Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-MS-2015-01

M.Sc. Thesis

Scalable Network Based Clock Synchronization for Digital PET System

Martijn Bijwaard B.Sc.

Abstract

Typically in synchronous digital systems time-based tasks are synchronized using hard-wired clock distribution networks. However in the growing application of large (wireless) sensor networks it is not always feasible to connect each sensor node to a hard-wired clock distribution network. Generally in such systems the sensor nodes are only connected via a (wireless) communication network, hence there have been implementations where clocks are synchronized via the network nodes, in general these solutions synchronize the network nodes in the range of milliseconds up to microseconds. Though some applications, like SPADnet, demand a network based clock synchronization in the range of picoseconds.

Currently the SPADnet system is equipped with a hard-wired clock distribution network, this clock network does not scale well. Since one of the goals of SPADnet is to create a scalable system, a network based clock synchronization solution is a better alternative.

In this thesis a hybrid solution is presented, where a hard-wired clock synchronization network is combined with a network based clock offset estimator. This combination enables scalability while maintaining high precision. A novel least squares synchronization algorithm is optimized and implemented in hardware equipped with a delayline FPGA TDC, allowing picosecond clock synchronization.

Faculty of Electrical Engineering, Mathematics and Computer Science **Delft University of Technology**

Scalable Network Based Clock Synchronization for Digital PET System
a picosecond hardware implementation for SPADnet

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

Martijn Bijwaard B.Sc.
born in Assendelft, The Netherlands

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **“Scalable Network Based Clock Synchronization for Digital PET System”** by **Martijn Bijwaard B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 13-03-2015

Chairman:	prof.dr.ir. E. Charbon
Advisor:	ir. C. Veerappan
Committee Members:	prof.dr.ir G. Leus
	dr. ir. S. Wong

Abstract

Typically in synchronous digital systems time-based tasks are synchronized using hard-wired clock distribution networks. However in the growing application of large (wireless) sensor networks it is not always feasible to connect each sensor node to a hard-wired clock distribution network. Generally in such systems the sensor nodes are only connected via a (wireless) communication network, hence there have been implementations where clocks are synchronized via the network nodes, in general these solutions synchronize the network nodes in the range of milliseconds up to microseconds. Though some applications, like SPADnet, demand a network based clock synchronization in the range of picoseconds.

Currently the SPADnet system is equipped with a hard-wired clock distribution network, this clock network does not scale well. Since one of the goals of SPADnet is to create a scalable system, a network based clock synchronization solution is a better alternative.

In this thesis a hybrid solution is presented, where a hard-wired clock synchronization network is combined with a network based clock offset estimator. This combination enables scalability while maintaining high precision. A novel least squares synchronization algorithm is optimized and implemented in hardware equipped with a delayline FPGA TDC, allowing picosecond clock synchronization.

Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this Master thesis project. Especially I would like to thank Edoardo Charbon for providing me the chance to work on this fascinating project, and Chockalingam Veerappan for his daily guidance throughout this thesis.

Also I want to thank the members of the Circuits and Systems group for inviting me to join the group activities, which I really enjoyed.

Finally I would like to thank my parents, brother and girlfriend for their support during my complete master study.

Martijn Bijwaard B.Sc.
Delft, The Netherlands
13-03-2015

Contents

List of Figures

List of Tables

Introduction 1

1.1 Motivation

In systems like wireless sensor networks (WSN) [1], SPADnet [2] and OLFAR [3] clock synchronization is of great importance. In these systems time-based tasks for instance: data sampling, time-stamping of data, data fusion, sleep and wake-up coordination and synchronized communication, requires that the clocks of each individual node are synchronized. Typically in synchronous digital systems the clock is distributed via a hard-wired clock distribution network to each component. But in (wireless) network based systems, it is not always feasible to connect each node to a hard-wired clock distribution network. In these systems each node will operate on their own local clock. However, to perform time-based tasks, the clocks of these nodes require to be synchronized [4]. The only way to do so is to synchronize via the network. Implementations exist for network based synchronization, in general these solutions synchronize the network nodes in the range of milliseconds up to microseconds [5] [6] [7] [8].

This thesis focusses on the SPADnet system [2]. This system uses a network of modules for photon-starved biomedical applications, such as positron emission tomography (PET). Each module relies on a hard-wired clock distribution network, driven by a single clock oscillator, to allow the execution of time-based tasks. One of the main goals of SPADnet is to construct a scalable system. Since the hard-wired clock distribution is not scalable, this thesis investigates if the SPADnet clock distribution network can be made scalable using network based approach.

1.1.1 SPADnet requirements

The requirements for the clock synchronization depends on the configuration of SPADnet. Three types of PET configurations exist: pre-clinical, brain and clinical. In time-of-flight brain and clinical systems clock synchronization better than 50 ps is required. For pre-clinical systems that do not use time-of-flight information, clock synchronization around 500 ps is sufficient.

1.1.2 Thesis goals

The goal of this thesis is to answer the following questions:

1. What is the performance of the hard-wired clock network of SPADnet?
2. Can a network based solution improve the scalability of SPADnet?

1.2 Contribution

The thesis presents a survey on various clock synchronization techniques (hard-wired and network based) for SPADnet. The existing hard-wired clock distribution network is characterized. Then it is investigated if network based clock synchronization can improve the scalability of SPADnet. For this the performance of novel clock synchronization techniques are examined, and a study is done on how to apply these techniques to SPADnet.

A hybrid clock synchronization solution is invented and implemented in a field programmable gate array (FPGA). This solution combines a hard-wired clock distribution circuit with a network based phase estimator, enabling scalability while maintaining picosecond performance. The performance of this hybrid solution is compared to the existing hard-wired clock distribution network of SPADnet, and tested in a three node system.

Although this thesis focusses on the SPADnet system, the findings can be used as a guideline for other systems as well.

The main contributions of this thesis are:

1. Characterization of the hard-wired clock distribution network of SPADnet.
2. Survey on network based clock synchronization techniques.
3. Implementation and verification of a hybrid synchronization solution.
4. Comparative study on hard-wired versus network based clock synchronization solution for SPADnet.

1.3 Related work

Probably the most used network based time synchronization protocol is the network time protocol (NTP) [9]. NTP is an internet protocol that synchronizes computers to within a few milliseconds of coordinated universal time (UTC). NTP is based on measuring the round-trip delay between the client and, preferably multiple, time-servers.

The growing interest of collecting physical and environmental data on a large scale using Wireless Sensor Networks (WSN), demands network based clock synchronization. Without synchronized sensor modules, time-stamping of data, data fusion, sleep and wake-up coordination and synchronized communication would fail. Multiple solutions for synchronization in WSN are available [5] [6] [7] [8]. In general these solutions synchronize the clocks in the range of milliseconds up to microseconds.

However some applications, for example SPADnet, demand more precise synchronization of the clocks. One of the most novel clock synchronization algorithms are presented in [10] and [11]. It is stated that these solutions approach the theoretical limits of clock synchronization. This thesis will investigate if they are applicable for SPADnet.

1.4 Application: SPADnet

The SPADnet FP7 European project aims to develop a new generation of smart, large area networked image sensors, based on a conventional CMOS fabrication technology, for photon-starved biomedical applications. SPADnet will build ring-assembly modules for positron emission tomography (PET) medical imaging, and carry out performance tests in a PET system evaluation test bed [2].

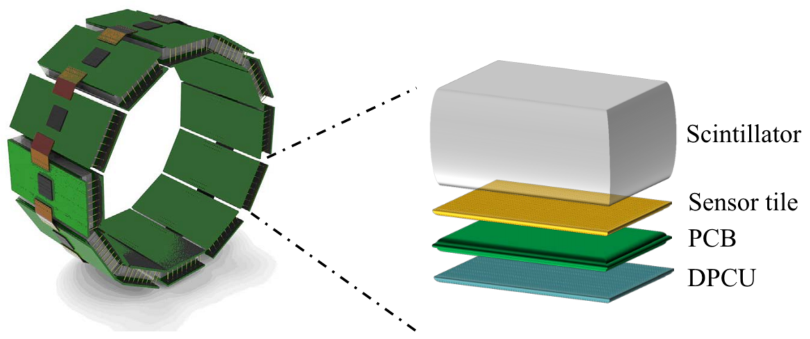


Figure 1.1: SPADnet system

1.5 Positron emission tomography

PET is a technique to create a three dimensional image of functional processes in the body. The patient is injected with a positron-emitting radionuclide on a biologically active molecule, also called a tracer. There are different kinds of biologically active molecules that can be used for different tissue of interest. For PET fludeoxyglucose

(FDG) is commonly used as an active molecule, FDG marks certain types of tissues metabolism for example: heart, lungs, brain and tumours.

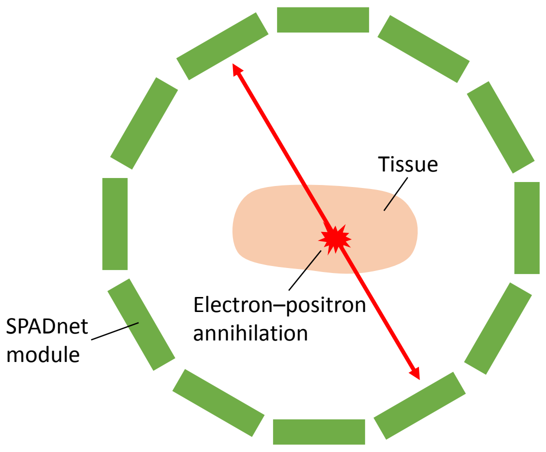


Figure 1.2: Annihilation of electron-positron pair, detected by the SPADnet modules

When the emitted positron of radionuclide collides with an electron, the two particles annihilate, the resulting energy is converted in to a pair of gamma rays. The gamma rays can be detected by the SPADnet modules. The pair of gamma rays are emitted back-to-back, using this property and time information of the detections, the origin of the annihilation can be determined. By recovering many events a three dimensional image can be reconstructed. The resolved image will only show the regions marked by the radionuclide. An example of an image recovered with PET of a human brain is shown in Figure 1.3.

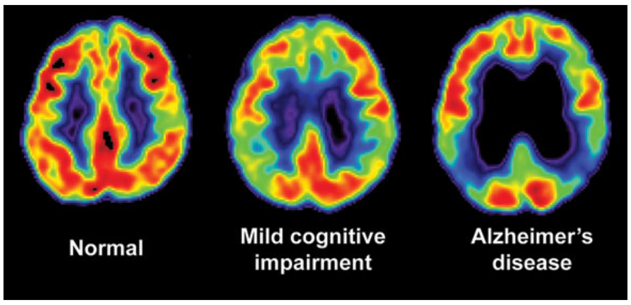


Figure 1.3: Example of functional neuroimaging, showing transaxial slice of the human brain recovered with PET. Blue regions indicate low accumulated tracer, red regions indicate show higher accumulated tracer. The tracer marks the active sections of the brain.

Clock synthesizers 2

The characteristics of the applied clock synthesizers are important factors for the eventual performance of the network based synchronization algorithm. The most important characteristic is the clock synthesizer stability. The stability will limit the maximum achievable synchronization resolution and the interval rate, at which the synchronization needs to be performed.

In this chapter the characteristics of quartz clock synthesizers are investigated. And a clock model is presented, that expresses the clock correction parameters.

2.1 Clock synthesizer imperfections

Ideally the output of a clock synthesizer would not change over time, and has the exact required frequency. Unfortunately, real clock synthesizers have phase output errors [12]. Clock synthesizers will not generate the exact specified frequency, and over time this frequency will drift. Besides drift, clock synthesizers suffer stationary noise on the frequency, know as jitter.

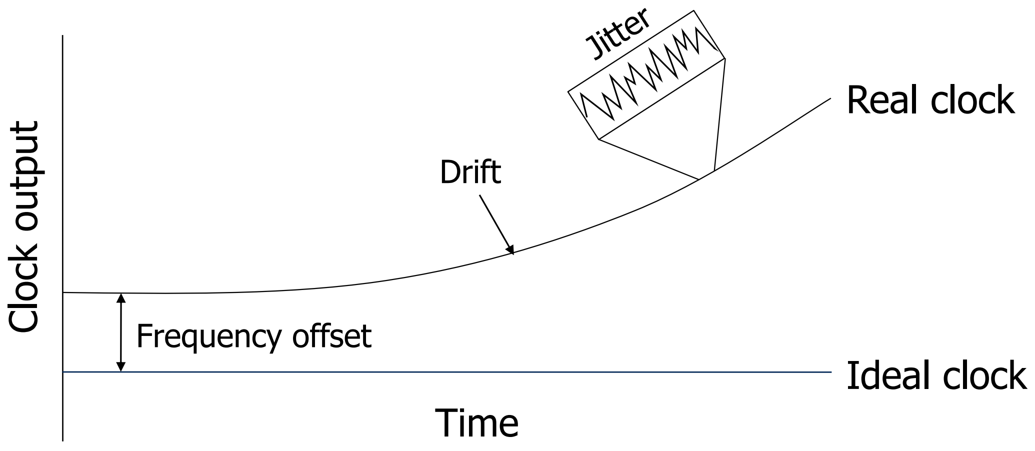


Figure 2.1: Comparison of ideal clock versus a real clock.

2.1.1 Quartz clock synthesizers

Quartz clock synthesizers are the most commonly used clock synthesizer, because of their low-cost and small size[13]. The SPADnet system also utilizes quartz oscillators.

Typically, a quartz clock synthesizer has an output frequency tolerance of 5-100 ppm. A quartz clock of 1 MHz with a frequency tolerance of 10 ppm, has an output frequency error of at most 10 Hz. The error in output frequency is the result of imperfections in the clock synthesizer.

The clock synthesizer frequency is also influenced by changes in environmental conditions. Clocks are affected by temperature, pressure, mechanical stress and humidity [14]. Typical quartz clock synthesizer frequency outputs change 20-100 ppm over 40 to +85 ℃. There exist solutions to compensate for temperature changes, two examples are: temperature compensated crystal oscillators (TCXO) and oven controlled crystal oscillators (OCXO). TCXO includes a temperature compensation circuit, and corrects to about 1 ppm error over the operating temperature range. OCXO oscillators are oscillators placed in temperature controlled chambers and achieve a temperature stability of 50-500 ppb over -55 to +85 ℃. The influences of the pressure, mechanical stress and humidity are controlled by packaging and the manufacturing process of the crystal oscillator.

Crystal oscillators undergo a slow gradual change of the frequency with time, known as ageing. This ageing process originates from small and slow changes of the physical aspects of the quartz crystal. Over time the ageing effect slowly decreases, for example the ageing effects in the first life year can be ± 3 ppm and the drift over 15 years life time a total ± 10 ppm.

Note that changes in the power supply, load, and other electrical interference will have an influence on the oscillator output.

2.1.2 Short-term clock stability

Besides the error introduced by ageing and environmental conditions on the output frequency, oscillators suffer from short-term variations of their signal, this is called jitter. Jitter comprises of two primary components: random jitter and deterministic jitter.

Random jitter is directly related to phase noise, the aggregate result of stochastic noise processes (thermal, shot and flicker noise). Random jitter typically shows a Gaussian distribution and can be quantified by its standard deviation (rms) value [15].

Deterministic jitter behaves predictably and repeatedly. The sources are generally related to imperfections in the behaviour of the device, transmission media, power supply noise and cross-talk. The range of the maximum phase deviations of this type of jitter can be bounded and expressed as a peak-to-peak value.

Not only the oscillator introduces jitter on the clock, but the clock interface also introduces phase noise. The analog clock signal is converted to digital by the clock interface. This clock interface does this by checking if the input signal is above or below a threshold value. Fluctuations in this threshold value, adds noise to the digital clock. The fluctuations are mainly induced by temperature and power supply instabilities. In this thesis it is assumed that this noise is random and has a zero mean.

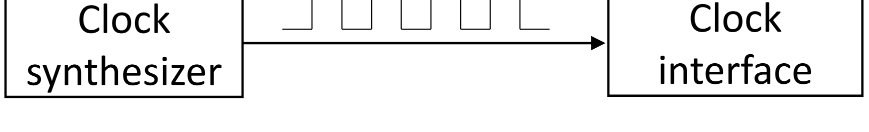


Figure 2.2: A complete clock system.

There are three commonly used methodologies for the measurement of different types of jitter, namely: period jitter, cycle to cycle jitter and time interval error measurement [16].

With period jitter, the deviation in period is measured with respect to the average period, with a random number of cycles between each measurement. From this measurement the RMS value is easily derived by taking the standard deviation of the measured data set.

Cycle to cycle jitter is the variation in cycle time of a signal between adjacent cycles. Cycle to cycle jitter is usually expressed as a peak value, defining the maximum deviation between the edges of two consecutive cycles.

Another methodology of jitter measurement is time interval error (TIE). TIE is a measure for the time deviation of a signal edge from its ideal position compared to a reference point. For this measurement an ideal signal is needed to compare the actual signal with. This signal is often created using software.

2.1.3 Long-term clock stability

Allan Variance [17] (AVAR), also called two-sample variance, is a technique to determine the long-term stability of a clock (or other types of signals, where stability is of interest) . Variance normally used in statistics is defined by the expected value of the squared deviation from the mean.

$$\mathrm{Var} = \sigma^2 = E\left[(X - E[X])^2\right]. \tag{2.1}$$

In contrast, AVAR calculates the variance based on the difference of adjacent fractional frequency measurements, calculating the two-sample variance. The AVAR method is shown in Equation 2.2.

$$\sigma_y^2(\tau) = E\left[(\Delta y)^2\right], \tag{2.2}$$

where Δy denotes the difference of adjacent fractional frequency measurements and τ the coherence time (space between measurements). So AVAR gives insight on the variation of two adjacent τ spaced samples. If we apply AVAR on a clock signal we can see how stable the clock is for a given coherence time. For clock synchronization this is of great interest, since it determines the maximum achievable performance of the clock synchronization solution.

In order to determine the interval time of the clock synchronization algorithm the long-term stability of the clock oscillators needs to be known. One way to infer the long-term drift is with the use of Allan variance. A Quartz crystal synthesizer has a frequency drift of $10^{-11} - 10^{-12}$ [17] per second. So the RMS error after 1 second is 1-10ps.

2.2 Clock synthesizer model

To be able to correct clocks a clock model is needed. This clock model will define the parameters that will be used to correct a clock. The short-term ($< 100s$) relation between local time and global time can be modelled by a first order clock model [10]:

$$t_i = w_i t + \phi_i, \tag{2.3}$$

where t_i is the local time at node i, t the reference clock, w_i the clock skew and ϕ_i the clock offset. The conversion from local time to global time can be written as:

$$t = \alpha_i t_i + \beta_i, \tag{2.4}$$

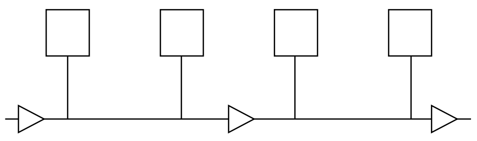
where $\alpha = w_i^{-1}$ and $\beta = w_i^{-1}\phi_i$ are the parameters needed to correct the local time of node i. These clock correction parameters will be used throughout the thesis.

3 Clock synchronization

This chapter first shortly covers the basic principles of hard-wired clock distribution networks. After that the basics of network based clock synchronization are addressed. Three novel network based clock synchronization algorithms will be covered. This chapter concludes with which of these three solutions can be best applied to the SPADnet system.

3.1 Hard-wired clock distribution

Typically, in synchronous digital systems the clock is distributed using a hard wired network. In these digital systems the clock originates from a single clock source and is distributed via a clock network to each component. There are different types of clock distribution networks, the simplest form is where the clock is daisy chained from each component to the next. A more sophisticated system, is the clock tree. In a clock tree the distance between the clock origin and each component is the same, and therefore the clocks are perfectly aligned at the leaves. In the daisy chained system on the contrary, the distance between the clock and component is not the same. Therefore there will be a clock offset at each node, which for some applications is not highly undesirable. The phase difference can be compromised by correcting the phases, with phase adjustment logic.


(a) Daisy chained clock network.

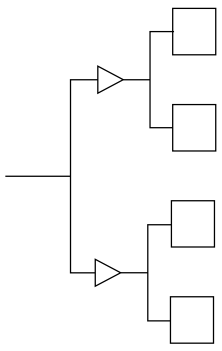

(b) Tree clock distributed network.

Figure 3.1: Example of typical hard wired clock distribution networks for digital systems.

3.2 Network based clock synchronization

In certain network based systems it is sometimes impossible to connect each node to a hard-wired clock distribution network. For example in wireless sensor networks

(WSNs). A WSN containing standalone modules, who are connected only via a wireless communication network, might need to fulfil synchronous tasks, and therefore need a way to synchronize each module in the system. Generally this is done by time-stamping of data packets.

3.2.1 Timestamp exchange protocols

To gain time information of each node data packets are used in network based synchronization. Time-stamping of data packets is the solution to get information about the clock drift and skew at each node. There are numerous timestamp exchange protocols.

Two-way timestamp exchanges

Most clock synchronization algorithms are based on two-way timestamp exchanges. Figure 3.2 shows the two-way timestamp exchange between a pair of nodes. Node 1 initiates the timestamp exchange, by sending a message to node 2. The message is marked with timestamp $T_{1,2}(1)$. Immediately when the message is received by node 2, the reception time is marked $R_{2,1}(1)$. Node 2 replies, again recording the transmission and reception times. This goes on until K two-way messages have been achieved.

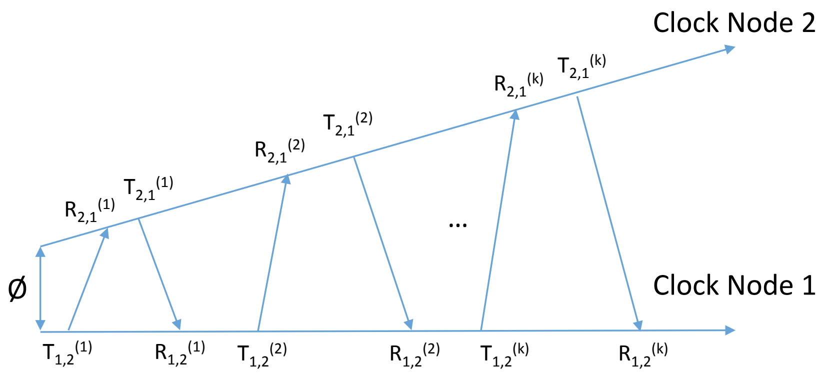


Figure 3.2: Conventional two-way timestamp exchanges between a pair of nodes.

After K two-way timestamp exchanges an algorithm can be applied to find the clock correction parameters [10].

Reference broadcast exchange

In the reference broadcast exchange protocol [18] one node transmits a broadcast message, marking the transmission time. The other nodes mark the reception times of this broadcast. After executing multiple broadcasts, this timing information can be used to estimate the clock drift and skew, assuming the distance is known.

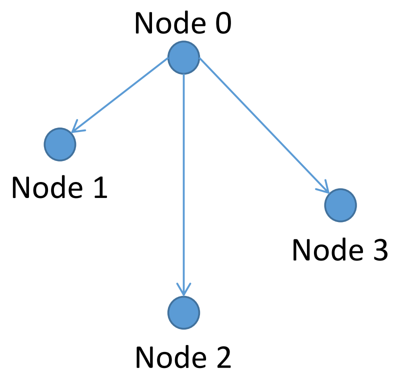


Figure 3.3: Reference broadcast protocol in a four node network. Node 0 transmits broadcast messages to the complete network, while capturing the transmission and reception times.

Passive listening protocol

The passive listening protocol is based on two-way timestamp exchanges, while the other nodes listen to this interaction of messages between the two nodes. The passive listening nodes also timestamp these overheard messages, which increases the amount of data collected which can contribute to a better estimation. A synchronization algorithm that uses passive listing is presented in [11].

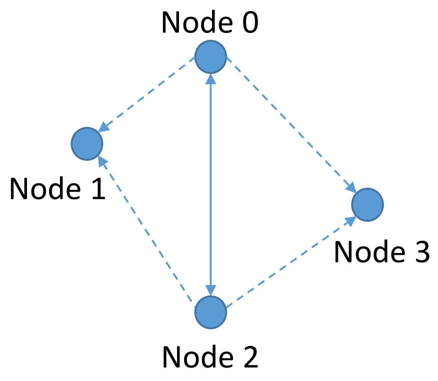


Figure 3.4: Passive listening protocol in a four node network.

3.3 Clock synchronization algorithms

There are various solutions to synchronize clocks in networks. In this section three novel clock synchronization solutions will be presented. The best solution for this application will be selected, based on the performance, complexity and scalability.

In [10] two solutions for clock synchronization for wireless networks are presented: pairwise least squares (PLS) and global least squares (GLS). The solutions efficiently estimate the first order clock parameters, clock skews (ω), clock offsets (ϕ) and pairwise distances ($\mathbf{d}$). Both solutions are based on timestamp exchanges via two-way messages and least squares. In PLS, pairs of nodes are synchronised per pair. GLS aims to find the global optimal solution for all unknown clock parameters and pairwise distances of the entire network.

The remaining novel clock synchronization algorithm is the asymmetrical time-stamping and passive listening (ATPL) protocol [11]. The ATPL solution uses passive listening and least squares in order to find all unknown clock parameters and pairwise distances of the entire network.

3.3.1 Least squares

All three novel clock synchronization algorithms use least squares to find the clock correction parameters. In general, least squares can be applied to problems where there are more equations than unknowns and, in general no solution exists [19]. Assume we have the set of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.1}$$

where $\mathbf{A}$ is a $n \times m$ matrix and $\mathbf{x}$ is an m-dimensional vector containing unknowns. In the case $m < n$, an arbitrary vector $\mathbf{b}$ cannot be represented as a linear combination of the columns of matrix $\mathbf{A}$. So the coefficients in vector $\mathbf{x}$ must be found, who will result in the best approximate of $\mathbf{b}$.

$$\hat{\mathbf{b}} = \sum_{i=1}^{m} x_i \mathbf{a}_i. \tag{3.2}$$

The least squares solution minimizes the norm of the error:

$$\parallel \mathbf{e} \parallel^2 = \parallel \mathbf{b} - \mathbf{A}\mathbf{x} \parallel. \tag{3.3}$$

This minimization problem has a unique solution, provided that the columns of the matrix $\mathbf{A}$ are linearly independent, given by solving the normal equations

$$\hat{\mathbf{x}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1} \mathbf{A}^T\mathbf{b}. \tag{3.4}$$

3.3.2 Pairwise least squares

The pairwise least squares (PLS) solution synchronizes a pair of nodes, one node being the reference. For networks with $N > 2$ nodes the PLS solution needs to be applied $N - 1$ times to synchronize all nodes. PLS can exploit only one link per pair. In Figure 3.5 a network of four nodes is shown, with node 0 as reference node. In this network the pairs where PLS is applied to are $\{Node0, Node1\}$, $\{Node0, Node2\}$ and $\{Node0, Node3\}$.

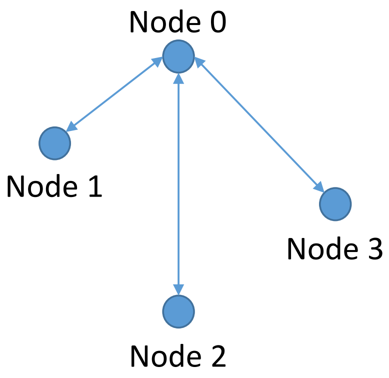


Figure 3.5: four node network for the PLS solution.

To synchronize a node pair, for example node 0 and node 1, first K two-way timestamp exchanges are executed, which results in time recordings at node 0 ($\mathbf{t}_{12}$) and node 1 ($\mathbf{t}_{21}$). Then, the time recordings of node 1 are sent to node 0. After that, node 0 can perform the PLS algorithm, Equation 3.5. Subsequently, node 0 sends the clock correction parameters $[\alpha_2 \; \beta_2]$ to node 1. Finally node 1 corrects its clock with the received parameters. The same holds for the other node pairs. More explicitly, we obtain

$$\hat{\boldsymbol{\theta}}_j = \left(\mathbf{A}_{ji}^T \mathbf{A}_{ji}\right)^{-1} \mathbf{A}_{ji}^T \mathbf{t}_{ij}, \tag{3.5}$$

with

$$\begin{aligned} \mathbf{A}_{ji} &= [\mathbf{t}_{ji} \; \mathbf{1}_{2K} \; \mathbf{e}]^T \in \mathbb{R}^{2K \times 3}, \\ \boldsymbol{\theta}_j &= [\alpha_j \; \beta_j \; \tau_{ij}]^T \in \mathbb{R}^{3 \times 1}, \end{aligned}$$

where $\mathbf{t}_{ij}$, $\mathbf{t}_{ji} \in \mathbb{R}^{2K \times 1}$ are the timestamps recorded at node i and j for all the two-way communications respectively, $\mathbf{e} = [-1, +1, -1, \cdots, +1]^T \in \mathbb{R}^{2Kx1}$, $\mathbf{1}_{2K} \in \mathbb{R}^{2K \times 1}$ is a vector of ones with length $2K$, and α_j, β_j are the correction parameters for the clock skew ω and the clock offset ϕ of node j.

3.3.3 Global least squares

Global least squares (GLS) is an extended version of the pairwise model. The GLS solution is capable of estimating all the clock correction parameters of all nodes in the network at once. GLS can make use of every link available in the network, as shown in Figure 3.6. GLS can be applied on any network configuration, as long as each node has atleast one link connected to the network. In GLS, on each link K timestamp exchanges are executed. When the message exchanges are completed, the timestamps are send to the reference node, and the clock correction parameters for each node are calculated, using Equation 3.6.

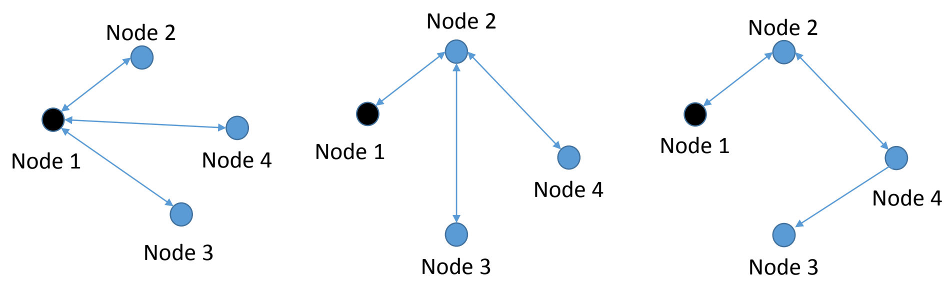


Figure 3.6: Example configurations of four node networks, where the GLS algorithm can be applied to

The global least squares solution is obtained by minimizing the least squares norm

$$\hat{\boldsymbol{\theta}} = \left(\bar{\mathbf{A}}^T \bar{\mathbf{A}}\right)^{-1} \bar{\mathbf{A}}^T \bar{\mathbf{t}}_1, \tag{3.6}$$

with

$$\begin{aligned} \bar{\mathbf{A}} &= \left[\bar{\mathbf{T}}\ \bar{\mathbf{E}}_1\ \mathbf{E}_2\right] \in \mathbb{R}^{2KM \times L}, \\ \boldsymbol{\theta} &= \left[\bar{\boldsymbol{\alpha}}\ \bar{\boldsymbol{\beta}}\ \boldsymbol{\tau}\right] \in \mathbb{R}^{L \times 1}, \end{aligned}$$

where $L = 2N + M - 2$ and $\bar{\mathbf{T}}, \bar{\mathbf{E}}_1 \in \mathbb{R}^{2KM \times (N-1)}$ are submatrices of $\mathbf{T}$ and $\mathbf{E}_1$ respectively, excluding the corresponding first columns. $\bar{\boldsymbol{\alpha}}$,$\bar{\boldsymbol{\beta}} \in \mathbb{R}^{(N-1)\times 1}$ represent the unknown clock parameters of all the nodes excluding node 1 (reference node). $\bar{\mathbf{t}}_1 \in \mathbb{R}^{2KM \times 1}$ is the first column of matrix $\mathbf{T}$ which contains the timing markers recorded at node 1.

The matrix $\mathbf{T} \in \mathbb{R}^{2KM \times N}$ contains all the timing vectors from all N nodes, $M = \binom{N}{2}$ number of unique pairwise distances, $\mathbf{E}_1 \in \mathbb{R}^{2KM \times N}$, $\mathbf{E}_2 - \mathbf{I}_M \otimes \mathbf{e} \in \mathbb{R}^{2KM \times N}$. For $N = 4$ $\mathbf{T}$ and $\mathbf{E}_1$ are of the form

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_{12} & -\mathbf{t}_{21} & & \\ \mathbf{t}_{13} & & -\mathbf{t}_{31} & \\ \mathbf{t}_{14} & & & -\mathbf{t}_{41} \\ & \mathbf{t}_{23} & -\mathbf{t}_{32} & \\ & \mathbf{t}_{24} & & -\mathbf{t}_{42} \\ & & \mathbf{t}_{34} & -\mathbf{t}_{43} \end{bmatrix}, \tag{3.7}$$

$$\mathbf{E}_1 = \begin{bmatrix} +\mathbf{1}_{2K} & -\mathbf{1}_{2K} & & \\ +\mathbf{1}_{2K} & & -\mathbf{1}_{2K} & \\ +\mathbf{1}_{2K} & & & -\mathbf{1}_{2K} \\ & +\mathbf{1}_{2K} & -\mathbf{1}_{2K} & \\ & +\mathbf{1}_{2K} & & -\mathbf{1}_{2K} \\ & & +\mathbf{1}_{2K} & -\mathbf{1}_{2K} \end{bmatrix}. \tag{3.8}$$

3.3.4 Asymmetrical time-stamping and passive listening

Asymmetrical time-stamping and passive listening (ATPL) [11] exploits the broadcast property to estimate the clock parameters. During communication between a pair of nodes timestamps are recorded and exchanged, like in the PLS and GLS solution. Besides this, the remaining nodes in the network passively listen and record the timestamps with their individual clocks. In this way more information is obtained to solve the unknown clock parameters. Figure 3.7, shows how the other nodes listen, while a two-way timestamp exchange is done between a pair of nodes.

In the ATPL there are no constraints on the sequence of forward links and reverse links, hence the reverse link does not always need to follow the forward link. This means that a node does not need to respond to the request from another node immediately. In the ATPL protocol the two-way communication always occurs between node i and the central node, where node i initiates the two way communication. The other nodes in the network passively listen and timestamp the messages.

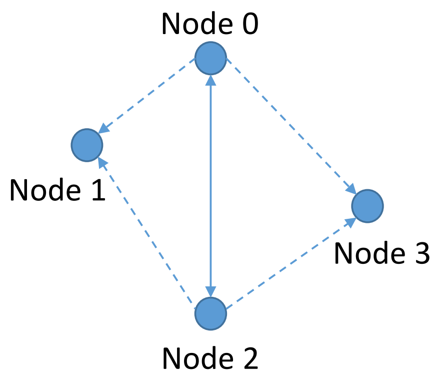


Figure 3.7: four node network for ATPL solution. The dotted arrows indicate the nodes passive listening to other nodes two-way timestamp exchange.

The clock parameters can be estimated using least squares as

$$\hat{\boldsymbol{\theta}} = \left(\bar{\mathbf{A}}^T\boldsymbol{\Sigma}^{-1}\bar{\mathbf{A}}\right)^{-1}\bar{\mathbf{A}}^T\boldsymbol{\Sigma}^{-1}\bar{\mathbf{x}}, \tag{3.9}$$

The construction of the matrices are presented in Figure 3.8; more information about this can be found in [11].

$$\begin{array}{cc}
\text{Node 0 responds} & j{=}1\\
(i{=}0) & j{=}2\\ \hline
\text{Node 1 transmits} & j{=}0\\
(i{=}1) & j{=}2\\ \hline
\text{Node 2 transmits} & j{=}0\\
(i{=}2) & j{=}1
\end{array}
\overbrace{\left[\begin{array}{cc|cc|cc}
\mathbf{t}_0 & \mathbf{1}_{K_0} & -\mathbf{r}_{0,1} & -\mathbf{1}_{K_0} & \mathbf{1}_{K_0} & \mathbf{0}_{K_0}\\
\mathbf{t}_0 & \mathbf{1}_{K_0} & \mathbf{0}_{K_0} & \mathbf{0}_{K_0} & \mathbf{0}_{K_0} & \mathbf{1}_{K_0}\\ \hline
-\mathbf{r}_{1,0} & -\mathbf{1}_{K_1} & \mathbf{t}_1 & \mathbf{1}_{K_1} & \mathbf{1}_{K_1} & \mathbf{0}_{K_1}\\
\mathbf{0}_{K_1} & \mathbf{0}_{K_1} & \mathbf{t}_1 & \mathbf{1}_{K_1} & \mathbf{0}_{K_1} & \mathbf{0}_{K_1}\\ \hline
-\mathbf{r}_{2,0} & -\mathbf{1}_{K_2} & \mathbf{0}_{K_2} & \mathbf{0}_{K_2} & \mathbf{0}_{K_2} & \mathbf{1}_{K_2}\\
\mathbf{0}_{K_2} & \mathbf{0}_{K_2} & -\mathbf{r}_{2,1} & -\mathbf{1}_{K_2} & \mathbf{0}_{K_2} & \mathbf{0}_{K_2}
\end{array}\right]}^{\bar{\mathbf{A}}\in\mathbb{R}^{2K\times 6}}
\overbrace{\begin{bmatrix}\mathbf{c}_0\\ \mathbf{c}_1\\ \boldsymbol{\tau}_0\end{bmatrix}}^{\boldsymbol{\theta}\in\mathbb{R}^{6\times 1}}
=
\overbrace{\left[\begin{array}{cc|c}
\mathbf{0}_{K_0} & \mathbf{0}_{K_0} & \mathbf{0}_{K_0}\\
-\mathbf{r}_{0,2} & -\mathbf{1}_{K_0} & \mathbf{0}_{K_0}\\ \hline
\mathbf{0}_{K_1} & \mathbf{0}_{K_1} & \mathbf{0}_{K_1}\\
-\mathbf{r}_{1,2} & -\mathbf{1}_{K_1} & \mathbf{1}_{K_1}\\ \hline
\mathbf{t}_2 & \mathbf{1}_{K_2} & \mathbf{0}_{K_2}\\
\mathbf{t}_2 & \mathbf{1}_{K_2} & \mathbf{1}_{K_2}
\end{array}\right]
\begin{bmatrix}\mathbf{c}_2\\ \tau_{1,2}\end{bmatrix}}^{\mathbf{x}\in\mathbb{R}^{2K\times 1}}
+ \mathbf{n}$$

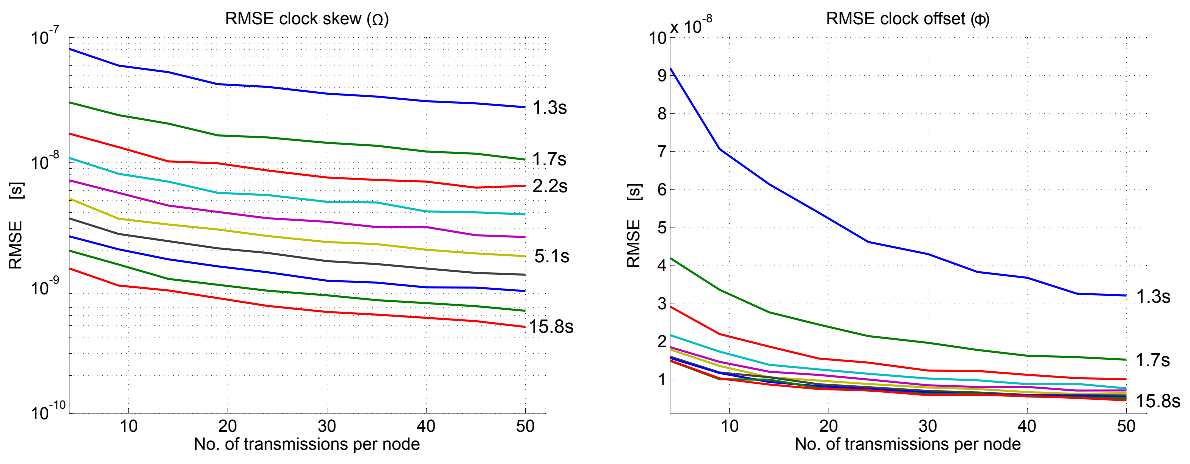
Figure 3.8: Construction of the matrices for the ATPL solution from [11]

3.3.5 Performance

The performance of the presented algorithms is bounded by the stability of the local clocks. Ideally, the two-way timestamp exchanges would be performed in the region where the clocks are most stable, to estimate the clock correction parameters. As shown in Figure 6.2 on page 34 the window where the clock is most stable is 2-16 seconds with a stability of 22 ns. No algorithms can compensate for this random noise on the clock output.

The performance of PLS, GLS and ATPL is the same, they all follow the Cramer-Rao lower bound, shown to approach the theoretical limits. Because GLS, and especially ATPL, can exploit more communication links they can be more efficient, and reach the lower bound faster than PLS.

The number of timestamp exchanges, K, has a strong influence on the performance. The more timestamp exchanges, the better the estimation. Not only the amount of message exchanges, but also the length of the observation window wherein the exchanges take place, effect performance. The estimation will be better for longer observation windows. Figure 3.9 show how the estimation is affected by the number of timestamp exchanges and length of the observation window.



(a) Estimation of the clock skew.   (b) Estimation of the clock offset.

Figure 3.9: Estimation of the clock parameters using PLS in a two node system, for different transmissions and observation windows. The algorithm was performed with MATLAB, with an observation noise of 22 ns, and infinite clock resolution.

3.3.6 Cramer-Rao lower bound

The algorithms follow the Cramer-Rao lower bound (CRLB), which defines the smallest variance of any unbiased estimator. If an estimator is found that achieves the CRLB, we call this a minimum variance unbiased estimator (MVUE).

In the case of PLS, GLS and ATPL the CRLB is most influenced by the noise on the time-stamping of the messages, this noise is called observation noise. This noise originates from the instability of the clock, used for time-stamping of the two-way message exchanges, and communication latency uncertainty of the network.

With simulations, using infinite clock resolution and averaging, it is possible to reach below the observation noise. In a practical situation, the random clock noise, cannot be compensated for by the algorithm, and therefore the estimation will be bound by the observation noise. The algorithm only estimates the first order clock parameters, which does not include any compensation for the clock noise.

3.3.7 Complexity

Table 3.1 shows the sizes of the operations needed to compute the least squares solution, as presented in Equation 3.4. To solve the least squares problem, one matrix multiplication, one matrix inversion and two matrix vector multiplications are necessary.

The most complex operation is the matrix inversion. For PLS a 3x3 matrix needs to be inverted, whereas for ATPL a 3Nx3N matrix needs to be inverted. Clearly we see that PLS has the lowest complexity, GLS having a moderate complexity and ATPL the highest complexity.

Table 3.1: Operation sizes of each least squares operation. With K number of messages, $K` = K_1 + K_2 + K_N$, $L = 2N + M - 2$, M the number of unique links in the network and with N nodes.

	$C = AA$	$d = Ab$	$E = C^{-1}$	Ed
PLS	(3x2K)(2Kx3)	(3x2K)(2Kx1)	3x3	(3x3)(3x1)
GLS	(Lx2KM)(2KMxL)	(Lx2KM)(2KMx1)	LxL	(LxL)(Lx1)
ATPL	(3NxK‘N)(K‘Nx3N)	(3NxK‘N)(K‘Nx1)	3Nx3N	(3Nx3N)(3Nx1)

3.3.8 Conclusion

For SPADnet PLS will be applied for clock synchronization for a number of reasons. First, PLS estimates the clock parameters per pair, which is easier to scale, since whenever an extra node is added, only an extra synchronization of a pair of nodes has to be executed. Also, PLS has the lowest complexity and will therefore be simpler to implement in hardware. PLS has the same eventual performance as GLS and ATPL. First, it will be shown that PLS can be implemented for SPADnet, in the future techniques like GLS or ATPL might be applied for a more efficient solution.

SPADnet system 4

The SPADnet system is composed of photonic modules. Each module, shown in Figure 4.1, contains scintillator crystals, sensor tiles and a data processing communication unit (DCPU). In this chapter each component will be explained.

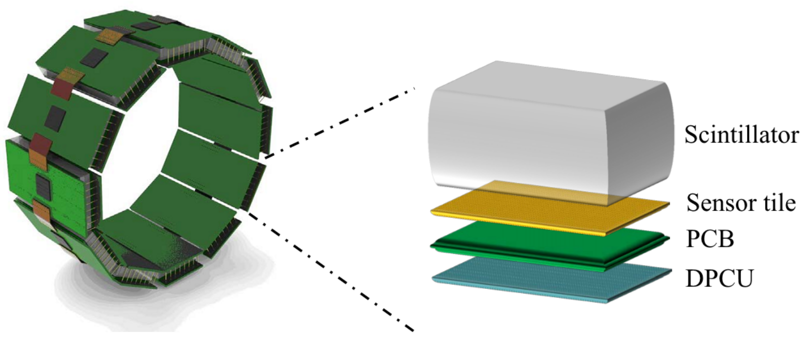


Figure 4.1: SPADnet ring system and photonic module.

4.1 Sensor tile

When the scintillator crystal is struck by a gamma photon, generated by the positron-electron annihilation, the energy is absorbed and a shower of visible photons is generated. The sensor tile detects this light emission in the scintillator crystal using digital silicon photo-multipliers (SiPMs). A digital SiPMs is an array of single-photon avalanche diodes (SPADs) whose individual outputs are combined digitally.

4.2 Data processing communication unit

As discussed earlier, every module in the SPADnet systems has its own data processing communication unit. The DPCU timestamps the gamma events received from the sensor tile. Then, using the timing information, a coincidence is identified by searching the packets in the network. These packets are known as coincidence packets. When a pair is identified, the pair information is send off the network to a remote computer. This computer uses the event pairs information to recreate a 3D image (tomography).

Because the pairs are created by means of their timing information, it is of most importance that the time-stamping at all nodes is synchronized, else wrong pairs of events could be combined, resulting in a false recovered 3D image.

A schematic of the DPCU is shown in Figure4.2. The core of the DPCU is the XC6SLX150T Spartan-6 FPGA from Xilinx. The connectors J3, J4, J5 and J6 are the connections to the SPADnet network. There are three available sources for the clock of the DPCU system: clock distributed via the network (J3), a local quartz oscillator or the hard-wired through coaxial cables (connectors J8 and J9). All clock sources, expect the local quartz oscillator, are cleaned by the LMK04906 clock jitter cleaner/multiplier.

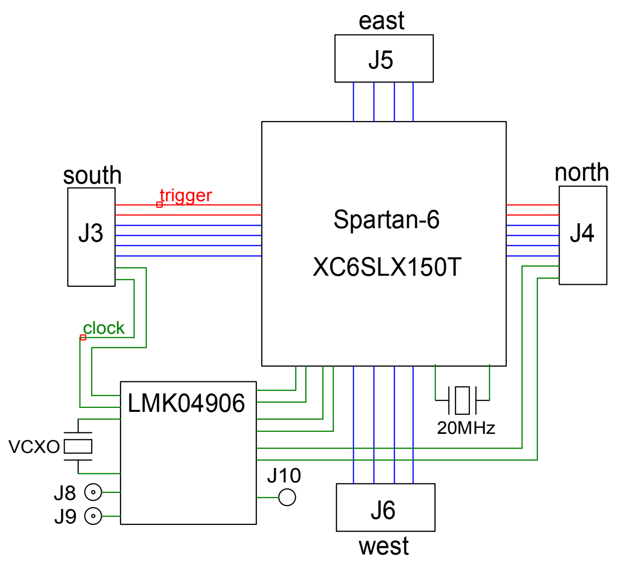


Figure 4.2: Schematic of the DPCU, the connector to the sensor tile is located at the backside of the DPCU (not shown in this figure).

4.3 Network architecture

The main purpose of the SPADnet network is to distribute the coincidence packets and transfer data out off the network. The SPADnet network is constructed out of rings of photonic modules, as can be seen in Figure 4.3.

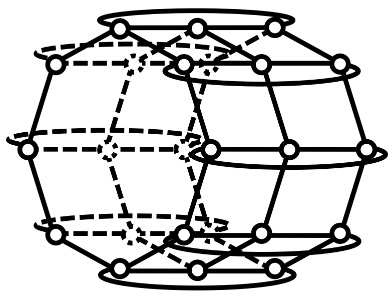


Figure 4.3: SPADnet network hierarchy. Each circle represents a photonic module

For distributed coincidence detection to occur, all network nodes are required to perform coincidence detection simultaneously, by comparing locally detected gamma events with those detected in other nodes, and across rings. This operation needs to handle up to 0.5 million gamma events per second [20].

To reduce the complexity of the search of pairs that needs to be performed simultaneously in every node, the search space is reduced by optimizing the network at system level, thereby achieving lower packet latency.

At system level, a two-stage data transfer approach was implemented, whereas in the first stage smaller data packets (32 bit) carrying gamma event timestamps, known as coincidence packets, are circulated in a dedicated network channel called coincidence network. In the second stage, coincidence packets are paired to form true-event packets and then transferred to the data-acquisition PC, using another network channel called data transfer network. These 64 bit packets contain node and scintillation energy/timing/position information, enabling to recover the positron-electron annihilation location.

The use of two dedicated network channels, along with the smaller data packet for coincidence detection, results in packet latency reduction. Furthermore, to lower the statistical variance on packet latency, a static packet routing technique was used along with the oldest-packet-first flow control algorithm. This approach to flow control was realized using a virtual channel in hardware. The architecture of these network channels are shown in Figure 4.4.

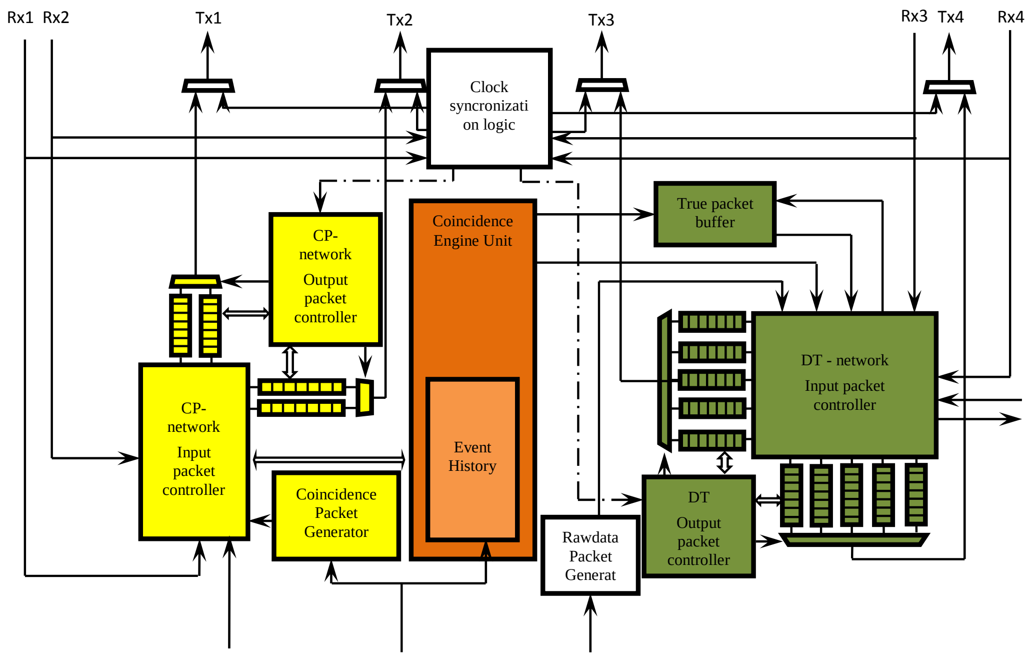

Rx1
Rx2
Tx1
Tx2
Tx3
Rx3
Tx4
Rx4
Clock syncronizati on logic
True packet buffer
Coincidence Engine Unit
CP- network Output packet controller
CP- network Input packet controller
Event History
Coincidence Packet Generator
Rawdata Packet Generat
DT - network Input packet controller
DT Output packet controller

Figure 4.4: Coincidence packet and data transfer architecture.

In the network architecture there is space reserved for the network based clock synchronization solution. The clock synchronization logic has the highest priority on the network. Whenever the clock synchronization logic requires to send a packet on the network, the coincidence and true data channels are disabled to send any data. The packets received by the module are directly available for the clock synchronization logic.

4.4 SPADnet system clock synthesizers

The modules of the SPADnet system run on a central clock source. The clock signal from a single quartz oscillator is spread with a 1-to-22 low skew buffer (ICS8534-01) to each module. The clocks are either distributed in a star hierarchy or daisy chained through each module. Each module has a local quartz oscillator, that is used during the power up sequence of the module. The specifications of these oscillators are shown in Table 4.1

Table 4.1: Specifications SPADnet system clock synthesizers.

	Frequency (MHz)	**Frequency stability (ppm)**	**Type**
Central clock oscillator	100	± 20	Quartz
Local clock oscillator	20	± 50	Quartz

All of the clocks, except for the initialization clock, are buffered by a LMK04906 clock cleaner.

4.4.1 LMK04906 Ultra low noise clock jitter cleaner

Every DCPU module is equipped with a LMK04906 [21]. This device cleans the noise from the hard-wired clock distribution networks. And it is also used to create different kinds of clock signals. The buffer generates five clock signals, as shown in Figure 4.5. CLKOUT0: clock for the next node in the ring system. CLKOUT1: easy accessible clock signal used for diagnostics. CLKOUT2: clock used for coincidence mode. CLKOUT3: user clock for the FPGA. CLKOUT5: clock signal that drives the embedded network transceivers, used for the communication network. The MICROWIRE bus is used to program the LMK04906. Either the buffer uses the clock from the ring or star configuration, for generation of the five clock outputs. All clock outputs have a frequency of 100 MHz, except for the 125 MHz Aurora clock (CLKOUT5).

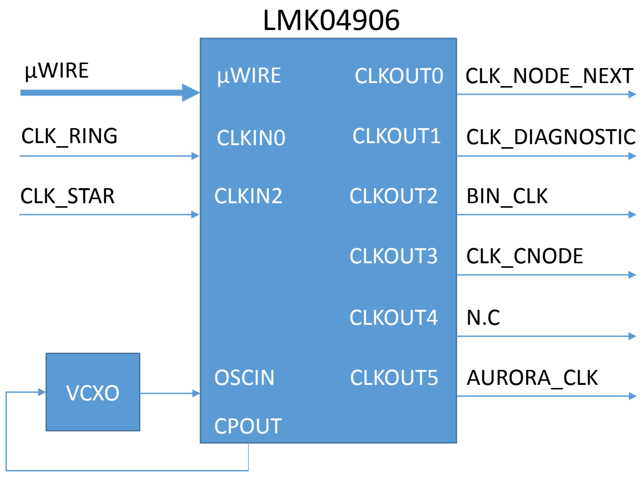


Figure 4.5: The LMK04906 and its clock outputs.

In the system the LMK04906 is used in 0-delay mode, in this mode the feedback of the first PLL is driven by a clock output. This loop-back ensures a deterministic phase between the clock in- and outputs. The functional block diagram of the LMK04906 in 0-delay mode is shown in Figure 4.6. The first phase locked loop (PLL) cleans the jitter with an external VCXO by using a narrow loop bandwidth. PLL1 serves as a reference for PLL2, this internal VCO drives the six divide/delay blocks which generate the six clock outputs.

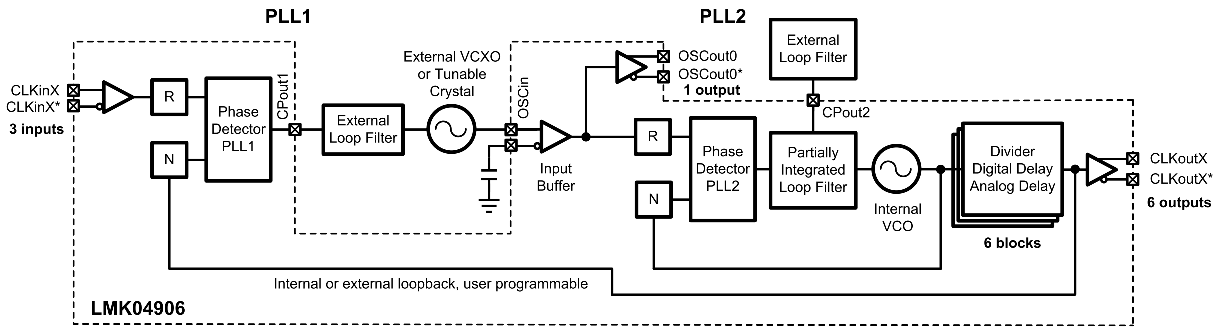


Figure 4.6: Simplified functional block diagram of the LMK04906 in 0-delay mode. Obtained from [21]

The, programmable, divider blocks scale down the output to a desired frequency. Delay blocks allow to adjust the output phase. There are two mechanisms to delay the output. A fine analog delay with a resolution of 25 ps and a range of 500 - 975 ps. The digital delay resolution depends on the VCO divider and VCO frequency. In this case the resolution is 400 ps as calculated in:

$$Digital\ Delay\ Resolution = \frac{VCO\_DIV}{2\,x\,VCO\_frequency} = \frac{2}{2\ x\ 2500\ MHz} = 400\ ps. \quad (4.1)$$

The LMK04906 is connected with the FPGA via a MICROWIRE Serial Interface. Via this interface the settings of the LMK04906 are programmed, including the analog and digital delay. During the start-up sequence of the FPGA the initial settings are loaded into the LMK04906.

4.4.2 ICS8534-01: 1-to-22 fanout buffer

The clock distribution network is driven by the ICS8534-01 fanout buffer. This buffer takes in a single clock signal and generates 22 clock outputs. This buffer adds a phase jitter of 0.04 ps and the skew between the outputs is 100 ps maximum [22]. In the SPADnet system 20 of the outputs are used for the star clock distribution. The other two clock outputs are used for the ring configuration. The printed circuit board, provided during this thesis, can support up to two rings of SPADnet modules.

5 Hard-wired clock distribution networks

5.1 Distribution topography

The SPADnet system has two hard-wired clock distribution options, a ring and star topography. In the ring configuration the clocks are daisy chained from each module to the next module. In the star configuration the clocks are wired from a single point to every module. In this section these hard-wired clock distribution networks are analysed and characterized.

5.1.1 Ring clock distribution network

In Figure 5.1 the schematics for a three node network, with a ring distributed clock, is shown. In this configuration the clock follows the same path as the communication network, and is physically placed on the same flat-cable. Due to this configuration the clock signal will not arrive at the same time at each node, because the distance to the clock origin (snooper) increases for every following node. Hence there will be a phase difference at each node. This phase difference can be compensated for by the LMK04906, this circuit is capable of delaying the clock outputs by programmable values.

If the user would like to extend the network with one or more nodes, the phase difference needs be measured for each extra node and adjusted by programming the LMK04906. This is not convenient since the user needs to have the appropriate measuring equipment. Measuring and aligning the phases can be time consuming, when multiple nodes are added to the network, and especially when the system is set up for the first time.

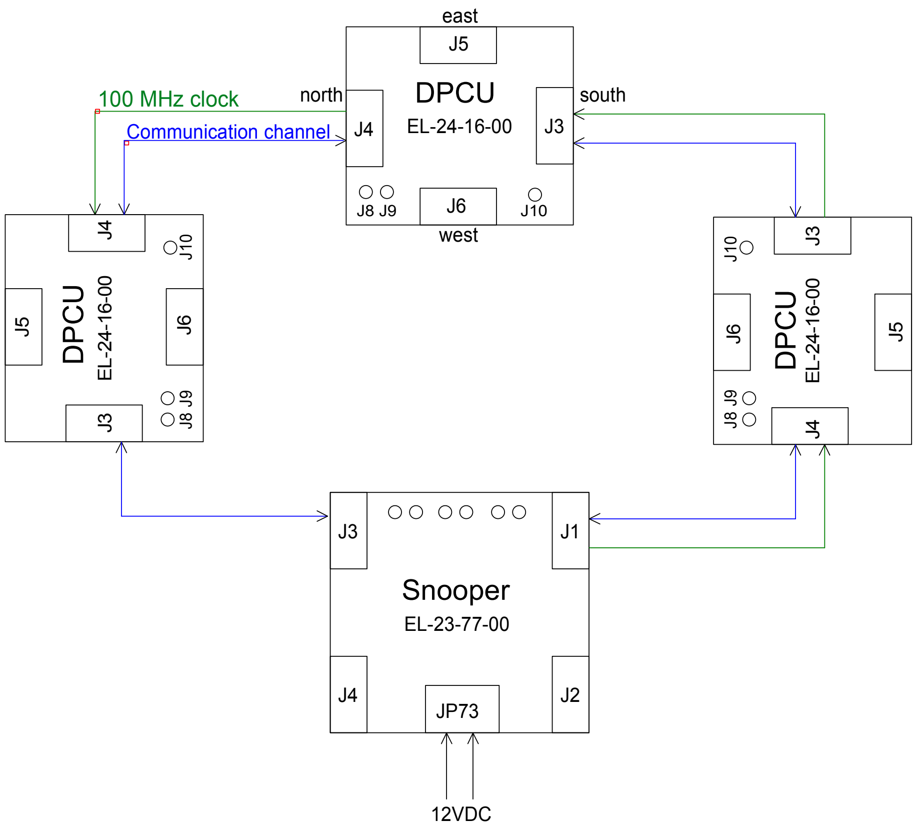


Figure 5.1: Clock distribution network: ring topography.

5.1.2 Star clock distribution network

The star clock distribution network is shown in Figure 5.2. In this configuration the clocks are connected from the snooper to each node using coaxial cables. If all the cables would have exactly the same length there would be zero phase offset in the clock signal at each node.

When the network is extended when using the star clock distribution network, the user needs to add the coaxial wires to distribute the clock. The number of connectors on the clock distribution module limits the maximum number of nodes in the network. The coaxial cables should have the exact same length, else a phase error will be introduced. The clocks are generated using the ICS8534-01 1-to-22 fanout buffer, as discussed in Section 4.4.2. This buffer introduces a maximum clock offset of 100 ps on its outputs [22].

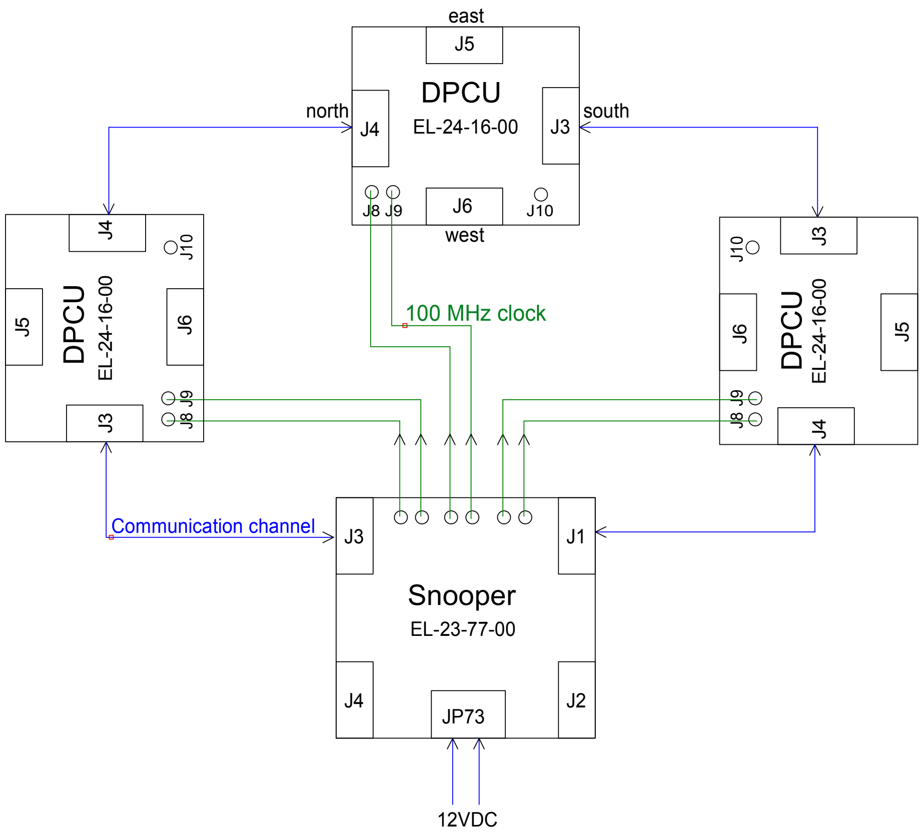


Figure 5.2: Clock distribution network: star topography.

5.2 Characterization

The performance of SPADnet partially depends on how good the module clocks are synchronized. Jitter and phase measurements were done to give insight on the performance of the hard-wired clock distribution networks. The jitter and phase measurements were measured after the jitter cleaner (LMK04906). The measurement point was the miniature RF connector $J10$ shown in Figure 5.3.

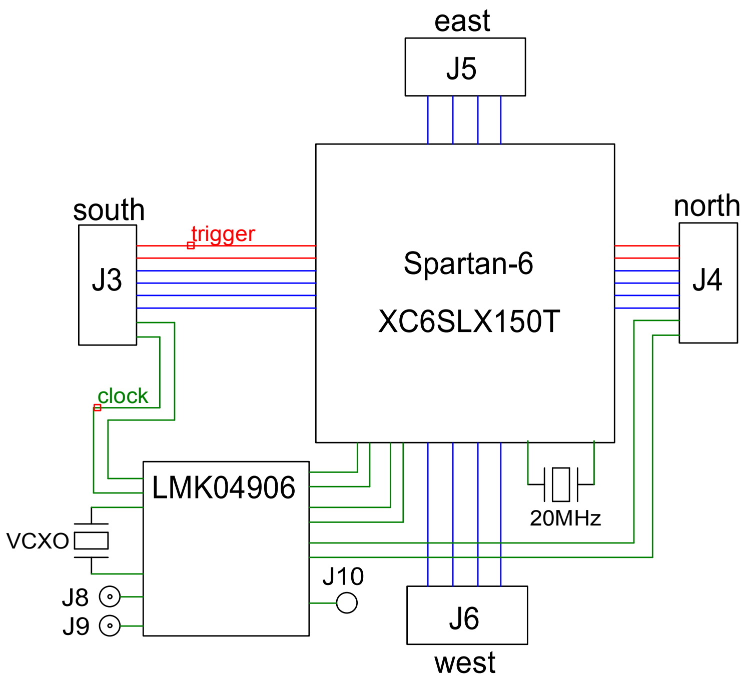


Figure 5.3: Simplified schematic of the SPADnet FPGA modules.

5.2.1 Jitter

For the jitter measurements a LeCroy WavePro 760Zi-A oscilloscope was used. This oscilloscope has a sampling rate of 40 GS/s and sampling jitter of 100 fs_{rms} for an acquired time range up to 10 μs.

With the oscilloscope the average period of the signal is measured, with a random number of cycles between each measurement. Then the deviation per sample is calculated. This deviation was accumulated in to a histogram of 10.000 samples. This histogram is shown in Figure 5.4. This measurement follows the JEDEC Standard 65B [16].

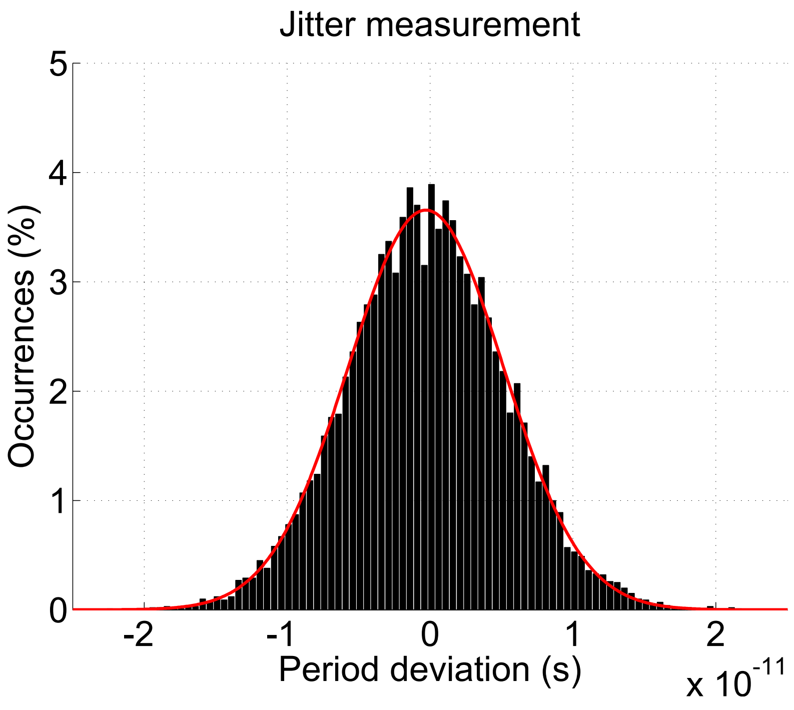


Figure 5.4: Normalized histogram of period jitter analysis, shows the period deviation of the clock signal over a short period of time. The histogram contains 10.000 samples and 100 bins.

The RMS value (standard deviation) of the jitter was calculated to be 5.52 ps. The data fits a Gaussian distribution and has a zero mean. There is no difference between the jitter with the star and the ring clock distribution network.

5.2.2 Multi node jitter

The presented jitter measurement was only for a single node system. In this section the jitter measurement is repeated in a system of four nodes. For the estimation, it is of great importance that the jitter does not increase significantly when the number of nodes in the system increases. The same measurement setup was used as in the previous case, only a different oscilloscope was used. This oscilloscope has slightly lower bandwidth (20 GS/s). The results are shown in Table 5.1, the jitter on the clock does not increase when the number of nodes increases.

Table 5.1: Jitter measurement in a four node ring system.

Node	Jitter (ps)
1	5
2	5
3	5
4	5

5.2.3 Phase difference star distributed clock network

The phase difference between two nodes was measured using the same oscilloscope used for the jitter measurement. The phase difference of two nearest clock edges (from the clock signals at each node) where accumulated in to a histogram of 10.000 samples and 100 bins.

The result of the phase difference measurement, of the star distributed clock network is shown in Figure 5.5. This small phase difference is introduced by the coaxial cables, that do not have exactly the same length. The center of the histogram lies at 191 ps, with a standard deviation of about 10 ps. This is about 2 times the standard deviation measured at a single node, as shown in Figure 5.4.

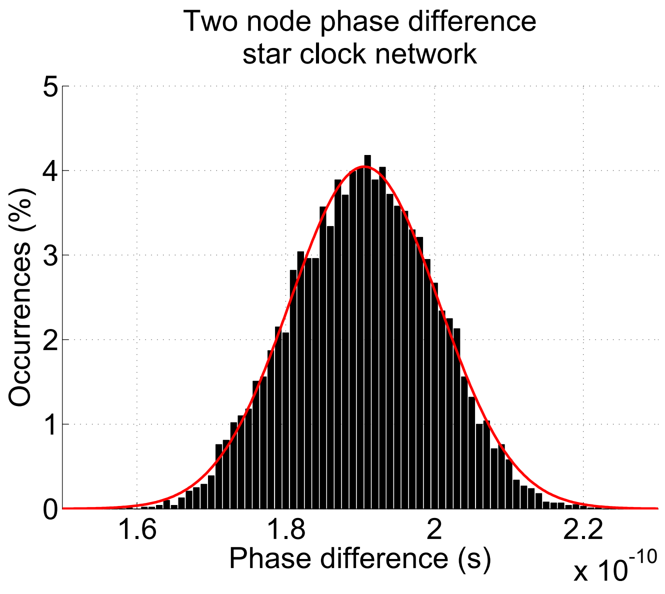


Figure 5.5: Phase difference between two adjacent nodes, using the star distributed clock network.

5.2.4 Phase difference ring distributed clock network

For the ring distribution network the same measurement was performed. The histogram is shown in Figure 5.6. The center of the histogram lies at 4.47 ns, with a standard deviation of about 11 ps.

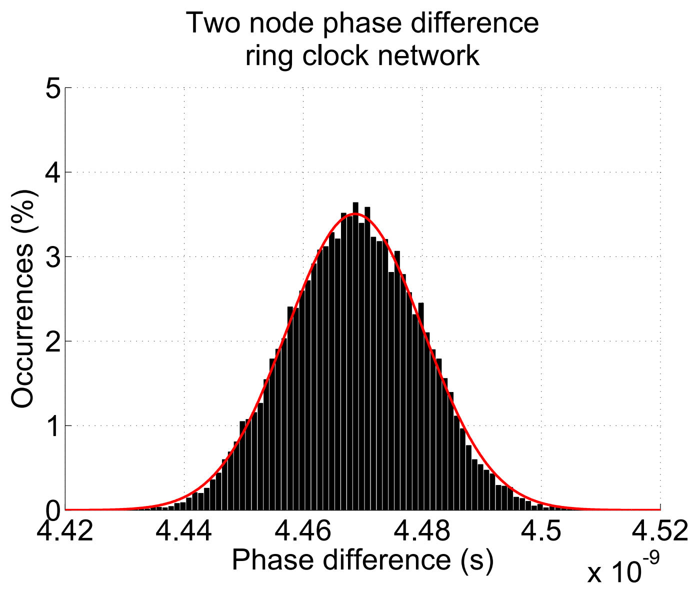


Figure 5.6: Phase difference between two adjacent nodes, using the ring distributed clock network.

Multi nodes ring

Table 5.2 shows the result of phase difference measurements in a system of four nodes. The noise on the phase slightly increases when the number of nodes increases.

Table 5.2: Phase difference in a four node system.

Node	Mean (ns)	SD (ps)
1-2	5.529	9
1-3	1.225	11
1-4	6.9548	14

To compensate for phase differences the LMK04906 can be used. It is capable of correcting the clock phase, with a resolution of 25 ps.

5.3 Conclusion

The network clock has a jitter of less than 6 ps, this jitter does not increase when the number of nodes increases.

5.3.1 Ring network

The phase jitter between two nodes, in the ring configuration is about 11 ps measured on two adjacent nodes, the phase jitter slightly increases (2-3 ps per node) when the system is scaled up. At each node there exists phase offset, that is depended on the modules position in the network. The phase offset difference can be compensated by the LMK04906, with 25 ps precision.

Adding an extra node to the system, while using the ring clock network, the phase difference of the added node needs to be manually measured and corrected.

5.3.2 Star network

The ring configuration has a phase jitter of about 10 ps between two nodes. The clock offset between two nodes is 191 ps, this phase difference is introduced by the output skew of the ICS8534-01 1-to-22 fanout buffer (max 100 ps) and the applied coaxial cables.

Whenever the system is extended with an extra node, a pair of coaxial cables needs to be connected between the added node and the clock distribution board. The amount of nodes is limited by the number of connections on this clock distribution boar. The coaxial cables can, and the ICS8534-01 1-to-22 fanout buffer will introduces a phase offset.

SPADnet solution 6

As discussed in Section 3.3.8, pairwise least squares (PLS) is considered to be the best solution for SPADnet for now. Also it was stated that the performance of the algorithms depends on the stability of the clock oscillators. In this chapter first the Allan variance will be measured, to see how stable the clock oscillators are. The chapter ends with a discussion of how the algorithm is mapped to the SPADnet network.

6.1 Allan variance measurement

As discussed in Section 2.1.3 the long-term stability of a signal can be measured using the Allan variance (AVAR). The AVAR was measured using the setup shown in 6.1. A 64 bit counter runs on the local clock oscillator. At fixed intervals an external clock generator triggers, this causes the system to store the current value of the counter in memory. After enough triggers, the counter values are sent to the PC via the serial network. This will be done for interval times from 0.01 s till 64 s.

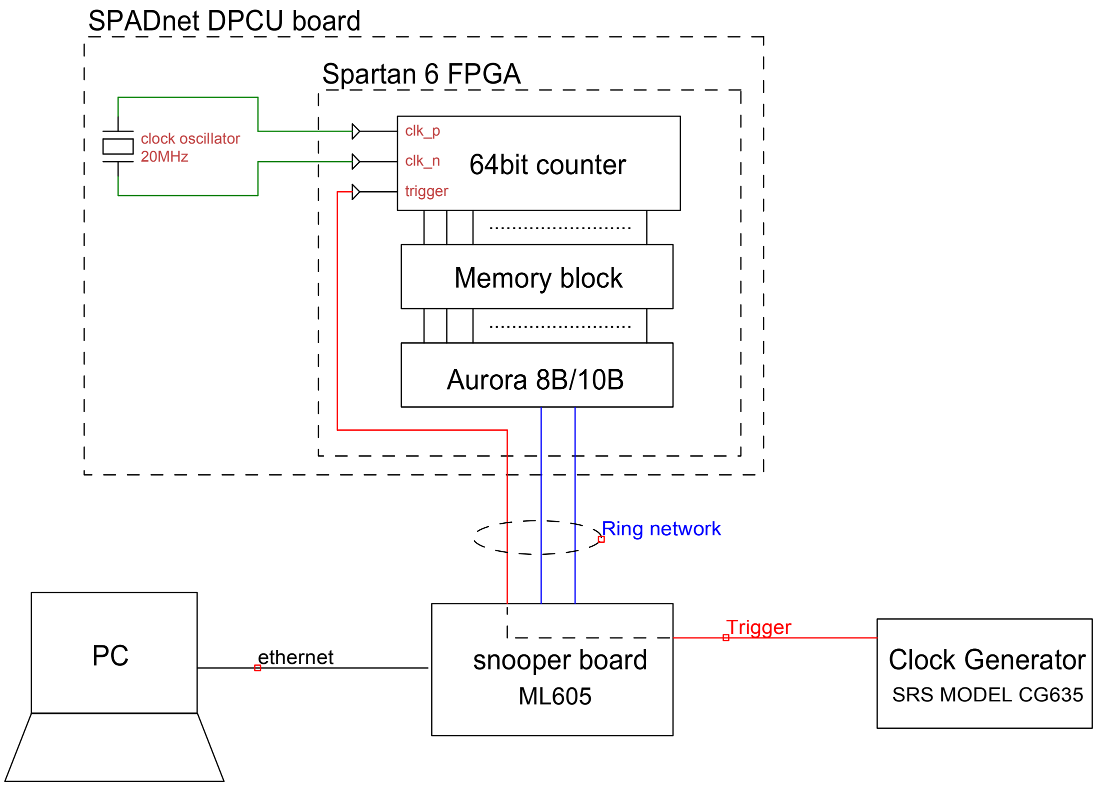


Figure 6.1: Hardware setup used to perform the Allan variance measurement.

The results of the Allan variance using the setup resented in Figure 6.1 are shown in Figure 6.2. The Allan variance is calculated as presented in Equation 2.2. The window where the clock is most stable is 2-16 s, with a stability of about 22 ns.

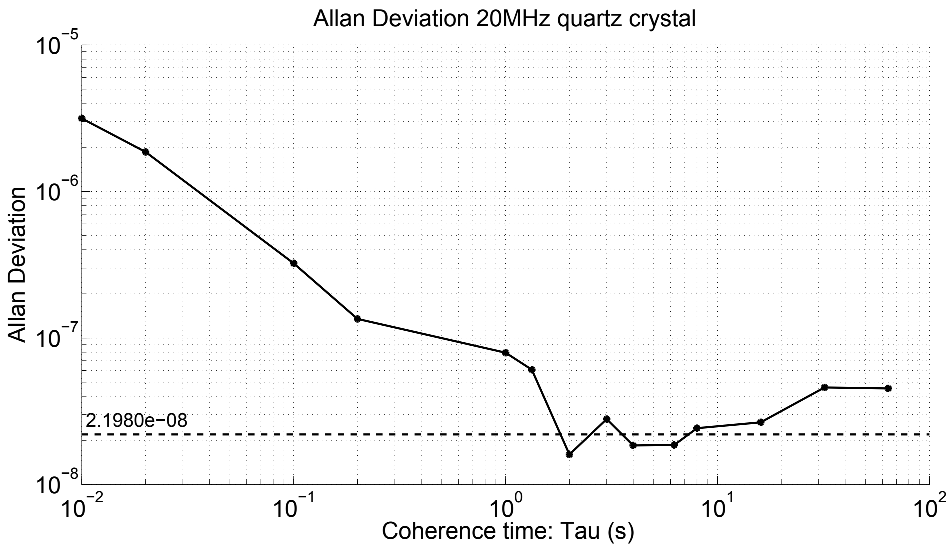


Figure 6.2: Result of the Allan variance measurement.

To compete with the hard-wired clock distribution approach, this lower bound of 22 ns is clearly unacceptable. The hard-wired clock distribution has 11 ps of jitter, and can compensate the phase up-to with 25 ps of accuracy. Therefore either the modules need to be equipped with more stable clock oscillators or another ingenious solution needs to be found. Table 6.1 shows alternative clock oscillators that have a much higher stability. An oven controlled crystal oscillator (OCXO) would be a good solution, even though this is the smallest in size, it is still too large for the SPADnet modules. The size of the PCBs of the modules are limited by the size of the sensor tiles, the sensor tiles of each module need to be as close as possible to each other, to decrease the blind spots in the system. The larger the gaps between each module, the lower the probability of detecting a gamma event pair.

Table 6.1: Alternative stable clock oscillators. Obtained from [23].

	Stabillity ($\tau = 1s$)	**Aging/year**	**size (cm^3)**	**Power (W)**	**price ($)**
OCXO	1 ps	5 ppm	20 - 200	0.6	200 - 2000
Rubidium	3 ps	0.2 ppm	200 - 800	20	2000 - 8000
Cesium	50 ps	0 ppm	6000	30	50000

6.2 Hybrid clock synchronization solution

Since we cannot rely on the local clocks of the module, which are too unstable to achieve satisfactory performance, another solution is required. First, let us recall why network based clock synchronization is needed.

One of the goals of SPADnet is to create a scalable product. In order to achieve this, network based clock synchronization would help. In the current situation, either an extra cable needs to be added (in case of the star distributed clock synchronization topography), or the phase has to be measured, and compensated for (in case of the ring clock distribution), whenever a module is added. With a network based clock synchronization, this process can be suppressed, thus making the system easy to scale.

The ring distributed clock, shown in Figure 6.3 has the interesting property that only the phase needs to be corrected, and since all modules run on the same clock we do not have to worry about the clock stability any more. The clock correction algorithms therefore can be used to estimated the phase, and with the LMK04906 the phase can even be physically corrected. The clock is available on the same flat-cable as the communication network. With this hybrid solution, of hard-wired distributed clock and network based phase estimation, the scalability issue can be resolved. PLS will be used to estimate the phase differences.

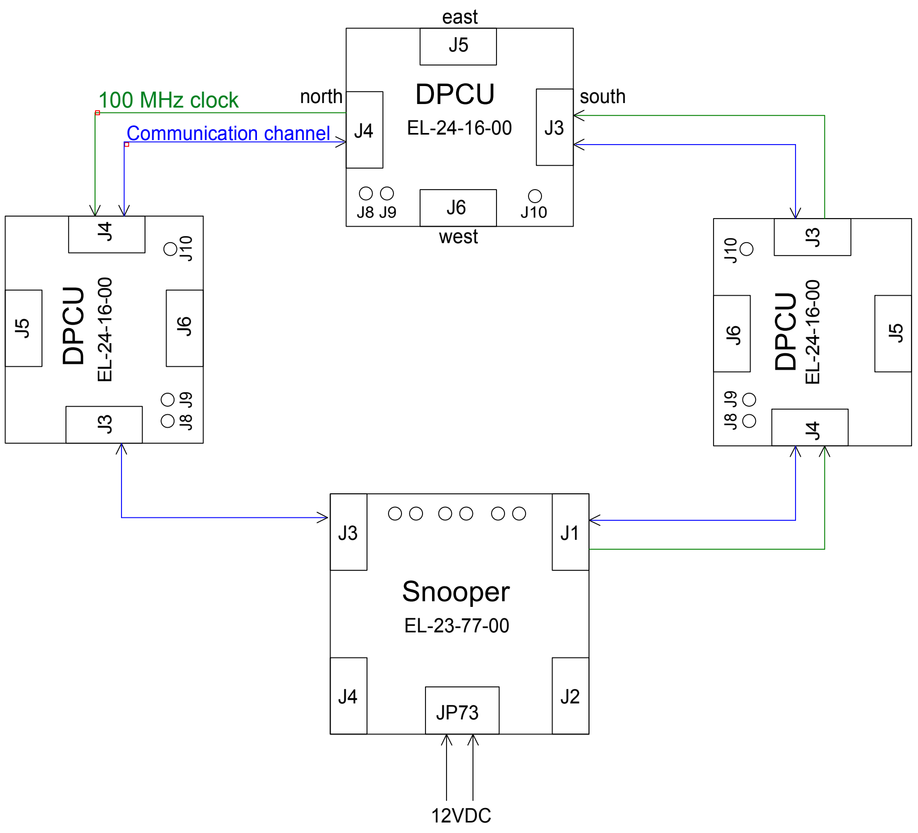
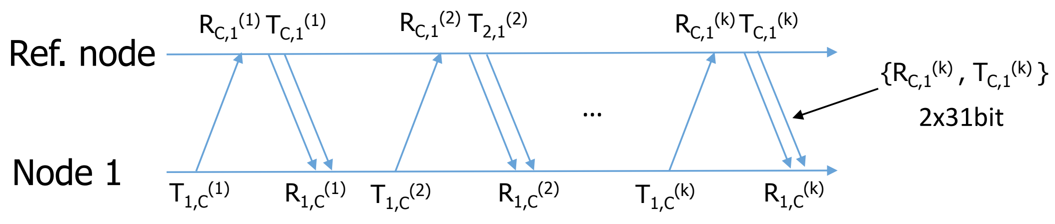


Figure 6.3: Clock distribution network: Ring topography. The phase of the clock will be different at each node, because of the distribution network. The phase will be estimated using pairwise least squares.

6.3 Phase synchronization protocol

Each ring of the SPADnet system will have one reference node, where all the other nodes of that ring synchronize with. The reference node only replies back, with the captured timestamps, to the caller. Two messages are sent back, one containing the reception marker, and the other the transmission time. All the other nodes in the ring, initiate the two-way timestamp exchange messages with the central node, and perform the PLS algorithm to estimate the phase difference. Figure 6.4 shows the two-way timestamp exchanges of a node with the reference node.



Ref. node
Node 1
$R_{C,1}^{(1)}$ $T_{C,1}^{(1)}$
$R_{C,1}^{(2)}$ $T_{2,1}^{(2)}$
$R_{C,1}^{(k)}$ $T_{C,1}^{(k)}$
$\{R_{C,1}^{(k)}, T_{C,1}^{(k)}\}$
2x31bit
...
$T_{1,C}^{(1)}$
$R_{1,C}^{(1)}$
$T_{1,C}^{(2)}$
$R_{1,C}^{(2)}$
$T_{1,C}^{(k)}$
$R_{1,C}^{(k)}$

Figure 6.4: Final two-way timestamp exchange protocol

The network supports the transmission of single packets with 32 bits of data. Each timestamp will consist of a 22 bits coarse value and 9 bit fine value. The most significant bit indicates if the packet holds a reception or transmission time marker. This bit is also used, to ignore the time-stamping of the second reply of the reference node.

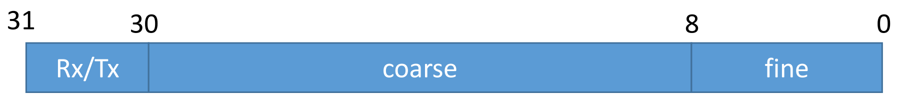
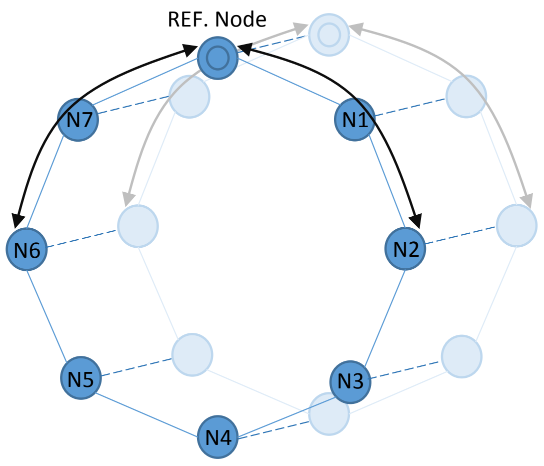


Figure 6.5: Two-way packet construction

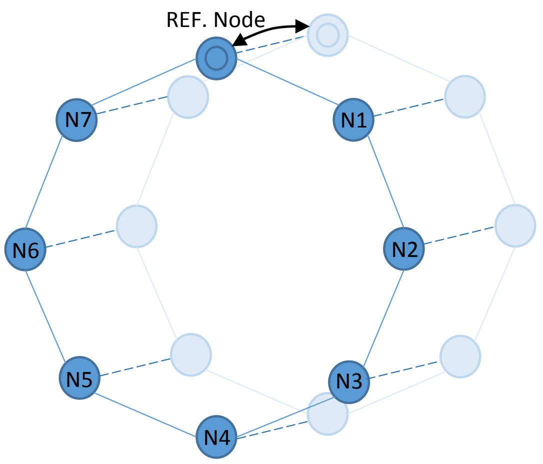
The phase estimation protocol is mapped on to the SPADnet in two stages. Each node in the system, has four network interfaces that can be individually operated (two links to the previous and next node in the same ring, and two links to the adjacent nodes of the other rings), this allows to synchronize multiple nodes to a single reference in parallel.

In the first stage of the synchronization, the individual nodes in the rings will synchronize to a reference node in that ring. Two nodes, one on the left hand and on on the right hand side of the reference node, will simultaneous synchronize to the reference node each interval, when the two nodes are synchronized the next two nodes start to synchronize with the reference node.

Figure 6.6 shows 2 nodes synchronizing with the reference node. The synchronization of all the nodes in the rings, will be done in parallel for each ring. When all the rings have synchronized their nodes, the central nodes will synchronize with each other. The time it takes to synchronize is equal to $(R/2 + N/2) \cdot t$ where N is the number of nodes in one ring, R the number of rings and t the average amount of time needed to synchronize a pair of nodes. This time depends on the observation window. The amount of messages sent is $3K \cdot N$.



(a) Stage 1: nodes synchronization.



(b) Stage 2: rings synchronization.

Figure 6.6: Stages of the synchronization protocol.

6.3.1 Broadcasting

The SPADnet network currently does not support broadcast messages. However if broadcasting messages would be enabled a more efficient synchronization scheme can be applied. Then instead of synchronizing per pair of nodes, the reference node could send broadcast messages around the network, while each node replies to the broadcasts. Using broadcasts reduces the amount of messages needed to synchronize the nodes.

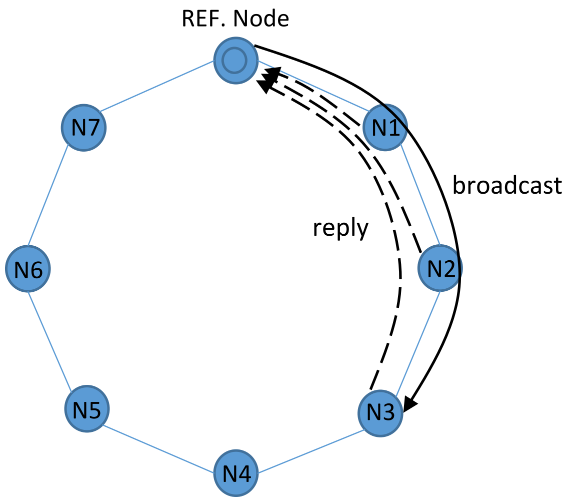


Figure 6.7: The use of broadcast messages to obtain timestamps. The reference node broadcasts a message to all nodes, while the individual nodes reply to the broadcast, while recording the transmission and reception times

7 Hardware implementation

In this chapter the implementation of the network based phase estimator is presented. The pairwise least squares (PLS) algorithm is simplified to only estimate the phase offset and distance. After that the phase estimator is adjusted to be implemented in hardware. In addition, the top-level module, that handles the two-way timestamp exchanges, controls the phase estimator and interfaces with the LMK04906, will be discussed in the second part of this chapter.

7.1 Phase estimator

For the estimation of the phase offset and distance between two nodes the PLS algorithm will be used. The distance estimation is solely used for debugging. The PLS algorithm 7.1 finds an estimate for the clock skew, phase and distance between pairs of nodes.

$$\hat{\boldsymbol{\theta}}_j = \left(\mathbf{A}_{ji}^T \mathbf{A}_{ji}\right)^{-1} \mathbf{A}_{ji}^T \mathbf{t}_{ij}, \tag{7.1}$$

where

$$\begin{aligned} \mathbf{A}_{ji} &= \left[\mathbf{t}_{ji}\ \mathbf{1}_{2K}\ \mathbf{e}\right]^T \in \mathbb{R}^{2K\times 3}, \\ \boldsymbol{\theta}_j &= \left[\alpha_j\ \beta_j\ \tau_{ij}\right]^T \in \mathbb{R}^{3\times 1}. \end{aligned}$$

Because the ring distributed clock network is used there will not be any clock drift in the system. Therefore we do not need to find an estimate for the clock skew α_j.

With $\alpha_j = 1$ we can simplify 7.1 to

$$\hat{\boldsymbol{\theta}}'_j = \left(\mathbf{A}_{ji}'^T \mathbf{A}'_{ji}\right)^{-1} \mathbf{A}_{ji}'^T (\mathbf{t}_{ji} - \mathbf{t}_{ij}), \tag{7.2}$$

where

$$\begin{aligned} \mathbf{A}'_{ji} &= \left[\mathbf{1}_{2K}\ \mathbf{e}\right]^T \in \mathbb{R}^{2K\times 2}, \\ \boldsymbol{\theta}'_j &= \left[\beta_j\ \tau_{ij}\right]^T \in \mathbb{R}^{2\times 1}. \end{aligned}$$

Now the elements of $\mathbf{A}'_{ji}$ all become constants, only the matrix size depends on k, the number of timestamp exchanges. $\mathbf{A}'^T_{ji}\mathbf{A}'_{ji}$ simplifies to

$$\mathbf{A}'^T_{ji}\mathbf{A}'_{ji} = \begin{pmatrix} -1 & -1 \\ -1 & -1 \\ \vdots & \vdots \\ 1 & -1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & -1 & \dots & 1 & 1 \\ -1 & -1 & \dots & -1 & -1 \end{pmatrix} = \begin{pmatrix} 2k & 0 \\ 0 & 2k \end{pmatrix}. \tag{7.3}$$

The inverse of this 2 by 2 matrix is easily found using its determinant

$$\left(\mathbf{A}'^T_{ji}\mathbf{A}'_{ji}\right)^{-1} = \frac{1}{4k^2}\begin{pmatrix} 2k & 0 \\ 0 & 2k \end{pmatrix} = \begin{pmatrix} \frac{1}{2k} & 0 \\ 0 & \frac{1}{2k} \end{pmatrix}. \tag{7.4}$$

Working out the multiplication of $\mathbf{A}'^T_{ji}$ times the matrix inverse from Equation7.4, reduces the least squares to

$$\hat{\boldsymbol{\theta}}'_j = \left(\mathbf{A}'^T_{ji}\mathbf{A}'_{ji}\right)^{-1}\mathbf{A}'^T_{ji}\mathbf{t}_{ij} = \begin{pmatrix} \frac{-1}{2k} & \frac{-1}{2k} \\ \frac{-1}{2k} & \frac{-1}{2k} \\ \vdots & \vdots \\ \frac{1}{2k} & \frac{-1}{2k} \\ \frac{1}{2k} & \frac{-1}{2k} \end{pmatrix} (\mathbf{t}_{ji} - \mathbf{t}_{ij}). \tag{7.5}$$

If the number of timestamp exchanges is $k = 2^i$, for a given i, solving the least squares only requires $2k$ logical shifts, $3k$ additions and $2k$ subtractions.

7.1.1 Hardware implementation

To efficiently implement the algorithm in hardware, the parameters need to be converted to fix point. The smallest value that needs to be represented is $\frac{1}{2k}$, for 8192 timestamp exchanges the smallest number is therefore

$$\frac{1}{2k} = \frac{1}{8192} = 2^{-13}. \tag{7.6}$$

To represent this value in two's complement fix point format, at least 13 fractional bits are required. The largest number that needs to be represented depends only on the communication latency between two nodes (expressed in clock ticks)

$$\sum_{i=1}^{2k} \left(\mathbf{t}_{ji}(i) - \mathbf{t}_{ij}(i)\right)\frac{1}{2k}. \tag{7.7}$$

The communication latency between two SPADnet nodes is about 100 coarse clock ticks, for a 100 MHz time-stamping clock, and 1.25 GB/s line rate. With 25 integer bits, a network of more than 500 nodes can be handled. In the implemented phase estimator registers of maximum 48 bits are used.

The phase estimator is implemented in a iterative fashion, whenever a new pair of timestamps is available, the phase estimator executes an iteration. After enough iterations the phase estimator returns the discovered phase offset.

The phase estimation algorithm is scheduled into 3 stages. In stage 1 the timestamps are subtracted from each other. In stage 2 and stage 3 the matrix vector multiplication, from Equation 7.5, is performed using logical shifts and additions. The Verilog implementation of the phase estimator is placed in Appendix A.1.

The synthesis report of the phase estimator is printed in Appendix A.2, it can be concluded that the phase estimator uses low amount of resources and can operate at high speed (161 MHz).

7.1.2 Coarse timer overflow

Since the coarse counters only has a limited number of bits, they will overflow during the two-way timestamp exchanges. In this design 22 bits were used, for a 100 MHz clock this holds that about every 40 ms the counters overflow. In Table 7.1 the timestamps of two-way timestamp exchanges are executed between two nodes. The overflowing of the counters will once in a while create an large error in the estimation algorithm. To overcome this fault, overflow correction logic is required.

Table 7.1: Two-way timestamp exchange, the transmission and reception times are marked using a 22 bit 100 MHz counter. Overflowing of the counter sometimes needs to be corrected, as we can see from the results of the first steps of the algorithm.

Timestamps				**Algorithm**	
R_{ij}	T_{ij}	R_{ji}	T_{ji}	$R_{ji} - R_{ij}$	$-T_{ji} + T_{ij}$
207	5	99	129	-108	-124
40435	40232	40326	40356	-109	-124
⋮	⋮	⋮	⋮	⋮	⋮
4187909	4187704	4187799	4187830	-110	-126
33832	33631	33724	33755	-108	-124
74059	73858	73951	73982	-108	-124
⋮	⋮	⋮	⋮	⋮	⋮
4154099	4153895	4153990	4154020	-109	-126
23	**4194125**	**4194219**	**4194249**	**4194196**	**-124**
40250	40048	40142	40172	-108	-124
⋮	⋮	⋮	⋮	⋮	⋮
103	**4194205**	**4194299**	**25**	**4194196**	**4194180**

To detect and correct for such overflow faults, the following property of the two-way timestamp exchange is used

$$\begin{aligned} R_{ij} &> R_{ji}, \\ T_{ji} &> T_{ij}. \end{aligned} \tag{7.8}$$

This property should always hold, as can be derived from the two-way timestamp exchange protocol, shown in Figure 7.1. If this property is violated, then overflow requires correction.

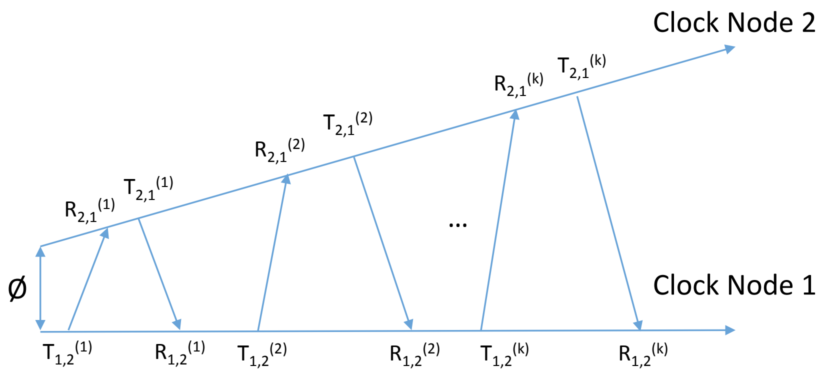


Figure 7.1: Two-way timestamp exchange protocol, with node i = 1 and node j = 2.

There are three incorrect overflows possible, violating the property of Equation 7.8, as shown in Table 7.2.

Table 7.2: Three possible moments of overflow generating a fault. The expected outcome of the first steps of the algorithm should be -108 and -124.

Timestamps				**Algorithm**	
R_{ij}	T_{ij}	R_{ji}	T_{ji}	$R_{ji} - R_{ij}$	$-T_{ji} + T_{ij}$
123	4194225	15	45	-108	4194180
103	4194205	4194299	25	4194196	4194180
23	4194125	4194219	4194249	4194196	-124

The timestamps can be corrected using the maximum coarse counter value ($2^{22} = 4194304$)

$$\begin{aligned} R_{ij} < R_{ji} \rightarrow R'_{ij} = R_{ij} + MaxCounterValue, \\ T_{ji} < T_{ij} \rightarrow T'_{ij} = T_{ij} - MaxCounterValue. \end{aligned} \tag{7.9}$$

Table 7.3, shows the timestamps after applying the overflow correction.

Table 7.3: Corrected timestamps, using the overflow correction.

Timestamps				**Algorithm**	
R'_{ij}	T'_{ij}	R_{ji}	T_{ji}	$R_{ji} - R_{ij}$	$-T_{ji} + T_{ij}$
123	-79	15	45	-108	-124
4194407	-99	4194299	25	-108	-124
4194327	4194125	4194219	4194249	-108	-124

7.1.3 Simulation results

The phase estimator was simulated with ISim simulator v. 14.7. Real timestamps were used, obtained by two-way timestamp exchanges using a two node network. The results are compared with the floating point PLS Matlab simulation, shown in Figure 7.2. There is no difference between the fix point implementation and the floating point Matlab simulation, hence enough fractional bits are used.

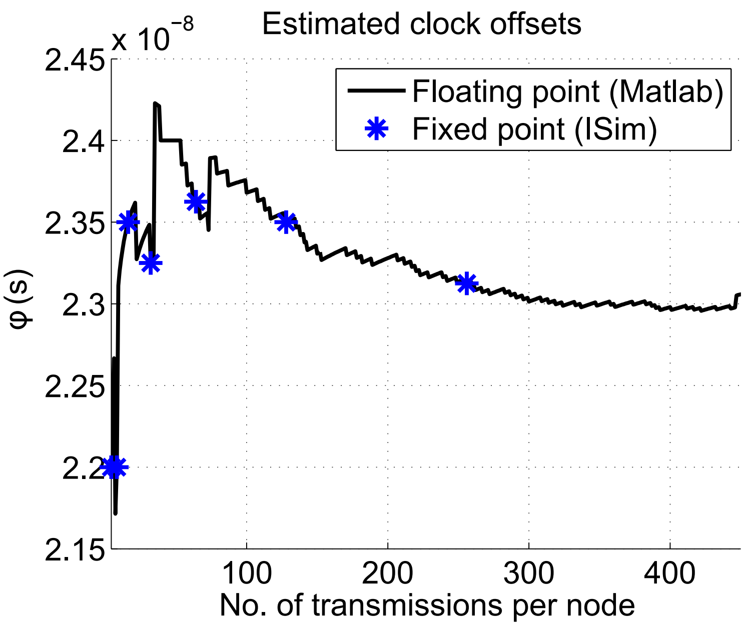


Figure 7.2: Verification of the fixed point phase estimator implementation, using ISim and Matlab.

The fix point implementation only gives a solution for $k = 2^i$ for any i, with k being the number of timestamp exchanges. Therefore the implementation only can give a solution for a limited amount of points, whereas the floating point Matlab solution can give an estimator for any number of timestamp exchanges.

7.2 High resolution time-stamper

The resolution of the phase estimator is the smallest phase offset that it can detect. The resolution of the phase estimation algorithm depends on the time-stamping precision. The Spartan-6 field programmable gate array (FPGA) can operate at a maximum frequency of 400 MHz (period of 2.5 ns). Since the goal is to compete with the hard-wired clock distribution network, a higher resolution is required then this maximum clock frequency. With interpolation it is possible to reach sub-clock resolution, in [24] a 100 ps resolution is achieved using interpolation of a counter in a FPGA. Still this is not enough, since the hard-wired clock distribution can correct a resolution of 25 ps.

In application-specific integrated circuits (ASICs) it is possible to reach a resolution in the order of a few picoseconds, for example in [25] a time-to-digital converter (TDC) is presented with a resolution of 1.12 ps. In those solutions a delayline is used to convert a time to digital code. This delayline is typically constructed out of delay elements, typically inverters, since they are the fastest logic elements in CMOS. The resolution of a delayline TDC depends on the propagation delay of that delay element. The time is measured by counting the number of switched delay elements, a start signal enables the signal propagation through the delayline. When a stop signal is given, the propagation is disabled, the number of switched elements is a scale for the time between the start and stop signal.

In the case of a FGPA the carry chain has the lowest propagation delay to speed up arithmetic operations. An FPGA implementation of a single delayline TDC, using the carry chain, is presented in [26][1]. This TDC has a resolution of about 21 ps and a range of 10 ns. This design will be used in this thesis. To increase the range of the time-stamping the fine delayline TDC is combined with a coarse counter. The coarse counter runs on a 100 MHz clock signal, and stores its current value in a register whenever the trigger is presented. The fine TDCs stop signal is driven by the same clock signal as the coarse counter, the trigger is used as the start signal for the fine TDC. The fine TDC measures the time between the trigger and the first rising edge of the coarse counter (t_f).

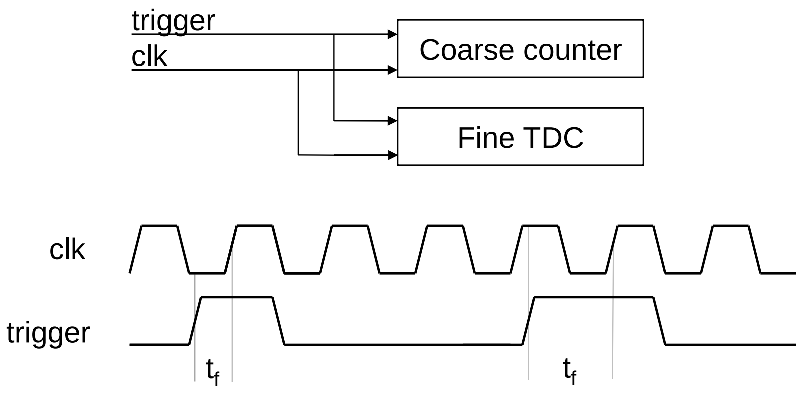


Figure 7.3: Combination of the fine delayline TDC and coarse counter.

[1]This design is available on: http://cas.tudelft.nl/fpga_tdc/

7.2.1 FPGA Time-to-digital converter

The FPGA TDC makes use of the carry structure embedded in the FPGA, the carry chain is used as the delay element of the delayline.

The single delayline structure is shown in Figure 7.4, the resolution of this TDC depends on the propagation delay (t) of the delay element. The thermometer code output of the TDC ($Q1 - Qn$) is converted to binary using a thermodecoder.

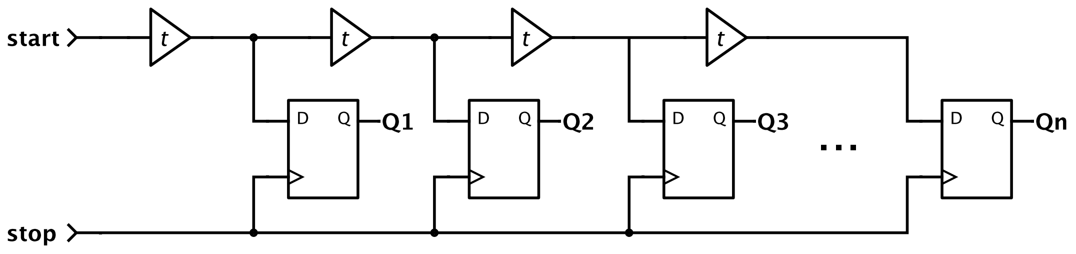


Figure 7.4: General single delayline time-to-digital converter. In the case of the FPGA TDC the delay elements are carry units.

7.2.2 TDC simulation

To compute the resolution of the TDC, a timing simulation (post place-and-route) was performed. This type of simulation takes into account of the timing delay information of the design. The input delay was increased with a step size of 1 ps over a range of 0 to 10 ns. The result of this simulation is shown in Figure 7.5.

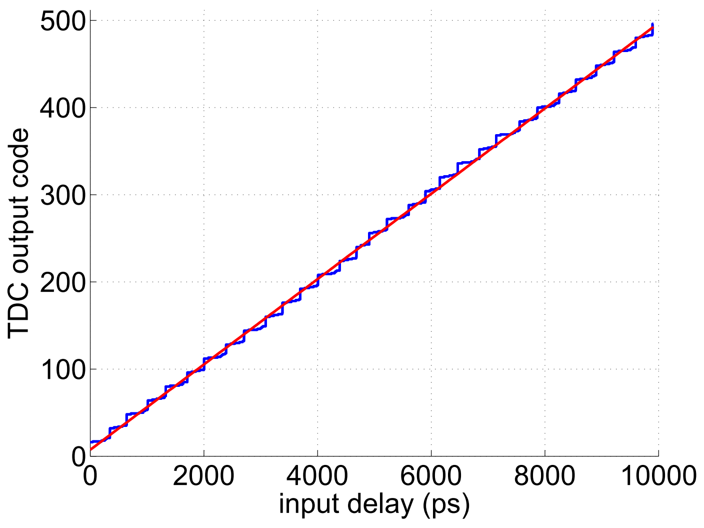


Figure 7.5: Timing simulation of the FPGA TDC.

The resolution of the TDC was computed by applying a linear fit and calculating its slope. The simulated resolution turned out to be 20.4 ps.

7.2.3 FPGA TDC implementation

After implementation, the actual transfer function of the TDC was verified. Although the post place-and-route simulations includes timing information, clock jitter is not taken into account for, imperfections in the FPGA and other system noise components are also not modelled. Unfortunately, it is not possible to characterize the TDC using a step size of 1 ps (like during the simulation). For the characterization the fine analog delay of the LMK04906 was used, this can generate phase differences with a resolution of 25 ps (the accuracy is unknown).

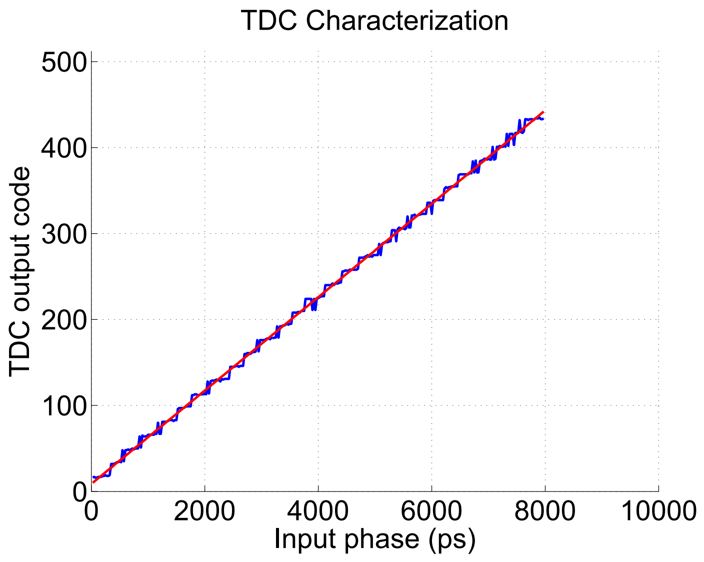


Figure 7.6: transfer function of the implemented FPGA TDC.

Using a linear fit and recovering its slope, the resolution of the TDC is computed to be 18.4 ps. The main contribution of this error, compared to the simulated resolution, is in the uncertainty of the accuracy of the injected phase.

non-linearities

Because of the structure of the FPGA the TDC has large linearity deviations. The main contribution to this is the gaps the carry4 elements, Figure 7.7. This repartition appears in the non-linearity as can been seen in Figure 7.8 and Figure 7.9.

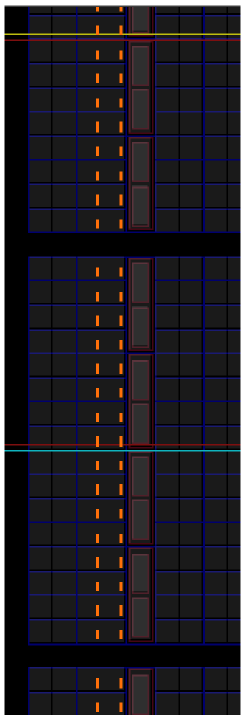


Figure 7.7: Floor plan of the TDC, between every 16 slices there is a gap. In the middle of the 16 slices there is a clock domain crossing. These gaps and crossings introduce delays, affecting the non-linearities of the TDC.

Differential non-linearity (DNL) was calculated using

$$DNL = |[(T_{D+1} - T_D)/T_{LSB-IDEAL} - 1]| \qquad where \;\; 0 < D < 2^N - 2. \tag{7.10}$$

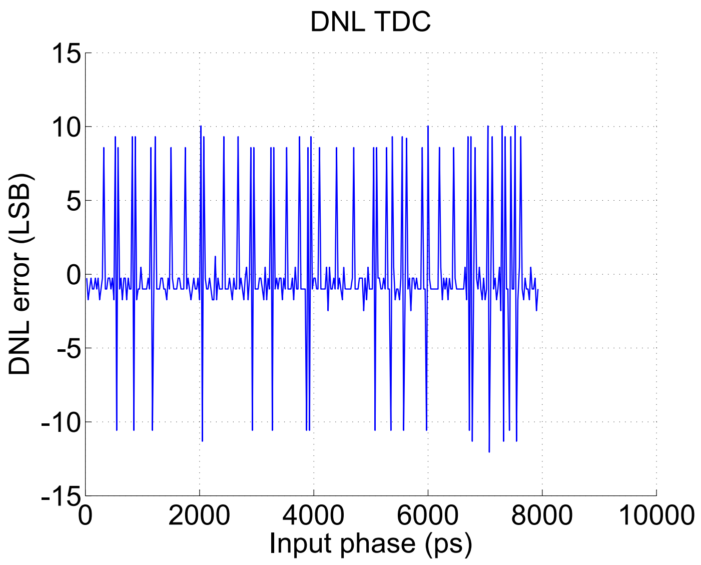


Figure 7.8: Differential non-linearity of the implemented TDC.

The integral non-linearity (INL) discribes the deviation of the transfer function from the ideal transfer function. INL is the integral of DNL, and is formulated as

$$INL = |[(T_D - T_{ZERO})/T_{LSB-IDEAL} - D]| \quad where \;\; 0 < D < 2^N - 1. \tag{7.11}$$

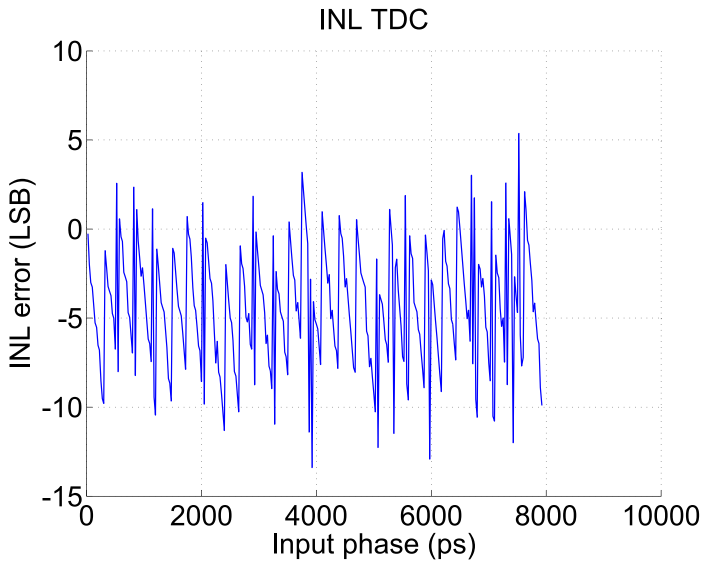


Figure 7.9: Integral non-linearity of the implemented TDC.

The INL and DNL of the implemented TDC are quite high, this non-linearity will increase the noise on the time-stamping of the two-way messages, this will effect the performance of the estimation. However this noise can be compensated by increasing the number of timestamp exchanges.

7.3 Time synchronization module

The time synchronization module handles two-way timestamp exchanges, and controls the phase estimation module. This module has two configurations, reference and normal. The modules are build out of general blocks.

The packet decoder, decodes the received messages. Depending on the header of the messages a trigger is send to the counters, and the time marker embedded in the packet is stored in the timestamp register. The transmission packets are constructed by the packet encoder. The coarse counter and fine counter are triggered whenever a message is received or transmitted, the outcomes of these counters are stored in the timestamp register. When, in the normal node, all timestamps of one two-way timestamp exchange iteration are collected, the phase estimator is updated. With the LMK adjuster the LMK04906 registers can be programmed, the LMK adjuster was mainly used to inject known phase offsets to the 100 MHz clock. The flow of data and control is handled by the status and control block.

7.3.1 Hardware implementation, normal node

The normal nodes initiate the two-way timestamp exchanges with the reference node. When the network is free the node sends a message to the reference. Then, it waits on the reply of that node. When the node does not answer within a given amount of time, it is assumed that a packet is lost in the network, the initiation is repeated. When a reply is received, the phase estimator is updated. Afterwards either enough two-way timestamp exchanges are fulfilled, or the complete sequence is repeated till enough timestamps are collected. These nodes are also capable of interfacing with the LMK04906 clock buffer. During this thesis this feature was mainly used to generate known phase offsets. But in the final system they could be used to physically correct the phase of the clocks, using the estimated clock correction parameters.

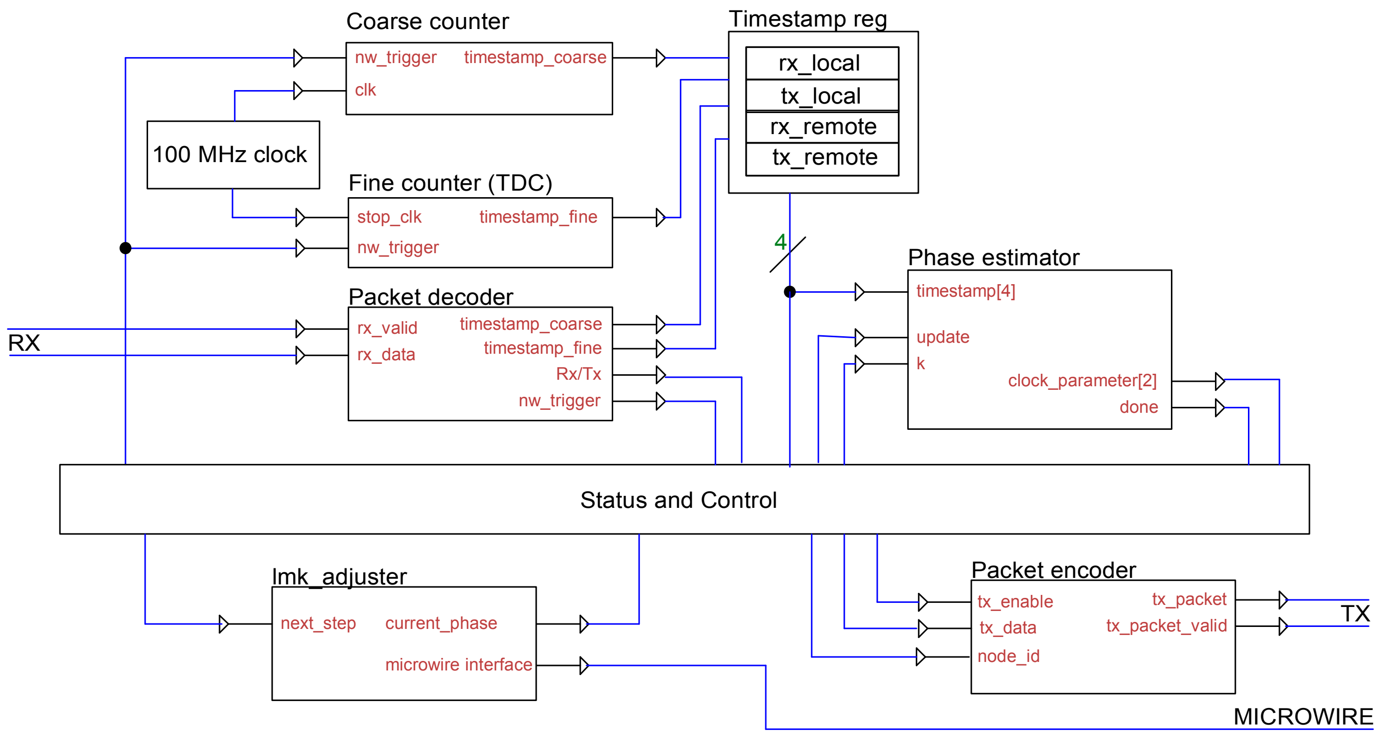


Figure 7.10: Simplified architecture of the normal node

7.3.2 Hardware implementation, reference node

The reference nodes are a simplified version of the normal node, they only reply with two messages whenever it has received a messages from a node. The received messages are timestamped. Just before replying the first message the transmission time is marked. The first message is filled with the reception timestamp, the second reply contains the transmission time of the first reply.

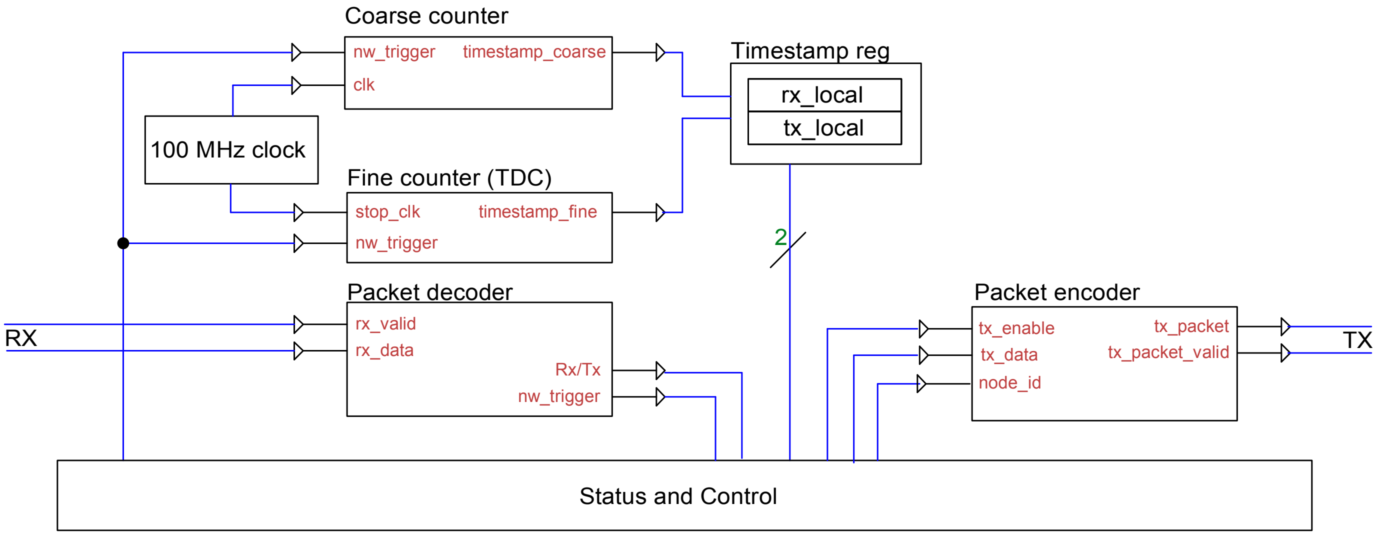


Figure 7.11: Simplified architecture of the reference node

Verification 8

In this chapter the performance of the phase estimator will be verified. First, the stability of the phase estimators will be verified. Later, the resolution of the phase estimator will be measured. And finally the hybrid solution will be evaluated.

8.1 Phase estimator stability

The stability of the estimator will give information about the estimation error. To measure the stability of the implemented phase estimator, the estimator was run 2000 times with a fixed phase difference between two adjacent nodes. The estimated phases were accumulated in a histogram, see Figure 8.1. The estimation was performed using 2048 two-way timestamp exchanges and an observation window of 328 ms. The standard deviation of the 2000 estimations is 124 ps.

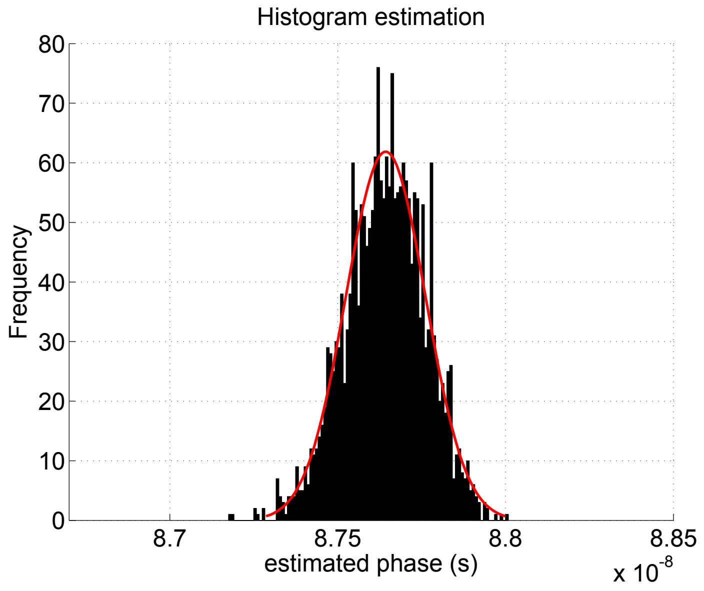


Figure 8.1: 2000 Executions of the hardware implemented phase estimation, 4096 two-way timestamp exchanges, 328 ms observation window, 124 ps standard deviation. For a fixed phase between two adjacent nodes

As mentioned in Section 3.3.5, the performance of the pairwise least squares (PLS) algorithm depends on the number of two-way timestamp exchanges and the observation window. To give insight on the performance of the phase estimator, the estimator was executed for a fixed number of two-way timestamp exchanges and increasing observation window($0.8-33$ ms). The result is shown in Figure 8.2, the estimator reaches a stability of 182 ps (standard deviation) with 2048 two-way timestamp exchanges and 329 ms observation window.

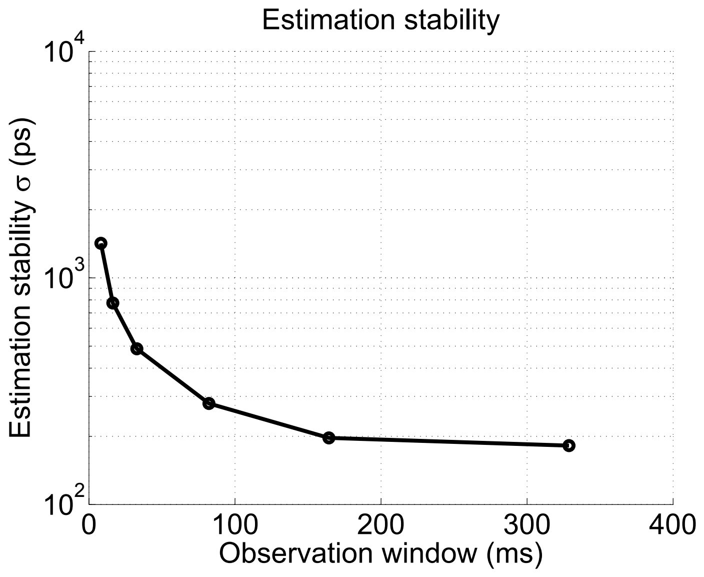


Figure 8.2: Estimation stability, standard deviation, of 2000 estimations. For fixed $k = 2048$ two-way messages, and increasing observation window ($8 - 328$ ms).

The performance of the algorithm is also influenced by the number of k two-way timestamp exchanges as can been seen in Figure 8.3. In this graph k was increased from 64 to 4096 with a fixed communication interval. Because of the fixed communication interval, also the observation window increases when k increases. With $k = 4096$ and observation window of 33 ms a stability of 124 ps standard deviation is reached.

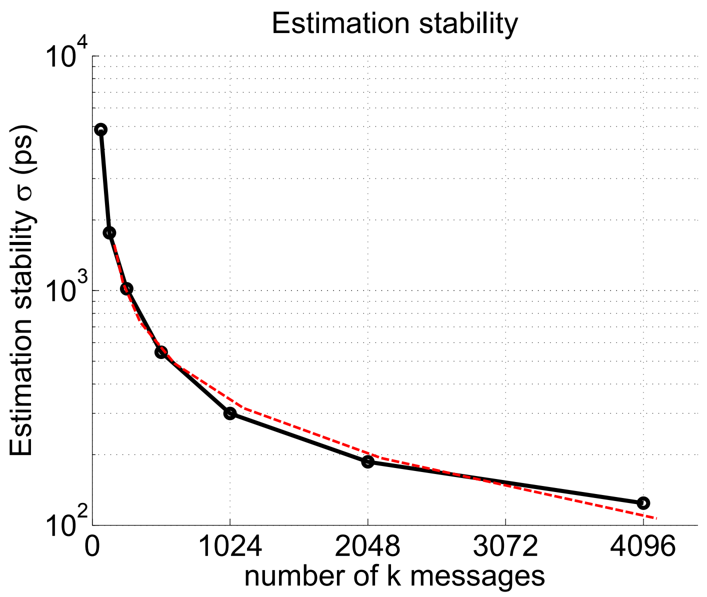


Figure 8.3: Estimation stability, standard deviation, of 2000 estimations. With increasing k two-way messages and fixed communication interval of 40 microseconds.

By the central limit theorem (CLT) [27], it is defined that when the sample size (n) increases the estimated sample mean gets closer to the true mean. And the distribution of this average will approximate a normal distribution, while the standard deviation gets smaller when the sample size increases. Since we use an unbiased estimator, and our process is assumed to be independent and identically distributed (IID) the estimator will follow this theorem. The standard deviation should follow the slope

$$\sigma_n = \frac{1}{\sqrt{n}}. \tag{8.1}$$

This slope was plotted in Figure 8.3, the estimator indeed approaches this slope. Therefore we can conclude that when k, sample size, gets increased further the stability of the estimator would increase following the slope from Equation 8.1.

8.2 Resolution (LSB)

In this section the resolution, or least significant bit (LSB), of the phase estimator is verified. The resolution of the phase estimator depends on the precision of the timestamping of the 2-way timestamp exchange messages. The timestamping has a resolution of 18.4 ps (resolution of the FPGA TDC).

The resolution was verified by estimating the phase difference between two nodes, while the coarse clock was shifted with steps of 25 ps over a range of 10 ns. The injection of this 25 ps phase offset was realized by the LMK04906 clock jitter cleaner. The transfer function of the phase estimator is plotted in Figure 8.4.

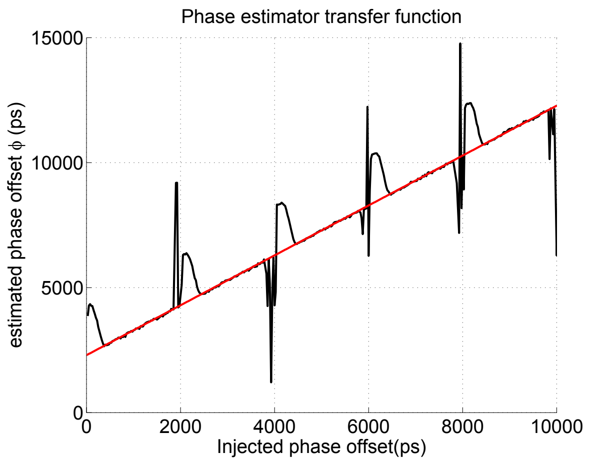


Figure 8.4: Transfer function of the phase estimator.

There are five large glitches in the transfer function of the phase estimator. The same pattern is also visible in the distance estimation (communication latency), see Figure 8.5, of the phase estimator.

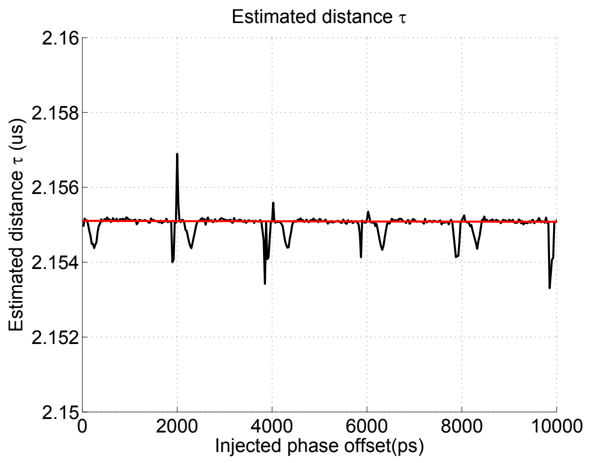


Figure 8.5: Transfer function of the estimated distance.

It was discovered that these glitches originate from the frequency difference between the coarse clock (100 MHz) and the network clock (125 MHz). The network clock is used for the high speed GTP transceivers, see Appendix A. Since these frequencies are not an integer multiple of each other, there is an uncertainty in the phase relation between these two clocks. Changing the frequency of the coarse clock to 125 MHz resolved this issue.

The transfer function of the phase estimator, with the 125 MHz coarse clock, is presented in Figure 8.6. The glitches now no longer appear. The resolution of this transfer function is 18.4 ps, matching the resolution of the FPGA TDC found in Section 7.2.

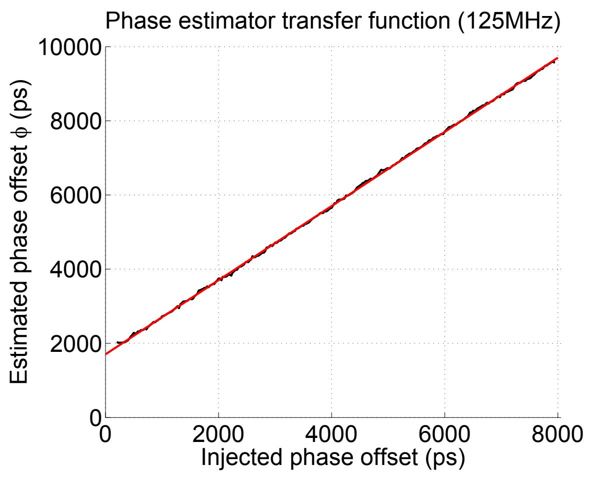


Figure 8.6: Transfer function of the phase estimator, showing the average of 25 estimations per injected phase offset. Using $k = 4096$ two-way timestamp exchanges over a observation window of 328 ms.

The error, with respect to the fitted transfer function, is shown in Figure 8.7. The maximum absolute error is 137 ps. This error corresponds to the stability of the estimator, as discussed in Section 8.1. Hence by increasing the number of two-way timestamp exchanges the error can be reduced.

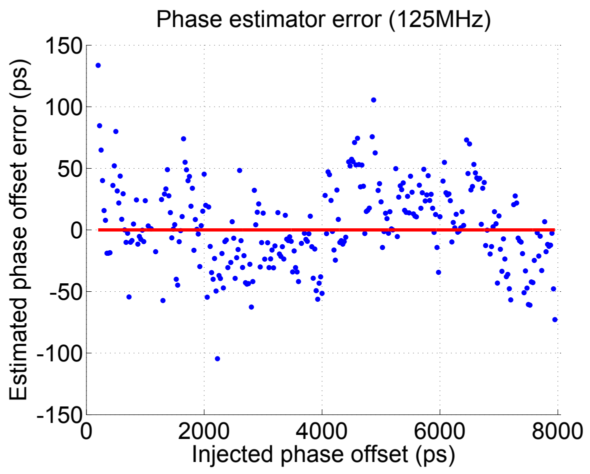


Figure 8.7: Estimation error compared to ideal transfer function. Presenting the average of 25 estimations per injected phase offset. Using $k = 4096$ two-way timestamp exchanges over a observation window of 328 ms.

8.3 Scalability study

In all previous measurements a two node system was used, and a stability of 124 ps was reached, with 4096 two-way timestamp exchanges. In this section the phase estimator will be tested in a three node system, as shown in Figure 8.8.

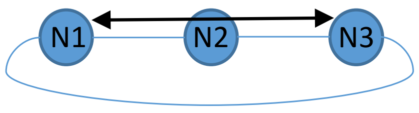


Figure 8.8: Three node ring system

The phase between node 1 and node 3 was estimated for different phase offset, generated by the LMK04906. The two-way messages traversed through node 2. During this experiment the number of two-way timestamp exchanges was 2048 with a observation window of about 165 ms. The result of this measurement is shown in Table 8.1.

In the case of a two node system, with the same settings, a stability of 187 ps was achieved, introducing a node in between decreased the stability to 233 ps. This extra instability, about 50 ps, is introduced by the forwarding of the messages in node 2, this process adds uncertainty to the latency of the message between node 1 and node 3, hence increasing the observation noise. The extra error can be compensated by increasing the number of samples. The latency uncertainty can be improved by using deterministic communication, as described in Appendix B.4.

Table 8.1: Phase estimation in a three node network for five different injected phase offsets. Results are obtained from about 2000 estimations per injected phase offset, with 2048 two-way timestamp exchanges over an observation window of 165 ms. ABS: absolute, SD: standard deviation. The injected phase does not match the estimated phase due to the error discussed in Section 8.2, this fault has been resolved.

Injected phase	**Estimated phase**	**Error (ABS)**	**SD**	**Units**
25	3	22	240	ps
200	515	315	239	ps
400	228	172	237	ps
425	234	191	233	ps
825	737	88	216	ps
Average		**158**	**233**	ps

8.4 Hybrid synchronization versus hard-wired synchronization

The hybrid solution uses network based clock synchronization to automatically estimate the phase offsets, in the ring hard-wired clock distribution network of SPADnet. This solution reduces the user proceedings whenever system is extended with extra nodes, compared to the hard-wired clock synchronization.

Before, the phases where only measured and corrected once, during the installation of the system, or when the system is extended. The presented solution enables the option to regularly monitor the phase offsets, this allows to compensate for temperature changes, ageing of the electronics and other sources that might affect the phase offsets in the system.

The presented solution achieves a resolution for phase offset estimation of 18.4 ps. The estimation error for a two node system is 124 ps, and for a three node system 233 ps, this error can be lowered by further increasing the two-way timestamp exchanges. Implementing deterministic communication, as presented in Appendix B.4, will further improve the performance.

The comparison of the presented hybrid solution versus the hard-wired synchronization is summarized in Table 8.2.

Table 8.2: Comparison of the hybrid solution versus the hard-wired clock synchronization solution.

	Hybrid solution	**Hard-wired solution**
Phase offset calibration	Automatic (18.4 ps)	Manual
Clock Jitter (SD)	<6 ps	<6 ps
Online phase monotoring	Yes	No
Scalable	Yes	No

Conclusion 9

9.1 Summary

Recapitulating, this thesis investigated if the hard-wired clock distribution network of SPADnet can be made scalable, using a network based clock synchronization solution.

9.1.1 Characterization of hard-wired clock distribution networks

First, in Chapter 5, the hard-wired clock distribution network of SPADnet was characterized. The clock jitter was measured in a four node system, and was found to be better than 6 ps (standard deviation).

9.1.1.1 Ring clock distribution network

In the ring clock distribution network, the clock is daisy chained from each node to the next. At each node the clock is cleaned using a LMK04906. Because of the daisy chained clock, there is an clock offset at each node. The clock offsets can be compensated, with a precision of 25 ps, by the LMK04906. However to do this the user needs to measure the phase offset and program the LMK04906. This operation can be time-consuming for large systems, and also the appropriate measuring equipment is required.

9.1.1.2 Star clock distribution network

By using the star clock distribution network, the phase offset at each node is minimized. The number of nodes is limited by the amount of available clock outputs on the clock distribution board, and therefore restricts the scalability. The coaxial cables, used to distribute the clock, can, and the ICS8534-01 1-to-22 fanout buffer will introduces a phase offset. The fanout buffer introduces a maximum phase offset error of 100 ps [22]. Depending on the quality of the coaxial cables, the phase needs to be measured and compensated, for satisfying results.

9.1.2 Network based clock synchronization

Three novel network based clock synchronization algorithms presented in [10] [11] where analysed in Chapter 3. It is stated that these algorithms reach the theoretical limit of network based clock synchronization. The eventual performance of network based synchronization is, however, bounded by the stability of the applied local clock synthesizers. In the case of SPADnet low-cost quartz oscillators are used. The best stability of these oscillators was measured to be 22 ns for the 2-16 s region (Chapter 6). This performance is unacceptable for SPADnet. The local quartz oscillators could be replaced by more stable clock synthesizers, for example oven controlled crystal oscillator

(OCXO). However OCXO oscillators are to large to be applied in the SPADnet system. Therefore clock synchronization solely based on network based clock synchronization is not possible for SPADnet.

9.1.3 Hybrid solution

This thesis presents a hybrid clock synchronization solution, this solution is presented in Chapter 6. The hybrid clock synchronization solution uses the ring hard-wired clock distribution network in combination with a network based phase estimator. This combination enables scalability while maintaining high precision.

The scalability of the ring clock distribution is improved by reducing the user proceedings whenever system is extended with extra nodes. Heretofore the user needed to measure the phase offsets at each node and correct this phase offset, using the programmable registers of the LMK04906. These operations can be time consuming, and appropriate measuring equipment is required. The hybrid solution makes these user operations unnecessary.

The phase estimator is based on the novel pair wise least squares (PLS) algorithm [10]. The algorithm was optimized and implemented in Spartan-6 FGPAs. A delayline FPGA TDC was deployed to enable high resolution phase estimation. The presented solution reaches a resolution of 18.4 ps. The stability, or error, of the phase estimator is 233 ps for a three node system. The estimation error can easily be improved by increasing the sample size parameter of the estimator.

For the brain and clinical SPADnet system a clock jitter of less than 50 ps is required. The hybrid solution makes use of the hard-wired clock distribution network, which has a clock jitter of less than 6 ps at each individual node. The worst-case phase jitter between two nodes, in a four nodes system, was measured to be 14 ps (Chapter 5).

Before, the phase offsets where only measured and corrected once, during the installation of the system, or when the system is extended. The presented hybrid solution enables the option to regularly monitor the phase offsets, this allows to compensate for temperature changes, ageing of the electronics and other sources that might affect the phase offsets in the system.

Table 9.1: Comparison of the hybrid solution versus the hard-wired clock synchronization solution.

	Hybrid solution	**Hard-wired solution**
Phase offset calibration	Automatic (18.4 ps)	Manual
Clock Jitter (SD)	<6 ps	<6 ps
Online phase monitoring	Yes	No
Scalable	Yes	No

9.2 Future work

To further improve the hybrid clock synchronization solution, the following work has to be done.

1. Enable dedicated phase-alignment of the GTP transceivers. This ensures fixed phase offset in the transceiver. Currently, after each programming of the FPGA a random phase offset is introduced in the GTP transceiver. See Appendix B.4.
2. Enable low-latency and deterministic communication as described in Appendix B.4. This will improve the performance of the hybrid solution.
3. Improve non-linearities of the FPGA TDC, to lower the observation noise, increasing the stability of the phase estimator.
4. Exploit the possibility of using, broadcasts, GLS or ATPL, to improve the efficiency of estimating the phase offsets in a large network at once.
5. Test the hybrid solution in a many node system.

Bibliography

[1] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52, April 2008.

[2] Edoardo. Charbon, Claudio Bruschini, Chockalingam Veerappan, Leo Huf Campos Braga, et al. Spadnet: A fully digital, networked approach to mri compatible pet systems based on deep-submicron cmos technology. *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pages 1–5, 2013.

[3] R.T. Rajan, M. Bentum, and A.-J. Boonstra. Synchronization for space based ultra low frequency interferometry. *Aerospace Conference*, March 2013.

[4] Nikolaos M. Freris, Hemant Kowshik, and P. R. Kumar. Fundamentals of large sensor networks: Connectivity, capacity, clocks and computation. *IEEE*, (11), November 2010.

[5] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. *Networked and Embedded Systems Lab*, November 2003.

[6] Kyoung-Lae Noh, Qasim Mahmood Chaudhari, Erchin Serpedin, and Bruce W. Suter. Novel clock phase offset and skew estimation usingtwo-way timing message exchanges for wireless sensor networks. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 55(4), April 2007.

[7] K. Arvind. Probabilistic clock synchronization in distributed systems. *IEE TRANSACTIOS ON PARALLEL AND DISTRIBUTED SYSTEMS*, 5(5), May 1994.

[8] Hui Dai and Richard Han. Tsync : A lightweight bidirectional time synchronization service for wireless sensor networks. *Mobile Computing and Communications Review*, 8(1), January 2004.

[9] David L. Mills. Internet time synchronization: The network time protocol. *IEEE TRANSACTIOS ON COMMUNICATIONS*, 39(10), October 1991.

[10] Raj Thilak Rajan and A-J. van der Veen. Joint ranging and clock synchronization for a wireless network. *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2011.

[11] Sundeep Prabhakar Chepuri, Raj Thilak Rajan, Geert Leus, and Alle-Jan van der Veen. Joint clock synchronization and ranging: Asymmetrical time-stamping and passive listening. *Signal Processing Letters, IEEE*, pages 51–54, 2013.

[12] Alper Demir, Amit Mehrotra, and Jaijeet Roychowdhury. Phase noise in oscillators: A unifying theory and numerical methods for characterization. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSI: FUNDAMENTAL THEORY AND APPLICATIONS*, 47(5), May 2000.

[13] John R. Vig. Introduction to quartz frequency standards, research and development technical rept:. *Army Lab Command Forn Monmouth NJ Electronics Technology and Devices Lab*, page 55, March 1992.

[14] Fred L. Walls and Jean-Jacques Gagnepain. Environmental sensitivities of quartz oscillators. *IEEE TRANSAUIONS ON ULTRASONICS, FERROELECTRICS. AND FREQUENCY CONTROL*, 39(2), March 1992.

[15] Alper Demir. Phase noise and timing jitter in oscillators with colored-noise sources. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSI: FUNDAMENTAL THEORY AND APPLICATIONS*, 49(12), December 2002.

[16] SiTime. Report: Clock jitter definitions and measurement methods, Januari 2014.

[17] David W. Allan. Time and frequency(time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 34(6):647–654, 1987.

[18] Jeremy Elson, Le wis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *Symposium on Operating Systems Design and Implementation*, December 2002.

[19] MONSON H. HAYES. *BOOK: STATISTICAL DIGITAL SIGNAL PROCESSING AND MODELING.* JOHN WILEY & SONS, INC., 1996.

[20] Claudio Bruschini. Nss-mic special focus workshop towards 10ps single soft photon detectors. 2014.

[21] Taxes instruments. Datasheet: Lmk04906 ultra low noise clock jitter cleaner/multiplier with 6 programmable outputs, May 2013.

[22] IDT. Datasheet: Ics8534-01 low skew, 1-to-22 differential-to-3.3v lvpecl fanout buffer, December 2007.

[23] Hui Zhou. Frequency accuracy & stability dependencies of crystal oscillators. *Technical Report SCE-08-12*, page 15, November 2008.

[24] Ryszard Szplet, Jozef Kalisz, and Rafal Szymanowski. Interpolating time counter with 100 ps resolution on a single fpga device. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 49(4), August 2000.

[25] KwangSeok Kim, Wonsik Yu, and Seong Hwan Cho. A 9 bit, 1.12 ps resolution 2.5 b/stage pipelined time-to-digital converter in 65 nm cmos using time-register. *Solid-State Circuits, IEEE*, 49(4):1007 – 1016, April 2014.

[26] Harald Homulle, Francesco Regazzoni, and Edoardo Charbon. 200 ms/s adc implemented in a fpga employing tdcs. *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, March 2015.

[27] David J. Goodman Roy D. Yates. *Probability and Stochastic Processes.* Wiley, 2005.

[28] A.X. Widmer and P.A. Franaszek. A dc-balanced, partitioned-block, 8b/10b transmission code. *IBM Journal of Research and Development*, 27(5):440–451, September 1983.

[29] Raffaele Giordano and Alberto Aloisio. Fixed-latency, multi-gigabit serial links with xilinx fpgas. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE*, 58(1), Februari 2011.

List of my Publications 10

1. M. Bijwaard, et al. Scalable Network Based Clock Synchronization for Digital PET Systems. Journal of Instrumentation (JINST). 2015.
2. C. Veerappan, M. Bijwaard, et al. Distributed Coincidence Detection Architecture in Digital PET Sensor Network. Journal of Instrumentation (JINST). 2015.
3. C. Veerappan, M. Bijwaard, et al. Data Acquisition in Digital PET Sensor Network. Journal of Instrumentation (JINST). 2015.

The publications are under preparation

Verilog Phase Estimator A

A.1 Verilog source

Verilog implementation of the phase estimator.

```
    always@(posedge clock) begin
        if(reset) begin
            x <= 0;
            c[0] <= 0;
            c[1] <= 0;
        end else if(markers_valid && i<k)begin
            //start update sequence
            x <= 1;
        end
        if(x == 1) begin
            a[0] <= ((Rji - Rij));
            a[1] <= ((-Tji + Tij));
            x <= 2;
        end else if (x == 2) begin
            b[0] <= (a[0] - a[1]) <<<(precision-shift);
            b[1] <= (a[0] + a[1]) <<<(precision-shift);
            x <= 3;
        end else if (x == 3) begin
            c[0] <= c[0] + b[0];
            c[1] <= c[1] + b[1];
            x <= 4; //Stall the SM
        end

    end

endmodule
```

A.2 Synthesis report

Synthesis report of the phase estimator for the Spartan-6 FPGA, created with Xilinx ISE 14.7.

```
=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                    : 8
 32-bit subtractor                                      : 2
 47-bit subtractor                                      : 1
 48-bit adder                                           : 3
 48-bit subtractor                                      : 2
# Counters                                              : 1
 13-bit up counter                                      : 1
# Registers                                             : 388
 Flip-Flops                                             : 388
# Comparators                                           : 1
 32-bit comparator greater                              : 1
# Multiplexers                                          : 5
 4-bit 2-to-1 multiplexer                               : 2
 96-bit 2-to-1 multiplexer                              : 3
# Logic shifters                                        : 2
 48-bit shifter logical left                            : 2

=========================================================================
 Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 6.191ns (Maximum Frequency: 161.513MHz)
   Minimum input arrival time before clock: 9.512ns
   Maximum output required time after clock: 3.634ns
   Maximum combinational path delay: No path found
=========================================================================
```

Aurora 8B/10B

B

SPADnet uses the Aurora 8B/10B protocol for the network, enabling high-speed serial communication. Aurora 8B/10B is a link-layer protocol that makes use of the high-speed serial transceivers embedded in the FPGAs. The protocol uses 8B/10B encoding to achieve DC-balance.

B.1 8B/10B encoding

The 8B/10B transmission code was published by IBM in 1983 [28]. The transmission code transforms 8 bit characters in to 10 bits. The two extra bits are used to keep the long-term ratio of ones and zeros transmitted equal, this is called DC-balance. De encoding also limits the maximum run length to five, this means that there will never be more than five successive ones or zeros. This increases the reliability of the link and allows easier clock recovery from the data. The protocol includes special characters to establish byte synchronization, indicated the beginning and end of packets and other signal control functions. The 8B/10B transmission code has an overhead of 25%. It is also possible to use 64B/66B transmission code which has less overhead and functions similar.

B.2 GTP transceivers

The Spartan-6 FPGAs are equipped with GTP transceivers, the transceivers act like serializer/deserializer (SERDES). The SERDES serializes the data on to the link, and recovers the serial data from the link. The general block diagram of a SERDES is shown in Figure B.1.

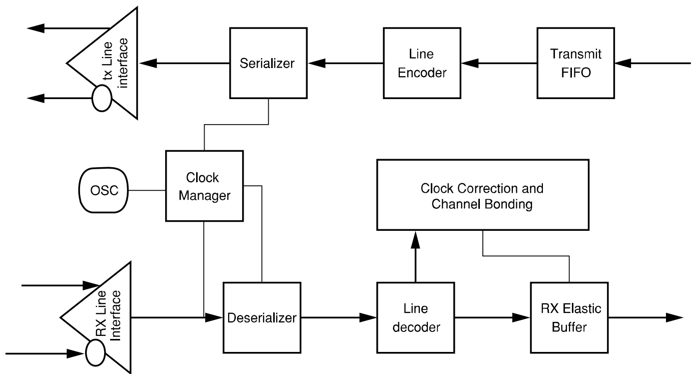


Figure B.1: Generic block diagram of a SERDES

Data enters the transmit FIFO buffer and gets encoded by the line encoder. In our case a 8B/10B encoder is used. The encoded data is converter to a serial stream by the serializer. The serializer runs on a clock generated by the clock manager, the clock speed depends on the line rate. The tx line interface maps the serial stream on the physical line.

The differential signal is converted to a single signal, this stream is deserialized and decoded. The decoded data is stored in a elastic buffer. This buffer is used to cross the data from the SERDES clock domain and the user clock domain. Clock correction is used to align the data correctly with the user clock, for this special characters are used called K-characters. Optionally channel bonding can be used to align multiple lanes.

B.3 Clock correction

The RX elastic buffer can also be used to compensate for frequency differences, between the PMA (physical medium attachment sub-layer) clock and the PCS (physical coding sub-layer) clock domains, by performing clock correction. Clock correction involves a unique character or unique sequence of characters who does not occur in the data stream. Typically K-characters are used for this. Clock correction works by monitoring the data in the RX elastic buffer, and preventing the RX elastic buffer from getting too full or too empty by deleting or replicating special idle characters in the data stream. Clock correction can be avoided when the frequency source for transmission and receiving and are the same.

B.4 Deterministic communication

To be able to estimate the clock parameters a deterministic delay in the communication is necessary. It was verified that the latency of the aurora 8B/10B protocol was constant during power on. However the latency randomly changes after reprogramming the device, therefore the latency is non-deterministic. The non-deterministic latency is due to a random phase difference between the PMA (physical medium attachment sub-layer) clock and the PCS (physical coding sub-layer) clock, the PMA and PCS domains of the Spartan-6 GTP transceiver are shown in Figure B.2.

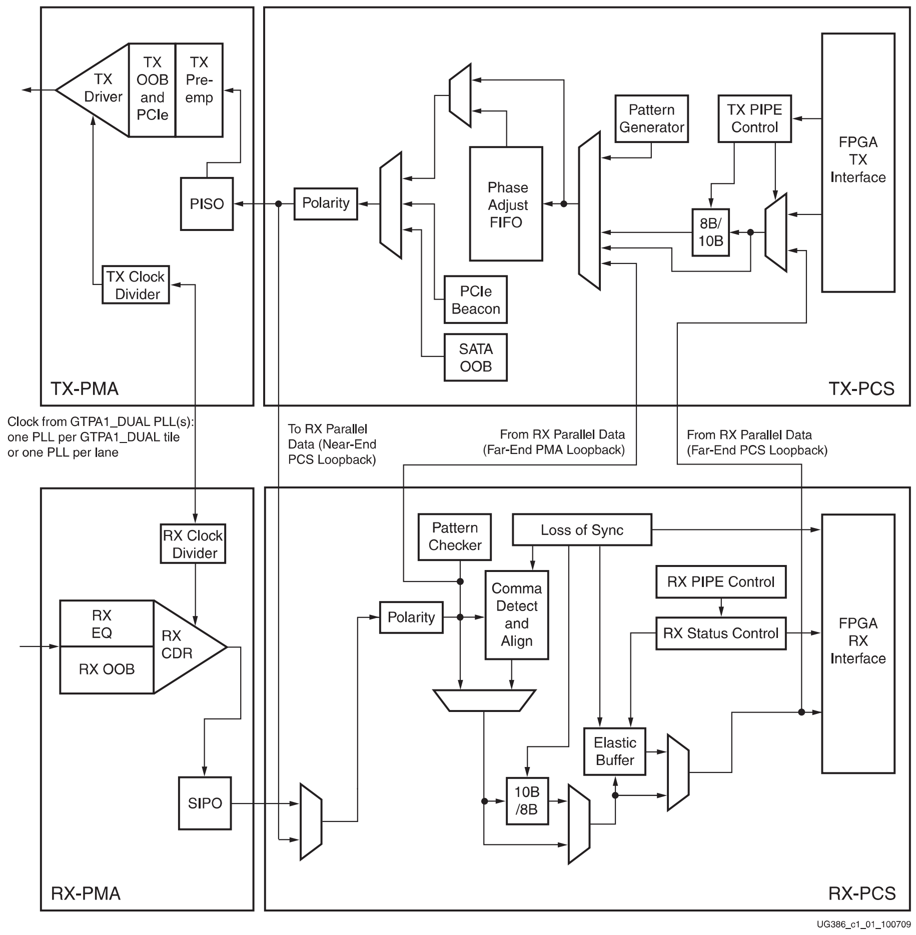


Figure B.2: Simplified Spartan-6 FPGA GTP transceiver block diagram

The GTP transceiver has dedicated phase-alignment circuits to match or adjust the phase difference between the PMA clock domain and the PCS clock domain, and will therefore be deterministic. The phase alignment should be performed every time the GTP is reset or after a power down sequence.

When using phase alignment there is no need to use the RX elastic buffer and therefore can be bypassed, this will lower the latency and create a deterministic RX datapath. Therefore the performance of the clock synchronization solution will increase, because there is less observation noise. However by bypassing the RX elastic buffer clock correction and channel bonding are not available. Therefore external logic is required to allow reliable communication. In [29] such a solution is presented. The creation of this external logic is beyond the scope of this research, and that is why the RX buffers are not bypassed in this design.

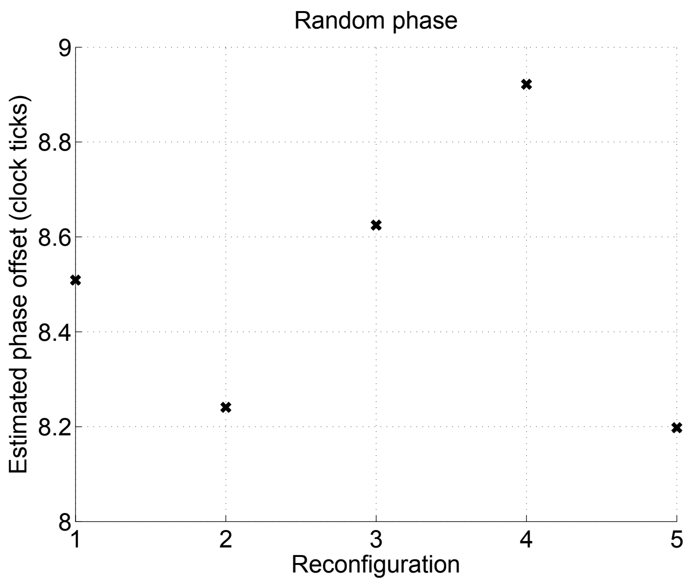


Figure B.3: After reconfiguring the FGPA a non-deterministic phase is introduced in the GTP transceiver. This random phase can be corrected by using the GTP transceivers dedicated phase-alignment circuits.

B.5 Communication latency

For proper clock synchronization using the Aurora 8b/10b protocol, constant latency is required. Without constant communication latency, the algorithms performance will drastically drop. To check if the communication latency is deterministic, the transmission and reception times where marked, while transmitting messages between two nodes. Using these markers the communication latency was calculated. Figure B.4 shows the communication latency between two nodes. The latency is either 0.976 μs or 0.96 μs, the difference between these two is exactly one user clock cycle (16 ns). This clock cycle is a result of jitter on the clock used to timestamp the reception and transmission times.

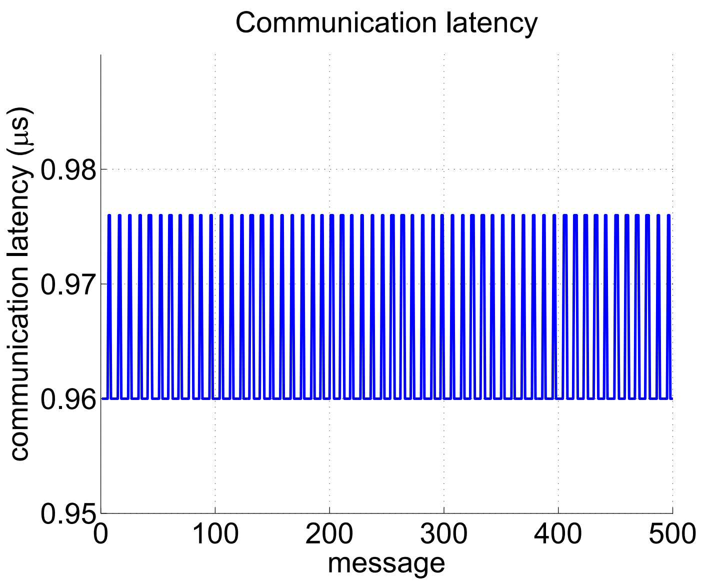


Figure B.4: Communication latency between two adjacent nodes, using a 2.5 GB/s Aurora 8b/10b network link