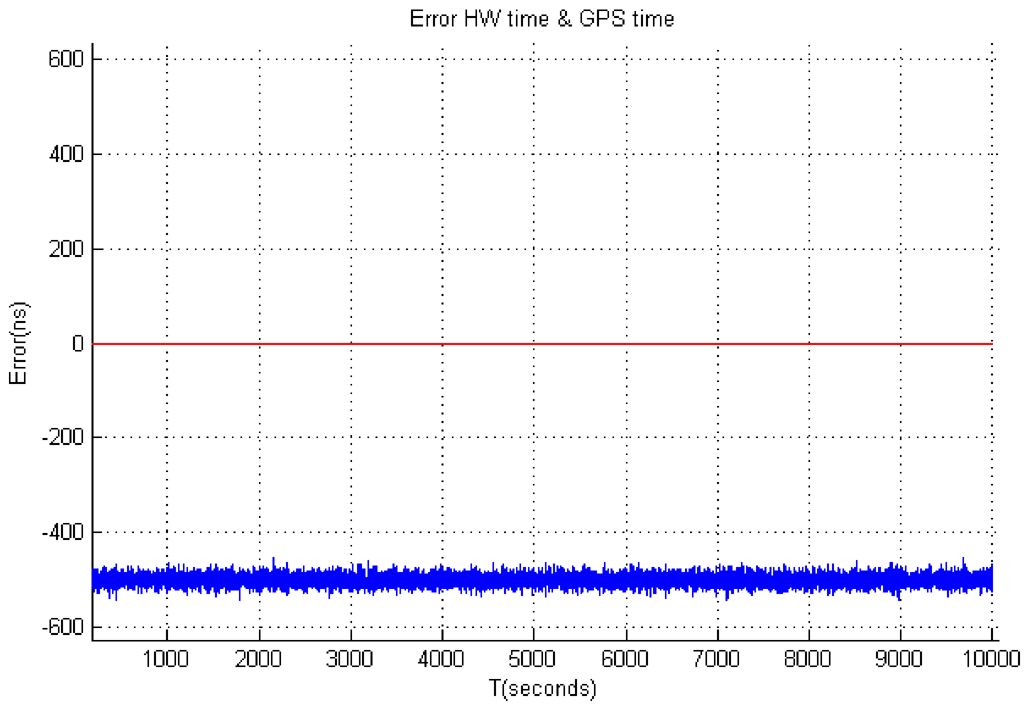


Figure 4-14: Accumulative Error: just time drift clock correction vs. GPS

Yes, the drift error is stuck at a constant point.

The last explanation is dedicated to clarify that the system works with 10 ns as our lowest unit. Moreover, it only admits integers as a reason of binary arithmetic. This is important because the smoothest tuning is a 10 ns change as well as our minimum error that of course it is also 10ns.

4.4.2 GPS communication

In the Section 3.2.3.3 an protocols introduction for working with the Trimble Resolution T receiver was done. Deciding to work under the NMEA 0183 or TSIP is not an easy decision to make. There are many positives aspects in both of them and also some disadvantages.

They key point is that TSIP offers quite more characteristics than its competitor. On the other hand, NMEA 0183 is a well known standard. This allows changing the vendor of GPS receiver almost without any problems. After analyzing the characteristics that TSIP offers, there are two that must be highlighted.

- **PPS error:** a brief summary about this error and its consequences was given in Section 3.4.1.1. TSIP presents a packet for minimizing this problem. The amount of quantization error present on each PPS output pulse is reported in packet 0x8F-AC. Here there are some figures which show the importance of this advantage. They can be compared against 3.4.1.1.1 .

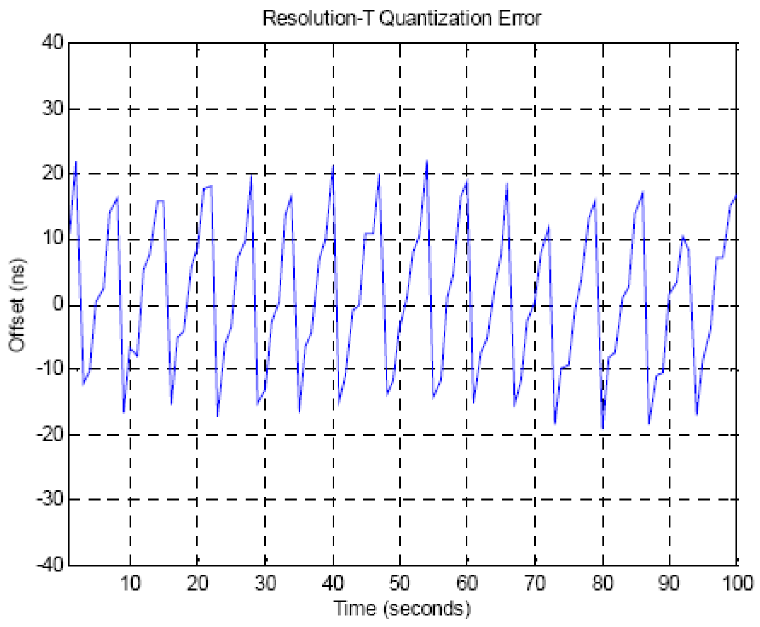


Figure 4-15: Resolution-T PPS Quantization Error [5]

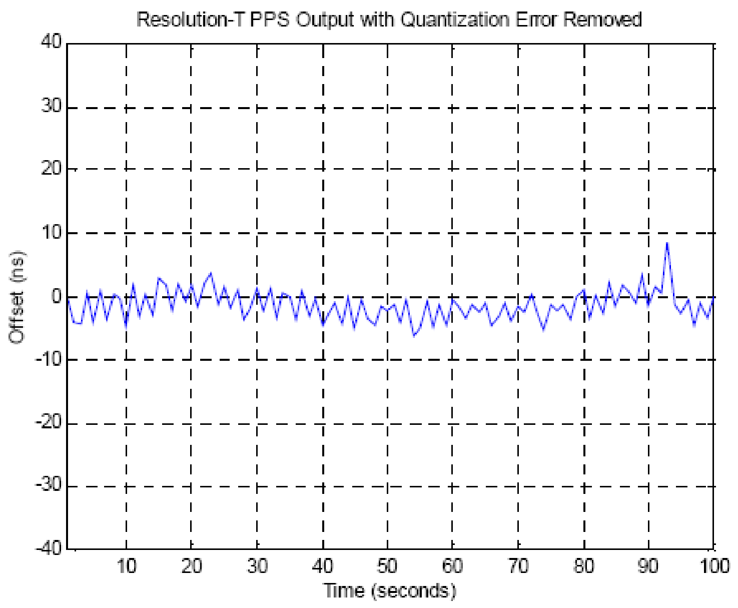


Figure 4-16: Resolution-T PPS Output with Quantization Error Removed [5]

Therefore, as has been explained with this packet a better clock accuracy can be achieved.

- **Seconds of the week:** NMEA 0183 ZDA time-packet shows the year, month, day, minutes and so on from and ancient date. This information must be translated to seconds, in order to have a compressive and optimum way of store this information in each captured packet, so each packet has a different time-stamp, in the same physical point of the analyze. However, TSIP brings again a nice solution. There is a packet 0x8F-AB that sends the number of seconds since Sunday at 00:00:00 GPS time for the current GPS week.
  It is easy, simple and warranties a perfect function of all the system. It does not need conversions either saving non-useful information.

TSIP has been chosen, as a consequence of these special characteristics of the protocol and of course, mainly because of this PPS error correction.

After this brief explanation about the protocol election, now is explained how communication works and which kind of information is transferred from the GPS receiver. All the information needed for the normal behaviour of the system is the seconds of week data. This is information is vital during the initialization of the time-stamp generator counter 4.2.

For getting this data and before starting the counter, the data is requested using a Uartlite device. The data received form the GPS receiver must pass several masks, in other to read the right data. If a number of attempts for reading the information are failed, then a critical error message which alert the user about the wrong initialization of the counter, and the lack of accuracy in its time-stamps.

Once that this procedure is finished, there is no real need of communication between the MicroBlaze and the Resolution-T, because the correction of the quantification error in the PPS signal has been postponed for future work.

4.4.3 Server Communications

In this section the socket thread is going to be explained.

The objectives for this thread are to set up the MAC, to open the target socket and last but not least to handle the reading of information from the Quintuple FIFO 4.3.2. and sending it to the aimed socket.

This thread is based on LwIP library and it follows a Berkeley socket model.
At the first point, the thread initializes the MAC of the system. Then it opens an UDP socket to the destination of the filtered information. .
And finally starts a polling loop. If the FIFO has any data for been read then it will be read and sent as an UDP packet.

5 Tests and results

5.1 CRC

CRC checker 4.3.1 has been tested in an exhaustive way. Specifically, it has passed four tests for showing a great reliability. The tests have executed in a Modelsim environment, so they aim to discard any logic problem in the CRC detection.

The first two tests have been extracted from the IEEE 802.3 manual, [53].

It is important to notice that these are not intended for checking the CRC performance. In fact in the manual it can be read "*The continuous random test pattern is a random test pattern intended to provide broad spectral content and minimal peaking that can be used for the measurement of jitter at either a component or system level*" [53].

Nevertheless, these tests offer a packet with its right CRC associated, which is more than enough for knowing how the CRC checker is performing.

The first packet and test:
"*Each packet in the long continuous random test pattern consists of 8 octets of PREAMBLE/SFD, followed by 1512 data octets (126 repetitions of the 12-octet modified RPAT sequence), plus 4 CRC octet*s" [53].

PREAMBLE/SFD:
55 55 55 55 55 55 55 D5

MODIFIED RPAT SEQUENCE (LOOP 126 TIMES)
BE D7 23 47 6B 8F B3 14 5E FB 35 59

CRC
94 D2 54 AC

The second:
"Each packet in the short continuous random test pattern consists of 8 octets of PREAMBLE/SFD, followed by 348 data octets (29 repetitions of the 12-octet modified RPAT sequence), plus 4 CRC octets"[53].

PREAMBLE/SFD:
55 55 55 55 55 55 55 D5

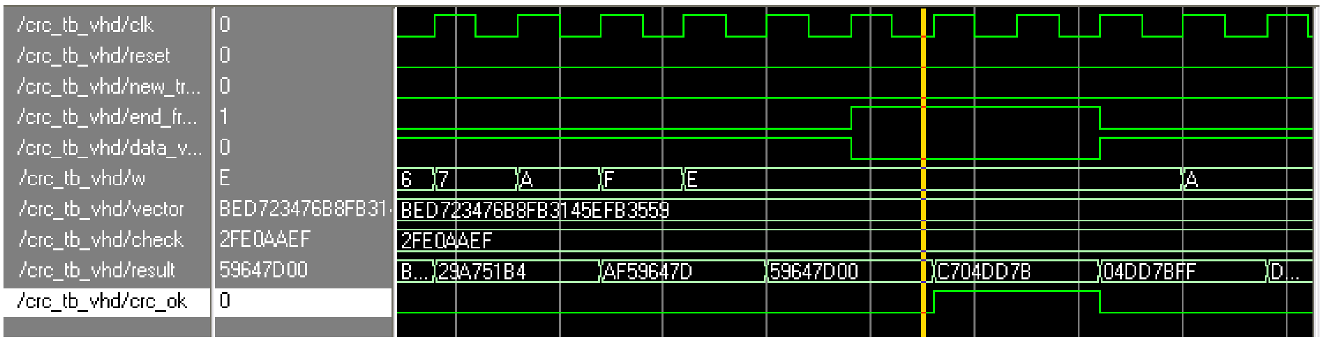
MODIFIED RPAT SEQUENCE (LOOP 29 TIMES)
BE D7 23 47 6B 8F B3 14 5E FB 35 59
CRC
2F E0 AA EF

After these two tests, it was the time to test a **raw capture packet**. This packet was captured using Chipscope [37] utility. Once more, the result was satisfactory confirming that the packet was without errors form the CRC point of view.

However, there are still some aspects that require a more complex test.

The ultimate test aims to show that the CRC checker is able to handle a stream of packets. This stream contains also a wrong packet in order to check that detecting bad CRC packets also works.
The stream is formed with the first packet tested, the second packet, intentionally modified for having a wrong CRC, and some raw capture packets.

The end of this test can be seen in the following figure.



```
/crc_tb_vhd/clk       0
/crc_tb_vhd/reset     0
/crc_tb_vhd/new_tr... 0
/crc_tb_vhd/end_fr... 1
/crc_tb_vhd/data_v... 0
/crc_tb_vhd/w         E
/crc_tb_vhd/vector    BED723476B8FB31
/crc_tb_vhd/check     2FE0AAEF
/crc_tb_vhd/result    59647D00
/crc_tb_vhd/crc_ok    0
```

Figure 5-1: Ultimate test: CRC end of packet Modelsim simulation

Please, notice that as was explained in Section 4.3.1, if the CRC code is correct, a 0xC704DD7B offset must appear as result in the following cycle, after the end of the frame.

5.2 Time-stamp accuracy

Achieving a good time-stamping accuracy is required for this project. This time accuracy can be measured in two contexts.

- **Digital measurement:**
  This analysis evaluates the time drift correction algorithm 3.4.1.2, 4.4.1.

  It consists on logging all the error and tuning average information during seven hours. So, when the MicroBlaze asks to the Time-stamp generator core 4.2 about the error committed in the last second, not only handles this information but send it trough a RS-232 to a PC.
  This PC logs all the RS-232 communication. After, this information is processed using Matlab. It is important to observe that the maximum resolution is 10 ns, due to the 100MHz clock. Moreover, only integers are allowed.

Here it can be seen one of the figures

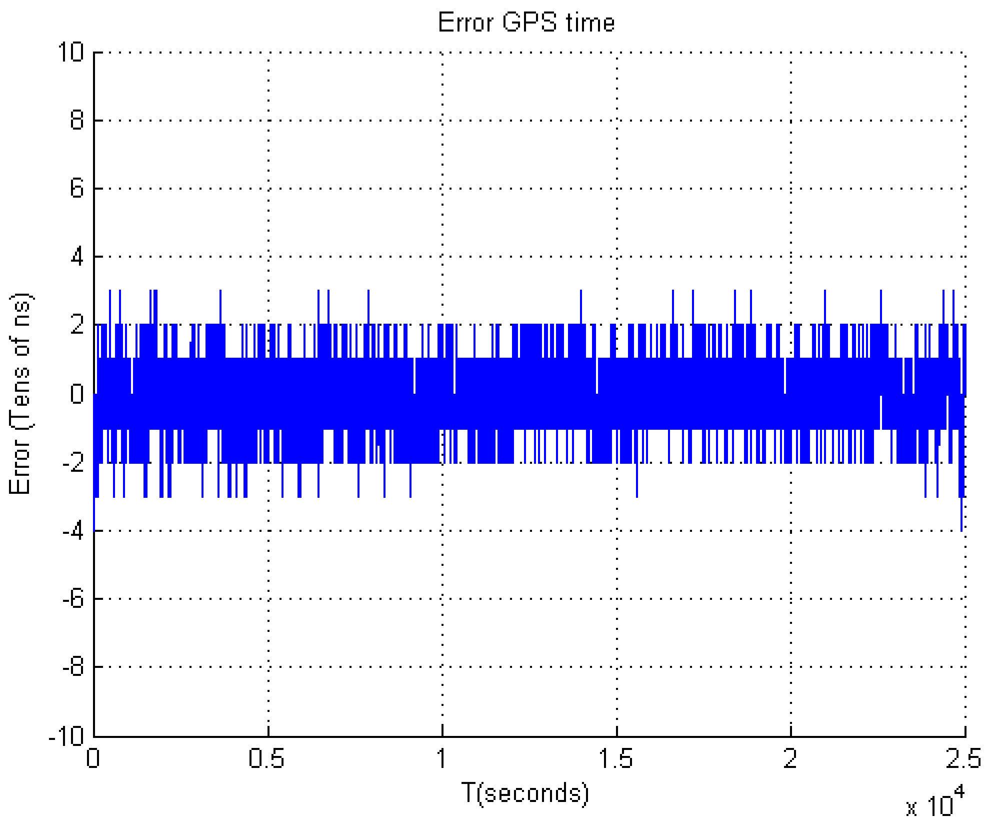


Figure 5-2: Error committed by the FPGA-generated PPS

It shows the registered error committed in 7 hours of log (~25000 samples).
For a better understanding of the waveform this figure is a zoom in. It represents the samples from 600 to 1000.

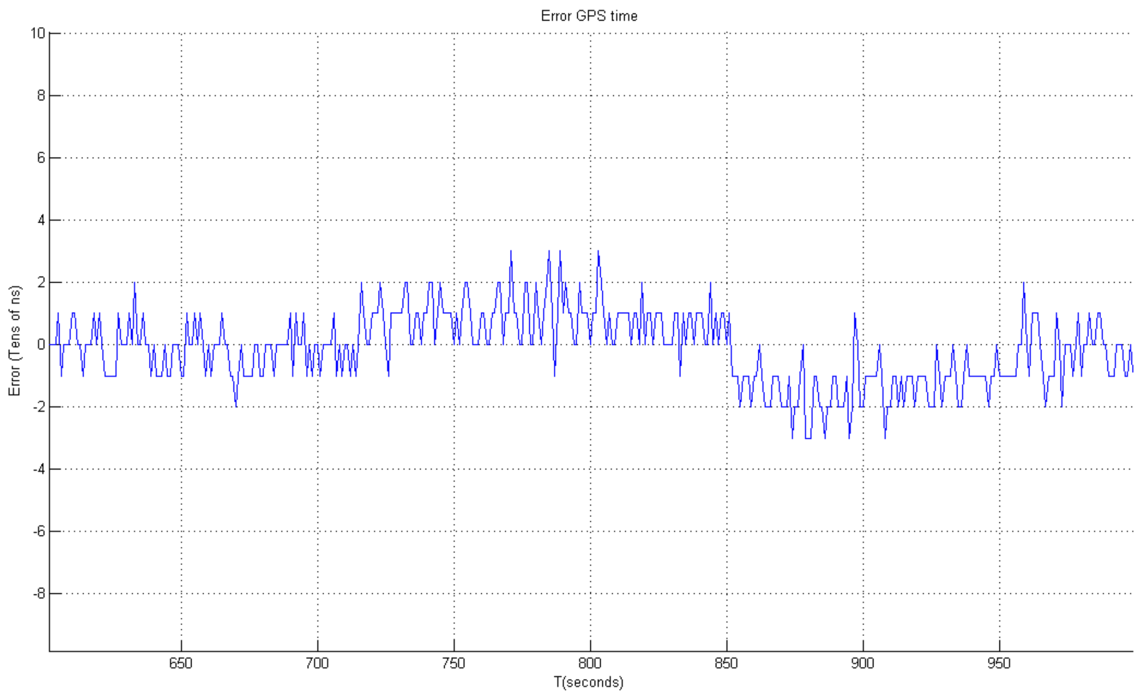


Figure 5-3: Detail of the Error committed by the FPGA-generated PPS

It is interesting to make a comparison with the figure 3-11. It can be seen that they share a similar waveform.

Another interesting figure is the importance of the thirty seconds window tuning average (mean error) calculation, which is explained in 4.4.1.

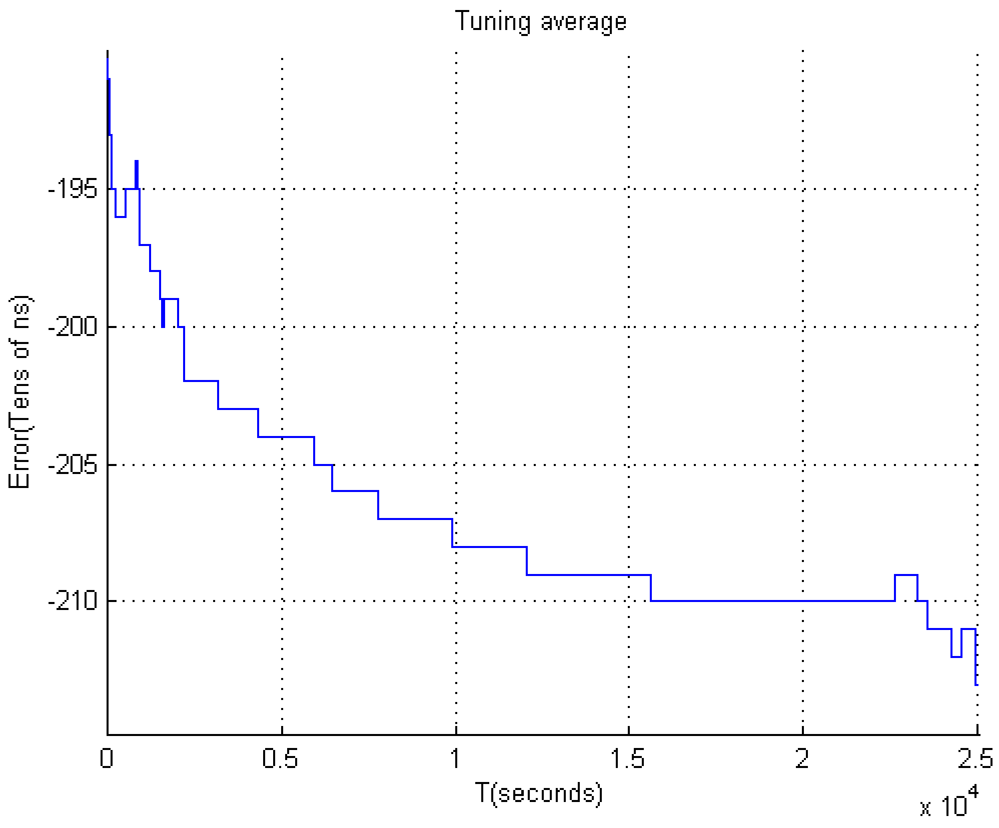


Figure 5-4: Mean error (30 seconds window) or tuning average values

It can be seen that this value changes during all the simulation and it shows the importance of this window algorithm for achieving a nice accuracy.

From the sample analysis some calculations has been done offering the following relevant information:

During the seven hours the clock was delayed 47690 ns, which means 1.91 ns per second. However, if it is just taken in consideration the last three hours and a half the clock is only delayed by 11860 or 0.95 ns per second, showing the importance of a well defined tuning average value.

Nevertheless, each time-stamp has an absolute average error of 6’5 ns, which is less than 10 ns or what is the same less than our unity.

- **Analog measurement:**

However, this digital analysis shows what the MicroBlaze eye can see. As it occurs with human eye, it can not see some relevant information.

As it was described in 4.2.1, a PPS output signal is generated when the system counter reach one second. This signal is measured with an oscilloscope and it is compared against the GPS one.
Therefore, this shows the **REAL** accuracy of the system time-stamps, which are based on following the GPS PPS signal. The better the system follows the PPS; better time-stamp accuracy is achieved.

In the following image a two hours test is displayed. The samples results are accumulated in green referring to the system generated PPS and in yellow for the GPS one.

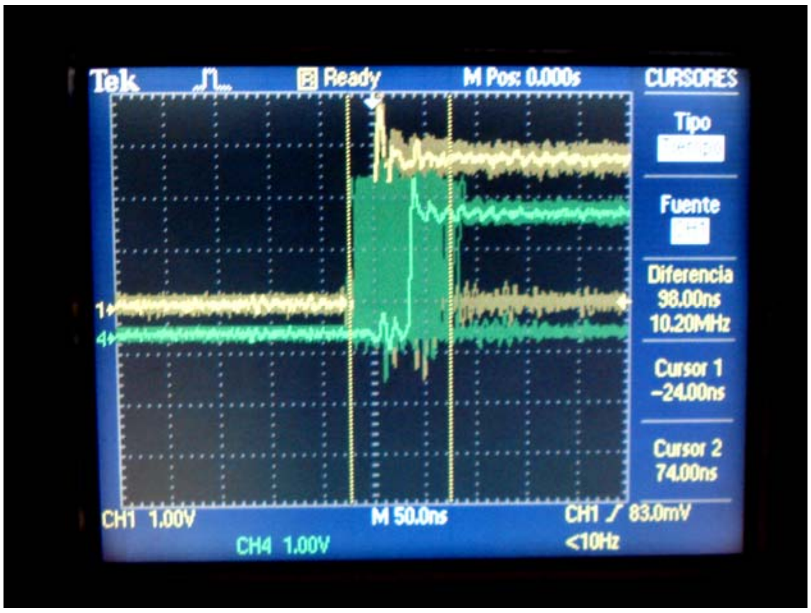


Figure 5-5: Oscilloscope capture showing GPS PPS vs. FPGA PPS

The image shows about 100 ns between the peaks errors. This is an about +/- 50 ns error which fulfils all the initial expectations.

Nevertheless, this error is not the committed vs. the UTC one. It has to be added a 2ns from the antenna wire, plus around 15 ns of the Resolution T module [5].

5.3 General performance

Finally a general test was realized for evaluating the general performance of the whole system.

This picture has been included for a better understanding.

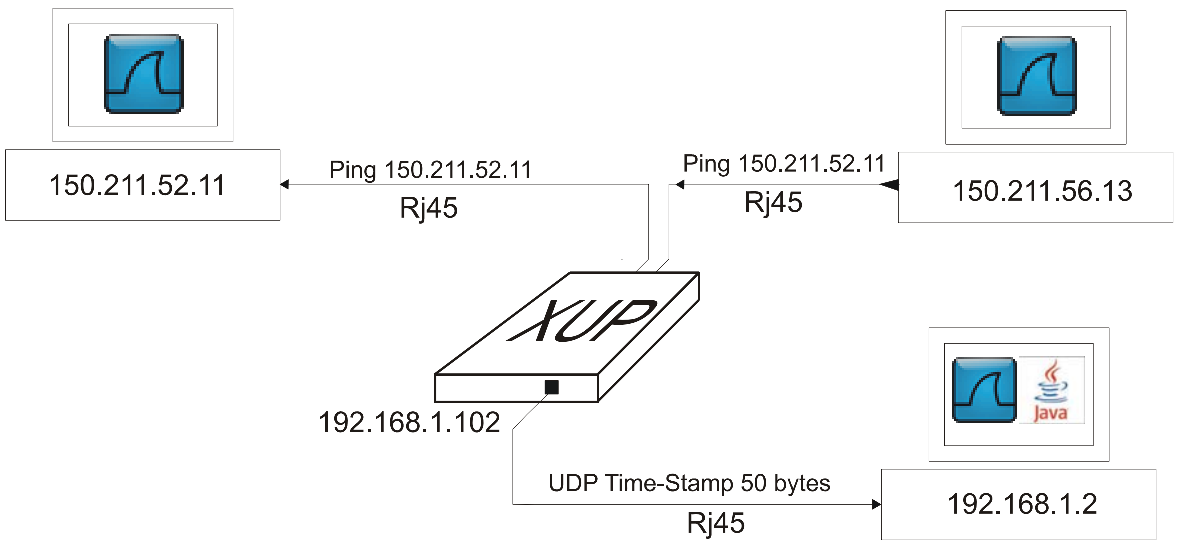


Figure 5-6: General performance test.

So, basically a computer is connected to another one trough the system Ethernet card interface.
The XUP development board is also connected using its own RJ45 interface with another computer. A network is established between them.

All the computers are running the "Wireshark" application in other to capture all the data running across its nets. However, 192.168.1.2 is also running a JAVA program. This program is based on BSD(Berkeley Sockets) and essentially runs an UDP socket on port 3000. It also prints the information that arrives to this socket, but only when receives a "0xABAB" sequence. As soon as is this sequence is detected, the next 4 bytes received are printed.

The test starts with a ping sent by 150.211.56.13 to 150.211.52.11. When this ping passes through the XUP then is filtered. Then, it is sent to the 192.168.1.2 divided in 8 Bytes words where both Wireshark and the java program register the incoming packet.

It must be said that some traffic is added between 150.211.56.13 and 150.211.52.11 just for showing that the filter is working fine.

5.4 FPGA characteristics and resources.

This section offers a summary about the utilization stats of the whole system. These have been taken from .mrp .par and .twr files. It also includes the ones concerning the IP cores designed for this project, 'TAP core' and 'Time-Stamp Generator core'.

5.4.1 The whole system

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	7761	13696	56%
Number of Slice Flip Flops	8102	27392	26%
Number of 4 input LUTs	9009	27392	32%
Number of bonded IOBs	172	556	30%
Number of BRAMs	43	136	31%
Number of GCLKs	8	16	50%
Number of PPC405s	0	2	0%
Number of MULT18X18s	3	136	2%
Number of DCMs	2	8	25%
Number of BSCANs	1	1	100%
Number of GTs	0	8	0%
Number of GT10s	0	0	0%
Number of BUFGMUXs	8	16	50%

Table 5-1 System utilization Summary

Timing characteristics:
Minimum period: 10.505ns (Maximum frequency: 95.193MHz)
Maximum path delay from/to any node: 3.009ns
Maximum net skew: 1.323ns

5.4.2 TAP core

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1258	13696	9%
Number of Slice Flip Flops	1590	27392	5%
Number of 4 input LUTs	1395	27392	5%
Number of bonded IOBs	166	416	39%
Number of BRAMs	2	136	1%
Number of GCLKs	6	16	37%

Table 5-2 TAP utilization summary.

Timing characteristics:

Minimum period: 6.472ns (Maximum Frequency: 154.507MHz)
Minimum input arrival time before clock: 5.048ns
Maximum output required time after clock: 4.530ns
Maximum combinational path delay: 5.075ns

5.4.3 Time-Stamp Generator core

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	107	13696	0%
Number of Slice Flip Flops	59	27392	0%
Number of 4 input LUTs	133	27392	0%
Number of bonded IOBs	170	416	40%
Number of GCLKs	1	16	6%

Table 5-3 Time-stamp generator utilization summary.

Timing characteristics:

Minimum period: 5.720ns (Maximum Frequency: 174.840MHz)
Minimum input arrival time before clock: 4.527ns
Maximum output required time after clock: 4.538ns
Maximum combinational path delay: 5.075ns

6 Conclusions and future work

6.1 Conclusions

A framework for the development of FPGA-based Ethernet monitoring system has been developed in this project.

An analysis of the state of art has also delivered. From this part it can be concluded that there is a extensive job to be done about improving the network monitoring systems.

Now it is time to highlight the advantages and disadvantages of the results.

- The system has reached a great time resolution 10ns. It is almost as great as the best DAG card [1] model which offers 7.5 ns.
- The time accuracy has also been quite precise. A +/- 50 ns error from the GPS PPS shows the evidence of the good results of both the time drift algorithm 4.4.1 and the clocking strategy. DAG 4.5G2 card offers +/- 100ns [1].
- A simple algorithm has been developed in order to handle the drift correction problem which saves developing time and FPGA resources.
- The time-stamp is selected when the Ethernet carrier sense signal rises to '1'. The Carrier sense signal rises to one sooner than the detection of the Preamble. Therefore, it does not matter how much time the packet takes for reaching the filter section.
- The system works for a 10/100Mbps velocity which is expected to increase in further versions. However, the architecture has been designed for supporting a 1 Gbps.
- The system works as an autonomous one. It has not been implemented in a PCI card which of course needs a computer. This helps to save money, to reduce the power consumption and it allows smaller dimensions, which adds versatility to the solution.
- Using a GPS source means that the antenna must have clear view of the sky with the aim of achieve a good performance.

6.2 Future work

A lot of future work has to be done because of the results in this project. These are some of the ideas and duties:

- Upgrading the system for allowing a 1 Gbps performance is a must. Scalability is one of the key points in surviving leather monitoring system design.

- Improving time accuracy removing the quantification error. The Trimble Resolution T has the key for reducing our overall error about +/- 10 ns. This fact, it is something that simply can not be forgotten in the incoming versions.

- The different time domains architecture allows a 175 Mhz (5.4.3) clock for time stamping generation. It the time-stamp generator core works at that speed it could offer a 5,7 ns resolution. The time-domain border should handle this problem, so it should not require a lot of work.

- A big hit would be to permit to change the filter rules, but without using reconfiguration or even download a new bit stream. This could be done sending from a client an UDP/TCP order to the monitoring system. In this order, a filter has to be selected and it also has to include the new rules. However, adding new filters or changing the structure of them, forces the need of downloading a new bit stream.

- Including a uLinux OS in the system could permit to process more the filtered data, instead of sending them almost in a raw condition.

- The implementation of a re-configurable PAM tree filtering could boost the versatility and capacity of the system. The PAM tree filter could be developed using several filtering cores 4.3.2 that will work as nodes of the tree.
  However, this structure brings new challenges that must be resolved. For example, could it be possible to share nodes of different trees? Which is the best hardware structure for the nodes?
  At least, there is an answer for the first question. Yes, perhaps it could be resolved employing FIFO's and the input of the node, and at the same time marking, in somehow, the nodes that the packet passes.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este proyecto se va a generar una plataforma para el desarrollo de un sistema de monitorización Ethernet.

Como complemento se ha entregado también un análisis del estado del arte. Del cual se puede ver la importancia en la mejora de estos sistemas.
Now it is time for highlighting the advantages and disadvantages of the results.

- El sistema ha alcanzado una buena resolución, concretamente de 10ns. Casi equiparable a la mejor de las tarjetas DAG [1] que ofrecen unos 7.5 ns.

- La precisión temporal a sido también bastante precisa. El algoritmo de corrección de la deriva temporal 4.4.1.ha cumplido con sus objetivos, ya que el error tan solo es una ventana de +/- 50 ns. Las tarjetas DAG 4.5G2 ofrecen +/- 100n [1] .

- Un algoritmo de corrección temporal ha sido propuesto. Este sencillo algoritmo permite ahorrar tiempo de desarrollo y recursos lógicos de la FPGA.

- El proceso de marcar temporalmente al paquete se produce cuando la señal “Carrier Sense”, encontrada en Ethernet, se active a ‘1’. Este “Carrier Sense” llega incluso antes que el preambulo. Lo cual permite que aunque el paquete se retrase en su procesado durante el filtrado, la marca temporal refleja la máxima fidelidad respecto al momento de llegada del paquete.

- El sistema permite capturar tráfico a una velocidad de 10/100Mbps. Pero la arquitectura ha sido pensada para alcanzar 1 Gbps

- El sistema trabaja de forma autónoma. Al no haber sido desarrollado mediante una tarjeta PCI, la cual requiere de un ordenador que la acoja. Esto permite que el dispositivo creado ahorre costes, tenga un consumo mas reducido y disponga de dimensiones más reducidas, generando versatilidad a la solución.

- El uso de una fuente temporal basada en GPS significa que existe la necesidad de ubicar su antena en algún lugar donde pueda ver el cielo abierto, ya que si no, trabajará por debajo de sus prestaciones.

6.2 Trabajo futuro

En esta parte se muestras algunas de las ideas para mejorar este proyecto en un futuro.

- Mejorar el sistema para llegar a trabajar con velocidades de 1 Gbps es una obligación para cualquier sistema de monitorización.

- Mejorar la precisión temporal eliminando el error de cuantificación. El dispositivo Trimble Resolution T puede permitir reducir el error en +/-10ns. Esto es algo que simplemente no se puede obviar para versiones futuras.

- La arquitectura con diferentes dominios de reloj puede permitir alcanzar 175Mhz al reloj empleado en la generación de marcas de tiempos. Si se logra que trabaje a esa velocidad se podría llegar a ofrecer una resolución de hasta 5,7 ns.

- Un gran avance sería el permitir cambiar las reglas de filtrado, pero sin usar la reconfiguración ni tampoco descargar de nuevo el "Bit stream". Esto podría ser realizado mandando una orden bajo UDP/TCP desde el cliente. El paquete solamente tendría que seleccionar el filtro al que modificar los parámetros y la información para las nuevas reglas. No obstante, si el cliente desea añadir o quitar filtros no hay mas remedio que reconfigurar la FPGA.

- Introducir un sistema uLinux puede acarrear múltiples beneficios. Entre ellos el mayor procesamiento de los datos filtrados, permitiendo la generación de histogramas, en vez de mandarlos casi sin procesar.

- La implementación de un árbol PAM reconfigurable podría disparar la versatilidad y capacidad del sistema. El árbol PAM podría ser desarrollado empleando varios "filtering cores" que serían los nodos del árbol. Esta nueva estructura traería nuevos retos que deberían de ser resueltos. Por ejemplo, ¿podrían ser posible compartir nodos de diferentes árboles? ¿Cual sería la mejor estructura hardware para integrar los nodos?
  Al menos, se puede responder a la primera pregunta. Si, quizás podría ser resuelto utilizando FIFOs a la entrada del nodo y al mismo tiempo marcando de alguna manera el paso de los paquetes por según que nodos transcurra.

References

[1] Endace Ltd., “Endace DAG Time-Stamping Whitepaper”. http://www.endace.com/resources/whitepapers.

[2] Stephen Donely, “High Precision Timing in Passive Measurement of Data Networks”, Ph.D. Thesis, Yhe University of Waikato. http://wand.cs.waikato.ac.nz/pubs.php.

[3] John Blyler, "Navigating the Silicon Jungle: FPGA or ASIC?", Chip Design Magazine, June/July 2005. http://www.chipdesignmag.com/display.php?articleId=115&issueId=11.

[4] Christian Peter, “Hardware Compilation and the Handel-C language”, Oxford University Computing Laboratory, UK. http://web.comlab.ox.ac.uk/oucl/work/christian.peter/overview_handelc.html.

[5] Navigation Ltd., “Resolution T System Designer Reference Manual”. http://www.trimble.com/tmg_resolutiont.shtml, 2005

[6] John C Eidson, “IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”. Agilent Technologies. 2003. Http://www.ieee.org/scv/ims/Meetings/IM_Society_IEEE_1588.pdf

[7] John C Eidson, “Measurement, Control and Communication Using IEEE 1588”.. April 2006.

[8] Silvana Rodrigues, “IEEE-1588 and Synchronous Ethernet in Telecom” 2007 International IEEE Symposium on Precision Clock Synchronization (ISPCS) for Measurement, Control and Communication Vienna, Austria, October 1-3, 2007

[9] MOMENT Project http://www.fp7-moment.eu/

[10] Javier Aracil, Jesus Martinez Garcia, Alfredo Salvador Gonzalez, Jorge E. López de Vergara, “Analysis of monitoring services”, MOMENT Project, March 2008

[11] SCAMPI Project http://www.ist-scampi.org

[12] LOBSTER Project http://www.ist-lobster.org/

[13] Paul Asthon, Algorithms for on-line clock synchronization, Deparment of Computer Science, University of Canterbury, Dec 1995

[14] Evangelos Markatos, Lessons learned from the LOBSTER project. Workshop on Learning from large scale attacks on the Internet - Policy Implications on 17 January 2008 in Brussels, Belgium http://www.ist-lobster.org/publications/presentations/markatos-attacks.pdf

[15] ETOMIC Project http://www.etomic.org/

[16] MOMENT D 2.1 Analysis of monitoring services http://www.fp7-moment.eu/images/FP7-MOMENT-WP2-D2.1.pdf

[17] Eduardo Magaña Lizarrondo, Técnicas eficientes de Filtrado para Monitorización de Redes de Comunicaciones, Universidad P´ublica de Navarra, Pamplona, July 2001.

[18] Young H.Cho, Shiva Navab, William H. Mangione-Smith, “Specialized Hardware for Deep Network Packet Filtering”, 2002

[19] D. Morat´o, E. Maga˜na, M. Izal, J. Aracil, F. Naranjo, F. Astiz and U. Alonso, “The European Traffic Observatory Measurement Infraestructure (ETOMIC): A testbed for universal active and passive measurements”, Universidad P´ublica de Navarra, 2006 Pamplona, Spain

[20] Jerry Case, Nupur Gupta, Jayant Mittal and David Ridgeway,“Design Methodologies for Core_Based FPGA Designs”,April 9, 1997

[21] Jason G. Tong, Ian D. L. Anderson and Mohammed A. S. Khalid,” Soft-Core Processors for Embedded Systems”, 1-4244-0765 University of Windsor - Department of Electrical and Computer Engineering Research Centre for Integrated Microsystems Windsor, Ontario, Canada,2006

[22] Grid Technology: http://www.sun.com/software/grid/overview.xml

[23] DIMES project: http://www.netDimes.org

[24] EVERGROW project: http://www.evergrow.org/news.php

[25] RIPE Test traffic Monitoring project http://www.ripe.net/ttm/

[26] Fotis Georgatos, Florian Gruber, Daniel Karrenberg, Mark Santcroos, Ana Susanj, Henk Uijterwaal and René Wilhelm. “Providing Active Measurements as a Regular Service for ISP’s”. Amsterdam, April 2001. http://www.ripe.net/projects/ttm/Documents/Presentations/2001_PAM.pdf

[27] The Network Time Protocol (NTP) Distribution http://www.eecis.udel.edu/~mills/ntp/html/index.html

[28] Cheryl Pellerin, http://www.america.gov/st/washfile-english/2006/February/20060203125928lcnirellep0.5061609.html, February 2006.

[29] NMEA 0183 Standard http://www.nmea.org/pub/0183/

[30] Teruhisa Tsujino, Effectiveness of the Quasi-Zenith Satellite System in Ubiquitous Positioning. http://www.nistep.go.jp/achiev/ftx/eng/stfc/stt016e/qr16pdf/STTqr1607.pdf

[31] Ron Wilson, “Will new ideas dim the future of FPGAs? Structured ASICs and microcontrollers renew the debate”http://www.edn.com/blog/1690000169/post/190031219.html, August 2008

[32] “Atmel’s Customizable Microcontrollers Outperform FPGAs, with Lower Power and Cost” http://www.industrial-embedded.com/news/db/?6975

[33] Koji Gardiner, “Comparing Power Consumption of FPGAs with Customizable Microcontrollers” , Stanford University, March 2008

[34] Steve Kilts, Advanced FPGA Design Architecture, Implementation and Optimization. Pages 84-97.

[35] D. Ficara, S. Giordano, F. Oppedisano, G. Procissi, F. Vitucci, “A cooperative PC/Network-Processor architecture for multi gigabit traffic analysis”, 2008

[36] Xilinx, “Xilinx University Program Virtex-II Pro Development System. Hardware Reference Manual” April, 2008

[37] Xilinx,ChipScope Pro Tool http://www.xilinx.com/ise/optional_prod/cspro.htm

[38] PowerPC 405 Processor Block Reference Guide http://www.xilinx.com/support/documentation/user_guides/ug018.pdf

[39] Xilinx,"MicroBlaze Processor Reference Guide Embedded Development Kit EDK 9.1i", MicroBlaze Processor Reference Guide UG081 http://support.xilinx.com/support/techsup/tutorials/index.htm , September 2007

[40] Xilinx, "Microblaze Processor" http://www.xilinx.com/products/design_resources/proc_central/microblaze.htm

[41] Xilinx, Xilkernel v 3.00

[42] Xilinx LibXil File, June 2006

[43] Xilinx, Platform Studio and the EDK http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

[44] Xilinx, On-Chip Peripheral Bus V2.0 with OPB arbiter (v1.10c), December 2005 http://www.xilinx.com/support/documentation/ip_documentation/opb_v20.pdf

[45] Xilinx, "OPB IPIF", Product Specification DS414, October 2006

[46] J. Arnold Douglas Stochastic Model Estimation of Network Time Variance, TrueTime, Inc, 2003

[47] Christ of Fetzer, Flaviu Cristian "An Optimal Internal Clock Synchronization Algorithm"

[48] B. Sliwczynski, "An automatic correction of time drift error in measurement systems" 1992

[49] Paul Ashton, "Algorithms for off-line clock synchronisation" Department of Computer Science, University of Canterbury, December1995

[50] Weidong Lu, "Designing TCP/IP Functions In FPGAs".DELFT University of Technology. July, 2003

[51] Chris Borelli, Xilinx, "IEEE 802.3 Cyclic Redundancy Check", March 2001

[52] Nair, R.; Ryan, G.; Farzaneh, F., "A symbol based algorithm for hardware implementation of cyclicredundancy check (CRC)", 1997. http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel3/4918/13570/00623934.pdf?arnumber=623934

[53] IEEE, "IEEE 802.3 standard section 3".

Glossary

API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BSD	Berkely Sockect
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
DRAM	Dinamic Random Access Memory
EDK	Embedded Design Kit
FIFO	First In First Out
FPL	Field Programmable Device
FPGA	Field Programmable Gate Arrays
FSL	Fast Simple Link
ICMP	Internet Control Message Protocol
IP	Intelectual Property
IPIF	Intelectual Property Interface
IOB	Input Output Block
LAN	Local Area Network
LCA	Logic Cell Array
MAC	Media Access Controller
OPB	On-chip Peripheral Bus
OS	Operating System
PCI	Peripheral Component Interconnect
PLL	Phase Locked Loop
PPS	Pulse Per Second

RISC	Reduced Instruction Set Computer
SoC	System on Chip
VHDL	VHSIC Hardware Description Language

Anexos

A PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)....... .... 2.000 €
- Compra de equipos Hardware .... 2100€
- Alquiler de impresora láser durante 6 meses .... 50 €
- Material de oficina .... 150 €
- Total de ejecución material .... 3.300 €

2) Gastos generales

- 16 % sobre Ejecución Material .... 352 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material .... 132 €

4) Honorarios Proyecto

- 640 horas a 15 € / hora .... 9600 €

5) Material fungible

- Gastos de impresión .... 60 €
- Encuadernación .... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto .... 14160 €

7) I.V.A. aplicable

- 16% Subtotal Presupuesto .... 2265.6 €

8) Total presupuesto

- Total Presupuesto .... 16425,6 €

Madrid, Mes Septiembre de 2008
El Ingeniero Jefe de Proyecto

Fdo.: Fernando Alonso Marcos
Ingeniero Superior de Telecomunicación

B PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un Sistema de monitorización de Ethernet. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es

obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.