Evaluation of a Reconfigurable Computing Engine for Digital Communication Applications

Jonas Thor

Civilingenjörsprogrammet

Institutionen för Systemteknik
Avdelningen för Datorteknik

1999:175 • ISSN: 1402-1617 • ISRN: LTU-EX--99/175--SE

Abstract

This thesis work intends to evaluate a reconfigurable computing engine, a Peripheral Component Interface (PCI) based Field Programmable Gate Array (FPGA) card, for use in digital communication applications. This was done by implementing three designs; a flexible high performance data acquisition system, a basic Global Positioning System (GPS) signal generator and a multi-channel GPS digital receiver.

The conclusion is that the configurable computing engine provides an excellent platform for developing and prototyping digital communication applications.

Acknowledgments

This report is written as a final thesis for the academic degree Master of Science in Electrical Engineering within the field of signal processing at the Division of Signal Processing, Luleå University of Technology. The project was carried out at the Division of Computer Engineering at the same university. The project covers areas in digital design applied to signal processing applications.

I would like to thank Dennis M. Akos, the supervisor of the thesis project, who has supplied invaluable help during the completion of this thesis work

Luleå, May 1999. Jonas Thor

Contents

1 Introduction **1**

1.1 Background 1

1.2 Goal 1

1.3 Outline 2

2 The Global Positioning System **3**

2.1 GPS Overview 3

2.2 GPS SPS Signal Structure 4

2.2.1 Frequency and Modulation Format 4

2.2.2 C/A Code Generation and Properties 5

2.3 GPS Receivers 6

2.3.1 Front End 6

2.3.2 Digital Receiver Channel 8

2.3.3 Acquisition 8

2.3.4 Tracking 8

2.3.5 Measurements 10

2.3.6 Computational Requirements 11

3 Hardware and Development Tools **12**

3.1 Field Programmable Gate Arrays 12

3.1.1 The Xilinx XC4000X Architecture 13

3.2 WILD-ONE^{TM} Reconfigurable Computing Engine 17

3.2.1 WILD-ONE^{TM} Architecture 18

3.2.2 Application Design Flow 19

4 Data Acquisition Application **21**

4.1 The AD9057 ADC Evaluation Board 21

4.2 The Design 21

4.2.1 Two's Complement Converter 23

4.2.2 Bit Packer 23
4.2.3 Asynchronous FIFO 23
4.2.4 Decimation 23
4.2.5 Control Logic 24
4.2.6 Host Program 26
4.2.7 Testing the ADC Application 26
4.2.8 Further Work 26

5 GPS Signal Generator **27**
5.1 The Design 27
5.1.1 Carrier NCO and Sine Look-up Table 27
5.1.2 Code NCO 28
5.1.3 Code Generator 29
5.1.4 Testing the Signal Generator 29
5.2 Further Work 30
5.2.1 Gaussian Noise Generator 30
5.2.2 Doppler Generator 33
5.2.3 Amplitude Profile 35
5.2.4 Code and Data Generation 35
5.2.5 Adder Tree 35

6 Multi-channel GPS Digital Receiver **36**
6.1 The Mitel Semiconductor GP2010 front-end 36
6.1.1 Downconversion 36
6.1.2 Analog to Digital Conversion and Automatic Gain Control 37
6.2 FPGA Implementation of a Multi-channel Digital Receiver 38
6.2.1 Introduction 38
6.2.2 Using the Distributed Edge-Triggered RAM for Interleaving 38
6.2.3 The Top-Level 39
6.2.4 Carrier NCO 39
6.2.5 Sine and Cosine Look-up Tables 42
6.2.6 Code NCO 42
6.2.7 Code Generator 44
6.2.8 Downmixing and Despreading 44
6.2.9 Accumulators 44
6.2.10 Control Logic 45

6.3 Further work 47

7 Conclusions 48

Chapter 1

Introduction

1.1 Background

During the winter of 1998 a project course took place at Luleå University of Technology, division of Computer Engineering. The goal with this project was to develop a Peripheral Component Interface (PCI) board for data acquisition applications. The project was split into four parts and groups of students participated in the different parts. The different project parts were:

- Design of a PCI core for a Field Programmable Gate Array (FPGA).
- Design of the back-end application, interfacing an Analog to Digital Converter (ADC).
- Design of a Printed Circuit Board (PCB).
- Development of a Windows NT device driver.

It was learned that designing a PCB for the complex PCI bus is not an easy task and given the time frame of the project course the actual card was never built. Instead, after the end of the project course, an FPGA based PCI board, WILD-ONE$^{\text{TM}}$, was purchased from Annapolis Microsystems Inc. The board is called a reconfigurable computing engine, since the FPGAs on the board can be reconfigured for different applications. The intention was that this board should be used to implement a data acquisition system. The implementation of the data acquisition system came to be a part of this thesis work at Luleå University of Technology, division of Computer Engineering during the winter and spring of 1999.

1.2 Goal

The goal with the thesis work is defined by its title - "Evaluation of a Reconfigurable Computing Engine for Digital Communication Applications". The WILD-ONE$^{\text{TM}}$ board was to be evaluated for digital communication applications for research as well as educational purposes. More precisely the work came to have three different parts:

1. Implementation of a flexible high performance data acquisition system.

2. Design of a basic Global Positioning System (GPS) signal generator.
3. Design of a multi-channel GPS digital receiver.

The three designs are targeted to the WILD-ONE^{TM} board. The reason why GPS applications were chosen is that is an interesting research area with a multitude of applications.

The conclusion is that the WILD-ONE^{TM} board provides an excellent developing and prototyping environment for our purposes, research as well as education. The three different parts of the thesis work turned out to be successful. In parallel with the thesis work we have also developed labs for an advanced undergraduate course in digital design using the WILD-ONE^{TM} board.

1.3 Outline

Chapter 2 gives an introduction to GPS and chapter 3 introduces the hardware and development tools used during the thesis work. Following the actual work done is presented. Chapter 4 describes the data acquisition implementation, chapter 5 describes the design of a GPS signal generator and chapter 6 deals with the design of a multi-channel GPS digital. Finally chapter 7 concludes the thesis.

Chapter 2

The Global Positioning System

In this chapter a brief overview of the Global Positioning System (GPS) is given. Most of the material for this chapter has been gathered from [1, 2, 3, 4].

2.1 GPS Overview

The Global Positioning System (GPS) is a satellite based navigation system owned by the Government of the United States. The U.S. Air Force (USAF) manages and controls the system. Even though GPS initially was intended for military use, the civilian community has found increasingly use of the system for a variety of applications. GPS provides two services: the Precise Positioning Service (PPS) and the Standard Positioning Service (SPS). SPS is available for the civilian user, but PPS is primarily intended for U.S. and allied military use.

GPS consists of three segments [1]: the control segment, the space segment and the user segment. The control segment tracks and maintains the satellites. The space segment consist of 24 satellites that continuously transmit navigation data at two frequencies, L1 (1575.42 MHz) and L2 (1227.6 MHz). The signals from the satellites are separated with Code Division Multiple Access (CDMA) technique [5]. Each satellite modulate L1 and L2 carriers with for the satellite unique Pseudo Random (PRN) codes. Because the cross correlations between the different PRN codes are low, the signals from the satellites do not significantly interfere with each other.

The user receiver equipment has three primary tasks:

1. **Acquisition**. The signal acquisition process consists of a three-dimensional search in time (code phase), frequency and satellite-specific PRN code. There are uncertainty in the carrier frequency due to Doppler effects and local oscillators errors.

2. **Tracking.** After the acquisition process is accomplished the receiver enters the tracking mode. A delayed locked loop (DLL) is typically used to track the code phase (or frequency) and a phase locked loop (PLL) or a frequency locked loop (FLL) is used to track the carrier phase or frequency, respectively.

3. **Data decoding and position solution**. When both tracking loops are in lock it is possible to decode the navigation data and determine the receiver's position, velocity and time (PVT). By synchronizing the receiver generated PRN code with the satellite

transmitted PRN code it is possible to determine the transit time of the signal and thereby determine the satellite-to-user range by scaling with the speed of light. Four satellite measurements are required to determine the three unknown x, y, and z user coordinates with a forth possible unknown which is the local receiver clock offset.

2.2 GPS SPS Signal Structure

The L2 carrier is spread in frequency by a precision code called P(Y) code. The L1 carrier is modulated by two ranging codes; the P(Y) code and a Coarse/Acquisition (C/A) code. The P(Y) code is available for PPS users, while the C/A code is available for PPS and SPS users. Only the L1 carrier and the C/A codes will be described here.

2.2.1 Frequency and Modulation Format

A simplified model of the L1 SPS modulator configuration is shown in figure 2.1. The L1 carrier is modulated by the modulo-2 sum of the C/A code and the navigation data. The modulo-2 sum is the same operation as the exclusive-or operation. The nominal chipping rate of the C/A code is 1.023 MHz and one code period is 1023 chips long or 1 ms. It is important to notice that the transitions of data bits are synchronized to code periods, but at a much slower rate, 50 bps.

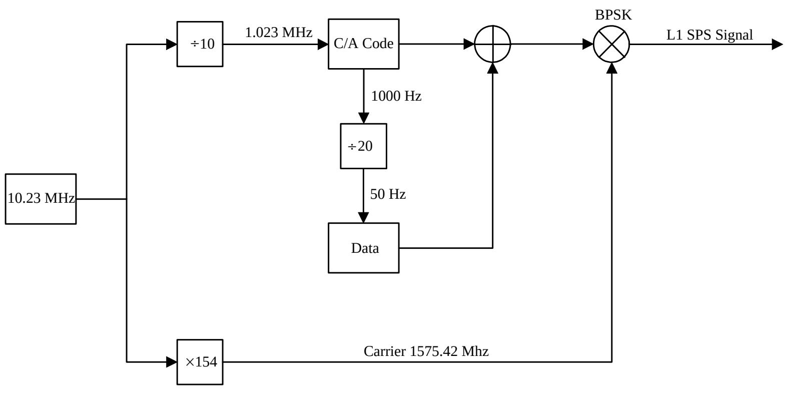

Figure 2.1: L1 SPS modulation format.

Before biphase shift Keying (BPSK) modulation is employed, the above described modulo-2 sum with values $\{0, 1\}$, are mapped to $\{-1, 1\}$. The transmitted L1 SPS signal is defined by

$$L_i\left(\omega_0 t\right) = AC_i\left(t\right) D_i\left(t\right) \cos\left(\omega_0 t\right), \tag{2.1}$$

where:	L_i	broadcast signal from the i^{th} satellite
	i	indicates satellite number
	A	signal amplitude
	C_i	C/A code for the i^{th} satellite (± 1)
	D	navigation data for the i^{th} satellite (± 1)
	ω_0	L1 radian freqeuncy ($2\pi 1.57542 \cdot 10^9$ rad/s).

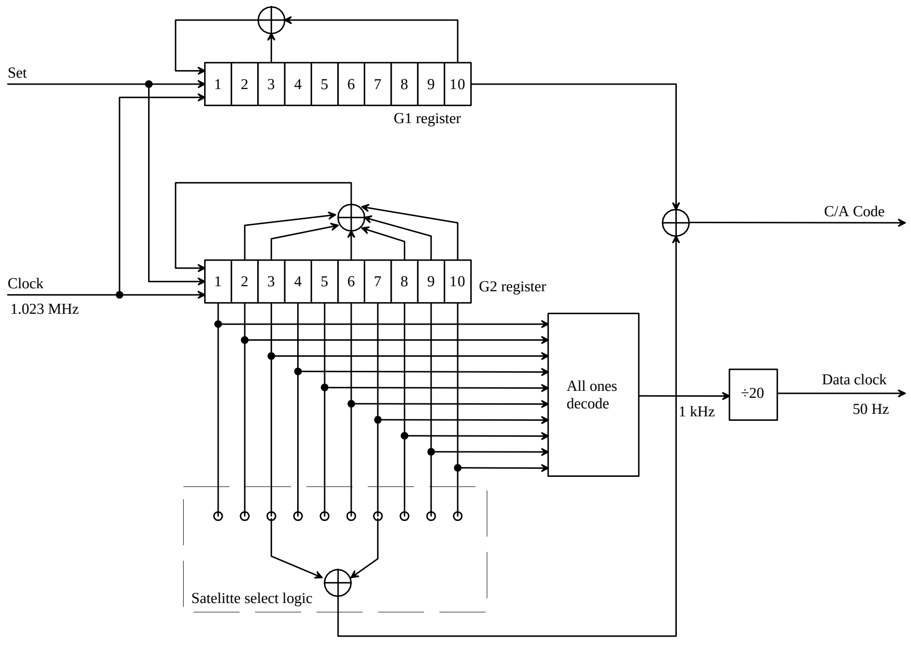

Figure 2.2: C/A Code Generator

2.2.2 C/A Code Generation and Properties

Each satellite uses a unique C/A code to implement the CDMA technique. The C/A codes belong to the family of Gold pseudorandom noise (PRN) codes. Gold codes are used because of their autocorrelation and cross-correlation properties. The architecture of a C/A code generator is shown in figure 2.2. Two 10-bit Linear Feedback Shift Registers (LFSR), G1 and G2, generate maximum length PRN codes with a length of $2^{10} - 1 = 1023$ bits. Initially, both G1 and G2 are all set to ones, the all-zero state is illegal. The G1 and G2 LFSRs feedback taps are defined by the generator polynomials

$$\begin{aligned} G1\left(X\right) &= 1 + X^3 + X^{10} \\ G2\left(X\right) &= 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}. \end{aligned} \tag{2.2}$$

The satellite specific C/A code is generated by taking the exclusive-or output of the G1 LFSR and a delayed version of the output from the G2 LFSR. The delayed effect for the G2 LFSR output is obtained by taking the exclusive-or of two selected stages from the G2 LFSR. This will work since adding a phase-shifted versions of a PRN sequence to itself will shift the phase of the code, while not changing the code [1]. The GPS code phase assignments are documented in [3].

The autocorrelation function of a C/A code is

$$R_{CA}\left(\tau\right) = \int_{-\infty}^{\infty} C_i\left(t\right) C_i\left(t + \tau\right) dt, \tag{2.3}$$

where C_i is the C/A code for the i^{th} satellite and τ is phase of the time shift. A normalized and simplified autocorrelation function of a typical C/A code is shown in figure 2.3. The correlation peaks repeats every code period and the correlation interval is two chips. As

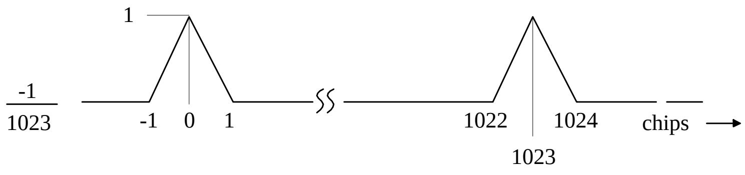

Figure 2.3: Simplified and normalized C/A autocorrelation function.

mentioned this is a simplified model of the C/A code autocorrelation function since there are shifts which produce non-zero values between correlation peaks. The properties of the autocorrelation function is used to synchronize the receiver replicated code with the received code. It is important that the cross-correlation of any two C/A codes are minimal for any phase or Doppler shift over the entire code period. The ideal cross-correlation is defined by:

$$R_{ij}(\tau) = \int_{-\infty}^{\infty} C_i(t)\, C_j(t+\tau)\, dt = 0 \tag{2.4}$$

where C_i is the C/A code for the i^{th} satellite and C_j C/A code for the j^{th} satellite where $i \neq j$. Due to the fact that multiple satellites signal are received with different delays and Doppler offsets the cross-correlation properties of C/A codes are not ideal, and may under certain conditions cause false acquisitions. For further information about C/A code properties refer to [1, 2].

2.3 GPS Receivers

2.3.1 Front End

A generic GPS receiver is illustrated in figure 2.4. The front end includes the blocks needed to provide the digital intermediate frequency (IF).

An antenna receives the radio frequency (RF) signals from all satellites in view. A low noise pre-amplifier amplifies the RF signals. There may be a bandpass filter to reject out-of-band RF interference [1]. The amplified (and filtered) RF signal is downconverted to an IF in the downconverter using signal mixing from local oscillators (LOs). There may be only one downconversion stage but two or more downconversion stages are also common. Recent research [8] also investigates direct digitization of the RF signal by bandpass sampling. With this approach no local oscillator is needed but the analog to digital converter (ADC) must have a high analog input bandwidth.

The analog IF output from the downconverter is sampled at a rate that is determined by the bandwidth of the analog IF. Depending on how the RF signal is filtered the sampling rate will vary from different receivers. However sampling rates of 4-10 MHz are typical. A simulated frequency spectrum of a typical C/A code is shown in figure 2.5. The width of the main lobe is 2.046 MHz and thus the sampling rate should be at least 4.048 MHz for the main lobe signal to be processed, assuming IF sampling.

GPS receivers require a small number of bits when sampling. 1-bit sampling is common for low-cost commercial receivers while 1.5-bits to 3-bits are common for high-end receivers [4].

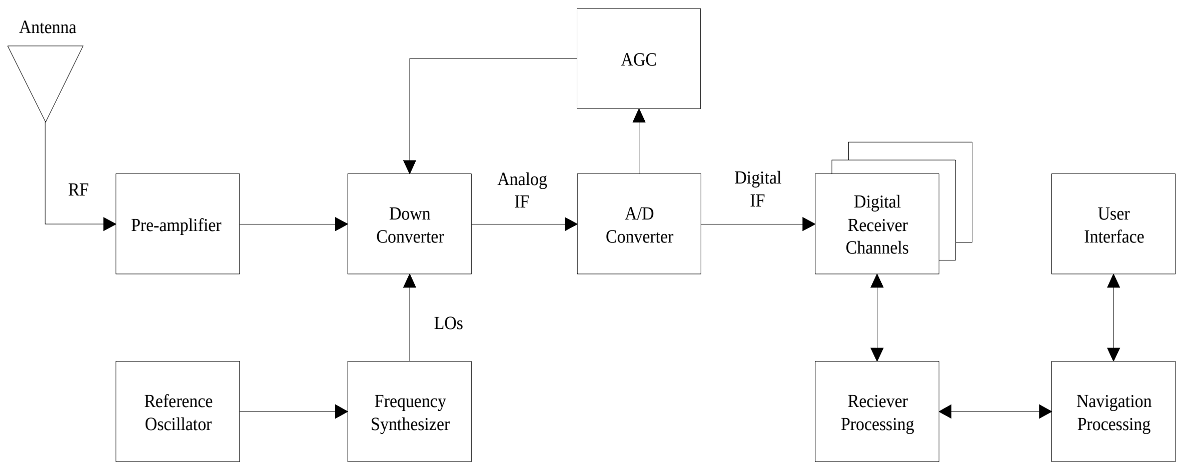

Figure 2.4: Generic GPS receiver block diagram (derived from [1])

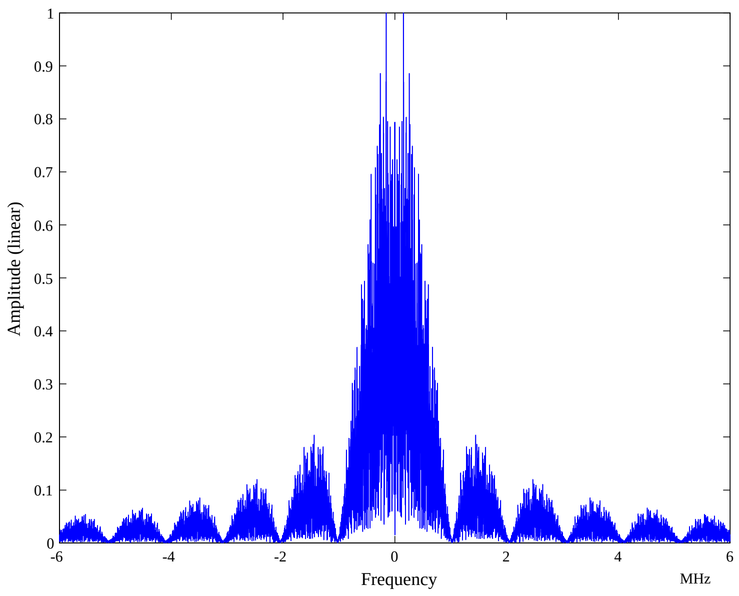

Figure 2.5: C/A code frequency spectrum.

The signal degradation of the finite bit quantization is described in [2]. The degradation does not only depend on how many bits are used, but also on the analog IF bandwidth and the ratio of the maximum ADC threshold to the RMS noise level. For the case of multi-bit sampling (more than 1-bit) automatic gain control (AGC) is needed to match the dynamic range of the sampled data to the dynamic range of the ADC.

2.3.2 Digital Receiver Channel

An FPGA implementation of a digital receiver channel will be explained in more detail in chapter 6.

A generic digital receiver channel is shown in figure 2.6. It is common for commercial receivers today have 12 or more separate digital receiver channels. The input to the receiver is the digital IF from the front-end. The digital IF is downconverted to baseband by multiplying with sin and cos. The result of the downconversion are the inphase (I) and the quadrature (Q) components. A carrier Numerically Controlled Oscillator (NCO) and lookup tables for the sinusoids generates sin and cos at desired frequency. Next the code removal is performed by multiplying the I and Q with replicated codes. For tracking purposes three versions of the code is provided, Early, Prompt and Late, each separated 1/2 chip apart. The code frequency is also controlled by an NCO. After the code removal the result is accumulated for usually one code period, 1 ms. The objective of the GPS receiver is to keep the prompt replica code inphase with the code in the received signal. When this is the case the output of the prompt accumulator is at maximum, i.e. the correlation is high.

2.3.3 Acquisition

Initially, when the receiver is powered up, the code phase and the carrier frequency are not known. Also not all 24 satellites are in view so initially the receiver must perform acquisition - search for satellites present in the received signal. For a stationary receiver it is common to assume a ± 10 kHz Doppler offset on the carrier. The acquisition process for each code period is usually stepped in the Doppler range in 500 Hz frequency increments. Choosing 500 Hz steps is a compromise in accuracy and speed. If the step was 1 kHz there would be less frequency bins to test, but with the possibility that the residual carrier of the I and Q components can cancel out the correlation, since 1 kHz corresponds to the code period of 1 ms. The code phase search may be done by searching over the code phases in 1/2 chip increments. This means that 2046 code phases are tested in each Doppler bin. When the correlation power exceeds some threshold the receiver channel starts tracking the satellite.

For further information about GPS acquisition refer to [6, 7].

2.3.4 Tracking

The tracking loops are usually closed in a microprocessor, which controls the code and carrier NCOs. The code loop is a Delayed Locked Loop (DLL) and the carrier loop is either a Phase Locked Loop (PLL) or a Frequency Locked Loop (FLL). Figure 2.7 shows a generic configuration for GPS tracking loops. The theory of tracking is described in [1, 2].

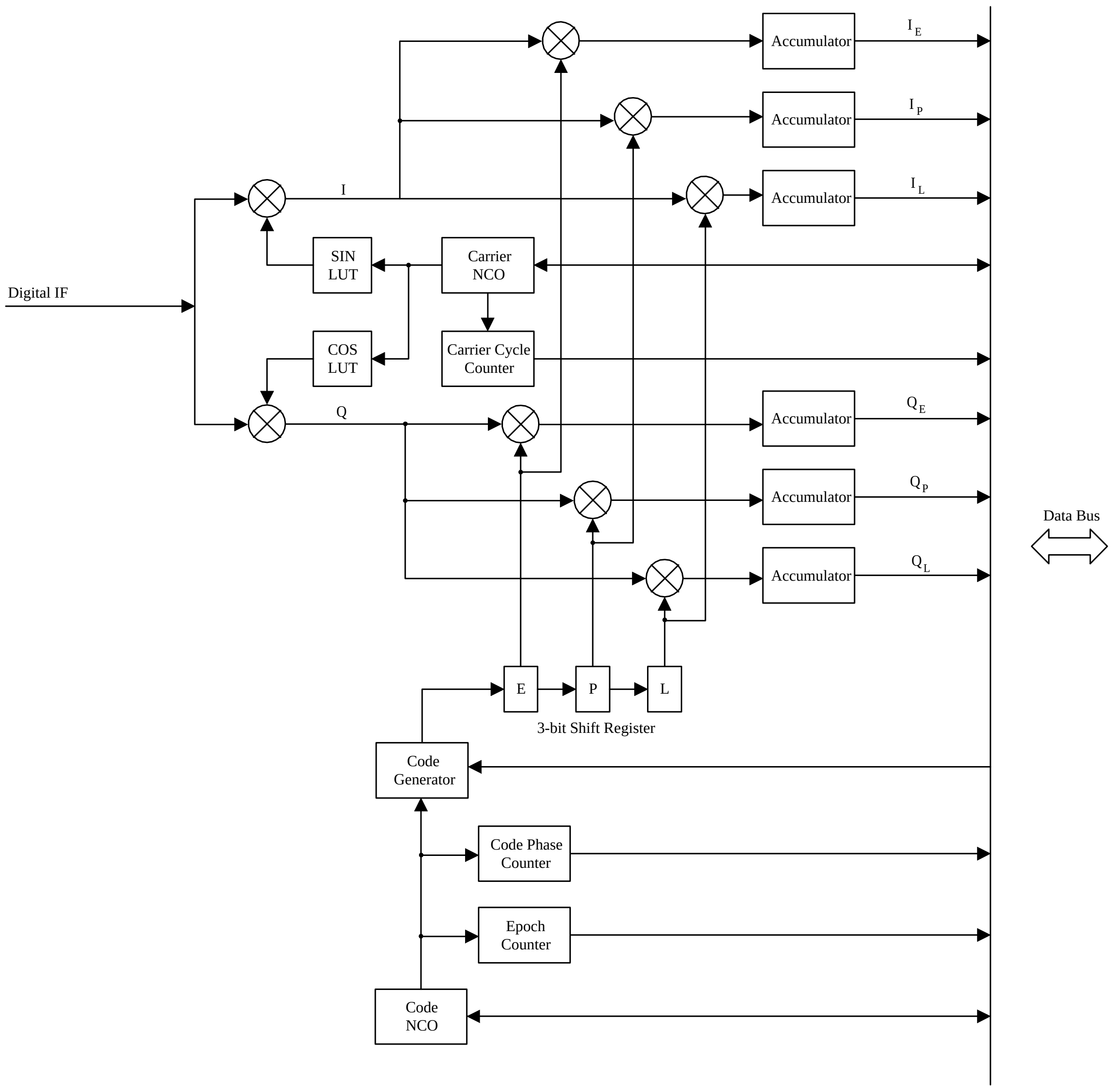

Figure 2.6: Digital receiver channel block diagram.

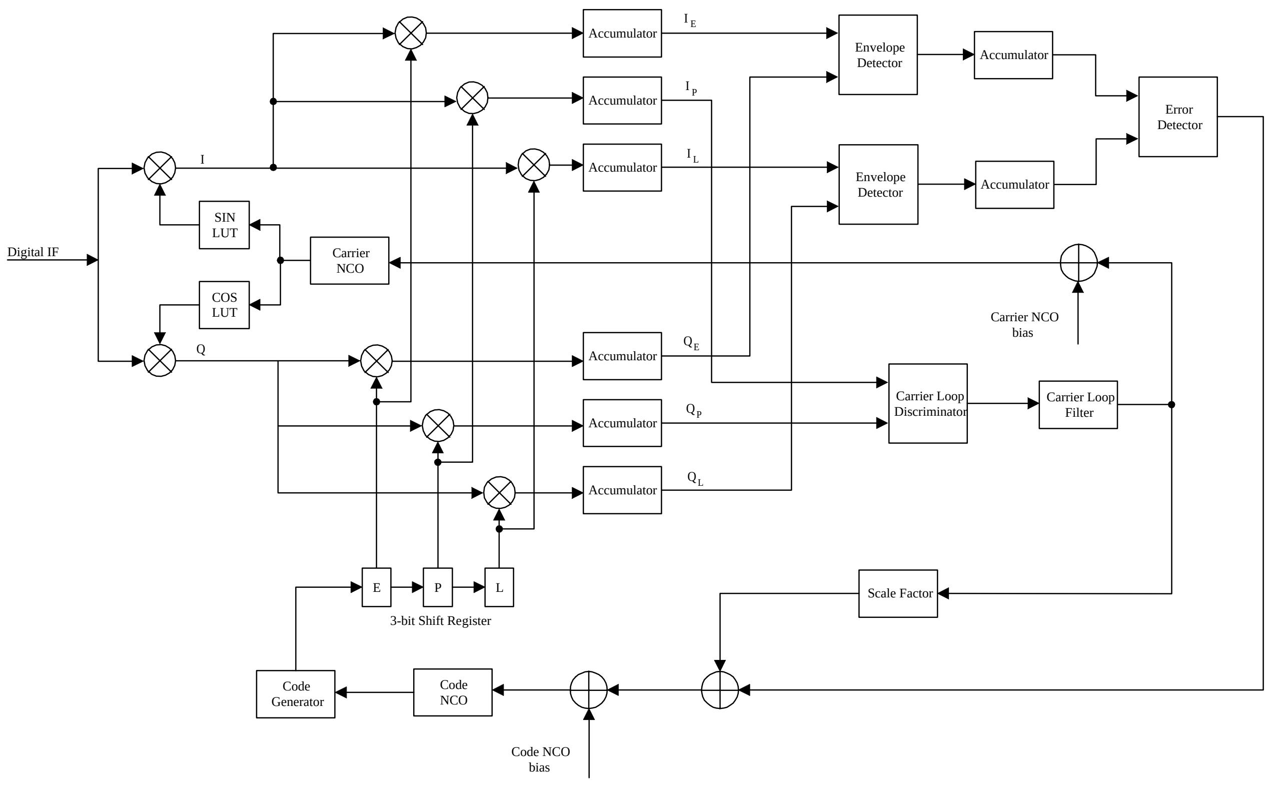

Figure 2.7: Generic GPS tracking channel.

2.3.5 Measurements

As previously mentioned position, time and velocity can be determined by tracking GPS satellites. The satellite data contains the satellites position and the Time Of Transmission (TOT). The receiver can determine the TOT by reading the states of the epoch counter, code phase counter and the code NCO (see figure 2.6) at the TOT. Since the receiver clock is not synchronized to the satellite clock an exact TOT can not be determined directly. The *pseudorange* measurement is defined by

$$PR = c\,(\mathrm{TOR} - \mathrm{TOT}) = R + c \cdot b \tag{2.5}$$

where c is the speed of light, TOT is the Time Of Reception, R is the satellite to user range and b is the receiver clock offset. The receiver position in three dimensions and clock offset bias may be determined by tracking four satellites.

The *delta range* and *accumulated delta range* measurements are given by keeping track of the satellite Doppler shifts on the carrier. The the line-of-sight (LOS) velocity may be approximated by [4]

$$v_{los} \approx \lambda_0 f_D \tag{2.6}$$

where λ_0 is the wave length of the transmitted carrier and f_D is the measured Doppler shift. The average LOS velocity, referred to as delta range, may obtained by counting the number of Doppler cycles under a short period of time and then scaling by the wave length and dividing by the duration of the integration interval.

The accumulated delta range is obtained by continuously accumulating the Doppler cycle counts. This effectively means that equation 2.6 is integrated, hence a measurement of the receivers relative change in placement to the satellite is given.

2.3.6 Computational Requirements

Most commercial receivers today are digital. The front-end, is of course analog, but the receiver processing is done digitally. A typical GPS receiver chipset consists of one IC for the analog front-end. This chip performs the filtering and downconversion to IF (or baseband). The output from the analog IC is 1-bit to 3-bit samples at a rate of 4-10 MHz. The second IC is a multi-channel digital receiver as illustrated in figure 2.6. Because of the relative high data rate the down conversion and despreading is performed in an Application Specific Integrated Circuit (ASIC) and not a programmable processor. However software radio approaches to GPS receivers have been suggested [9]. The output rate from the accumulators seen in figure 2.6 is at a much lower rate than the sampling frequency. If the accumulation is performed over one code period the rate is 1 kHz. It may even be lower, 200 - 500 Hz, if the accumulation is allowed to be more than one code period. For these lower rates the data can be processed by programmable processors. The tracking loops can be closed in a programmable processor, the receiver measurements decoded, and PVT solutions are computed in the processor.

Chapter 3

Hardware and Development Tools

In this chapter the hardware and software tools used in the thesis work will be presented. All designs have been targeted towards a Peripheral Component Interface (PCI) based reconfigurable computing Engine board, WILD-ONE^{TM}, developed by Annapolis Micro Systems Inc. A short introduction to Field Programmable Gate Arrays (FPGA) will be given, with the architecture Xilinx XC4000X FPGA explored in more detail. Lastly the WILD-ONE^{TM} board and the development tools will be covered.

3.1 Field Programmable Gate Arrays

When implementing a digital system the designer has to make a choice on what type of digital devices that should be used. Factors that influence this choice are cost, flexibility, performance and time-to-market. Microprocessor are the ideal choice if they are they provide adequate computation requirements. Processors such as micro controllers and DSPs are relatively cheap and very flexible to use. Another alternative is to use dedicated chip sets if they exist for the specific application. These chip sets are cheap but they may lack flexibility. ASICs provide the best performance in terms of speed and power consumption, but the time-to-market is long and because of the cost only high volume applications can be considered. Programmable Logic Devices (PLD) is another alternative and can, in many cases, be the best alternative. PLDs are more flexible than ASICs and usually provide better performance than microprocessors. However one should note that the best choice strongly depends on the application to be developed.

One type of PLDs is the FPGA. In 1985 Xilinx Corporation Introduced XC2000 Series, the first family of FPGAs [10]. The XC2000 device came to serve as a prototype for the following generations of FPGAs.

An FPGA consists of an array of Configurable Logic Blocks[1] (CLB), surrounded by programmable IO Blocks (IOB), and connected with programmable interconnect as illustrated in figure 3.1. The end user configures the FPGA for a particular design implementation by programming the FPGA "in the field". The number of CLBs in an FPGA varies from 64 to several thousands [11]. The performance of an FPGA is highly dependant on the functionality of the CLBs and the routability of the programmable interconnect, i.e. the ability to interconnect CLBs and IOBs with as small delay as possible.

[1]Here the Xilinx terminology has been adopted, different vendors have different terminology.

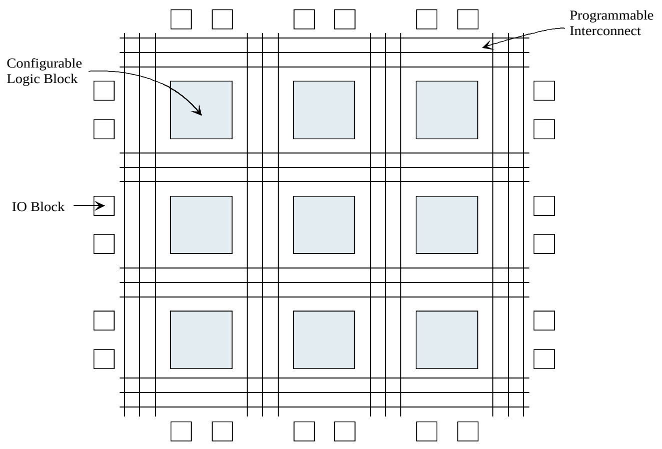

Figure 3.1: Generic FPGA structure.

Depending on the architecture of the CLBs FPGAs can be divided in two classes of FPGAs, fine grained and coarse grained. The CLBs in a fine grained architecture simply have less logic in a CLB compared to a coarse grained architecture. Another distinction between different FPGA architectures is how the user-programmable switches are implemented. These can be either SRAM based or antifuses. An SRAM cell can control a pass transistor or the select line of a multiplexer in order to implement a single user programmable switch. The SRAM based FPGAs can be reprogrammed unlimited times, unlike FPGAs where the programmable switches are implemented with antifuse technology [12]. Antifuse FPGAs are one time programmable.

3.1.1 The Xilinx XC4000X Architecture

In 1991 Xilinx introduced their third generation FPGA, XC4000 [13] a SRAM-based coarse grained FPGA. The device has been improved over the years. Architectural improvements and progresses in process technology has benefited the XC4000 device. The latest in the XC4000 family is the XC4000X series. Since this device was used during the research it will be further described here. For a full description of Xilinx FPGAs refer to [14].

As of May 1999 the smallest device in the XC4000X series is XC4005XL with 100 CLBs, the largest is XC40250XV with 8464 CLBs. Xilinx also offers other FPGA families such as XC3000, XC5000, Spartan and the new Virtex family [10].

Configurable Logic Block

Figure 3.2 illustrates the XC4000 CLB. There are two 4-input function generators, F and G, and one 3-input function generator, H, implemented as memory look-up tables (LUT). A K-input LUT can implement any Boolean function of K variables. As illustrated in figure 3.3 and listed below a CLB can implement any of the following functions [14]:

1. any two functions of up to four variables each and any third functions of up to three variables (one function output must be registered since there are only two unregistered function outputs from a CLB)

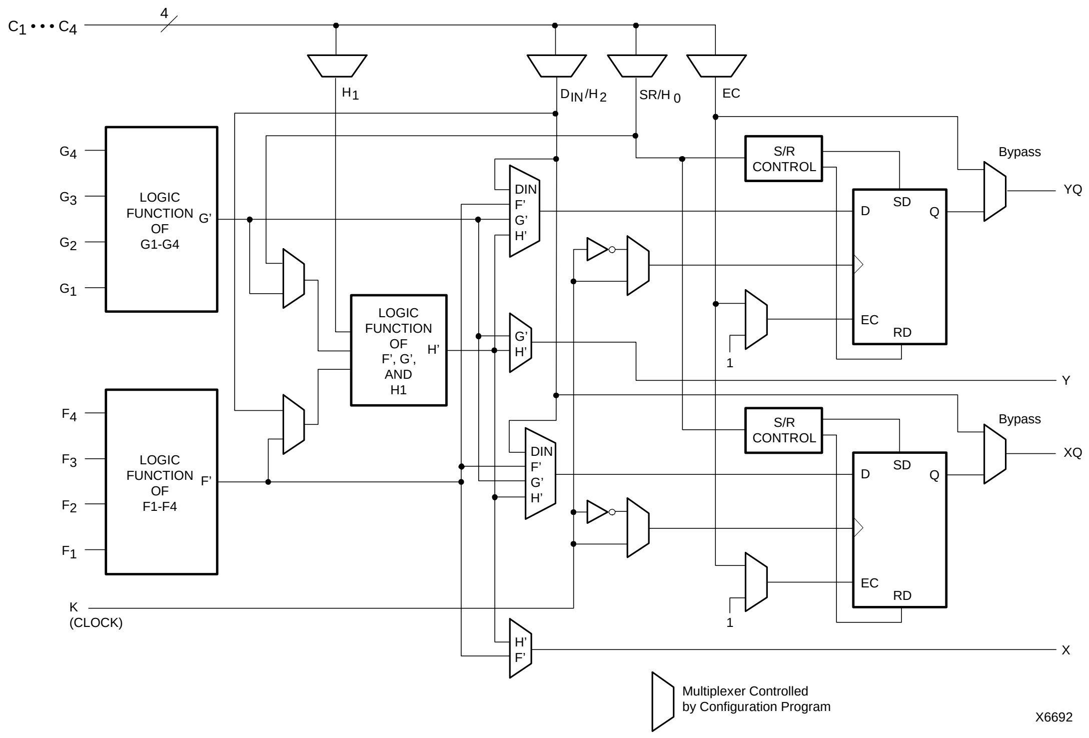

Figure 3.2: The Xilinx XC4000 familiy CLB [14].

2. any single function of five variables
3. any function of four variables together with some functions of six variables
4. some functions of up to nine variables.

Each CLB contains two storage elements that can be used to store the output of the function generators. It is also possible to use the storage elements and the function generators independently. Each storage element can be configured either as an positive or negative edge triggered D flip-flop or as a latch. The storage elements have a common clock (K), common clock enable (CE) and a common asynchronous set/reset input (SR). The set/reset state can be independently configured for each storage element. After the device is configured the flip-flops and latches are initialized to a known state determined by the set/reset state. Apart from the local reset, SR, there is also a Global/Reset net with dedicated routing resources.

The function generators may also be configured as level-sensitive, edge-triggered, and dual-port edge triggered RAM. A single CLB can be configured as either a 16×2, 32×1 or 16×1 bit array. Figure 3.4 shows CLBs configured as 16×2 edge-triggered single-port RAM and 16×1 edge-triggered dual-port RAM.

To improve arithmetic operation each CLB also contains dedicated programmable carry logic for the F and G function generators, as shown in figure 3.5. The carry in and carry out signals have direct connections to adjacent CLBs for highest speed.

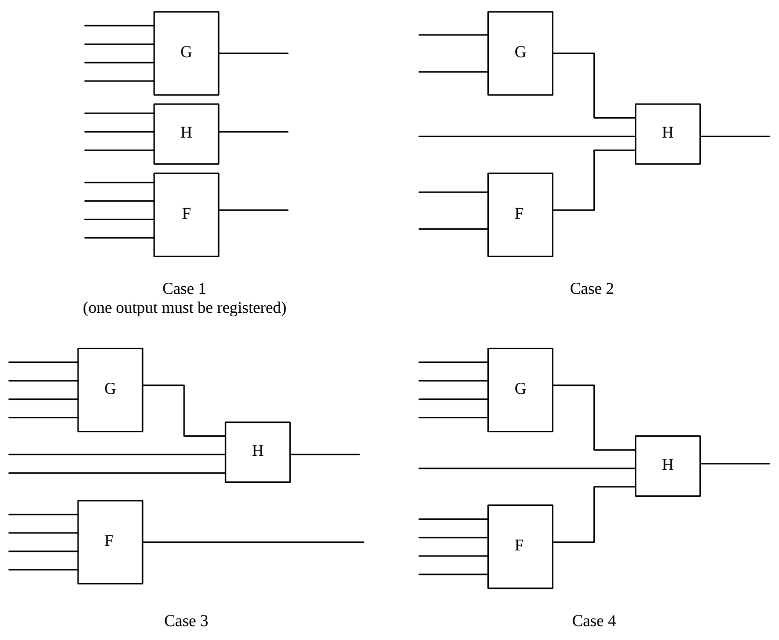

Figure 3.3: Possible function generator configurations.

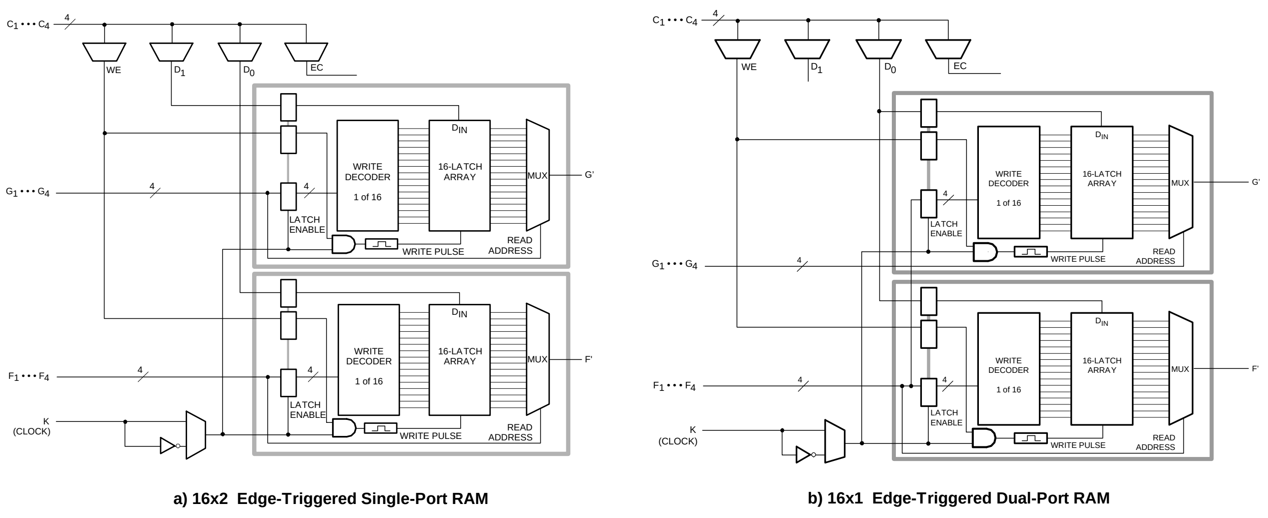

Figure 3.4: Two possible RAM configurations of a CLB [14].

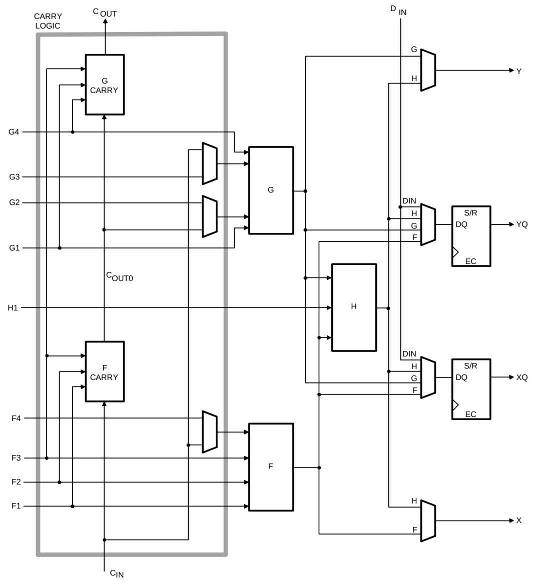

Figure 3.5: XC4000X carry logic [14].

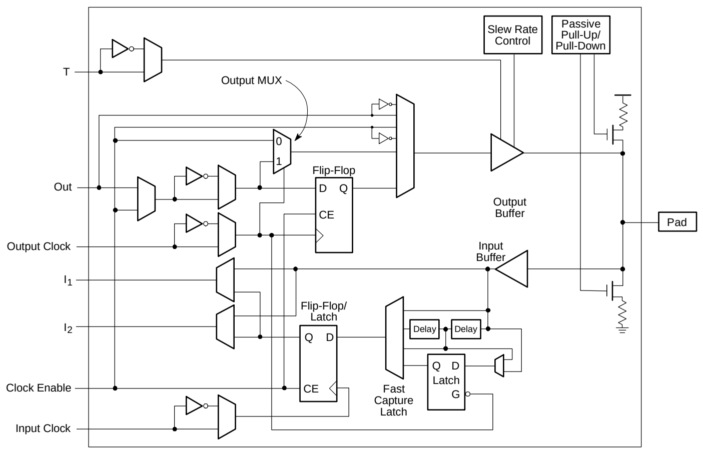

Figure 3.6: XC4000X IO block [14].

Input/Output Blocks (IOB)

As illustrated in figure 3.6, the IOBs have several configuration options. The input may be taken from I_1 or I_2 and can be passed through the storage element or not. The storage elements, as in the CLBs, can be configured as latches or flip-flops and have a common clock enable (CE). The registered input has an optional delay to assure zero hold time. The "Fast Capture Latch" may be used to allow fast capture of the input data. All inputs are 5V tolerant.

Output signals may be inverted before they are routed directly to the pad or to the pad via an edge triggered flip-flop. The output may be tristated locally or globally for all IOBs on the FPGA. The slew rate control allows the user to minimize ground bounce.

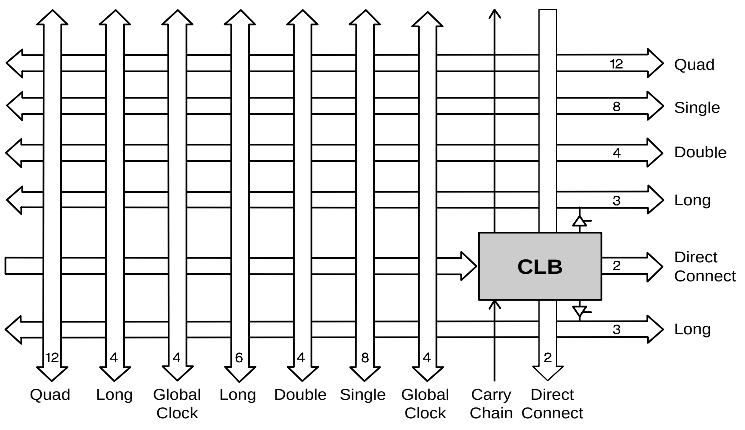

Figure 3.7: XC4000X routing diagram [14].

Three state buffers

Two tristate buffers are placed outside each CLB, see figure 3.7. The tristate buffers may be used to drive the horizontal longlines and can be used to drive bi-directional buses or to multiplex data. The tristate buffers can also be configured to implement wired-AND and wired OR-AND functions.

Wide Edge Decoders

Since high fan-in functions may be inefficient to implement with LUTs, the XC4000X devices have four wide edge decoders located on each edge of the device.

Programmable Interconnect

The internal connections consist of metal segments with programmable switching points and switching matrices. The XC4000 series has a hierachial segmented routing architecture. Five different type of connection segments are named after he relative length of a segment; direct connect, single-length and double-length lines, quad and octal lines and long lines. A high level diagram of the routing for a CLB is shown in figure 3.7.

There are eight global nets primarily intended for clock distribution. The XC4000 series also have additional routing around the IOBs forming a ring, called VersaRing.

3.2 WILD-ONE^{TM} Reconfigurable Computing Engine

The WILD-ONE^{TM}[15] card is the smallest member of the WILDFIRE^{TM} [16] reconfigurable computing engines, provided by Annapolis Micro Systems, Inc [17]. The boards are intended to speed up computational intense application by employing parallelism and reconfiguration. The Processing Elements (PE) on the boards are Xilinx XC4000X series FPGAs.

3.2.1 WILD-ONE^{TM} Architecture

The WILD-ONE^{TM} card communicates with a host computer via a Peripheral Component Interface (PCI) bus. The PCI bus is a high bandwidth 32-bit bus capable of transferring data at a rate of 33 MHz, which gives a peak bandwidth of 132 MBps. A PCI device can be either a target or initiator. An initiator device, or bus master, can take control of the PCI bus and does not have to rely on the host CPU. The WILD-ONE^{TM} is a bus master capable device capable of Direct Memory Transfers (DMA). The PCI chip is a Xilinx FPGA, but the configuration of this FPGA is not available for the user. A PROM stores the configuration for the PCI FPGA.

A block diagram of the WILD-ONE^{TM} is shown in figure 3.8. There are two Processing Elements (PE) on the card, PE0 and PE1. The PEs are programmed over the PCI bus by the aid of software libraries.

An optional mezzanine card can be connected to each PE. These cards can hold SRAM, DRAM, DSPs, FPGAs, microprocessor or ASICs. The board used for the thesis work has the configuration indicated in table 3.1.

PE0	XC4036XLA
PE1	XC4013XLA
Mezzanine Card 0	0.5 MB SRAM
Mezzanine Card 1	0.5 MB SRAM

Table 3.1: Card Configuration

Host Communication

The PEs may communicate with the host by using:

- the dual-port memory cards if available. A dual-port memory controller (DPMC) for each memory controls access to its respective the memory.
- the 512 by 36-bit bi-directional First In First Out (FIFO) memories.
- interrupt interfaces to both PEs.

IO Ports

There are up to four 36-bit bi-directional ports that may be used to communicate with external devices. One of these ports is the SIMD port with a direct connection to PE0. If an external IO card is used the other three ports may be accessed.

Inter PE Communication

There are a number of connections between PE0 and PE1. The DirectData bus is a 36-bit bi-directional bus between PE0 and PE1. The PE_Right systolic bus also provides a connection between the two PEs as well as it is present on the external IO connector.

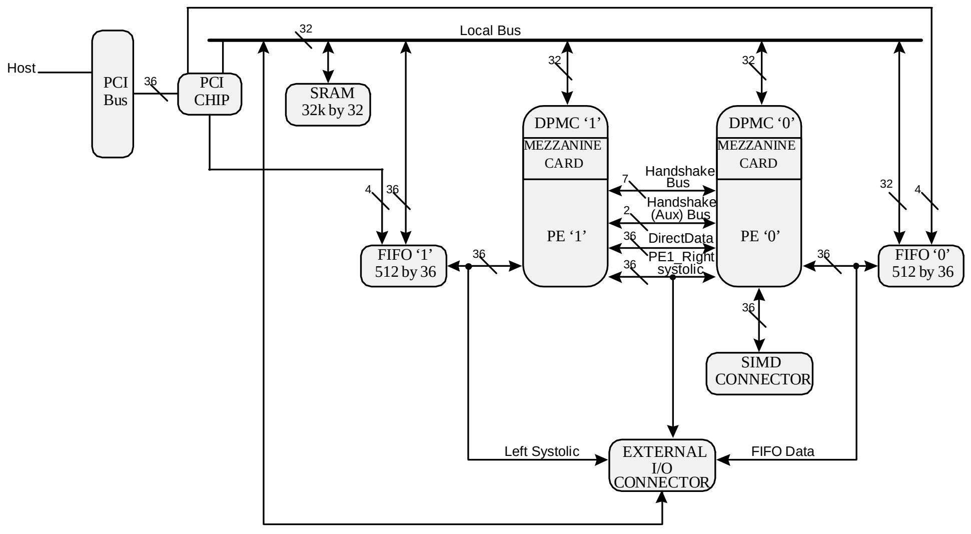

Figure 3.8: Block diagram of WILD-ONE^{TM} [15].

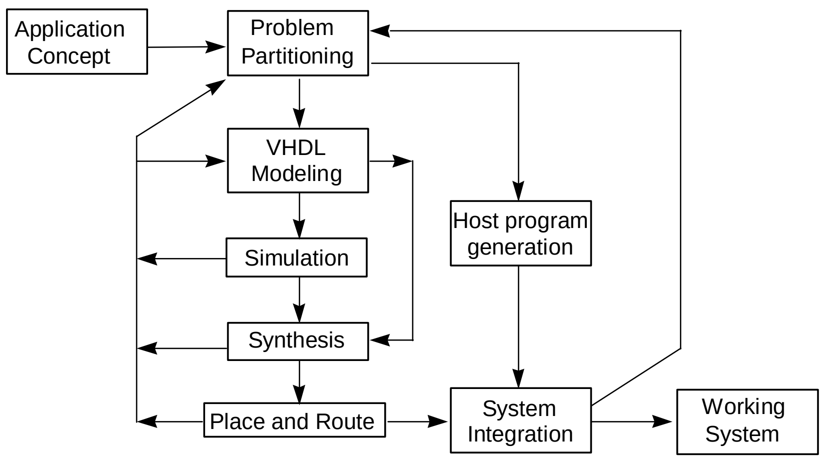

Figure 3.9: Application design flow [15].

The handshaking and auxiliary signals provides the user with another 9-bits for inter PE communication.

3.2.2 Application Design Flow

Figure 3.9 shows a typical application design flow for the WILD-ONE^{TM} system. First the application must be understood and algorithms must be developed that performs the desired tasks. The algorithms are then partitioned between PEs and the host processor. The hardware description language VHDL is used to model the behavior of the complete system. Behavioral VHDL libraries are provided with the WILD-ONE^{TM} development tools. The user only has to model the internal behavior of the PEs, referred to as Logic Cores. The other parts of the board are modeled in the provided VHDL libraries. In order to verify the functionality of the system ModelSim Elite Edition [18] , a VHDL and Verilog simulator, is used. The VHDL model of the PEs are then synthesized with Synplify [19]. The output from Synplify is a netlist that is passed to the Xilinx implementation tools. The implementation tools performs a place and route and then generates a binary stream that is used to configure the FPGA.

The WILD-ONE^{TM} development tools also contain C libraries for interfacing the board to the host. Application programs are compiled with Microsoft Visual C++ [20].

Chapter 4

Data Acquisition Application

The first application developed for the WILD-ONE^{TM} board was a data acquisition system. An 8-bit Analog to Digital Converter (ADC) evaluation board from Analog Devices is connected to the WILD-ONE^{TM} via a cable and the collected data is transferred to the host by using DMA.

The hardware design is a rather simple, but since two asynchronous clock domains are interfaced careful attention must be given to that interface. The main part of the design was modeled in VHDL, but one component, an asynchronous FIFO was created with schematic entry using the Xilinx Foundation software. The reason why schematic entry was used for creating the FIFO is simply that it is easier to floorplan designs using schematic entry than it is using VHDL entry. The FIFO is a timing critical part of the design and was therefore floorplanned.

4.1 The AD9057 ADC Evaluation Board

The AD9057/PCB [21] evaluation board is equipped with an 8-bit 60 MSPS ADC. There is also a Digital to Analog Converter (DAC) on the board. The DAC is connected to the output of the ADC, that way the original analog signal can be compared to the output of the DAC. On the board there are BNC connectors for clock input, analog input and DAC output. The digital output form the ADC and the clock are available from a cable connector

The analog bandwidth of the ADC is 120 MHz, thus direct digitization of the GPS L1 signal is not possible with this component. One limitation of the ADC is that the minimum sampling rate is 5 MHz. In order to have lower sampling rates with this device, decimation must be done in the FPGA.

4.2 The Design

A simplified block diagram showing the data flow in the data acquisition system is shown in figure 4.1. All logic resides in PE0. PE1 is only used as a route-through. The control logic is not shown. An analog signal is low pass filtered with an anti-aliasing filter, i.e. a low pass filter, and fed into the ADC evaluation board. The ADC is clocked by an HP signal generator. The 8-bit samples and the sampling clock is passed into PE0 on the WILD-ONE^{TM} board

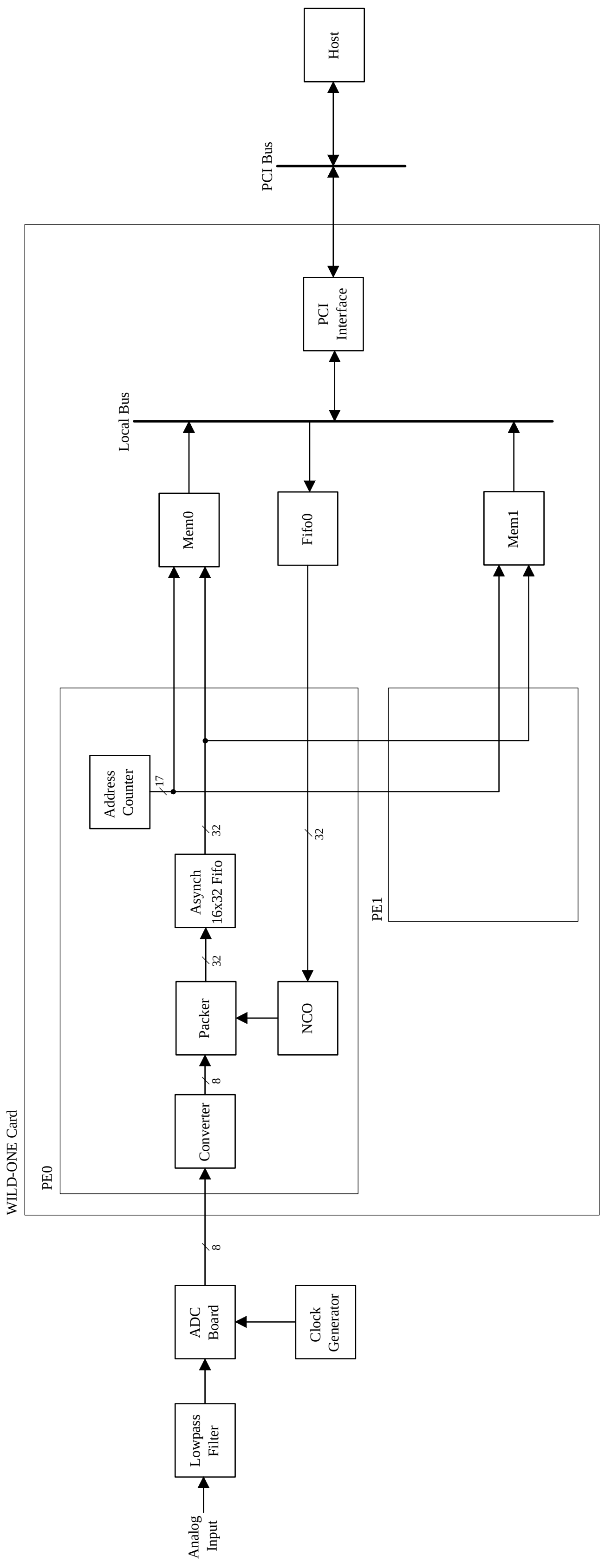

Figure 4.1: Simplified block diagram of data acquisition application.

via a cable connected to the SIMD bus.

4.2.1 Two's Complement Converter

The ADC data is converted to two's complement format within PE0. The sampled signal level is bipolar, between -0.5 V and $+0.5$ V. The output format of the ADC samples may be considered as an unsigned byte where 00000000 represents the negative input full scale -0.5 V and 11111111 represents the positive full scale $+0.5$ V. Thus in order to facility further processing on the host PC the samples are converted to 8-bit two's complement which is the native signed character number format in the host PC. The converter is very simple, only the most significant bit needs to be inverted.

4.2.2 Bit Packer

The 8-bit two's complement samples are packed to 32-bit words for efficient transfers over the PCI bus. This is implemented with an 8-bit wide 4-bit deep shift register.

4.2.3 Asynchronous FIFO

The 32-bit words are fed into asynchronous 16×32 FIFO [22] that resides inside PE0. This FIFO is necessary since two asynchronous clock domains must be interfaced - the ADC clock domain and the PE clock domain. The ADC clock can not be used to clock the SRAM buffers on the WILD-ONE$^{\mathrm{TM}}$ board since these are clocked by programmable oscillator on the board.

The FIFO memory is implemented by configuring 32 CLBs as 16×1 edge-triggered dual-port RAM, hence the size of the FIFO. However it is possible to build deeper FIFOs as explained in [22].

4.2.4 Decimation

Since the minimum sampling frequency of the ADC is 5 MHz a programmable decimation was implemented. This done by a 16 bit Numerically Controlled Oscillator (NCO), see figure 4.2. The NCO is programmed by a 17-bit control word via the board FIFO connected to PE0. The host initiates the buffering of samples by writing this control word. If bit 17 is set then that indicates that no decimation should be done. If bit 17 is not set then the remaining 16 bits specifies the decimation rate. The resulting sampling frequency in case of decimation is

$$f_D = f_s \frac{N}{2^{16}} \tag{4.1}$$

where N is the 16-bit control word and f_s is the sampling rate. Note that the factor $2^{16}/N$ should be an integer in order to avoid jitter in the in the NCO output. Assume that $N = 4095$, then we have $2^{16}/4095 = 16.004$. This is not an integer and jitter in the output will be the result since it is not possible to decimate by fractions of samples. In this case $N = 4096$ would make sense since here only every 16th sample is kept.

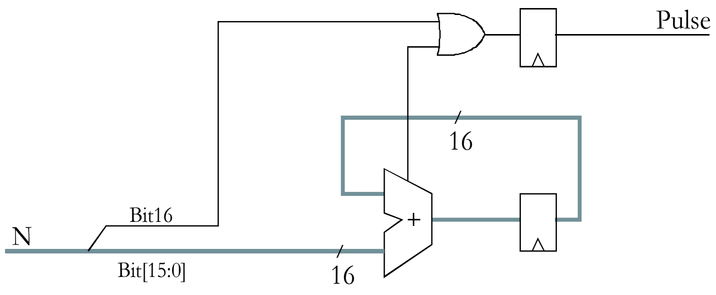

Figure 4.2: Numerically controlled oscillator

When decimation is active it is assumed that the analog low pass filter prevents aliasing. No digital filtering is performed.

4.2.5 Control Logic

The 32-bit words output from the asynchronous FIFO are written to one of the two memory buffers, depending on the state of the system. When any of the two memory buffers is full the interrupt for PE0 is activated, the host must then read the correct buffer and acknowledge the interrupt. Since the PE0 interrupt is used for both buffers, the host must keep track of what buffer to read from. The PE1 interrupt is used to signal failure caused by overrunning the asynchronous FIFO. The operation is based on the Finite State Machined (FSM) described below.

The control logic mainly consist of two FSMs - the **main FSM** and the **secondary FSM**. The Main FSM handles the initialization, memory writes and requests and the “Failure” interrupt (PE1). The Secondary FSM handles the “Buffer ready” interrupt (PE0) and makes sure that no interrupt is requested as long as the acknowledge signal from the host is active. The bubble diagram of the Main FSM is shown in figure 4.3.

The main FSM sits in an idle state until a decimation control word is written to FIFO0. The control word is read from FIFO0 and the NCO is updated. Memory buffer 0 (Mem0) is then requested. When the request is acknowledged Mem0 will be written to until it is full. When the Mem0 is full, the Main FSM enters a wait state where it checks for a pending PE0 interrupt.

However at this point no interrupt as been requested so Mem1 is requested the next clock cycle. When Mem1 is full the FSM enters a wait state for Mem0. The FSM sits in this state until the host has finished reading Mem0 and acknowledged the interrupt. Assuming the host acknowledges the interrupt in time, before the asynchronous FIFO overruns, Mem0 is requested and the cycle repeats.

The Failure state is reached whenever the FIFO overruns, that is when the Full signal from the FIFO is active when data needs to be written to the FIFO. In the Failure state the PE1 interrupt is activated. In order to restart the sampling all interrupts must be acknowledged and then board must be reset.

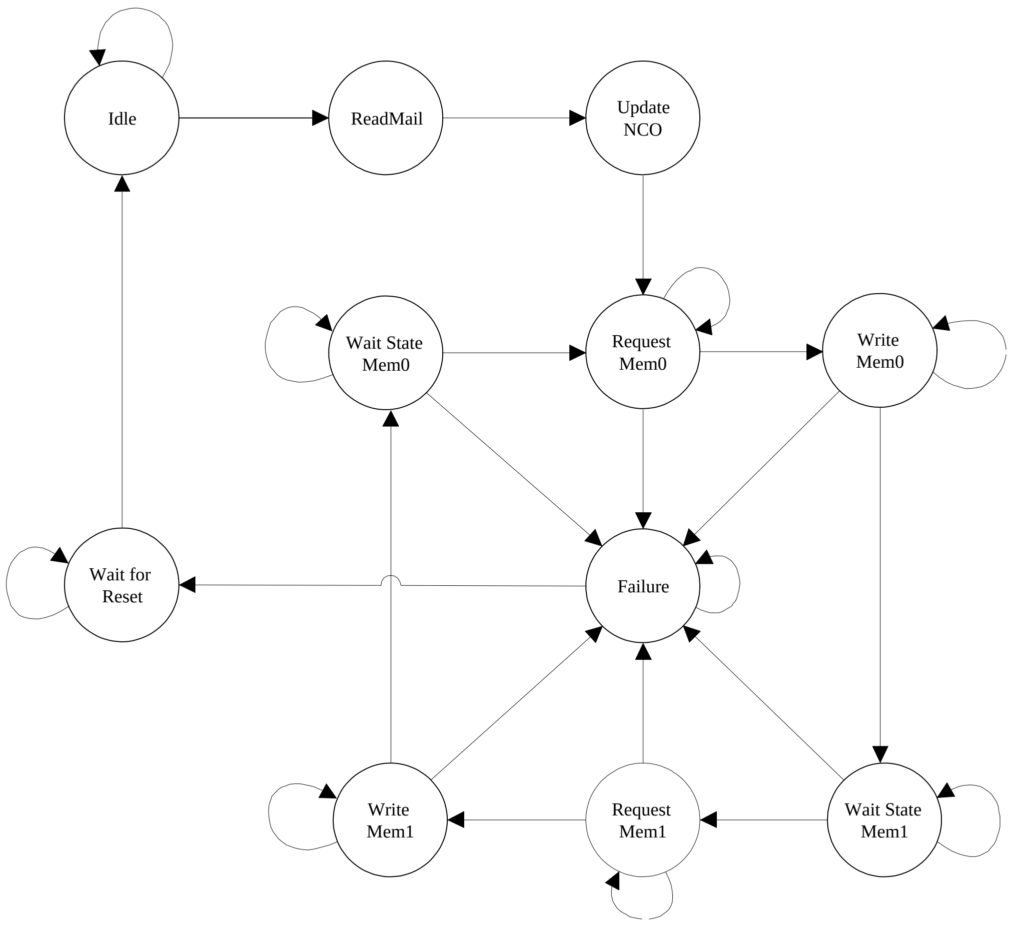

Figure 4.3: Main FSM.

4.2.6 Host Program

DMA transfers are used to transfer the sampled data to the host memory. The actual transferring consumes very little CPU time, which allows the data to be processed efficiently in a host CPU.

4.2.7 Testing the ADC Application

The design has been successfully tested over a range of sampling frequencies. Under 5 MHz with decimation was validated. Also sample rates up to 20 MHz, the upper limit of available clock sources, were successfully tested. Based on static timing analysis sample rates well over 50 MHz should be reachable. However at this point it is not verified that the cable connecting the ADC board and the WILD-ONE$^{\mathrm{TM}}$ card can handle such high rates.

4.2.8 Further Work

The data acquisition design has proved to be flexible and to have a high performance. Interfacing other ADCs with various bit-widths will require minimal changes to the design. Only one component, the "bit packer", has to be changed in order to interface other ADCs. Possible further work would be to test the application at sampling rates higher than 20 MHz.

Chapter 5

GPS Signal Generator

For research purposes a GPS signal generator was implemented. When testing and developing GPS receivers it is convenient to work with simulated GPS data. Also at the time a GPS front-end was not available, so a basic GPS signal generator was implemented on the WILD-ONE$^{\text{TM}}$ board. Possible improvements of the GPS signal generator are explored in detail.

5.1 The Design

The two channel GPS generator is illustrated in figure 5.1. One channel consists of five sub modules: a carrier NCO, a sine look-up table, a code NCO, a code generator and an adder/subtracter. The frequency of both NCOs are fixed, so the signal does not model Doppler effects. The code generators are fixed to two different satellites and can not be changed at run-time. The BPSK modulation is implemented with a simple adder/subtracter. The output of the code generator controls the adder/subtracter so the replicated carrier is either passed through or negated. No data modulation is present on the signal. The output of the two channels are added to form the simulated 8-bit GPS signal.

5.1.1 Carrier NCO and Sine Look-up Table

The carrier NCO and sine look-up table are illustrated in figure 5.2. The input to the NCO is, apart from the clock, an i-bit word M, that determines the frequency. The phase is accumulated in an i-bit register. Each time the adder in the NCO overflows a carrier cycle is completed. Thus the frequency of generated carrier is

$$f = f_c \frac{M}{2^i} \tag{5.1}$$

where f_c is the clock frequency. The frequency output resolution is

$$\Delta f = \frac{f_c}{2^i}. \tag{5.2}$$

It is possible to generate frequencies up to the Nyquist rate $f_c/2$.

The pipelined sine look-up was created with Xilinx Core Generator, a module compiler for Xilinx FPGAs. Not all i bits are used to address sine look-up table, instead the j

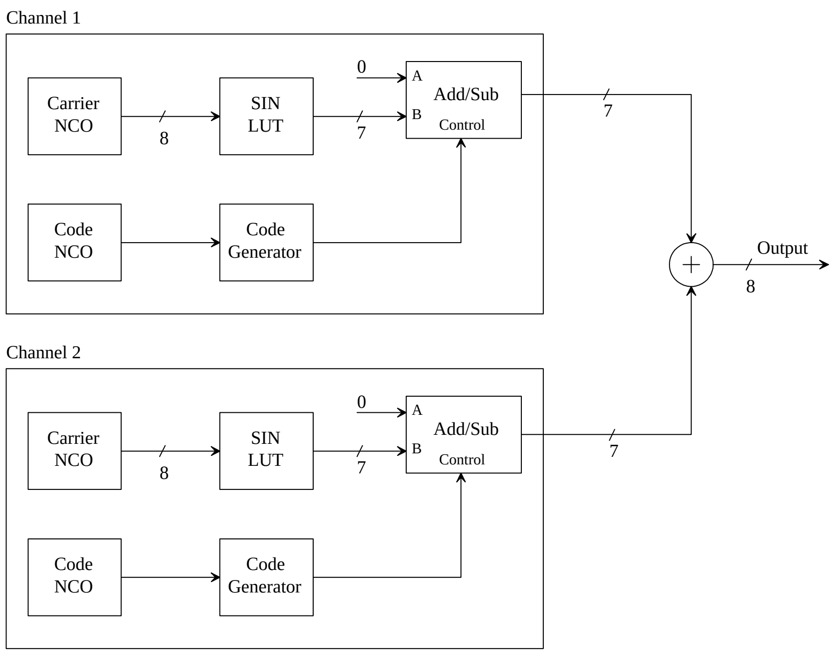

Figure 5.1: GPS signal generator.

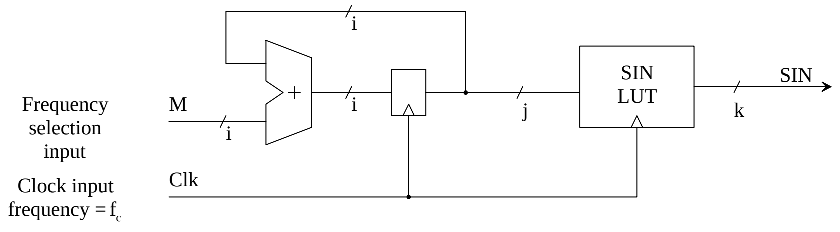

Figure 5.2: Carrier NCO and sine look-up table

most significant bits are used. The phase plane is subdivided into 2^j phase points and the amplitude of the output is quantized to a k-bit representation.

Specifying the bit-widths i, j and k is a trade-off between accuracy and hardware resources. For this application values of $i = 32$, $j = 8$ and $k = 7$ were utilized. The output was chosen to seven bits, since the output from the both channels combined, initially was determined to be 8 bits.

5.1.2 Code NCO

The code NCO, shown in figure 5.3, resembles the carry NCO. The only difference is that the output is the registered carry from the adder. This registered carry output, is a pulse train used to enable the code generator at a nominal rate of 1.023 MHz. Also here the width of the NCO is 32 bits ($i = 32$).

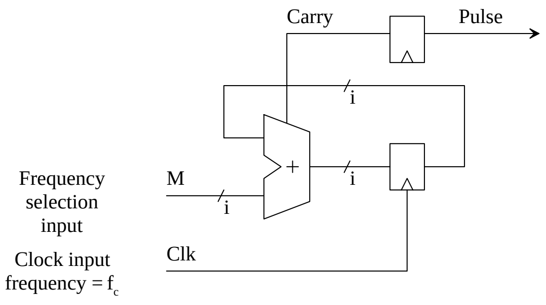

Figure 5.3: Code NCO.

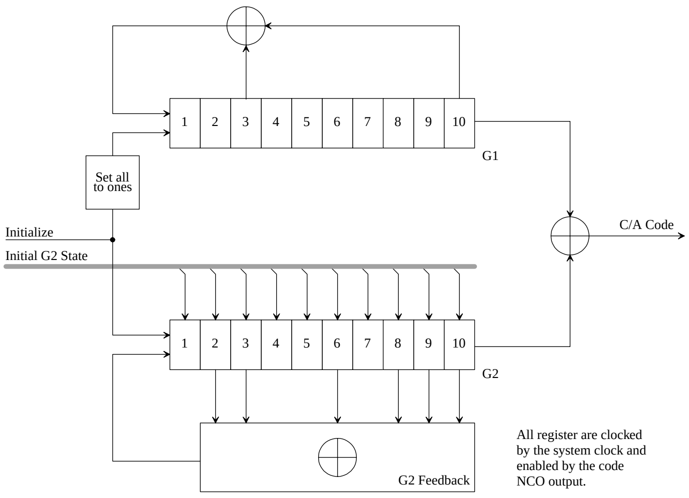

Figure 5.4: C/A code generator.

5.1.3 Code Generator

The code generator is a bit different from the code generator illustrated in figure 2.2, where the phase shift of G2 is accomplished by taking the exclusive-or of two taps of the G2 register. This shift can also be implemented by loading the G2 register with a specific pattern at initialization [2] as shown in figure 5.4. This pattern is the inverted bits of the ten first chips of the C/A code sequence. In the current implementation the G2 register is not loadable but when the FPGA is powered up or when it reset the G2 register is initialized to a specific pattern. The patterns for the two channels are obviously different since two separate satellite signal are generated. The LFSRs are clocked by the system clock, but are enabled by the output of the code NCO.

5.1.4 Testing the Signal Generator

The signal generator and the data acquisition design were combined in order to test the signal generator. Instead of taking data from the SIMD bus, the GPS signal generator was connected to the asynchronous FIFO. Data was generated at a sampling rate of 5 MHz and was written to disk for later processing in Matlab. In Matlab the acquisition process was simulated with a noncoherent correlator. The correlation is given by

$$C[m] = \left(\left[\sum_{n=0}^{L-1} x[n] \cdot CA[n+m] \cdot \cos[\Omega n] \right]^2 + \left[\sum_{n=0}^{L-1} x[n] \cdot CA[n+m] \cdot \sin[\Omega n] \right]^2 \right)^{1/2}, \tag{5.3}$$

where L is the length of a CA code in samples, $x[n]$ is the generated GPS data, and CA is the C/A code and Ω is radian frequency.

In the generated GPS data, two satellite signals are present namely satellite 1 at IF 1.25 MHz and satellite 2 at IF 1.248 MHz. Figure 5.5 shows the correlation for two cases. Satellite 3 was searched for at IF 1.25 MHz and satellite 2 was searched for at the same IF. As seen in the plots satellite 3 is not detected while satellite 1 is.

5.2 Further Work

Since the implemented design is simple it is possible to do a number of improvements. To improve the simulated signal to more resemble true GPS signals, the following should be done:

- Data modulation should be added. Depending on the application random data might suffice.
- Doppler effects should be modeled, and the Doppler profile should be loadable.
- The code generators should be loadable in order to generate any satellite code sequence at any code phase.
- The amplitude of satellite signals varies with time, so an amplitude profile should also be loadable.
- More than two satellite signals should be modeled. Four satellites are needed to do a position solution so a minimum of four channels should be implemented.
- A programmable level of Gaussian noise should be added to the signal.

A suggested architecture with the mentioned improvements is illustrated in figure 5.6. Note that the bit widths have not been specified simply because the design has not been implemented and is proposed for future work.

5.2.1 Gaussian Noise Generator

The GPS code generator is implemented with LFSRs as described in chapter 2. LFSRs can also be used to generate pseudo random sequences for other purposes than code generators for CDMA applications. A number of LFSRs can be used to implement a pseudo Gaussian noise generator. The central limit theorem states that under general conditions the sum of a large number of random variables are approximately normally distributed [23]. By counting

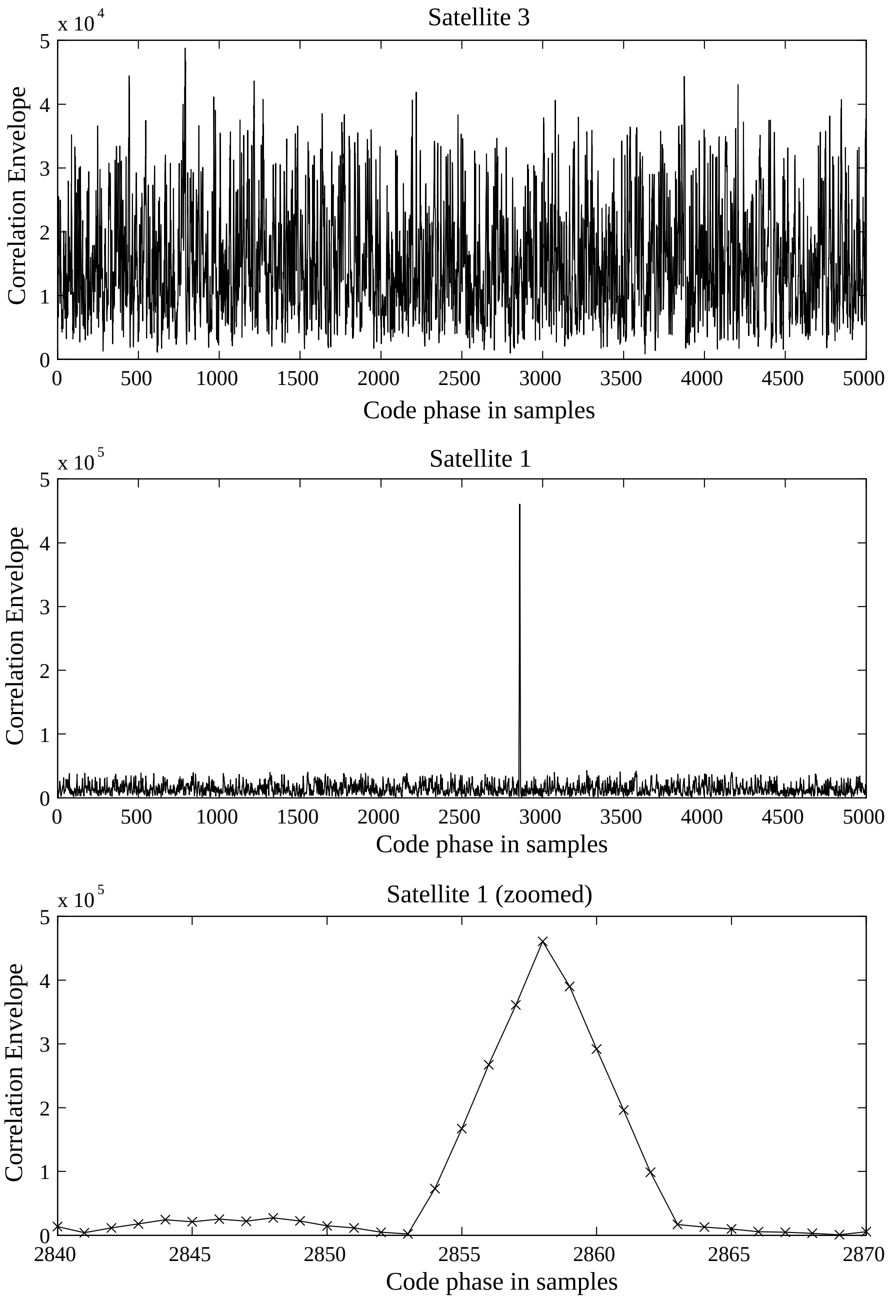

Figure 5.5: Correlation envelopes obtained when performing acquisition in Matlab on generated GPS signals.

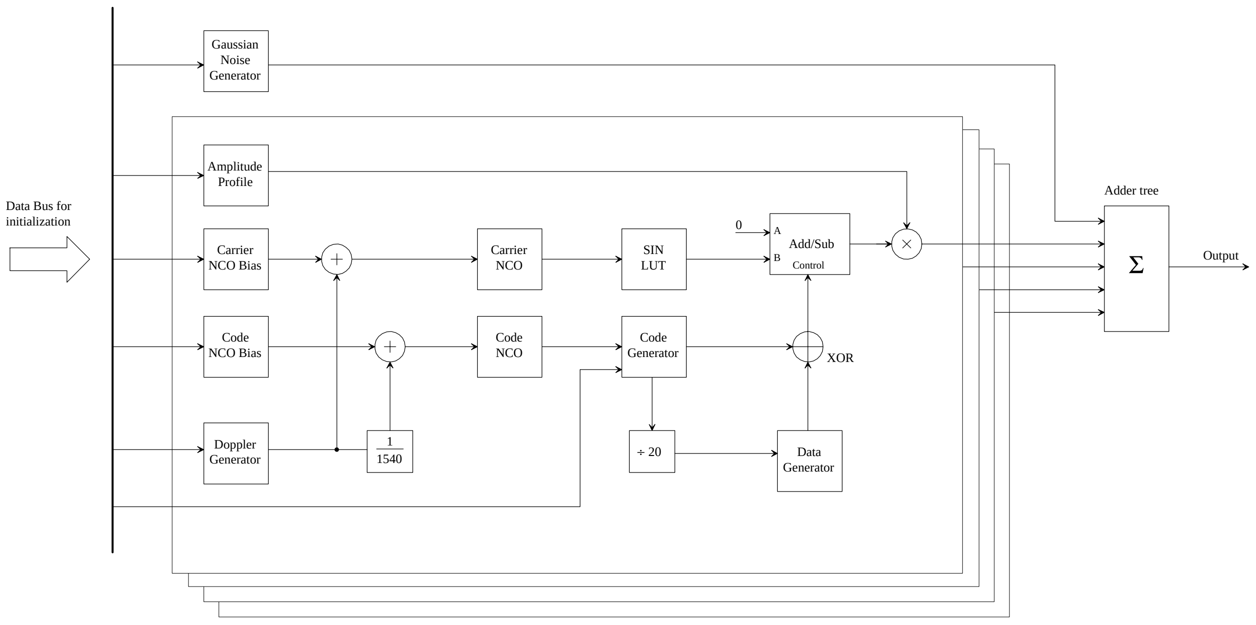

Figure 5.6: A suggested architecture for a GPS signal generator.

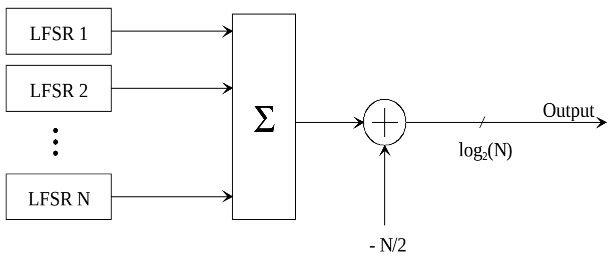

Figure 5.7: Gaussian Noise Generator.

the output bits N LFSRs a pseudo Gaussian sequence can be generated, as illustrated in figure 5.7. This sequence does not have zero mean so $N/2$ is subtracted from the result in order to get a zero mean sequence.

A 63-bit LFSR can efficiently be implemented in only four XC4000 CLBs [24]. The output of this maximum length LFSR repeats after $2^{63} - 1$ clock cycles, which translates to 58494 years at 5 MHz clock frequency. To generate a n-bit random sequence 2^n LFSRs are needed. Assuming we want a 6-bit noise generator we will need $2^6 = 64$ LFSRs. It is also possible to take more than one bit from each LFSR to generate the random number and thereby reducing the number of CLBs.

The variance of the noise should be variable, which can be implemented in a number of ways. The most obvious implementation is to scale the output by multiplying by a factor. However multipliers with variable operands require a substantial amount of logic resources in a FPGA. If we can allow to noise level to be scaled in levels of 3 dB a simple shifter can implement the variable noise variance. Another option would be to only sum a certain number of LFSR outputs depending on a control word. This would insert some logic between the LFSRs and the summer shown in figure 5.7. As usual it is a trade-off between accuracy and logic resources.

5.2.2 Doppler Generator

The Doppler offset [1] due to relative motion of the satellite and receiver is

$$\Delta f = -f_0 \frac{(\mathbf{v} - \mathbf{u}) \bullet \mathbf{a}}{c} \tag{5.4}$$

where f_0 is the transmitted frequency, $\mathbf{v}$ is the satellite velocity, $\mathbf{u}$ is the user velocity, $\mathbf{a}$ is the unit vector pointing along the line-of-sight from the receiver to the satellite, and c is the speed of light. It should be noted that Δf is the relative Doppler on the received carrier frequency and it is not changed when the satellite signal is downconverted. The Doppler offset on the received C/A code is a factor 1540 smaller than the Doppler offset on the received carrier. Figure 5.8 shows a typical received Doppler frequency by a stationary receiver. When the satellite is at its closest position to the receiver the Doppler offset is zero.

One way to model various Doppler offsets in hardware is illustrated in figure 5.9. Initially the carrier and code NCO biases are loaded into registers. A RAM is loaded with a Doppler profile were each entry represents the increment/decrement of Doppler offset compared to the previous Doppler offset. When in operation the RAM address counter is incremented at a programmable rate. The output of the RAM is fed into an accumulator that accumulates the carrier Doppler offset relative to the bias value. The Doppler offset is then added to the bias carrier in order to create a simulated Doppler effect. The code Doppler offset is determined by dividing by a factor of 1540.

The bit-widths of the various components must be chosen with respect to what is feasible to implement and the desired precision. Assuming we want 32 bit NCOs and a maximum carrier Doppler offset of ± 10 kHz at 5 MHz system clock rate. Then the width of the Doppler

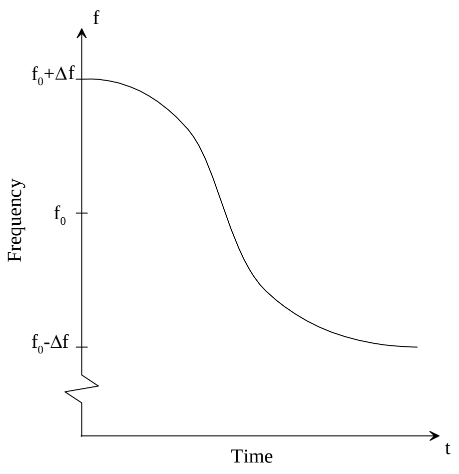

Figure 5.8: Typical received Doppler frequency by a stationary receiver.

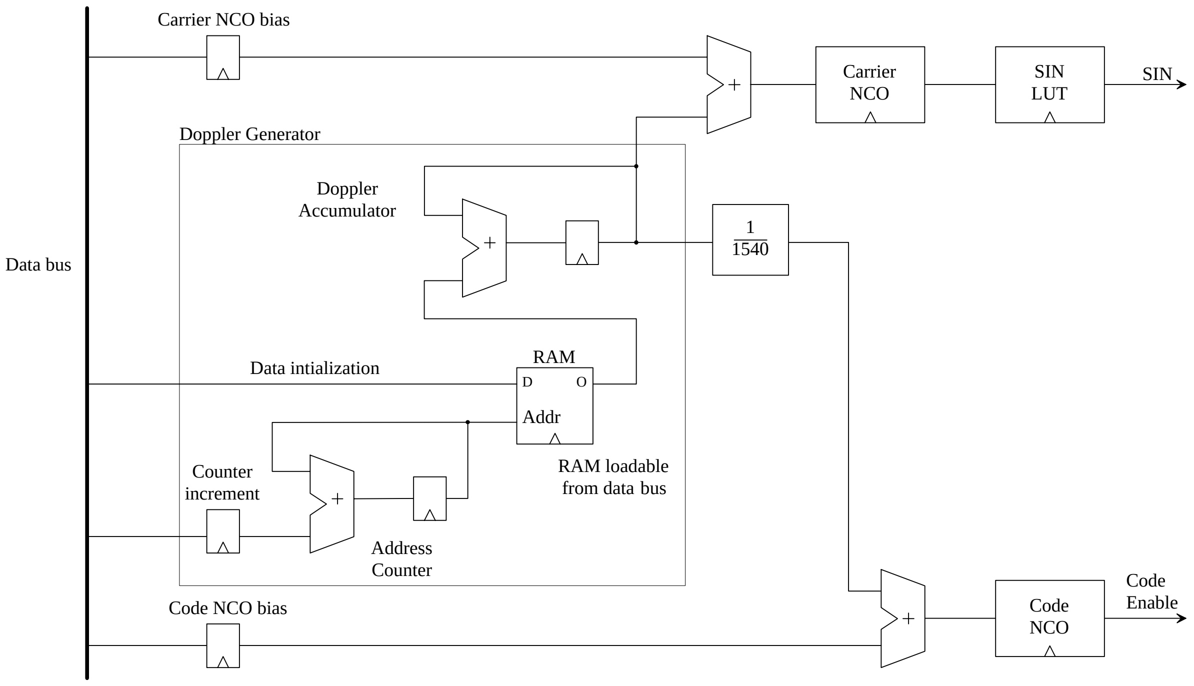

Figure 5.9: Doppler generator

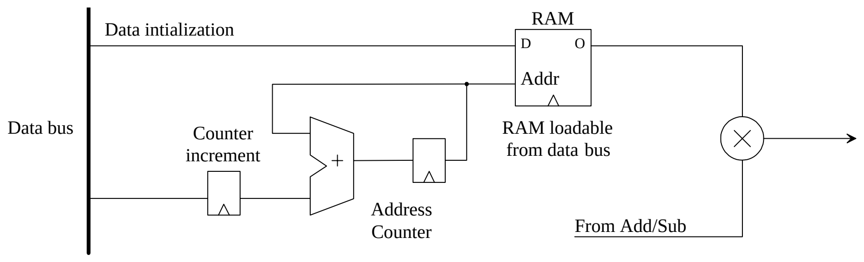

Figure 5.10: Variable amplitude implementation.

accumulator is determined by

$$n = \left\lceil \log_2 \left(\frac{2^{32} \cdot 20 \cdot 10^3}{5 \cdot 10^6} \right) \right\rceil = 25 \text{ bits.} \tag{5.5}$$

The depth and width of the RAM and the width of the address counter are other parameters that have to be determined.

5.2.3 Amplitude Profile

The received signal power varies as the satellite moves. The received signal levels depends on the satellite elevation angle [1]. A variable signal power can be implement similar to the proposed Doppler generator, as in figure 5.10. The difference here is that the amplitude scaling factor is not accumulated. Fewer bits are needed for the amplitude multiplication compared to how many bits are needed to offset the carrier, so the RAM stores the complete factor.

5.2.4 Code and Data Generation

The code generator should be implemented as illustrated in figure 5.4 with the addition of epoch decoding of the G1 register. When G1 is all ones one code period has passed. Every 20th code epoch the data generator should be triggered. Both divide by 20 counter and the data generator can be implemented in two simple LFSRs. This is of course not true if “real” data is required.

5.2.5 Adder Tree

The output of the four channels and the noise generator are summed to create the composite GPS signal. The adder should be structured as a tree for optimal speed.

Chapter 6

Multi-channel GPS Digital Receiver

The last project in the thesis work was to implement a multi-channel GPS digital receiver in a single FPGA. The intention is that the digital receiver should get its data from a GPS front end from Mitel Semiconductor. A prototype of the receiver has been implemented in hardware and has initially been tested to perform an acquisition with previously collected GPS data. The data was stored in RAM and read by the digital receiver. The test was positive, i.e. the design is working as intended.

Following the Mitel front-end will shortly be described and after that the digital receiver design will be explored in more detail.

6.1 The Mitel Semiconductor GP2010 front-end

The Mitel Semiconductor GP2000 chipset [25] consist of the GP2015 RF front end and the GP2021 12-channel correlator. In the thesis work we wanted to develop a digital receiver in a FPGA, which could replace the GP2021 correlator chip so that additional user configurarability can be added.

The GP2015 RF front end is illustrated in figure 6.1.

6.1.1 Downconversion

An external 10 MHz reference oscillator drives an on-chip phase locked loop. The intermediate frequencies, 1400 MHz, 140 MHz and 31.111 MHz needed to downconverts the L1 C/A code signal is generated within the chip. The downconversion is done in 3 stages. In the first stage the L1 signal at 1575.42 MHz is mixed with the 1400 MHz signal giving a first IF at 175.42 MHz and placing an image at 1224.58 MHz. The first IF filter should reject the image.

At the second stage the 175.42 IF is mixed with the 140 MHz LO resulting in a second IF at 35.42 MHz and an image at 104.845 MHz. The filter after the second stage is crucial to the system performance since it effectively sets the bandwidth of the system. Mitel recommends using a SAW filter with tight passband and sharp cut-off. The width of the passband should be the bandwidth of the C/A code.

At the third stage the 35.42 MHz IF is mixed with the 31.111 MHz LO, resulting in an IF

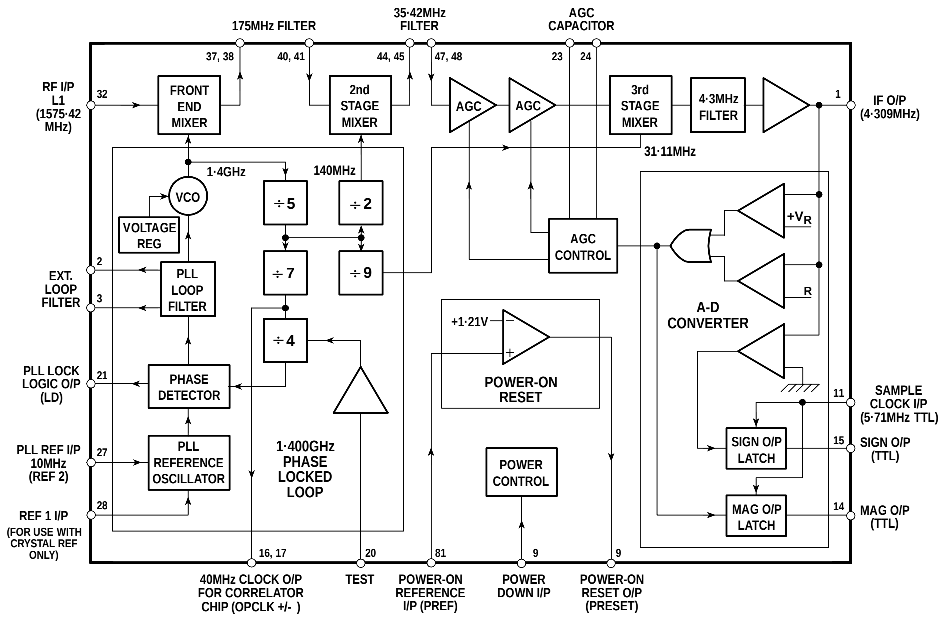

Figure 6.1: GP2015 block diagram 6.1.

at 4.309 MHz and an image at 26.802 MHz. An on-chip 4.3 MHz passband filter rejects the image.

6.1.2 Analog to Digital Conversion and Automatic Gain Control

The GP2015 uses a 2-bit ADC to produce SIGN and MAG signals that are synchronized to the rising edge of the sample clock. The sample clock at 5.714 MHz is usually derived from the GP2021 correlator, but in our case we do not want to use the correlator and will provide another clock source. The sampling aliases the GPS signal down in frequency from 4.309 MHz to 1.405 MHz.

The analog 4.309 MHz IF signal is used to control the gain of the AGC which operates such that the SIGN bit is high for 50 % of the time and the MAG bit is high for 30 % of the time. Table 6.1 shows the distribution of the sampled signals and the binary encoding.

Level	Percentage of time	Binary encoding
-3	15%	01
-1	35%	00
1	35%	10
3	15%	11

Table 6.1: Card Configuration

6.2 FPGA Implementation of a Multi-channel Digital Receiver

6.2.1 Introduction

Implementing a single channel GPS digital receiver in an FPGA is not very interesting. Commercial receivers today usually have up to 12 or more parallel digital receiver channels and a single channel FPGA design can therefore not be motivated. So we have developed a prototype for a multi-channel FPGA digital receiver design. The most obvious approach to this problem is to first design a single channel and then re-use that channel to create a multi-channel digital receiver. However this approach is not really such a good idea since large and expensive FPGAs will be required for the implementation. Considering the relatively low sample rate of a typical L1 GPS receiver, like the 5.714 MHz for the GP2000 chipset, we see a possibility to interleave hardware resources in time. Assume that we can clock a single receiver channel at 50 MHz then interleaving that channel in time would give us virtually ten 5 MHz channels. This approach has been suggested before for an ASIC implementation of a multi-channel GPS receiver design [26]. It should be mentioned that the Mitel Semiconductor GP2021 correlator chip has served as a basis for the FPGA design.

6.2.2 Using the Distributed Edge-Triggered RAM for Interleaving

As a result of the distributed edge-triggered RAM in the Xilinx XC4000 devices it is possible to implement the interleaving or multiplexing of channels in a straight forward manner. Instead of using D flip-flops to store data, the CLBs can be configured as SRAM allowing 32 bits to be stored instead of 2 bits in a single CLB. Also if the flip-flops are used then 34 bits can be stored in one CLB. Combined with an address counter the CLB configured in a RAM mode can be viewed as a shift register. The address counter can be shared between many CLBs. This allows an efficient implementation of shift registers or delay queues. Since this SRAM can be placed close to other required logic the access to the SRAM will ensure minimal delay. No external RAM is needed to interleave a channel.

An "ordinary" design would have registers instead of RAM to store signals. The interleaved multi-channel design can be seen as an "ordinary" design where all registers have been replaced with a 16×1 edge-triggered RAM. A global address determines which channel is active during a specific clock cycle. Assume we need N channels, then the multi-channel receiver needs to be clocked at a rate that is N times the data sampling rate. Up to 16 channels can be used and $\lceil \log_2(N) \rceil$ address bit are needed. To better understand the interleaving or multiplexing of channels consider figure 6.2, which is a simplified view of the 16×1 RAM in figure 3.4. Input registers synchronize the data, write enable and write address inputs while the output address is not registered. When the address counters cycles through the addresses corresponding to channel 1 up to channel N the 16×1 RAM may be considered as a circular buffer. When the pointer to this circular buffer wraps around all channels have been activate during an *address* cycle. One address cycle is then N sampling clock cycles.

The design is limited to eight channels, thereby reducing the required clock rate to eight times the clock rate of a single channel. This implies that the Gray address counters only have eight states. Gray counters are used to reduce the toggling of the address lines, and as

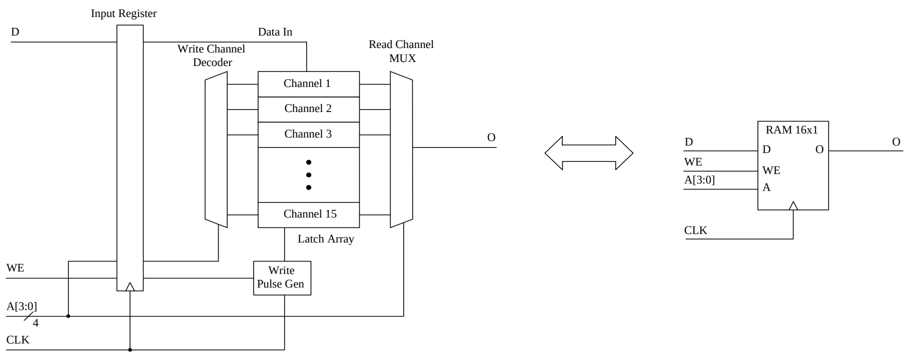

Figure 6.2: Simplified view of 16×1 edge-triggered RAM configuration.

a result the power consumption will be decreased. If more channels are required later, the address counters can be changed to count over more states. As mentioned, the front-end is designed for a sampling rate of 5.714 MHz, so the design needs to be clocked at a rate of 45.712 MHz. In order to have some timing margin the frequency constraint was set to 50 MHz for the complete design.

Not all FPGAs have the same type of distributed synchronous RAM as Xilinx XC4000 have. Generally when designing for FPGAs it is a good idea strengths and weaknesses of the target architecture before starting a new design. The key point is that the fastest and densest design come from designers with a good knowledge of the target architecture.

6.2.3 The Top-Level

The data path of the design was created with schematic entry, while the control logic was modeled in VHDL. The design is very sensitive to efficient placement, so floorplanning for the data path was done in the design entry stage. A high-level block diagram of the design is shown in figure 6.3. The input to the receiver channel is the two bit data ±1 or ±3 at 5.741 MHz. First the inphase, I, and quadrature components, Q, are generated after mixing with cosine and sine. The cosine and sine are also have a two bit representation and can take the values ±1 and ±2, so I and Q can be represented with three bits and can take the values ±1, ±2, ±3, and ±6. The I and Q components are then multiplied with the Early, Prompt and Late codes separated 1/2 chip apart. The codes can be either $+1$ or -1 so three bits can still represent the downconverted and despread signals. However before the signals are passed to the accumulators they are converted to two's complement representation.

The different sub-modules will be described next.

6.2.4 Carrier NCO

The 32-bit carrier multi-channel carrier NCO is illustrated in figure 6.4. Two 16×16 RAMs stores the input control word, Ref[31:0], for the NCO when Load is high. The 32-bit adder in the NCO is split up into two 16-bit adders in order to insert a 16×1 RAM to pipeline the carry path. Since the carry is pipelined two extra 16×16 RAMs are required to balance the two paths - the 16 most significant bits and the 16 least significant bits.

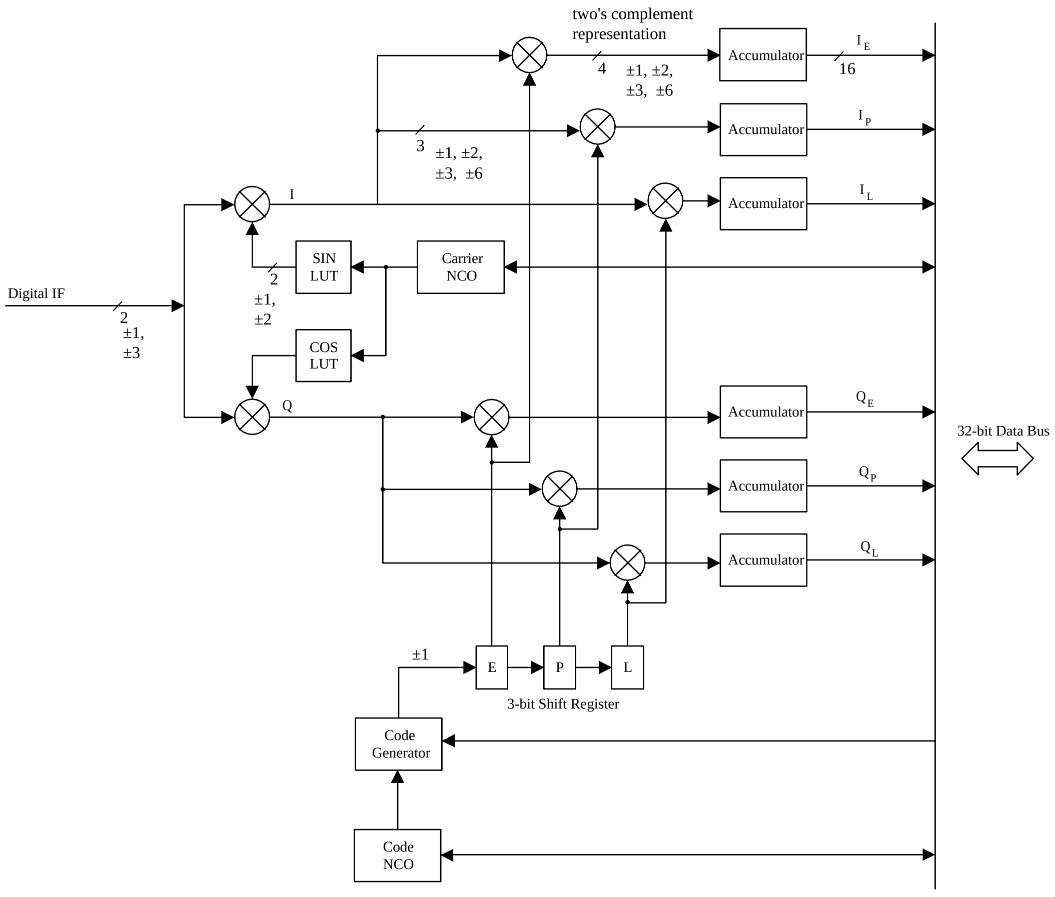

Figure 6.3: Implemented digital reciever channel.

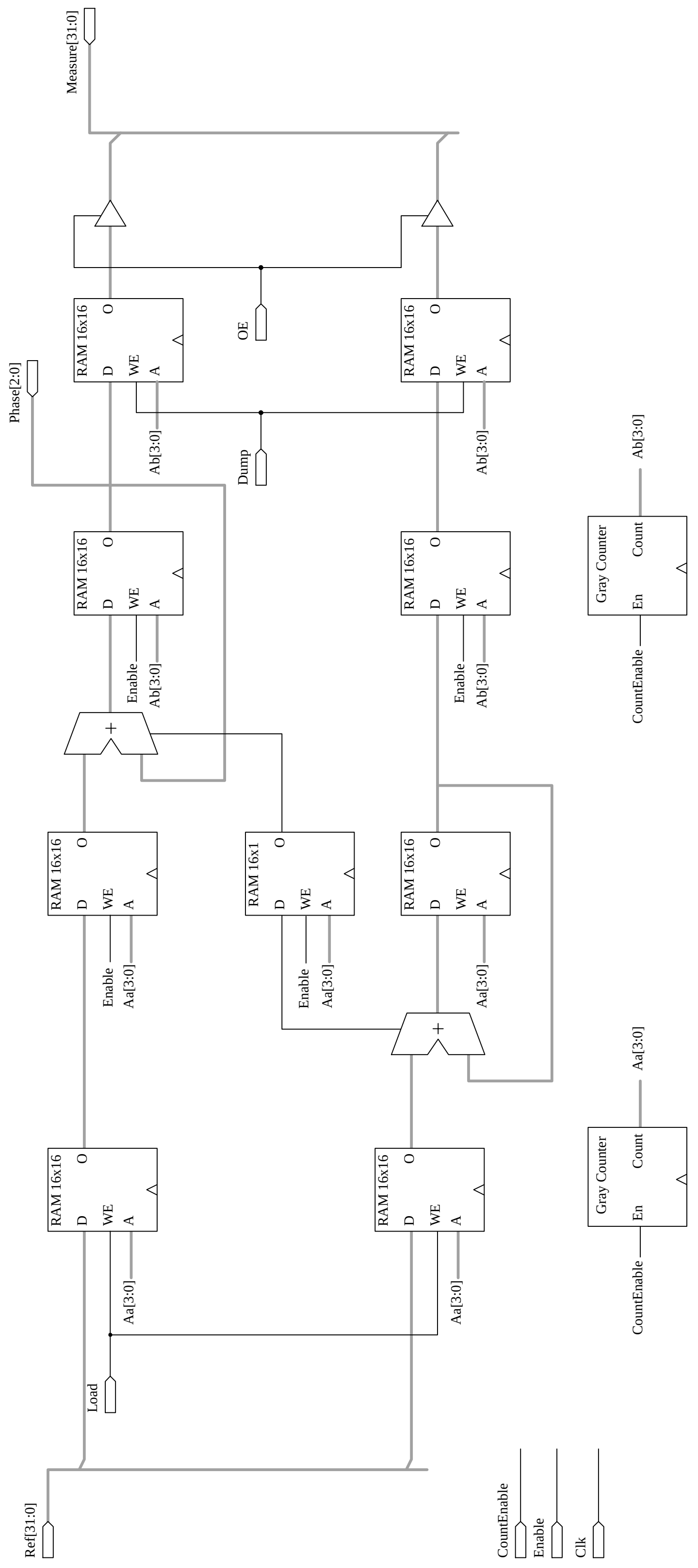

Figure 6.4: Multi-channel carrier NCO.

Three most significant bits of the NCO, Phase[2:0], represent the phase and are used to address the sine and cosine look-up tables. By setting the Dump input high the complete 32-bit representation of the phase is written to the RAM at the output of the NCO. The stored phase may be activated on the output bus, Measure[31:0], by holding OE low. When OE is high the output bus is tri-stated.

As seen in figure 6.4 there are two address counters. These are identical counters and the reason for having more than one source is to reduce the fan-out of the address lines. This is as critical for the design as well floorplanning. The design require lots of routing due to the RAM based approach and without careful floorplanning it is likely that it would be impossible to meet the timing constraints. For illustration the floorplan of the carrier NCO is shown in figure 6.5. The upper 16 bits and lower 16 bits of the NCO are separated apart with a few CLB places. The address counter are placed between the two halves in order to reduce routing delays. Since the design have a 32 bit bi-directional bus it is important that the bits of the bus are placed at the same row throughout the design. The individual tri-state buffers in the carrier NCO drives one longline each. The code NCO has the same layout as the carrier NCO and is aligned with the carrier NCO. The accumulator have 16 bit output to the bus, and they are aligned so the inphase accumulators drive the upper 16 bits of the internal data bus while the quadrature accumulators drives the lower 16 bits.

The multi-channel NCO require a total of 88 CLBs. Eight 16×16 RAMs uses 64 CLBs, the adders use a total of 19 CLBs, the address counters use two CLBs each and the 16×1 RAM used to pipeline the carry occupy one CLB. So this results in $64 + 19 + 4 + 1 = 88$ CLBs. For comparison a register-based carrier NCO would require a total of 67 CLBs. The reduced amount of CLBs comes from the fact that 33 registers can be placed in the same CLB as the adders occupy and that no address counters are needed. If the 8-channel carrier NCO is implemented by replicating 8 NCOs this would require $67 \times 8 = 536$ CLBs which is a dramatic waste of resources since the RAM-based multi-channel carrier NCO can be implemented in only 88 CLBs.

6.2.5 Sine and Cosine Look-up Tables

The representation of the sinusoids is identical to the GP2021 correlator chip - four levels and eight phases, as indicated in table 6.2. The look-up tables require only one CLB each.

Sinusoid	**Sequence**
Sin	-1 +1 +2 +2 +1 -1 -2 -2
Cos	+2 +2 +1 -1 -2 -2 -1 +1

Table 6.2: Sinusoids distribution over one cycle

6.2.6 Code NCO

The code NCO is almost identical to the carrier NCO. The only difference is the primary output is the carry from the upper 16-bit adder. The output of code NCO have a nominal rate of twice the C/A code rate. This is required to produce the early, prompt and late codes separated 1/2 chip apart.

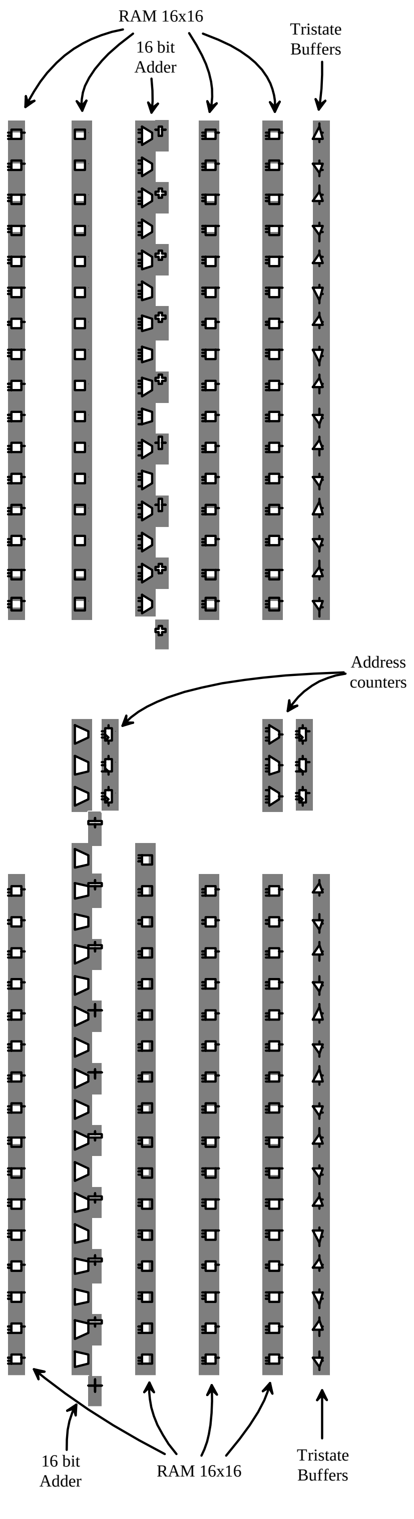

Figure 6.5: Floorplan of carrier NCO.

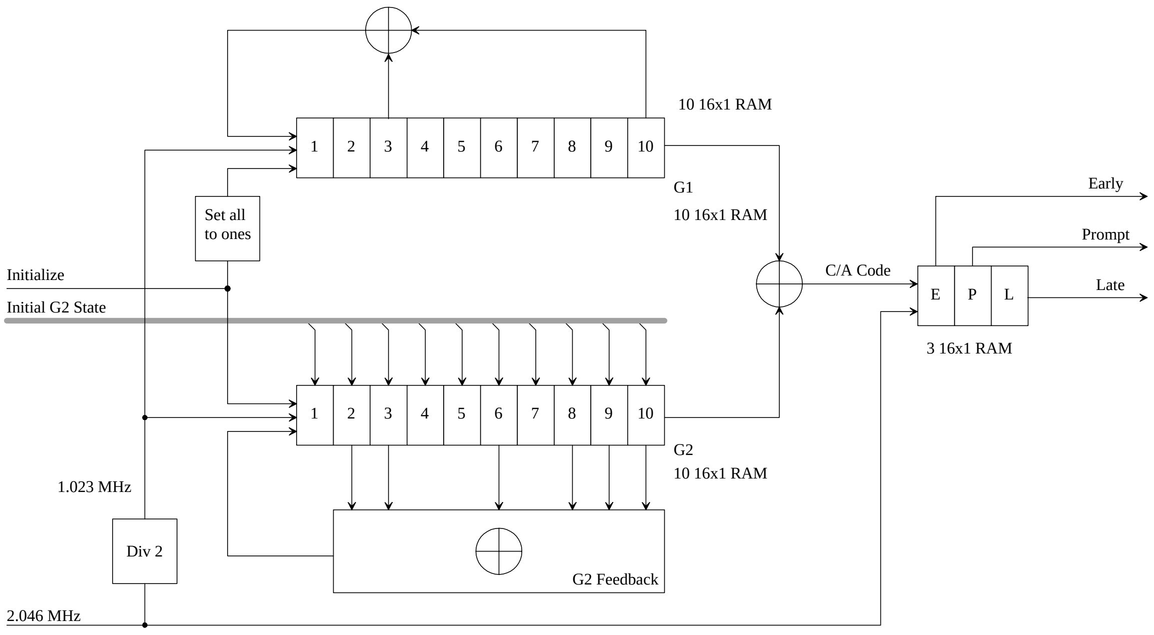

Figure 6.6: Code generator.

6.2.7 Code Generator

Obviously the code generator needs to be RAM-based since different codes are used for different channels. A simplified block diagram of the code generator is shown in figure 6.6. The LFSRs are implemented in RAM instead of registers. Not shown in the figure is the epoch decoding. The epoch decoding detects when G1 is all ones and outputs a pulse synchronized to when the prompt arm has completed a code cycle. The epoch pulse triggers the dump of the accumulators of the active channel. Another thing that is not shown is that it is possible to skew the code 1.5 chips. This feature can be used during acquisition when the code phase needs to be changed.

6.2.8 Downmixing and Despreading

The multipliers shown in figure 6.3 have low complexity because of the low bit widths. The output is a four bit two's complement number that can take the values ± 1, ± 2, ± 3, and ± 6.

6.2.9 Accumulators

In the design there are six accumulators per channel. Some receivers like the GP2021 only have four accumulators per channel. This is possible since the early and late codes are subtracted prior to despreading. However this limits the types of discriminators that can be used in the tracking loop. The multi-channel accumulator is shown in figure 6.7. The input is the four bit result from the downmixing and despreading. The result of the accumulation is dumped at each code epoch and a new accumulation is then initiated.

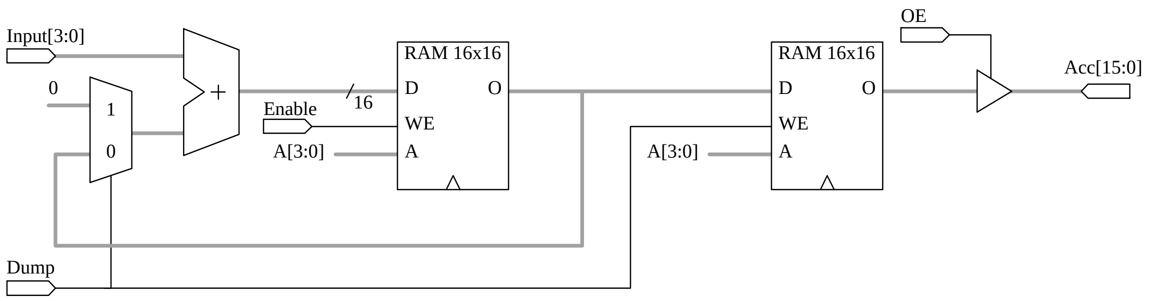

Figure 6.7: Accumulator

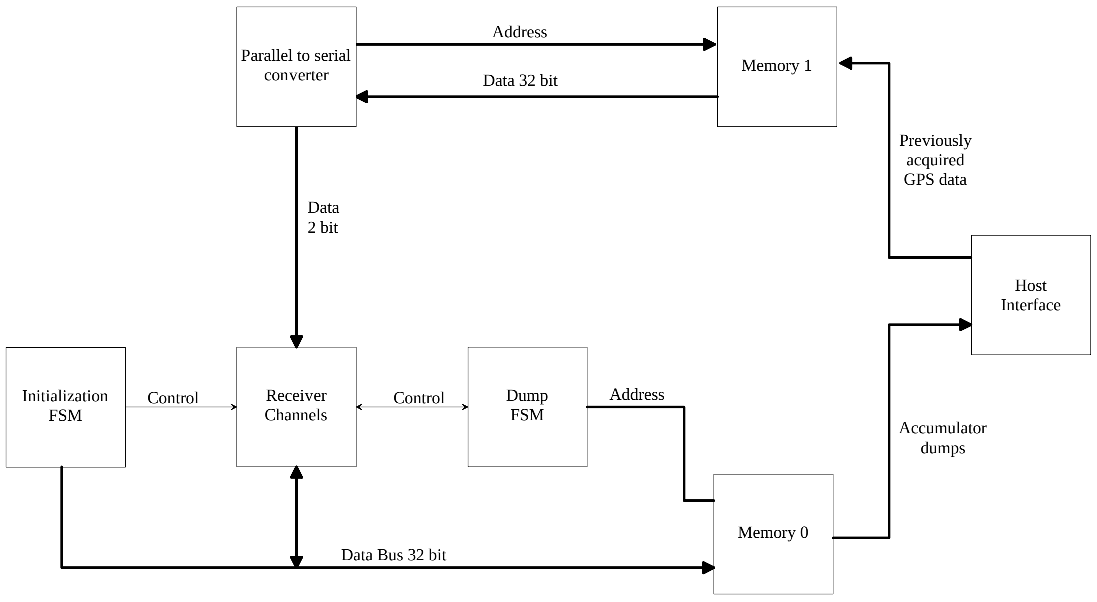

Figure 6.8: Test of acquisition.

6.2.10 Control Logic

The control logic is implemented to allow testing acquisition of previously collected data. The setup for testing is shown in figure 6.8. The multi-channel digital receiver and the control logic is implemented in PE0 on the WILD-ONE^{TM} board. Initially the PE1 memory buffer is filled with two-bit GPS data. The data is sampled at 5 MHz and about 0.4 seconds of data is stored in the memory buffer for PE1. A serial to parallel converter in PE1 feeds the receiver channels with 2-bit data.

A finite state machine in CPE0 initializes the receiver channels to perform acquisition on the data. A second finite state machine monitors the epoch signals from the receiver channels and an interrupt is triggered when the channels have dumped the accumulations in the memory for CPE0. The host then reads the accumulated results for each channel and writes the result to disk.

Correlation peaks were found as expected, hence we may say that the acquisition was successful. Figure 6.8 shows correlation envelope for one of the eight channels.

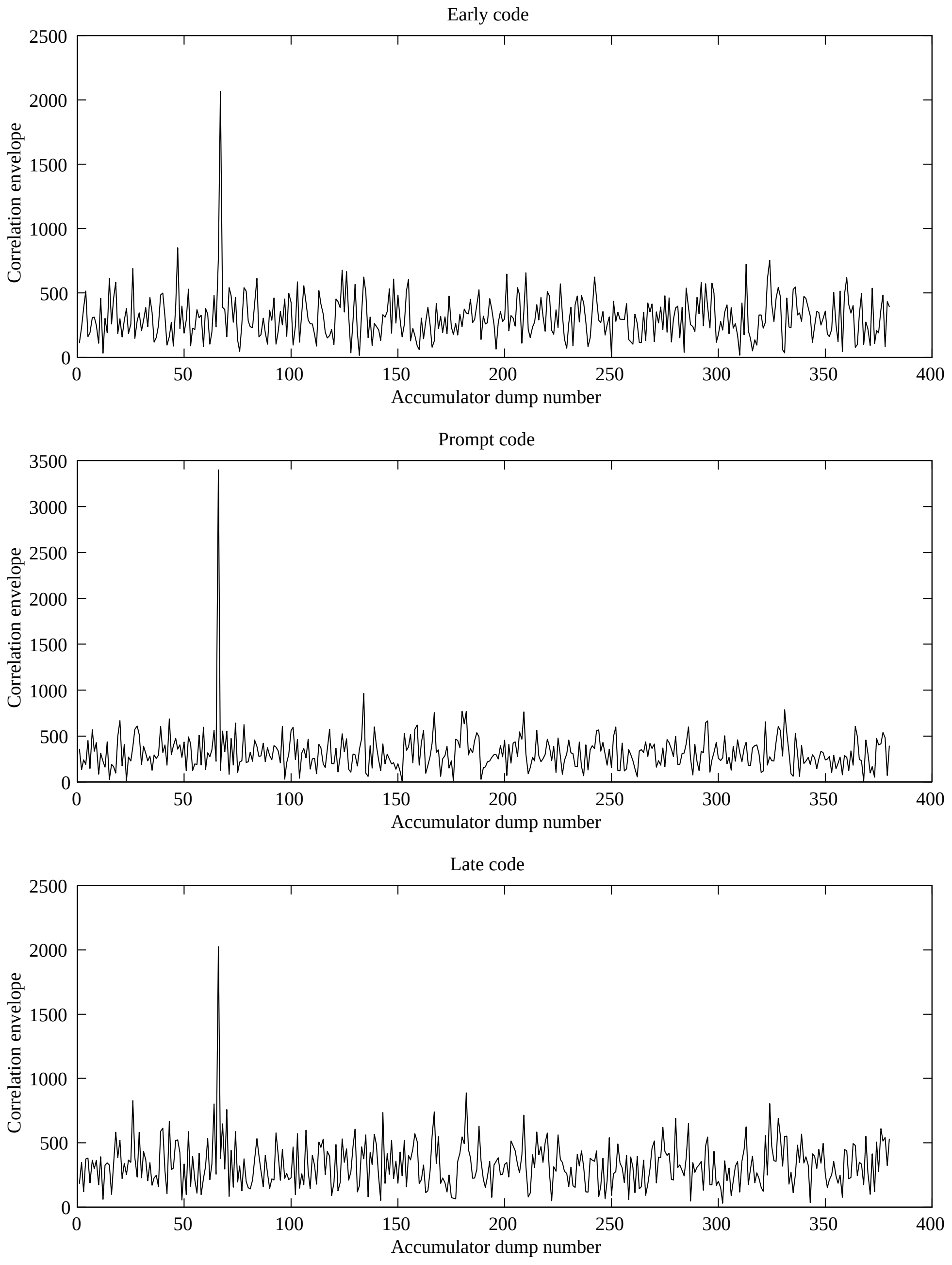

Figure 6.9: Correlation envelopes for one channel obtained when testing acquisition with the multi-channel GPS digital receiver.

6.3 Further work

Some further work remains to be done before the FPGA implementation can be useful as a fully functional digital receiver. The control logic should be extended so it is fully functional. Not all measurements are available code phase, epoch and carrier cycle counters should be added. There is plenty of space left in the XC4036XLA that was used as the target device. Since the current implementation only uses 35% of the available CLBs, there should be no problems to implement control logic and interfaces. The most obvious would be to implement a processor interface and let the processor close the tracking loops. It could even be possible to close the tracking loops in the FPGA. Then the loop discriminators and filters have to be implemented in the FPGA. This approach would decrease the load of the microprocessor, but of course a microprocessor will always be required.

It has been shown that it is possible to implement a multi-channel GPS digital receiver in an XC4000X FPGA. Implementing more than 8 channels is not unrealistic.

Chapter 7

Conclusions

The intention with this thesis work is defined by its title - Evaluation of a Reconfigurable Computing Engine for Digital Communication Applications. Three designs are implemented on the FPGA based WILD-ONE^{TM} PCI card:

- An 8-bit data acquisition system.
- A GPS signal generator.
- A multi-channel GPS digital receiver.

The 8-bit data acquisition system proved to have high performance. It is also flexible in that sense it is easy to modify for other ADC boards than the board that was used. Tests revealed that sample rates up to 20 MHz are achievable. The bus mastering capabilities of the WILD-ONE^{TM} board allows efficient transfers over the PCI without significantly loading the host PC CPU.

The GPS signal generator is basic. Two satellite signals are generated at fixed carrier and code frequencies. Further work on the GPS signal generator will however make it more useful.

The the multi-channel GPS digital receiver is implemented by interleaving hardware resources in time. The interleaving is accomplished by using the distributed synchronous RAM in the XC4000X FPGA. The RAM allows fast local access and efficient delay queues can be implemented. Having the target architecture in mind while designing with FPGAs will result in faster and denser designs.

The WILD-ONE^{TM} system has proved to be a very flexible platform for developing digital systems in general. The architecture is very flexible which gives the user many configuration options. Applications are not only limited to digital communication, the WILD-ONE^{TM} system is well suited for research projects involving digital systems in general.

Bibliography

[1] Kaplan, Elliott D.,Editor, *Understanding GPS : Principles and Applications*, Artech House, ISBN: 0890067937, March, 1996.

[2] Parkinson, Bradford W.; Spilker, James, J. Jr, Editors, *Global Positioning System: Theory and Application, Volume 1*, American Institute of Astronautics and aeronautics, ISBN: 156347106X, June 1996

[3] *ICD-GPS-200 (GPS Interface Control Document)*, GPS Joint Program Office (prepared by ARINC Research), October 1993.

[4] Braasch, Michael;Van Dierendonck, A. J., *GPS Receiver Architectures and Measurements*, Proceedings of the IEEE , Vol. 87, No. 1, January 1999, pp. 48-64

[5] Peterson, Roger L.; Ziemer, E. Rodger; Borth, David E., *Introduction to Spread Spectrum Communications*, Prentice Hall, ISBN: 0-02-431623-7, 1995.

[6] Ward, Phillip W., *GPS receiver Search Techniques*, IEEE PLANS, Position Location and Navigation Symposium, April 1996, pp. 604-611

[7] Coenen, A.J.R.M.; Van Nee, D.J.R., *Novel fast GPS/GLONASS code-acquisition technique using low update rate FFT*, Electronics Letters Vol 28 No. 9, Apr 23 1992, pp. 863-86

[8] Akos, Dennis M; Tsui, James B.Y., *Design and implementation of a direct digitization GPS receiver front end*, IEEE Transactions on Microwave Theory and Techniques, Vol. 44, No. 12/2, December 1996, pp. 2334-2339

[9] Akos, Dennis M.; Braasch, Michael S., *Software radio approach to global navigation satellite system receiver design*, Proceedings of the National Technical Meeting, Institute of Navigation, 1997, pp. 532-542

[10] Xilinx Inc, `http://www.xilinx.com`

[11] The programmable Logic Jump Station, OptiMagic Inc, `http://www.optimagic.com`

[12] Brown, Stephen; Rose, Jonathan, *FPGA and CPLD architectures: A tutorial*, IEEE Design & Test of Computers, Vol.13, No. 2, Summer 1996, pp 42-57.

[13] Alfke, Peter, *Third-Generation Architecture Boosts Speed And Density of FPGAs*, Electro International, 1991, pp 340 -345.

[14] The Programmable Logic Data Book, Xilinx Inc, 1999.

[15] WILD-ONE^{TM} Reference Manual ver 3.2, Annapolis Micro Systems, 1999.

[16] Fross, Bradley K.; Hawver, Dennis M.; Peterson, James B., *WILDFIRE^{TM} Heterogenous Parallell Processing Systems*, Annapolis Micro Systems Inc, Proceedings of the International Parallel Processing Symposium, 1998, pp. 611-615

[17] Annapolis Micro Systems Inc, `http://www.annapmicro.com`

[18] Model Technonology Inc, `http://www.model.com`

[19] Synplicity Inc, `http://www.synplicity.com`

[20] Microsoft Inc, `http://www.microsoft.com`

[21] AD9057 Datasheet, Analog Devices, 1997.

[22] Alfke, Peter, *Synchronous and Asynchronous FIFO Designs*, Xilinx Inc, Application note XAPP052, ver 2.0, September 1997.

[23] Råde L.; Westergren, B, *Beta Mathematics Handbook*, Studentlitteratur, Lund, 1990.

[24] Alfke, Peter, *Efficient Shift Registers, LFSR Counters, and Long Pseudo Random Sequence Generators*, Xilinx Inc, Application note XAPP052, July 1996.

[25] *GP2000 - GPS Receiver Hardware Design*, Mitel Semiconductor, Application Note AN4855, Issue 1.4, February 1999.

[26] Aardom, Eric, *A Pipelined Multiplex Spread Spectrum Demodulator an its Applications in a VLSI Multi-System Navigation Receiver*, IEEE Second International Symposium on Spread Spectrum Techniques and Applications, November 29 - December 2, 1992.