

Copyright © 2002, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**ULTRA WIDE-BAND BASEBAND  
DESIGN AND IMPLEMENTATION**

by

Mike Shuo-Wei Chen

Memorandum No. UCB/ERL M02/42

18 December 2002

**ULTRA WIDE-BAND BASEBAND  
DESIGN AND IMPLEMENTATION**

by

Mike Shuo-Wei Chen

Memorandum No. UCB/ERL M02/42

18 December 2002

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering  
University of California, Berkeley  
94720

---

# **Ultra Wide-band Baseband Design and Implementation**

by Mike Shuo-Wei Chen

---

## **Research Project**

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### **Committee:**

---

Professor Robert W. Brodersen  
Research Advisor

---

(Date)

\* \* \* \* \*

---

Professor David Tse  
Second Reader

---

(Date)

---

# Ultra Wide-band Baseband Design and Implementation

by Mike Shuo-Wei Chen

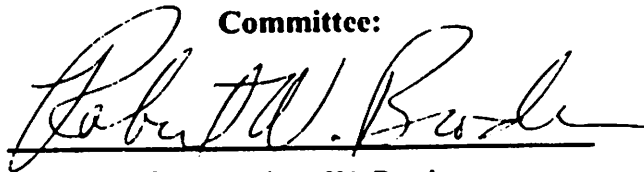
---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

**Committee:**

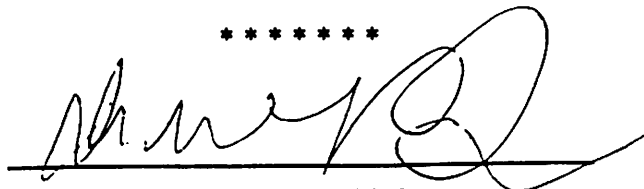


Professor Robert W. Brodersen  
Research Advisor

12/13/02

(Date)

\*\*\*\*\*



Professor David Tse  
Second Reader

12/16/02

(Date)

# TABLE OF CONTENTS

• CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION .....	1
1.2 SCOPE OF THIS WORK .....	2
1.3 ORGANIZATION OF THIS THESIS .....	2
• CHAPTER 2 UWB COMMUNICATION BACKGROUND	3
2.1 INTRODUCTION .....	3
2.2 CHANNEL CAPACITY .....	3
2.2 MAXIMIZE SYSTEM THROUGHPUT.....	5
2.2.1 Modulation Schemes .....	5
2.2.2 Spectral Efficiency and Processing Gain.....	6
2.3 POWER SPECTRAL DENSITY .....	8
• CHAPTER 3 SYSTEM SPECIFICATION	11
3.1 INTRODUCTION .....	11
3.2 DETECTION RULE .....	11
3.3 DERIVATION OF RECEIVED SIGNAL STATISTICS .....	14
3.4 PROCESSING GAIN AND THRESHOLD DETERMINATION .....	17
3.5 MONTE-CARLO SIMULATION .....	18
3.5 FINITE WORD LENGTH .....	20
• CHAPTER 4 FLEXIBLE UWB BASEBAND DESIGN	23
4.1 INTRODUCTION.....	23
4.2 BRIEF SYSTEM OVERVIEW: .....	23
4.3 FLEXIBLE SYSTEM PARAMETERS.....	25
4.4 OPERATION MODES (SYNCHRONIZATION):.....	27
4.4.1 Receiving Chain: .....	28
4.4.2 Acquisition Mode: .....	29
4.4.3 Code Tracking.....	38

4.5 UWB BASEBAND COMPONENTS .....	43
4.5.1 Matched Filtering.....	44
4.5.2 Correlation Block.....	46
4.5.3 Peak Detector.....	46
4.5.4 Data Recovery.....	47
4.5.5 Symbol Clock Generation .....	48
4.5.6 Control Logic.....	50
• CHAPTER 5 IMPLEMENTATION OF ASIC DIGITAL BACKEND	53
5.1 PULSE MATCHED FILTER.....	53
5.2 PN CORRELATOR AND DOWNSAMPLER .....	61
5.3 COEFFICIENTS REGISTERS .....	64
5.4 PN GENERATOR .....	65
5.5 PEAK DETECTOR .....	66
5.6 CONTROL LOGIC.....	67
5.7 PMF CONTROL SIGNAL DECODER .....	69
5.8 TOP LEVEL I/O.....	70
• CHAPTER 6 SYSTEM SIMULATION	72
6.1 INTRODUCTION.....	72
6.2 COMMUNICATION CHAIN MODELING .....	72
6.3 XILINX IMPLEMENTATION OF BASEBAND .....	76
6.3 FUNCTIONALITY SIMULATION .....	81
• CHAPTER 7 CONCLUSIONS	85
• REFERENCES	87

## LIST OF FIGURES

FIGURE 2-1 MODULATION SCHEMES AFFECT INPUT SNR V.S. THROUGHPUT .....	8
FIGURE 2-2 FCC REGULATION ON INDOOR SYSTEMS.....	10
FIGURE 3-1 BINARY DETECTION FOR SIGNAL PLUS NOISE.....	11
FIGURE 3-2 NEYMAN-PEARSON TEST .....	14
FIGURE 3-3 FINITE WORD LENGTH V.S. $E_b/N_{in}$ .....	21
FIGURE 4-1 UWB COMMUNICATION SYSTEM OVERVIEW .....	23
FIGURE 4-2 TARGET PARAMETERS OF THE SYSTEM .....	27
FIGURE 4-3 COHERENT DETECTOR.....	28
FIGURE 4-4 A) GPS SYSTEM SEARCH PATTERN. B) UWB SEARCH PATTERN. ....	29
FIGURE 4-5 MULTIPLE-DWELL DETECTION FLOW.....	32
FIGURE 4-6 A) ACQUISITION TIME V.S. SEARCHING PHASE B) SIZE V.S. SEARCHING PHASE. ....	37
FIGURE 4-7 PRODUCT OF ACQUISITION TIME AND AREA .....	38
FIGURE 4-8 EARLY-LATE TRACKING PROFILE A) RX PULSE MATCHES ON TIME PHASE B) RX PULSE COMES EARLIER .....	40
FIGURE 4-9 A) DLL TRACKING LOOP B) UWB TRACKING LOOP.....	41
FIGURE 4-10 BASEBAND OVERVIEW.....	43
FIGURE 4-11 PULSED MATCHED FILTER .....	46
FIGURE 4-12 DATA RECOVERY DIAGRAM.....	47
FIGURE 4-13 PROPOSED PACKET FORMAT .....	49
FIGURE 4-14 SYMBOL BOUNDARY.....	49
FIGURE 4-15 SYMBOL CLOCK GENERATION.....	50
FIGURE 4-16 EXAMPLE OF MAIN CONTROL LOGIC .....	52
FIGURE 5-1 SEQUENTIAL ADDER.....	54
FIGURE 5-2 LINEAR ADDER ARRAY .....	55
FIGURE 5-3 BINARY ADDER TREE.....	56
FIGURE 5-4 CARRYSAVE ADDER TREE.....	57
FIGURE 5-5 ARCHITECTURE EXPLORATION FOR CORRELATOR .....	61
FIGURE 5-6 DOWNSAMPLER ARCHITECTURES .....	62
FIGURE 5-7 ARCHITECTURES OF PN GENERATOR.....	65
FIGURE 6-1 COMMUNICATION CHAIN OVERVIEW .....	73



FIGURE 6-2 A) IDEAL DOUBLET PULSE BY CURRENT LOOP ANTENNA B) MONOCYCLE PULSE BY MONOPOLE ANTENNA .....	73
FIGURE 6-3 MULTI-PATH CHANNEL MODEL AT CORY 2ND FLOOR .....	74
FIGURE 6-4 FRONT END AMPLIFIER MODELING.....	74
FIGURE 6-5 SAMPLE AND HOLD CIRCUIT AND QUANTIZER .....	75
FIGURE 6-6 SIGNAL WAVEFORM ALONG THE CHAIN .....	76
FIGURE 6-7 XILINX IMPLEMENTATION OVERVIEW.....	77
FIGURE 6-8 1-BIT MULTIPLICATION .....	78
FIGURE 6-9 PMF BLOCK .....	79
FIGURE 6-10 A) ACCUMULATOR B) DOWNSAMPLER .....	80
FIGURE 6-11 XILINX DATA RECOVERY BLOCK.....	80
FIGURE 6-12 A) MAX CELL B) MAX CELL TREE (PEAK DETECTOR).....	81
FIGURE 6-13 SIMULINK SIMULATION WITHOUT DRIFTING .....	82
FIGURE 6-14 SIMULINK SIMULATION WITH DRIFTING .....	83
FIGURE 6-15 RESULTS FROM FPGA SIMULATION .....	84

## LIST OF TABLES

TABLE 3-1 BER SIMULATION.....	19
TABLE 3-2 PROBABILITY OF ERROR SIMULATION IN ACQUISITION MODE.....	19
TABLE 3-3 FINITE WORD LENGTHS AT PMF OUTPUT V.S. BER.....	22
TABLE 5-1 ARCHITECTURE COMPARISON OF ADDER TREE.....	58
TABLE 5-2 AREA/DELAY VERSUS SIZE AND BIT WIDTH OF PMF .....	60
TABLE 5-3 I/O OF PMF .....	60
TABLE 5-4 AREA/DELAY OF PN CORRELATOR .....	63
TABLE 5-5 I/O OF PN CORRELATOR .....	64
TABLE 5-6 I/O OF MAIN CONTROL BLOCK.....	69
TABLE 5-7 I/O OF READOUT BLOCK.....	69
TABLE 5-8 TOP LEVEL I/O .....	71

## **Acknowledgment**

There is no way I could come through the course of the project without the help from these people and institutions. First of all, I would like to thank my advisor Professor Bob Brodersen for his support and guidance ever since I joined the group. His trust and insight has helped me out at the various time of this project. Also his personality and vision has offered me a great model of being a good researcher. The environment of BWRC that Bob and Jan have created is really unbeatable for graduate studies. I would also like to thank Prof. David Tse, who had many useful discussions with me. His lectures solidified my communication background, which proved to be very useful for this project.

Next, I would like to thank my great colleagues in BWRC. I'd like to thank our UWB fellows, Ian O'Donnell, and Stanley Wang. Ian has kindly helped me a lot in many aspects during the course, including carefully proofreading this thesis. His wide knowledge and unique viewpoints made this research project more interesting. Stanley was a great partner of both research and class projects. I also thank Engling Yeo for almost instantaneous response to all my questions in many digital chip implementation issues. I enjoyed very much to join his mountain biking team. Of course, I have to thank Ada Poon for many useful advice and discussions. There are of course these intelligent people that I could always discuss with. They are Yun Chiu, Chinh Doan, Brian Limketkai, Luns Tee, and Hans-Martin Bluethgen. People who have been kindly helping me use the in-house design flows, Rhett Davis, Brian Richards, Chen Cheng, Kimmo Kuusilinna, and Tina Smilkstein.

Last but not the least, my parents, younger brother, and my girlfriend, Wan-Hsuan, have always been the best supports in my life. Without their endless care and love, it is impossible for me to accomplish these tasks. I am so fortunate to be born in this family and have my brother, Wei-Ming, to grow up together and also Wan's accompany makes my life more complete and enjoyable. Of course, I will not forget all my best friends in Taiwan and those great times we had created together before. Thank you all.

# Chapter 1 Introduction

## 1.1 Motivation

Ultra-Wideband (UWB) technology has gradually drawn people's attention recently. The technology itself already existed since the 1980's, and was mostly used for radar applications. In 1998, the FCC first proposed UWB transmission under Part 15 rules (limits on intentional and unintentional radiators in unlicensed bands), and asked for the comments from the industry. In Feb. 2002 it issued a First Report and Order [FCC02] that permits the marketing and operation of certain types of new products incorporating ultra-wideband. UWB becomes promising in various application areas, such as imaging systems, vehicular radar systems, and communication and measurement systems.

UWB was first defined by FCC as communication bandwidth that is more than 25% of the carrier frequency. This is much wider than any existing communication system. This wide bandwidth allows UWB to not only have a fine timing resolution but also more degree of freedom to use. Thus, one application in positioning has been under progress [Aetherwire]. Another main consumer application is targeting short-range and high-speed communication [XtremeSpectrum] due to the larger degree of freedom for UWB.

From a hardware implementation perspective, an UWB transceiver inherently has lower complexity given the fact that RF carrier is eliminated. The information is directly modulated on the pulses. It reduces the cost for both analog front-end and baseband design. Building a low power and low cost UWB system is the goal of this project. Another concern about UWB is the interference with other systems that reside within the same bandwidth. Therefore, our focus is so called "undetectable" UWB, which is transmitting the pulses under noise floor to avoid degrading existing systems.

## **1.2 Scope of This Work**

One objective of this project is to implement the UWB digital backend. The system is mainly digital, and analog data is sampled with sub-nanosecond spacing. The basic functionality of digital backend is to do signal acquisition and data tracking. Among the many issues, this report will cover from system design to hardware implementation in Module Compiler and FPGA.

The research target of the baseband design is low power consumption and high flexibility. Low power consumption comes from the low duty-cycle communication mode where one could slow down the pulse transmission rate and relax timing constraint for baseband. Then, we could reduce the supply voltage and push the unused blocks into sleep mode for power saving. The flexibility issue arises from our desire to experiment with UWB, such as changing the data rate or pulse shape, etc. The baseband has to fulfill these missions.

## **1.3 Organization of This Thesis**

The remainder of the thesis has six chapters. The structure of this report starts from communication perspectives and turns into circuit design. Chapter two gives a basic communication background for UWB, such as data rate and modulation scheme. Chapter three determines the system specifications for the baseband. Issues like number of PN chips and the detection threshold required for the baseband will be covered. Chapter four focuses on system design. It first describes the functionality and operational modes of the baseband and then brings out how we implemented the system. Chapter five goes into the core of each system block including the architectural explorations in Module Compiler. Chapter six provides an implementation on FPGA and Simulink system simulation. Finally, chapter seven concludes with the status of the current work and future tasks.

# Chapter 2 UWB Communication

## Background

### 2.1 Introduction

This chapter will give some background about UWB communication. Since UWB communication makes use of an ultra-wide bandwidth, the first question would be how fast it could communicate. What kind of modulation should we use to approach this capacity? These issues will be discussed in the following sections.

### 2.2 Channel Capacity

In order to get some idea how fast UWB could communicate, let's start from AWGN channel. The capacity is already given by well-known Shannon capacity equation,

$$C = W \log_2 \left( 1 + \frac{P_{av}}{W \cdot N_0} \right)$$

If we transmit at the thermal noise level for 1 GHz bandwidth, the transmission power,  $P_{av}$ , is  $-84$  dBm given a 50-Ohm front-end [Razavi98]. Constant noise power spectral density,  $N_0$ , depends on the receiving antenna. From some measurement data taken in the Berkeley Wireless Research Center (BWRC) lab (courtesy of Ian and Stanley), the noise power integrated from 0 to 1GHz band is about  $-56$  dBm and is treated as white Gaussian noise. Under this scenario, the channel capacity is about 2 Mbps.

As a matter of fact, the noise is colored due to those narrowband interferers such as cellular phones, wireless TV channels, etc. Assume the transmitted power is evenly distributed over the entire band, the channel capacity is calculated as,

$$C = \int_w \log_2 \left( 1 + \frac{P_{av}}{W \cdot N(f)} \right) df$$

The channel capacity goes up to 20 Mbps, almost ten times faster than AWGN channel.

Furthermore, if the transmitter knows the channel response, the channel capacity could be higher according to information theory, using “water-filling” [Cover91]. The idea is to put more power on those bands whose noise is lower while remain the total transmission power the same.

$$C = \int_{-w}^w \log(1 + S(f) |C(f)|^2) df$$

$$\text{where, } S(f) = \left( \lambda - \frac{1}{|C(f)|^2} \right)^+$$

$$\text{while satisfies, } \int_{-\infty}^{\infty} S(f) df = P \text{ the total transmission power}$$

$\lambda$  is the water level for weighting transmission power, while  $C(f)$  is the channel response. The channel capacity according to the measured data is about 40 Mbps as long as the transmitter could do power budgeting over different frequency bands. Of course, the channel measurement should be done in advance and be known to the transmitter. Power control in pulse-based communication is difficult to implement. Therefore, this water filling method simply gives us an idea how fast such a wideband system can transmit at the time being.

## 2.2 Maximize System Throughput

In the following sections, we will discuss the method to approach AWGN channel capacity calculated from previous section. This will depend on different modulation techniques and spectral efficiency we choose.

### 2.2.1 Modulation Schemes

For this pulse-based digital communication, the modulation schemes considered here are PAM (Pulse amplitude modulation), PPM (Pulse position modulation) and Biorthogonal signaling (PAM plus PPM) [Proakis00].

- PAM

PAM modulates the information on the amplitude of a transmitted pulse. 2-PAM, also known as antipodal signaling or BPSK, sends a positive pulse when bit “one” is sent, and negative one for bit “zero”. Due to its simplicity compared to M-PAM, antipodal signaling is the targeted modulation in order to avoid pulse amplitude control and automatic gain control circuitry. Under 2-PAM modulation, a processing gain may be needed to reduce error probability. This is achieved by adopting direct sequence spread spectrum (DSSS) technique [Proakis00]. The transmitted signals are modulated with a certain pseudo random code. At the receiver side, it correlates with the same code, thus gaining more energy than just a single pulse.

- PPM

PPM is an orthogonal signaling technique in time. The information modulates on the position of pulses [Scholtz93]. The more positions it has, the more information contained per pulse. Because the signal set is orthogonal, the Euclidean distance between any two signals is less than BPSK. Therefore, the processing gain required for PPM should be larger than BPSK in order to keep the same performance. However, PPM is more



difficult to get acquired and has a more stringent constraint on timing since the information is carried by the timing offset. Processing gain in PPM is achieved by modulating the position of pulses with a certain pseudo random code, as with PAM above.

#### • BIORTHOGONAL SIGNALING

M-ary biorthogonal signaling is constructed from  $1/2M$  orthogonal signals by including the negatives of those. In our case, the modulation is using positive and negative  $1/2M$  PPM signals. Thus, the information carried on a single pulse is doubled compared to purely PPM. And the pseudo random code could be modulated on both amplitudes and positions.

### 2.2.2 Spectral Efficiency and Processing Gain

In this section, we will try to relate the data transmission rate with pulse rate,  $R_c = 1/T_{rep}$ , and input SNR. First of all, the processing gain needs to be determined.

The signal to noise ratio per bit,  $E_b/N_o$ , is the parameter people use to relate to error probability, which will be discussed more thoroughly in the next chapter. The figure that shows the relation between the two is known as waterfall curve. From the curve, we can know the detection  $E_b/N_o$  that receiver has to achieve for a certain error probability. For instance, one needs 7 dB  $E_b/N_o$  at the correlator output for  $1e-3$  error probability in BPSK modulation scheme [Proakis00]. Obviously, the required processing gain depends on input SNR. The stronger the received signal is, the less processing gain is needed. The following equation tells the relationship between the input SNR,  $P_{av}/(W \cdot N_o)$ , and the corresponding input  $E_b/N_o$ .  $R_c$  is pulse rate, and  $W$  is transmission bandwidth.

$$\frac{E_b}{N_o} = \frac{W}{R_c} \left( \frac{P_{av}}{W \cdot N_o} \right)$$

$R_c/W$  is also known as spectral efficiency,  $\rho$ . As the pulse rate  $R_c$  goes low, the efficiency drops due to the lower usage of bandwidth (degree of freedom). However, this provides some extra gain,  $1/\rho$ , to input  $E_b/N_0$ .

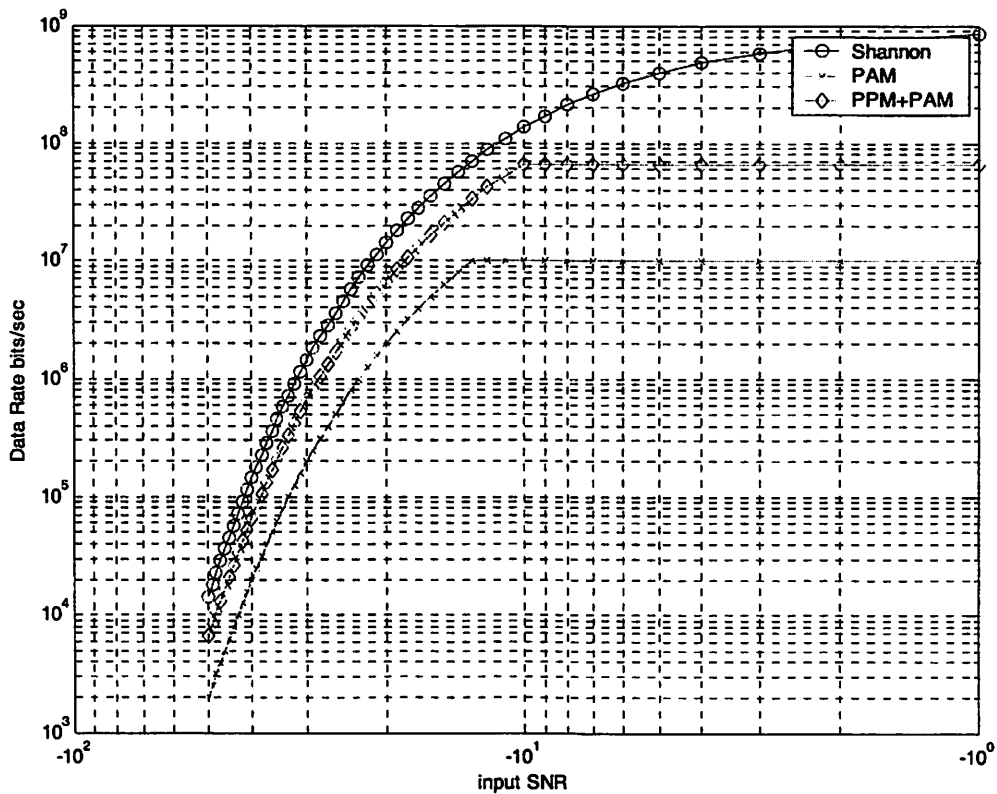
The gap between input SNR and detection  $E_b/N_0$  (10 dB in BPSK) should be compensated for by the DSSS technique, i.e. PN spreading. In other words, one information bit spreads over several pulses or chips. For example, if the gap is 20 dB, the system will require a hundred PN chips. Consequently, the data rate is a function of pulse rate and processing gain. When the required processing gain is not larger than unity, the data rate is equal to  $R_c$ , pulse rate.

$$Data\_rate = \frac{R_c}{num\_PN\_chips}$$

Therefore, given a fixed input SNR, if  $R_c$  is halved, the processing gain required will also be halved as long as it is still larger than unity such that the data rate remains the same. Note that in order to keep the same input SNR, the pulse energy should increase by the same ratio as  $R_c$  is decreased. This will keep the power spectral density the same. FCC regulates UWB emission according to power spectral density profile, which will be discussed in the next section.

So far, we have already gained enough background for thinking about how to maximize the throughput: PPM modulates the information on the pulse position within one pulse repetition period. The longer the repetition period, the more positions can be used for modulation. Plus, when we decrease the pulse rate, the required processing gain is also less (because the energy per pulse could scale up proportionally to keep the same power spectrum density). Therefore, in order increase the throughput, one should reduce the pulse repetition rate, and do biorthogonal signaling on the positions within the period. The optimal throughput would be achieved when the required processing gain is equal to unity.

Figure 2-1 shows the relationship between input SNR and throughput using different modulation schemes. The pulse rate was 10 MHz and Shannon capacity here is simply using AWGN channel model. Note that as the input SNR goes higher than a certain level, the data rate saturates. The reason is that no process gain is needed beyond that point, where the optimal throughput is achieved at that particular input SNR. If we keep reducing the pulse rate, the saturation point will move towards lower input SNR.



**Figure 2-1 Modulation schemes affect input SNR v.s. throughput**

### 2.3 Power Spectral Density

From the previous section, we learn that one should increase the pulse energy while decreasing the pulse rate in order to enhance capacity. However, the power spectral density is regulated by the FCC [FCC02]. Figure 2-2 shows the emission limit for indoor systems. Therefore, one has to reduce any spikes in the frequency domain.

Since the pulses are sent periodically, it could be expressed as,

$$s(t) = \sum_n a_n \cdot p(t - nT)$$

The power spectral density of these pulse train is the Fourier transform of the autocorrelation function of  $s(t)$ , and could be expressed as,

$$\Phi(f) = FT\{R_{ss}(t)\} = \frac{\sigma_a^2}{T} |P(f)|^2 + \frac{\mu_a^2}{T^2} \sum_{n=-\infty}^{\infty} |P(\frac{n}{T})|^2 \delta(f - \frac{n}{T})$$

$P(f)$  is the Fourier transform of  $p(t)$ . From the equation, one has to decrease the mean of  $a_n$  in order to suppress those spikes. If the spectral density is flat, then the transmission power can be maximized under the FCC regulation. This is another reason we introduce a pseudo random code to modulate with pulse amplitudes if PAM is used. Furthermore, these frequency spikes could be flattened even more if a pseudo random code is modulated on the position of pulse train (PPM).

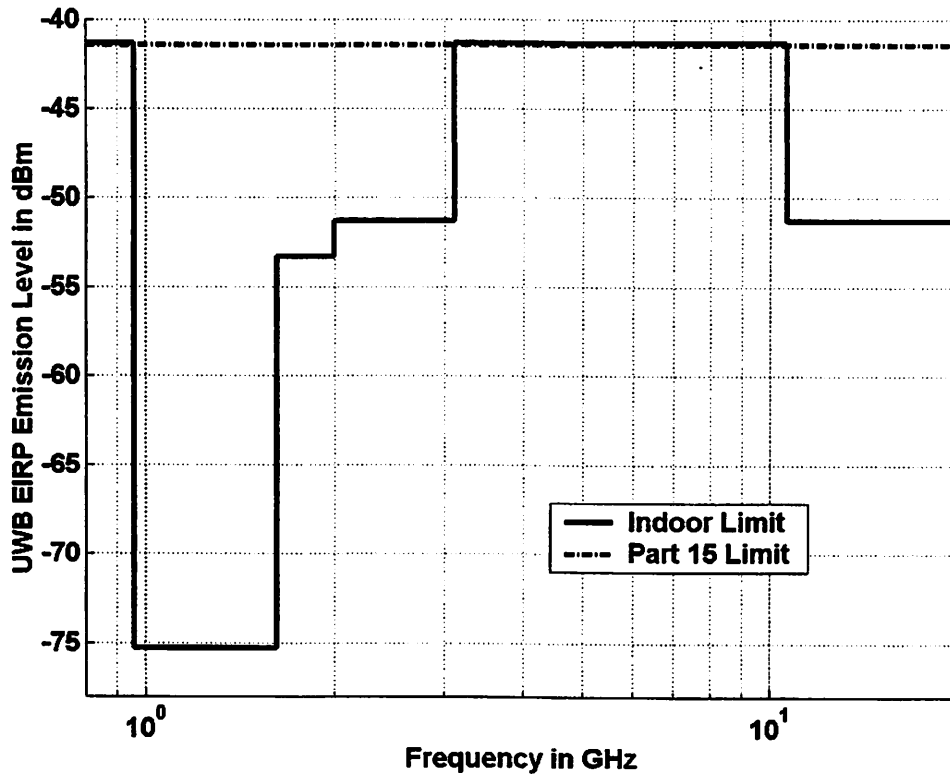


Figure 2-2 FCC regulation on indoor systems

# Chapter 3 System Specification

## 3.1 Introduction

This chapter will go through the determination of system specifications, including the number of PN chips, detection threshold, and effect of finite word length. The work has been done via hand calculation and Monte-Carlo simulation.

## 3.2 Detection Rule

From a communications point of view, the receiver is essentially a binary detection problem or two hypotheses testing given the BPSK modulation scheme. At the transmitter side, it sends "one" or "zero". During the synchronization mode, "one" means the presence of the signal while "zero" means absence. During data recovery mode, "one" and "zero" represent the information bit "one" and "zero" individually.

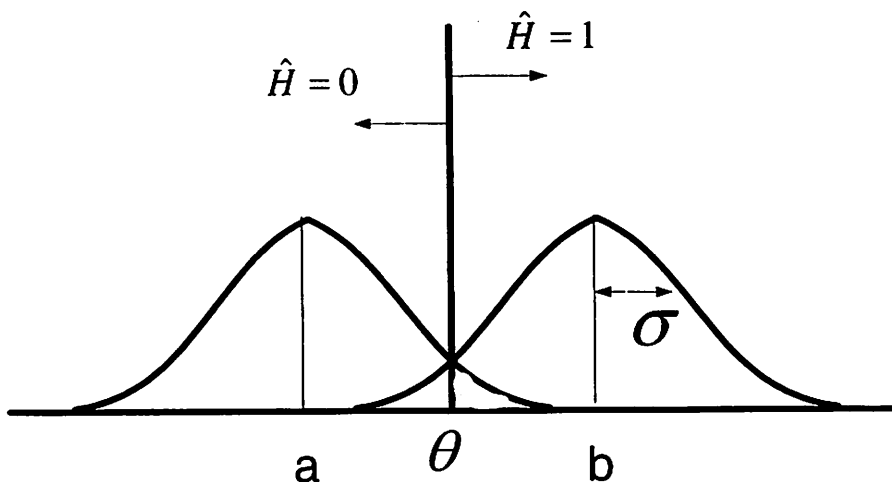


Figure 3-1 Binary detection for signal plus noise

Shown in Figure 3-1 is an illustration of the probability profile for two hypotheses testing. If “one” is sent, the detector will have value b. However, additive noise adds to the error probability. Here, we assume the noise is white and Gaussian, which is the worst-case scenario. Therefore, in UWB communication, the total power of interferers is approximated with a Gaussian noise of same power. The error probability is categorized as follows:

$$\Pr(\hat{H} = 1 | H = 0) \text{ False alarm probability;}$$

$$\Pr(\hat{H} = 0 | H = 1) \text{ Miss detection probability;}$$

The area under the little triangular area shown in Figure 3-1 is exactly the error probability. Now, the problem is how to set the detection threshold to minimize the aggregate error probability or any cost function of the system. Several detection schemes are commonly used. They are MAP detection, ML detection and Neyman-Pearson Test for optimizing a certain cost function [Tse01]. A generic detection problem setup is observing a random vector Y, and makes the best guess on the transmitted signal, H.

- MAP DETECTION

The maximum a posteriori probability (MAP) rule is to maximize the probability of guessing correctly. The rule may be written as a mathematical equation:

$$\hat{H}(\vec{y}) = \arg \max_i [P_{H|\vec{y}}(i | \vec{y})]$$

$$\Rightarrow \Lambda(\vec{y}) = \frac{P_{\vec{y}|H}(\vec{y} | 1)}{P_{\vec{y}|H}(\vec{y} | 0)} \geq \frac{P_0}{P_1} = \eta$$

MAP detection assumes the detector knows the prior probability of sending “one” or “zero”. Comparing the likelihood ratio,  $\Lambda(\vec{y})$ , to a certain threshold is called likelihood ratio test (LRT). Since the noise is Gaussian, we could replace the likelihood ratio with an exponential function, and take logarithmic of both sides. Then we could get a formula

for detection threshold  $\theta$ . Whenever observation,  $y$ , is larger than  $\theta$ , the receiver will choose “one”, and vice versa.

$$y \geq \frac{\sigma^2 \ln(\eta)}{b-a} + \frac{b+a}{2} = \theta$$

A slight deviation from the MAP rule is the Bayes test, which includes the cost function that the system may have. The MAP rule tries to minimize the total error probability. However, the false alarm and miss detection may cause different severity of penalty to the system. This is the reason why the Bayes test should be adopted. MAP rule suppresses both error probability to the minimum possible extent; however, it is possible to trade-off one of them while keeping the final cost function as small as possible. In UWB communication, this cost function could be acquisition time or power consumption. Once the cost function has been determined, one could set an optimal threshold.

Let  $C_{ij}$  be the cost function of guessing  $i$  while sending  $j$ . Then Bayes test is:

$$\hat{H}(\vec{y}) = \arg \min_i [C_{i0} P_{H|\vec{y}}(0|\vec{y}) + C_{i1} P_{H|\vec{y}}(1|\vec{y})]$$

$$\Rightarrow \Lambda(\vec{y}) = \frac{P_{y|H}(\vec{y}|1)}{P_{y|H}(\vec{y}|0)} \geq \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})} = \eta$$

- ML DETECTION

Similar to the MAP detection rule, but when the prior probability is unknown; the best thing that the receiver could do is simply depend on the maximum likelihood ratio,  $\Lambda(\vec{y})$ .

If  $\Lambda$  is larger than one, then receiver guesses one and vice versa.

- NEYMAN-PEARSON TEST



This is similar to the Bayes test except the prior probability is unknown. The Neyman-Pearson test fixes an upper bound for either one of the error probabilities, because one of them may have a much higher cost than the other one. For example, if the miss detection probability has a certain upper bound,  $\theta$ ; the problem is how to minimize the false alarm probability. It turns out that LRT is still the best rule to use. From LRT, if we vary the detection  $\eta$ , we could draw a tradeoff curve between miss detection and false alarm, Figure 3-2. When  $\eta$  is zero, Pfa is always one, and vice versa. Since Pm is bounded by  $\theta$ , in this case. We should sweep  $\eta$  from infinity to zero until Pm meets the bound. Then the LRT based on this  $\eta$  is the optimal detection rule.

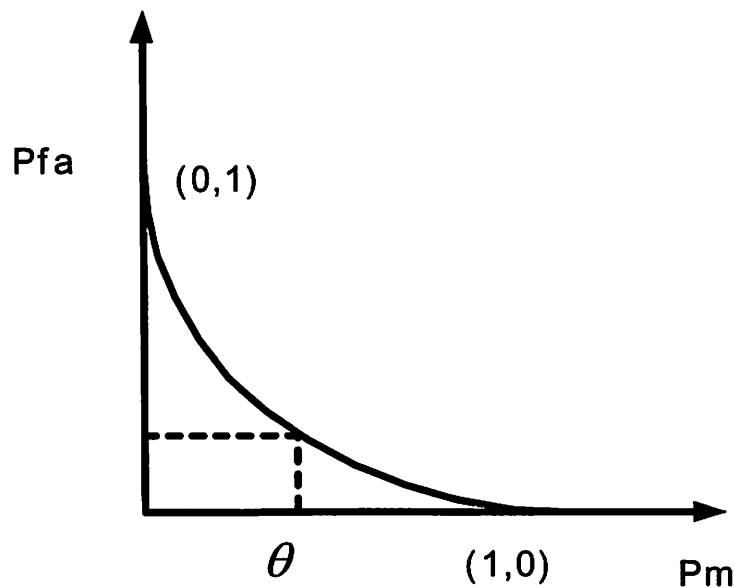


Figure 3-2 Neyman-Pearson Test

### 3.3 Derivation of Received Signal Statistics

Before we could really apply those detection rules described in the previous section, one needs to know the statistics of the received signal, i.e. mean and variance if the noise is

Gaussian distributed In other words,  $a$ ,  $b$  and  $\sigma$  has to be determined. So as to prior probability  $P_0$  and  $P_1$ , it's optional to the designer. In UWB, this prior probability depends on the pulse rate and sampling window size during the acquisition mode. If it is not that obvious what the prior probability is, one could simply use those detection rules without prior knowledge. To calculate  $a$ ,  $b$ , and  $\sigma$  is the topic of this section. Let's start from the infinite precision case and then examine 1-bit quantizer case.

Before we go, some parameters are used in the following derivation.

$N$ : number of PN chips

$L$ : number of matched filter taps

$S_i$ :  $i$ th sampled value from quantizer

$P_i$ :  $i$ th coefficient of matched filter ( $P_i=S_i$ , if matched filter is perfect)

$E_b$ : signal energy

$\sigma_n$ : square root of noise power

Corr: final correlation output assuming totally synchronized

- INFINITE PRECISION

The final stage correlator output, Corr, could be written as follows:

$$Corr = \sum_{i=1}^N \sum_{j=1}^L (S_j + n_j) P_j$$

The 1<sup>st</sup> and 2<sup>nd</sup> moment of the random variable, Corr, could be derived very straightforwardly:

$$E(Corr) = N \sum_{j=1}^L (S_j \cdot P_j)$$

$$Var(Corr) = E((Corr - \overline{Corr})^2) = N \sigma_n^2 \sum P_j^2$$

The mean of Corr will be equal to  $(N \cdot E_b)$  as long as  $P_i$  is  $S_i$ , i.e. a perfect matched filter. Therefore, the output SNR at this perfect situation is:

$$SNR_{out} = \frac{[E(Corr)]^2}{Var(Corr)} = N \left( \frac{E_b}{\sigma_n^2} \right) = N \cdot SNR_{in}$$

This equation tells exactly how DSSS could help the system performance. The SNR is proportional to the number of chips.

- ONE BIT QUANTIZER (SLICER)

As the quantizer moves from infinite precision to lower precision, the impairment to the system could be modeled as another additive white Gaussian noise source (quantization noise). As it is eventually reduced to 1-bit quantizer, the system will become nonlinear. A more detailed derivation of mean and variance values is required.

-Calculate  $\|a\|$  (only AWGN noise existed):

$$Corr = \sum_{i=1}^N \sum_{j=1}^L sign(n_j) P_j$$

$$E(Corr) = N \sum_{j=1}^L (E(sign(n_j)) \cdot P_j = 0 = \|a\|$$

-Calculate  $\|b\|$

$$E(Corr) = N \sum_{j=1}^L (E(sign(S_j + n_j)) \cdot P_j$$

$$E(sign(S_j + n_j)) = 1 \cdot \Pr(n_j \geq -S_j) + (-1) \cdot \Pr(n_j < -S_j)$$

$$\Pr(n_j \geq -S_j) = Q\left(\frac{-S_j}{\sigma_n}\right)$$

$$\Rightarrow Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz$$

$$\therefore \|b\| = N \cdot \sum \left( Q\left(\frac{-S_j}{\sigma_n}\right) - (1 - Q\left(\frac{-S_j}{\sigma_n}\right)) \right) P_j = N \cdot \sum (2 \cdot Q\left(\frac{-S_j}{\sigma_n}\right) - 1) P_j$$

-Calculate Var(a)

$$\sigma_a^2 = E\left(N \sum_{j=1}^L E(\text{sign}^2(n_j)) \cdot P_j^2\right) = N \sum P_j^2$$

-Calculate Var(b)

$$\begin{aligned} \sigma_b^2 &= E\left(\sum_{i=1}^N \sum_{j=1}^L [( \text{sign}(S_j + n_j) P_j - \overline{\text{sign}(S_j + n_j) P_j} )^2]\right) \\ &= \sum_{i=1}^N \left( \sum P_j^2 - \sum \overline{\text{sign}(S_j + n_j) P_j^2} \right) = N \cdot \left[ \sum P_j^2 - \sum \left( (2 \cdot Q\left(\frac{-S_j}{\sigma_n}\right) - 1) P_j \right)^2 \right] \end{aligned}$$

So far,  $\|a\|$ ,  $\|b\|$ , Var(a), Var(b) have been derived. These values correspond to the binary detection plot shown in Figure 3-1.

### 3.4 Processing gain and Threshold Determination

Given the input statistics, we've already gained enough information to determine the number of chips required and the detection threshold. They are all related to each other.

The first thing to do is to decide what the error probability is targeted. This probability has a slightly different meaning in the two operational modes. In tracking mode, this error probability is translated into BER. Since the modulation scheme is BPSK, the output SNR could be determined by looking into waterfall curve given a certain BER. The output SNR can be expressed as,

$$SNR_{out} = \frac{\|b\|^2}{\sigma_b^2} = f(N, S_j)$$

Therefore, given a certain waveform, one could decide the number of PN chips that the system requires. This detection rule is simply doing sign detection, in other words, the threshold equals to zero since the transmitted signal is either one or negative one.

In acquisition mode, the probability of error means missing detection or false alarm. The error probability depends on the detection threshold, number of PN chips and pulse shape. If one wants to keep the same error rate as tracking mode, usually it requires double PN chips because the system has to choose between one and zero, instead of one and negative one. In other words, the Euclidean distance is halved. The determination of miss detection or false alarm probability should depend on the cost function of the system, such as average acquisition time, which will be described in the next chapter. In the baseband design, we keep the threshold programmable by the user so that we can test out different detection rules.

### **3.5 Monte-Carlo Simulation**

If we constraint ourselves to hide the UWB pulse spectrum under the thermal noise floor, the peak value of UWB pulse in time would be comparable to RMS value of the noise floor.

For the worst case of doing synchronization, we first treat thermal noise and interferers as white Gaussian noise, which is proved to be the worst-case scenario for the capacity of a channel. Therefore, we measure the power spectrum received by the receiving antenna that we are going to use in the Lab. And sum up the total noise power over the interested band, as the variance of white Gaussian noise.

As we mentioned in the previous section, for reliable communication, BPSK requires a 7dB input  $E_b/N_0$  for BER of  $10^{-3}$ . However, in order to keep the acquisition error probability at roughly the same level, one needs 3 dB more signal energy. In allowance for a safe margin, the targeted output  $E_b/N_0$  is set to be 10 dB. Therefore, (10dB - input  $E_b/N_0$ ) is the extra processing gain required by using DSSS. Once we have the worst-case input  $E_b/N_0$ , then the maximum PN chips required for the baseband will be determined as well.

Running the BER simulation for a monopole antenna and testing for 50000 information bits, the signal energy,  $E_b$  is equal to  $3.6727e-16$  ( $V^2$ ), which corresponds to a peak of 1mv. From the oscilloscope, the noise plus interference floor is mostly around 10 mV. That is, we transmit the signal underneath the thermal noise floor, which should be under the sensitivity of most narrowband systems. The noise power is measured from spectrum analyzer, and we found the noise power density is around  $-70\text{dBm/MHz}$ . Therefore, the  $E_b/N_0$  used in this Monte-Carlo simulation is  $-11.34$  dB. Table 3-1 shows how precision of PMF and number of chips affect BER.

Another long run simulation for miss detection and false alarm probability has been done in Table 3-2 assuming receiver could lock the signal once it's present. There, we simply set the threshold to be the middle point of  $\|a\|$  and  $\|b\|$  in order to get an idea of how system performs. As described in detection rule, one could optimize the performance even more by tuning the detection threshold.

Chips	BER(perfect pmf)	BER(5-bit pmf)	BER(3-bit pmf)
1	0.38	0.3424	0.3842
10	0.165	0.1663	0.1849
100	0.0011	0.0011	0.0069
200	~0	$2e-5$	~0
1000	~0	~0	~0

**Table 3-1 BER simulation**

Chips	False alarm rate	Miss detectoin rate	$E_b/N_0$ @ detection
300	0.0041	0.0037	14.4245 dB
400	0.0013	0.00086	15.6643 dB

**Table 3-2 Probability of error simulation in acquisition mode**

If we start from input  $E_b/N_0 \sim -10\text{dB}$ , the simulation shows that 400 chips are needed to suppress the miss lock rate below  $1e-3$ . Once the signal gets locked, then we could

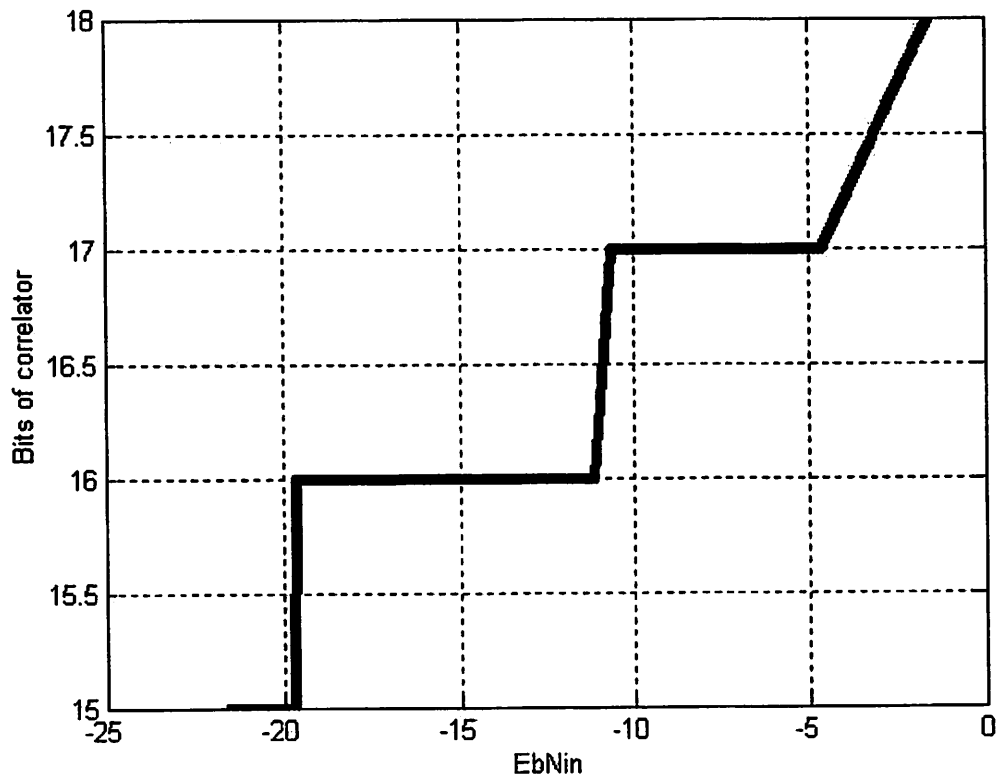
simply apply BPSK, meaning we could reduce the required spreading gain to 100 chips. The reason why false alarm probability is higher than miss detection is because the variance of noise-only inputs is larger than that of signal plus noise inputs, which matches to the hand calculation.

### 3.5 Finite Word Length

Up to this point, we have specified the input signal to be 1-bit and the PMF coefficients 5-bit wide. And we are summing 128 5-bit numbers in PMF while correlating over a maximum of 1024 chips. If we do not have the prior knowledge about the input statistics, one needs a  $(5+7+10)=22$ -bit precision at the correlator output in order to prevent overflow.

Since we have already derived the input statistics in section 3.3, we should be able to save some hardware. Usually we use the three-sigma rule for finite word length determination. Equivalently, if the signal on a certain node has a standard deviation,  $\sigma$ , the finite word length should keep a range of  $+3\sigma$  to  $-3\sigma$ . The probability of overflow will be kept to less than 1% given a Gaussian distribution. For those who are interested in more details in finite word length effects, [Shi02] is a good reference.

Input statistics (mean and variance since we assume a Gaussian distribution) depends on pulse shape, the number of chips and the input SNR. Under the three-sigma rule and with a PN length of 1024 chips, one needs 16 bits for the correlator and 6 bits for the PMF output with the input waveform previously used for BER simulation. However, it varies with the input noise level and pulse shape. Given the same pulse shape and BER, bit widths increase from 15 bits to 18 bits as the SNR increases, shown in Figure 3-3. Bit width does not increase too much over the input SNR variation. If input  $E_b/N_0$  is around  $-10$  dB, the bit width is around 17.



**Figure 3-3 Finite word length v.s. EbNin**

However, if the received pulse shape changes, it will cause the most variation on the bit precision. Say the environment has more multipaths or the pulse width is wider, then the bit width could grow to 22 bits at most.

Since the simulated signal to noise ratio at the output is roughly 15 dB. The number of bits required at the PMF and PN correlator do not need to keep the full precision. Therefore, we could simply keep several MSBs at the output of PMF while not deteriorates the performance too much. Table 3-3 is the long run simulation for 100 chips. It shows 5 bits is enough given the pulse shape. Note there is a growing discrepancy between miss detection and false alarm owing to the fact that slicing the LSB of PMF outputs will make the value smaller than expected. In order to balance the two error probabilities, again, one could simply tune down the threshold. Here, the threshold is somewhat too large.



<b>Bits @ PMF output</b>	<b>BER</b>	<b>Miss detection</b>	<b>False alarm</b>
8	0.0018	0.0729	0.0699
7	0.0018	0.0777	0.0655
6	0.0019	0.0822	0.0624
5	0.0019	0.0934	0.0541
4	0.0023	0.1254	0.0415
3	0.0049	0.2267	0.0263
2	0.0513	0.7173	0.0069

**Table 3-3 Finite word lengths at PMF output v.s. BER**

# Chapter 4 Flexible UWB Baseband Design

## 4.1 Introduction

This chapter will first give a brief system overview of the project, from transmitter through multipath channel, and analog frontend to the digital backend. A system simulation environment has been built in Simulink and will be discussed in Chapter 6. The emphasis of this chapter is designing a low power and flexible baseband. System parameters will be defined first as a basis for designing the UWB baseband. And then the most important issue, i.e. synchronization, will be brought up with a thorough discussion of operation modes, i.e. signal acquisition and data tracking, and how one could coordinate between the two. After understanding how the system synchronizes, we will go through each functional block of the whole baseband, including the pulse matched filter, PN correlation block, peak detector, control logic, and data recovery block, etc.

## 4.2 Brief System Overview:

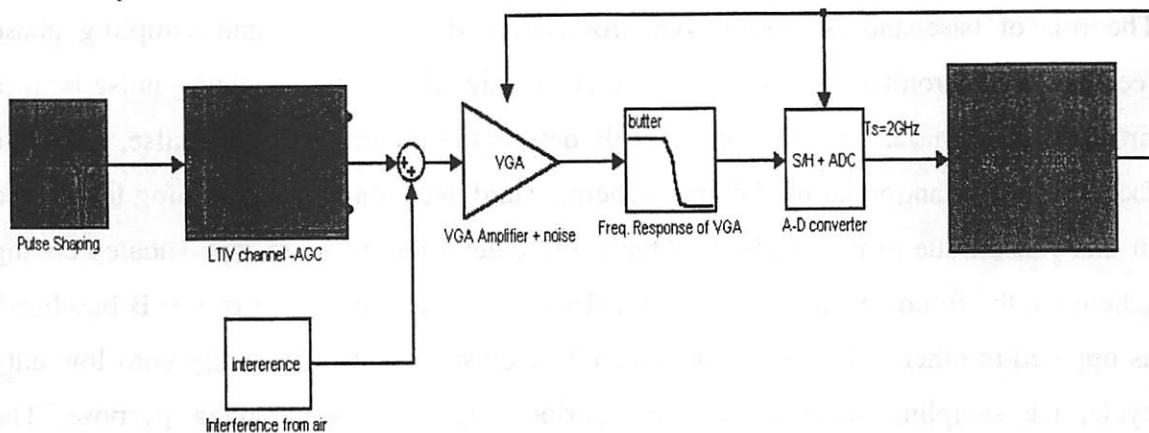


Figure 4-1 UWB communication System Overview

The architecture of this system is essentially a digital radio in the sense that the digital backend is brought to antenna as close as possible. Only amplification, filtering and quantization are left in analog frontend. Figure 4-1 shows the whole chain of UWB communication system. A UWB pulse is generated via a pulser and a wideband antenna. The pulse shape can be a Gaussian-like pulse or a monocycle one, depends on which type of antenna being used (current mode versus voltage mode antenna). The bandwidth of the pulse is from 10's of MHz to a couple of GHz range. Such a wideband signal goes through a multipath channel, which may distort the pulse shape but offers the possibility of using Rake receiving. Theoretically, due to the wideband signal, the resolvable paths should be more than narrowband signal, which implies more energy can be collected via Rake receiving.

The signal is received through antenna, then amplified by the LNA, gain stages and finally filter to reduce out of band noise or interference. And the signal is directly sampled at fairly high speed but low resolution. Current specification is aiming at a 2 GHz sampling rate, with 1-bit resolution, which is basically a zero crossing detection. At such a high sampling rate, there is no way to serially feed the data stream into digital backend in 2 Gbps rate. Therefore, the sampled data is buffered right after ADC stage, and feeds into baseband in a parallel array of 1-bit data at pulse repetition rate which may vary from 100 MHz to 1 MHz.

The role of baseband is mainly synchronization, data recovery and sampling phase feedback. Synchronization lines up the reference signal with the incoming pulse both in time and code phase. Data recovery simply detects the polarity of UWB pulse, given the fact that we use antipodal modulation scheme. Hard decision is the one going to be used in this version due to its simplicity. One could extend this to more sophisticated coding scheme in the future. Sampling phase feedback is a different feature of UWB baseband, as opposed to other DSSS ones. Because UWB pulse is sent periodically with low duty cycle, the sampling window activates periodically for power saving purpose. The

sampling phase feedback contains the information of when to activate the sampling window. More details will be covered in the later part of this chapter.

### 4.3 Flexible System Parameters

In order to test and understand the capability of an impulse-based UWB system, the goal of this project is to build a flexible receiver. Flexibility in baseband includes the following items:

1) Different random codes:

Different codes, such as Barker codes, Gold codes, Frank codes, etc, have their own cross-correlation properties. In terms of good acquisition, the Barker code has a very good cross-correlation, with sidelobe values less than or equal to  $1/N$  in size and uniform distribution [Taylor00].

2) Different length of codes:

From previous chapter, we know the length of code depends on how bad the environment is. If the interference is large, a long PN code will be needed. The worst-case is 1024 chips.

3) Pulse Repetition Rate ( $1/T_{rep}$ ):

The pulse repetition rate may range from 100MHz to 1 MHz. As the pulses are close to each other, the delay spread may overlap with the next chip pulse, which reduces the signal energy the receiver can capture given the topology we use and increases the inter chip interference. Also the periodical on-off type of receiving strategy can be no longer used. Instead, one has to always keep the radio alive all the time. The fastest pulse repetition rate depends on pulse duration. The shortest pulse duration we are targeting could be 10ns, which corresponds to 100MHz maximum rate.

4) Pulse Width ( $T_{pulse}$ ):  $< 16ns$  (32 samples)

The pulse width depends on pulser and antenna. If the pulse is Gaussian, then 1 GHz bandwidth corresponds to 2 ns half cycle pulse duration. Plus the duration between the first and second half cycle, the total length of UWB pulse estimated is around 10ns.

5) Pulse ripple length (Tripple):  $\leq 64\text{ns}$  (128 samples)

The pulse ripple length is the length of received signal, which is the convolution response of pulse shape with multipath channel response. Some measurements of UWB channel response have been done in [Molisch02], [Scholtz98], [Foerster01] and [Rusch01]. The result shows the RMS delay spread of indoor environment is ranging from 20ns to 60ns. Most of the energy is within 100ns. Again, this delay spread depends on the setup of environment. However, we could still get a rough idea of how long the delay would be.

6) Sampling period:

The sampling rate is 2 GHz, due to the 1GHz bandwidth of UWB pulse. Therefore, the system samples at Nyquist rate, which preserves all the sufficient statistics for the digital backend. Recent research explored the possibility of doing sub-Nyquist sampling as shown in [Vetterli02]. Although it inevitably degrades the system performance by some degree, however, it may imply a simpler implementation for receiver. In this project, we still implement Nyquist sampling.

7) Sampling window length ( $T_{\text{win}}$ ):  $\leq 128\text{ns}$  (256 samples)

In order to take advantage of repetitive sending pattern of UWB pulse, the sampling circuit should also be activated periodically. The baseband will search through all possible PN code phases given a certain sampling window phase. If no signal has been detected at the current sampling window phase, it will shift by a fixed amount of time offset, and then continue another cycle of the detection procedure.

To avoid losing the incoming signal, the amount of time in the sampling window has to be less than ( $T_{\text{win-Tripple}}$ ). In this report, the time offset is referred as effective sampling window size,  $T_{\text{weff}}$ , because this is the effective time that has been searched through. The procedure of signal acquisition will be discussed more in the next section.

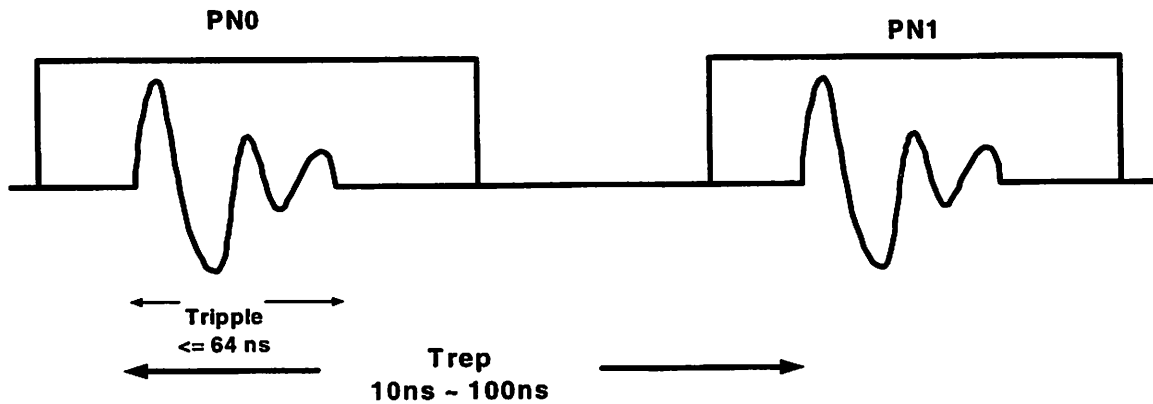
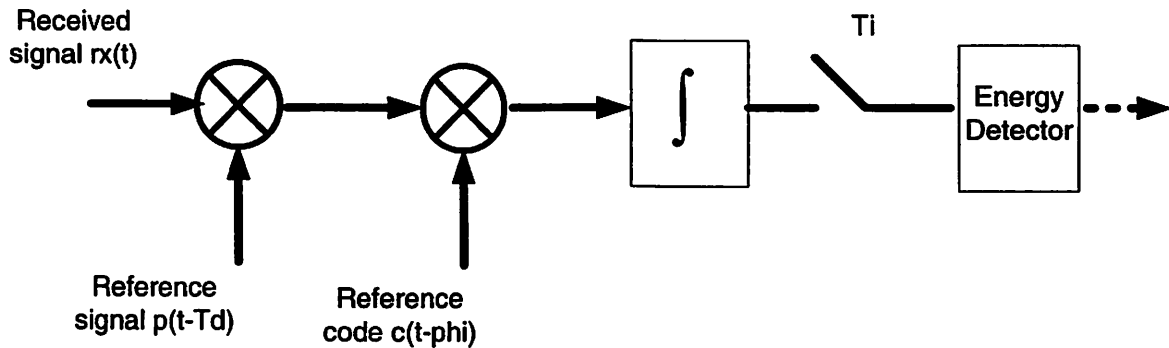


Figure 4-2 Target parameters of the system

#### 4.4 Operation Modes (Synchronization):

The most important role of the UWB baseband is synchronization, which is composed of initial acquisition of the signal, and code tracking and the transition between the two. The synchronization mechanism is similar to a GPS receiver [Kaplan96] or a CDMA cellular system (IS-95), due to the usage of DSSS technology. The main difference is the absence of carrier in UWB communication. In narrowband system, like GPS, the acquisition and tracking is a two-dimensional (code and carrier) matching process. One has to consider the carrier frequency offset caused by Doppler effect or oscillator mismatch. Therefore, carrier recovery or frequency compensation could be ignored in UWB, which is one of the simplicities that UWB possess

In this section, we will discuss acquisition and tracking individually, and see how these could dramatically affect our circuit size. But before we dig into the details of synchronization, let's look at the receiver chain first.



**Figure 4-3 Coherent Detector**

#### 4.4.1 Receiving Chain:

Shown in Figure 4-3, is a conceptual flow of a coherent detector. Received signal  $r_x(t)$  is the signal captured by antenna. A UWB pulse,  $p(t)$ , is generated at the transmitter and sent through the wireless channel, whose impulse response is  $h(t)$ . Thus,  $r_x(t)$  is equal to the convolution of  $p(t)$  and  $h(t)$ , assuming the channel is a linear system. In a multipath environment, the pulse may be distorted, no longer maintaining the original shape. That is the reason we need to put extra efforts to do the pulse matched filtering, or adopt Rake receiving. This will be addressed later in pulse matched filter section.

Because the pulse goes through the wireless channel,  $r_x(t)$  has an unknown time delay and code phase. Time delay,  $T_d$ , is due to the physical propagation latency through the wireless channel, and unknown code phase,  $\phi$ , is due the asynchronous property of any communication system. If the spreading code has a length of 1024, it will then have 1024 possible code phases. Therefore, as the length of spreading code gets longer, the more time it will be required to get synchronized. After two de-spreading mixers, the signal integrates over a period of time,  $T_i$ , also named as dwell time or correlation time. The integrated value will be compared with a threshold.

The first task of the receiver is trying to determine the existence of an incoming signal. If there is a signal, the correct time offset and code phase needs to be determined. As shown

in Figure 4-3, the first de-spreading mixer is actually a pulse match filter, while the second one is a PN correlator. Therefore,  $T_d$  will be resolved via PMF, and PN correlators will help to determine code phase,  $\phi$ .

#### 4.4.2 Acquisition Mode:

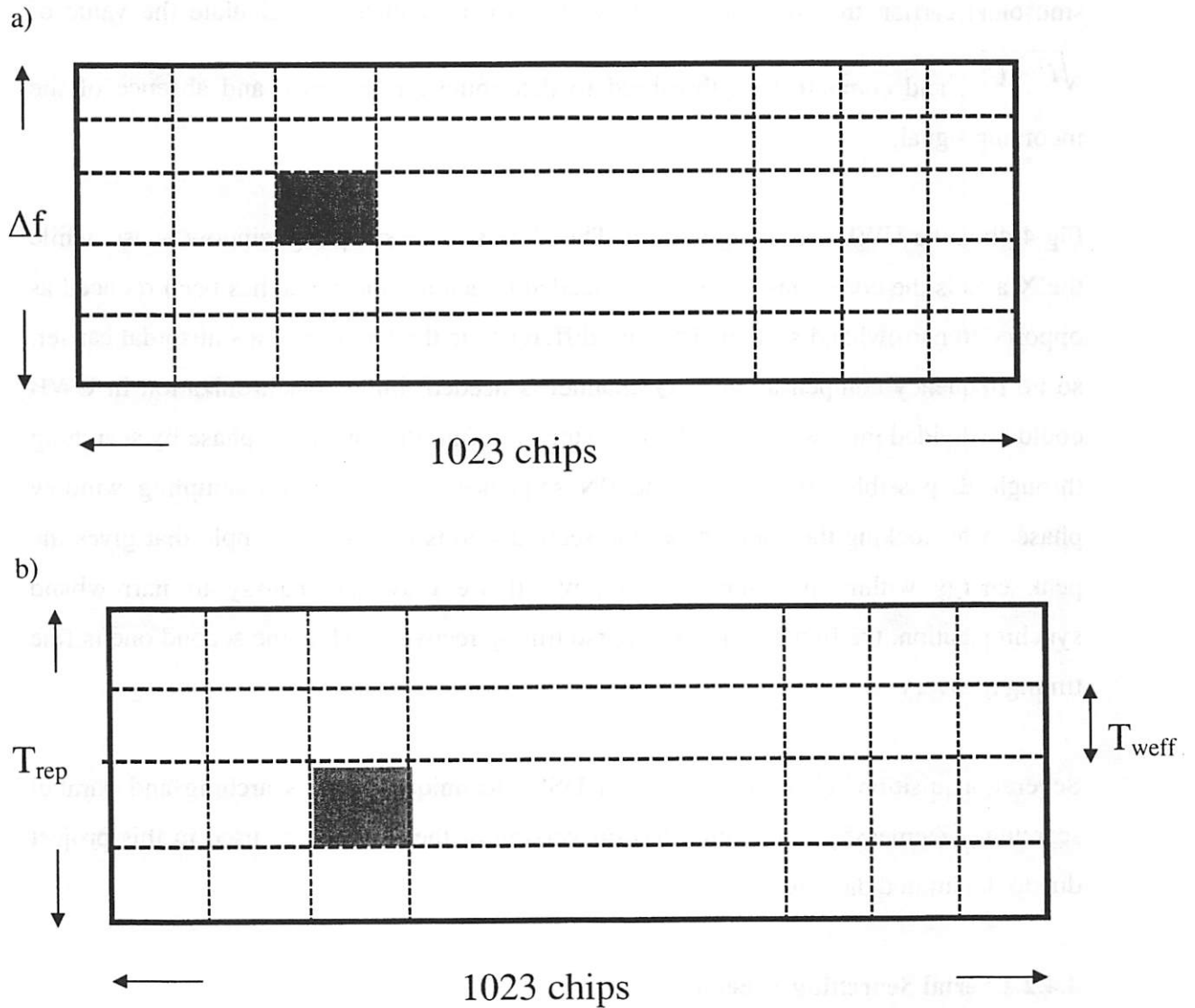


Figure 4-4 a) Gps system search pattern. b) UWB search pattern.

Acquisition is essentially a searching process. Shown in Figure 4-4a, is an example of the GPS 2-D searching pattern. The Y-axis is the carrier frequency, while the X-axis is the code phase. A GPS receiver has to search over all possible carrier frequencies along with



a certain code phase. Let's say the shaded cell is the actual receiving frequency and phase. The searching process begins with a certain cell, and then goes through all possible cells in some way, finally locking with the correct one via the detection threshold. Once the correlation value is above the threshold, then the receiver is assumed to be synchronized. In narrowband system, there are I and Q channels due to the use of sinusoidal carrier, therefore, an envelope detector is required to calculate the value of  $\sqrt{I^2 + Q^2}$ , and compare to a threshold to determine the presence and absence of the incoming signal.

Fig 4-4b is the UWB searching pattern. The Y-axis is the sampling window phase, while the X-axis is the code phase. The effort needed to acquire the signal has been reduced as opposed to narrowband system. The key difference is the absence of a sinusoidal carrier, so no frequency compensation or Q channel is needed. Initial synchronization in UWB could be divided into two steps. The first step is to find the right code phase by searching through all possible orderings of the PN sequence given a certain sampling window phase. After locking the code phase, the second step is to find the sample that gives the peak energy within the sampling window. If we draw an analogy to narrowband synchronization, the first step is like coarse timing recovery, while the second one is fine timing recovery.

Several acquisition schemes are used in DSSS technique: serial searching and parallel searching [Ziemer85]. And then a hybrid version of the two will be used in this project due to the limited die size.

#### **4.4.2.1 Serial Searching Scheme**

The most commonly used scheme is a serial search. Basically, the receiver serially evaluates each cell, shown in Figure 4-4, until the correct one is found. The different code phase is due to the different time that receiver begins to listen. Therefore, the probability of each code phase is assumed to be equal, a uniform distribution. Uniform distribution implies that no matter which phase you begin with, or which searching pattern will be

used, the acquisition time should be the same on average. That is the reason, in this project, the search pattern is simply one chip delay offset between any two subsequent cells. The receiver goes through every cell in the row, and repeats the same sequence if all the cells are dismissed, i.e. no signal is detected.

Besides the sequential searching pattern, other important issues are how to determine UWB signal is present or absent within a cell, and when to dismiss a cell and move into the next one. Three different schemes are usually used in serial search technique.

#### 1) Fixed correlation time detector

This method fixed the correlation time or dwell time,  $T_i$  shown in Figure 4-3. The integrated signal is compared to a fixed threshold  $V_i$ . If the value is above  $V_i$ , the signal is regarded present, and absent otherwise.

#### 2) Multiple-dwell detector

As opposed to fixed correlation time, one could do something smarter than using only  $T_i$  and  $V_i$  as the system parameters, which affect detection probability and average acquisition time, etc. The synchronization process is essentially trying to locate the signal-presented cell and reject all other noise-only cells. Therefore, we don't need to put equal amount of efforts into every cell. Instead, one could reject most of the cells rapidly using shorter correlation time,  $T_1$  shown in Figure 4-5, and then turn into the second round of re-evaluation to reduce the false-alarm probability with a longer correlation time,  $T_2$ , and so on so forth until the final Pfa is satisfactory. It can be shown that the average acquisition time goes down by choosing several different length of integration time (PN sequence). After the system gets locked, it enters tracking mode. The data detection is based on correlation ( $T_3$ ,  $V_{t3}$ ). To avoid miss detection, it adds another longer correlation ( $T_4$ ,  $V_{t4}$ ), because the penalty of miss detection is serious (system has to go through all searching procedures). The detection flow is shown in Figure 4-5. Some modifications could be made to detection flow, like miss detection of correlation ( $T_2$ ,  $V_{t2}$ ) can be connected back to correlation ( $T_1$ ,  $V_{t1}$ ) instead of assigning new code phase.

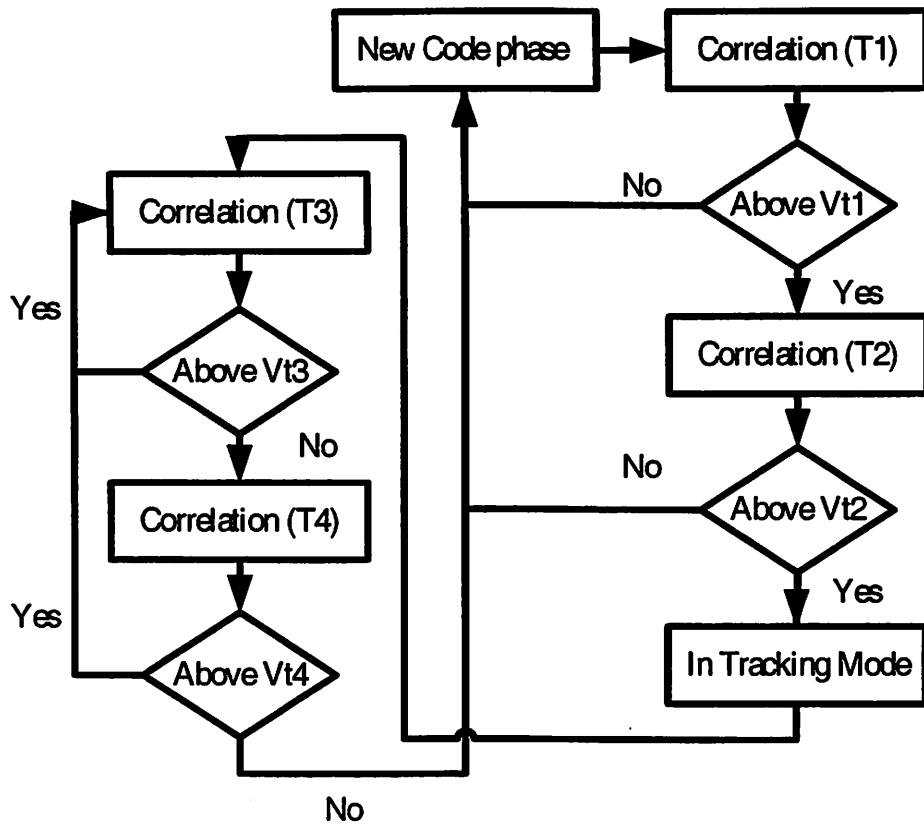


Figure 4-5 multiple-dwell detection flow

### 3) Sequential detector

The idea of doing sequential detection is to calculate the logarithmic likelihood ratio of every de-spreading sample instead of integrating over a certain period. After de-spreading mixers, the samples are  $s_1, s_2, \dots, s_n$ , and the likelihood ratio of detection is as follows:

$$\Lambda(s_1, s_2, \dots, s_n) = \prod_{i=1}^n \frac{P_s(s_i)}{P_n(s_i)}$$

$P_s$  is the probability of presence of signal plus noise, while  $P_n$  is the probability of the noise-only case. If  $\Lambda$  is greater than one, one will have more confidence to claim that signal is present than absent. In a sequential detector, there are two thresholds,  $H_I$  and  $L_O$ . If  $\Lambda$  is above  $H_I$ , the signal is assumed to be acquired, similarly, when  $\Lambda$  is lower than  $L_O$ , the system will dismiss the current cell and move into the next one. However,

this technique requires more complex computation, log function, which has to be taken into account.

#### 4.4.2.2 Fully Parallel Searching Scheme

##### 1) Correlator based

A fully parallel searching scheme is similar to the serial searching scheme, except the system searches all of the cells at the same time. By using the same receiver structure, one simply duplicates the correlators in order to search all possible code phases at the same time. Not hard to imagine, this trades-off a lot more circuit size with the fastest possible acquisition time.

##### 2) Matched-filter based

The output of the correlator,  $y(t)$  could be written as a linear function of input signal,  $x(t)$  and code phase  $c(t)$ ,

$$y(t) = \int_0^{T_i} x(t) \bullet c(t) dt$$

The equation is essentially the same with a linear filter response, except the time-reversed filter response,

$$y(t) = \int_0^t x(\tau) \bullet c(t - \tau) d\tau$$

The similarity gives us a hint that the implementation of a correlator could be replaced with a filter whose response is the time reversed code sequence. The biggest advantage is that the MF continuously correlates the input sequence with an expected spreading code sequence. It outputs a new correlation value whenever a new sample comes in. It will generate a maximum value whenever the incoming signal coherently matches with the code phase. Over the period  $T_i$ , the MF equivalently searches over all the possible code

phases, which is fully parallel searching scheme. Therefore, as long as the speed of data stream is not prohibitively too fast, MF is a very promising method of doing acquisition.

#### **4.4.2.3 Hybrid Searching Scheme**

This design decision comes along with the trade-offs between the chip size and acquisition time. One could achieve the fastest acquisition time using a fully parallel design. In a correlator-based, fully parallel design one could either duplicate correlators for each code phase or adopt time sharing technique to reduce the chip size. But in terms of keeping the flexibility of pulse rate, we didn't use any time-sharing technique so that we could push to the limit of the communication speed. On the other hand, if we use a MF acquisition scheme, the filter has to run at the sample rate, i.e. 2GHz. Of course, it is still impractical in the current CMOS technology and also has too much power consumption. As a result, we adopted a correlator-based design in UWB baseband.

For detection, a fixed correlation time detector is used for the ease of testing and design. Given this scheme, the length of the PN sequence and the threshold setting have been determined in the previous chapter. They are all related to the system specification, such as BER, false alarm probability, etc.

A hybrid searching scheme, i.e. duplicate correlators for several cells (code phases) searching simultaneously, can be adopted in the acquisition as opposed to serial or fully parallel searching strategy. Thus, we could get a compromise between the acquisition time and chip size. The tradeoff compromise is made using the first order acquisition time estimation and chip area estimated by Module Compiler.

- **Acquisition Time v.s. Hybrid Strategy**

The mean acquisition time is a function of the probability of detection,  $P_d$ , probability of false alarm,  $P_{fa}$ , number of parallel searching cells, and length of PN sequence. It is a bit complex since many parameters are involved. Let's focus on the worst-case acquisition

time given the Pd and Pfa are zero in order to get a quick idea of how fast the system could get locked. The parameters are (please refer to section 4.3 for more details):

N: length of PN code;

M: number of code phases being searched simultaneously;

$T_{rep}$ : pulse repetition rate;

$T_{weff}$ : effective sampling window period;

$T_s$ : sampling period;

$$Tacq\_worst = N \cdot \left[ \frac{T_{rep}}{T_{weff}} \right] \cdot \frac{N}{M} \cdot T_{rep}$$

$$Tacq\_ave = \frac{Tacq\_worst}{2}$$

The average acquisition time is half of the worst-case time given the assumption that the probability distribution of each searching cell is uniform. If the distribution is not uniform, one should start from the most likely cell and then spread out from the center of the mean value so that the average acquisition time is reduced.

Certainly, it's possible to derive a more accurate equation considering more parameters to a first-order analysis.

$P_{false}$ : false alarm probability;

$T_{false}$ : extra time required to reject the cell when a false alarm happens;

$P_{miss}$ : miss detection probability;

$C_{total}$ : total cells on 2D searching matrix ( $= T_{rep}/T_{weff} \cdot N$ );

$T_i$ : correlation time ( $= N \cdot T_{rep}$ );

$$T_{acq\_ave} = \frac{C_{total} \cdot T_i}{2M} + \sum_j C_{total} \cdot T_i \cdot j \cdot P_{miss}^j (1 - P_{miss}) / M + T_{false} \cdot \sum_k \binom{C_{total}}{k} \cdot P_{false}^k (1 - P_{false})^{C_{total}-k}$$

A more accurate acquisition time analysis can also be done; however, it's already beyond the scope of this thesis. Some analysis has been done using Markov model, [Holmes77] [Simon94]. The first order estimation is good enough for making design decisions, and more exploration with different algorithms could be done in the future.

- **Size v.s. Hybrid Strategy**

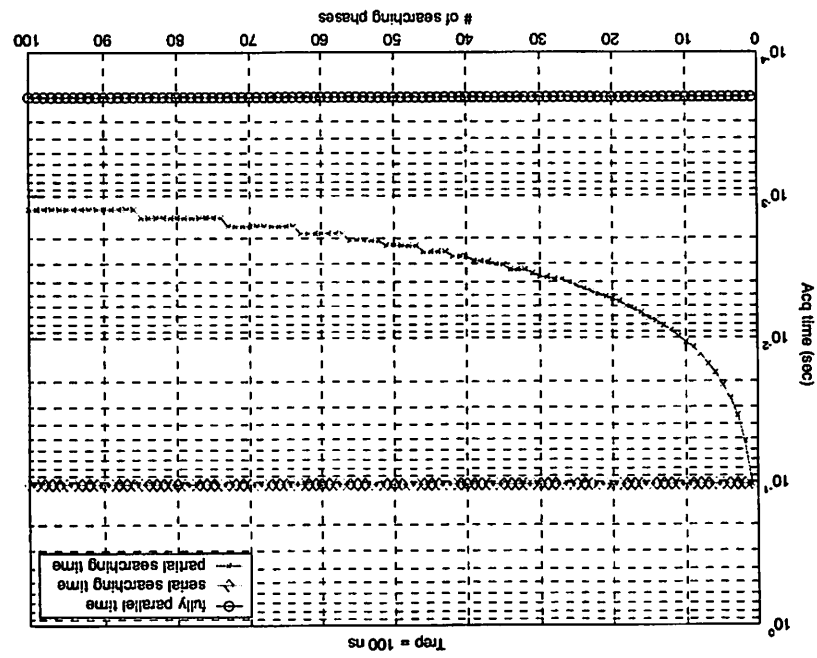
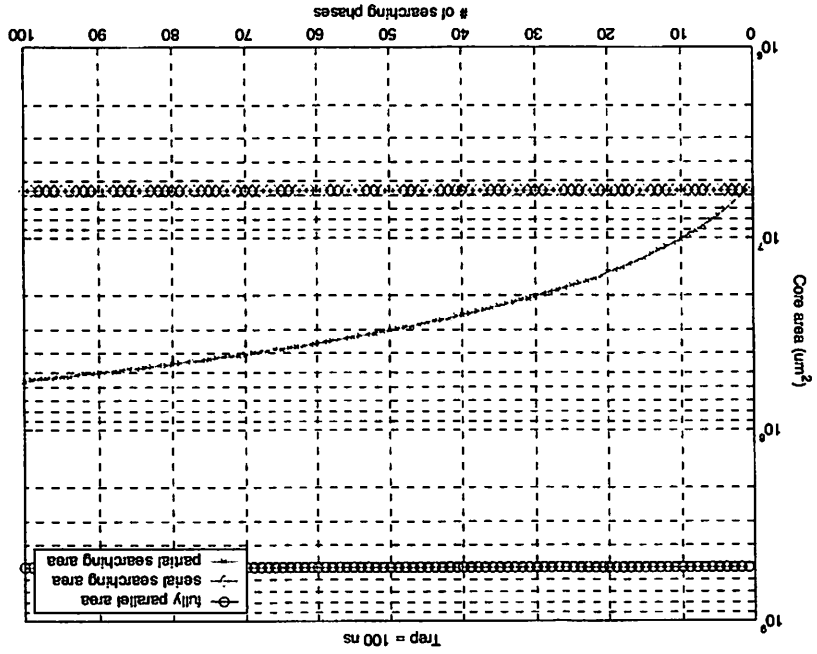
The amount of circuit required is estimated using Module Compiler. In chapter 5, we will dig into more details. In order to investigate how hybrid strategy could affect the chip size, the area report for each fundamental slice was done in MC. For example, we get the area for one correlator, and then assume the area scales up proportionally with the number of correlators we put on the chip. Thus, the total transistors on the chip could be parameterized by number of simultaneous searching cells.

- **Simulation Results**

The design decision has been made using the first-order average acquisition time analysis and size reports from MC. The question is simply how many correlators we could fit on chip, given the PMF is taking 256 samples, which corresponds to a sampling window of 128 ns.

Figure 4-6a shows a fully parallel searching scheme will only require 0.1 ms, given the pulse rate ( $T_{rep}$ ) of 10 MHz. A serial searching scheme gives a 0.1 second acquisition time. The effect of the acquisition time affects the QoS of the communication network, the target here is trying to achieve as fast as possible acquisition time given a reasonable area cost.

Figure 4-6 a) Acquisition time v.s. searching phase b) Size v.s. searching phase

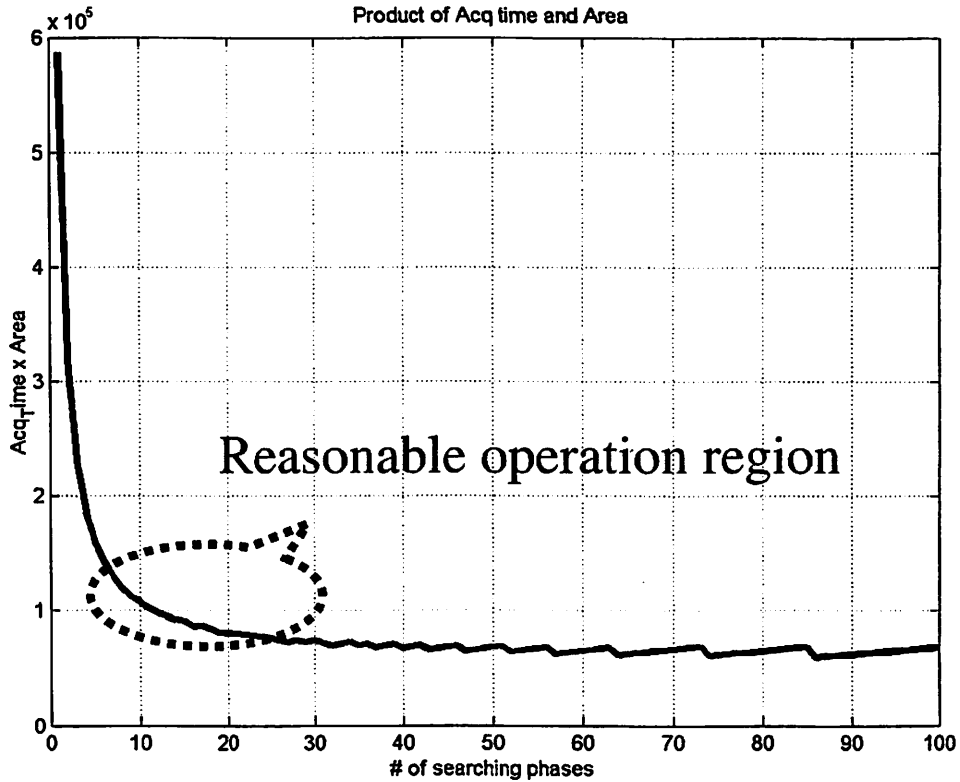


b)

a)



Figure 4-6b shows 500 mm<sup>2</sup> chip size is required for fully parallel correlator based acquisition scheme, while serial searching takes only 5.8 mm<sup>2</sup>, roughly a hundred times smaller.



**Figure 4-7 Product of acquisition time and area**

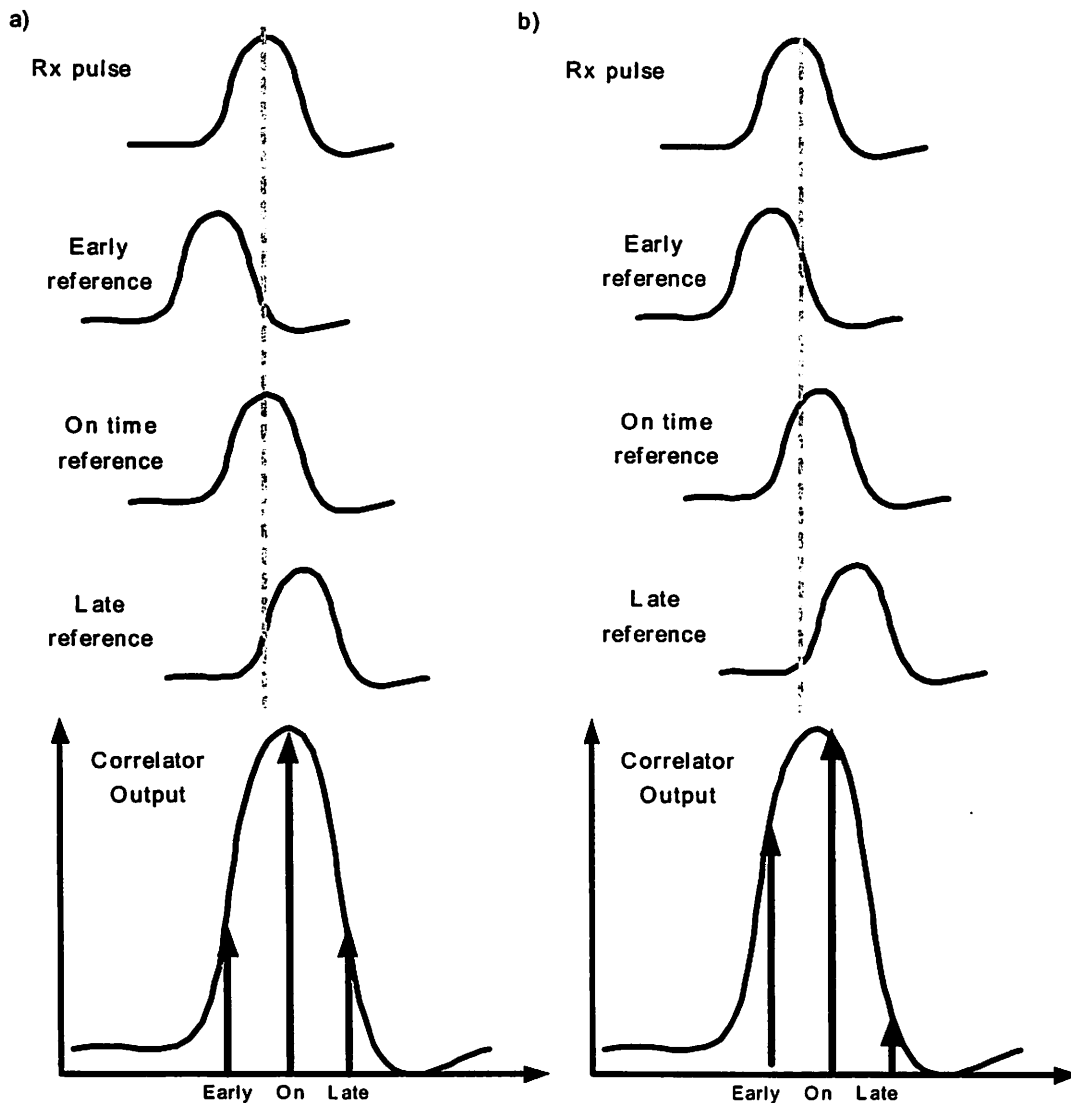
If we consider only the tradeoffs between area and acquisition time, Figure 4-7 shows that as searching phase goes beyond ten, the acquisition time saturates. Therefore, in our design, we chose eleven searching phases, which corresponds to eleven correlation blocks. Under this design decision, the correlators and detection blocks together are nearly of equal size as the PMF block.

#### 4.4.3 Code Tracking

Once the system passes the acquisition mode, it enters into the code tracking mode. During acquisition, the receiver has an estimated signal arrival time and code phase.

However, due to the discrepancy of the transmitter and receiver clock frequencies, and movement of the user or environment, the signal arrival may drift over time. The previously estimated arrival time may be no long valid. Therefore, we need a mechanism to track the drifting of the signal, so called code tracking. The tracking strategy in this project is essentially early-late tracking, which will keep the reference signal within a fractional of chip period.

In order to track the direction of a drifting signal, let's first look at the autocorrelation function of the incoming pulse. In Figure 4-8, the received pulse is correlated against a reference signal with different time offset: on-time, early and late. After the correlator, the value of on-time correlation has the maximum value, while early and late correlation has smaller value (usually around half). If the received pulse matches the on time reference, shown in Figure 4-8a, the early and late correlation values are equal. As the received pulse drifts towards the early reference signal, shown in Figure 4-8b, the early correlation value starts to be larger than the late one. And the reference signal should be shifted back.



**Figure 4-8 Early-late tracking profile a) Rx pulse matches on time phase b) Rx pulse comes earlier**

The performance of the tracking loop could be described by the dynamic locking range and tracking jitter. Given the early and late correlation, somehow, we need to tell the direction of the drifting signal. In the code tracking loop of a DSSS system, this is done by a code loop discriminator, loop filter, VCO, and PN generator, shown in Figure 4-9a. It is essentially a DLL. Tracking range and tracking jitter are used to describe the loop performance. Tracking range is the maximum allowable input delay error that system could still track back. And the tracking jitter is the variation of delay error due to the input noise.

Usually, the relationship between the delay error and discriminator output should be found first. Based on this relationship, one could correct the input signal by making use of discriminator output together with appropriate loop gain and bandwidth. The delay error should be subtracted from the original signal. This is the way a system could keep tracking the incoming signal with the reference within a close enough time value. Not surprisingly, different discriminator and loop filters will lead the system to different tracking performance. For more details, readers are referred to [Kaplan96].

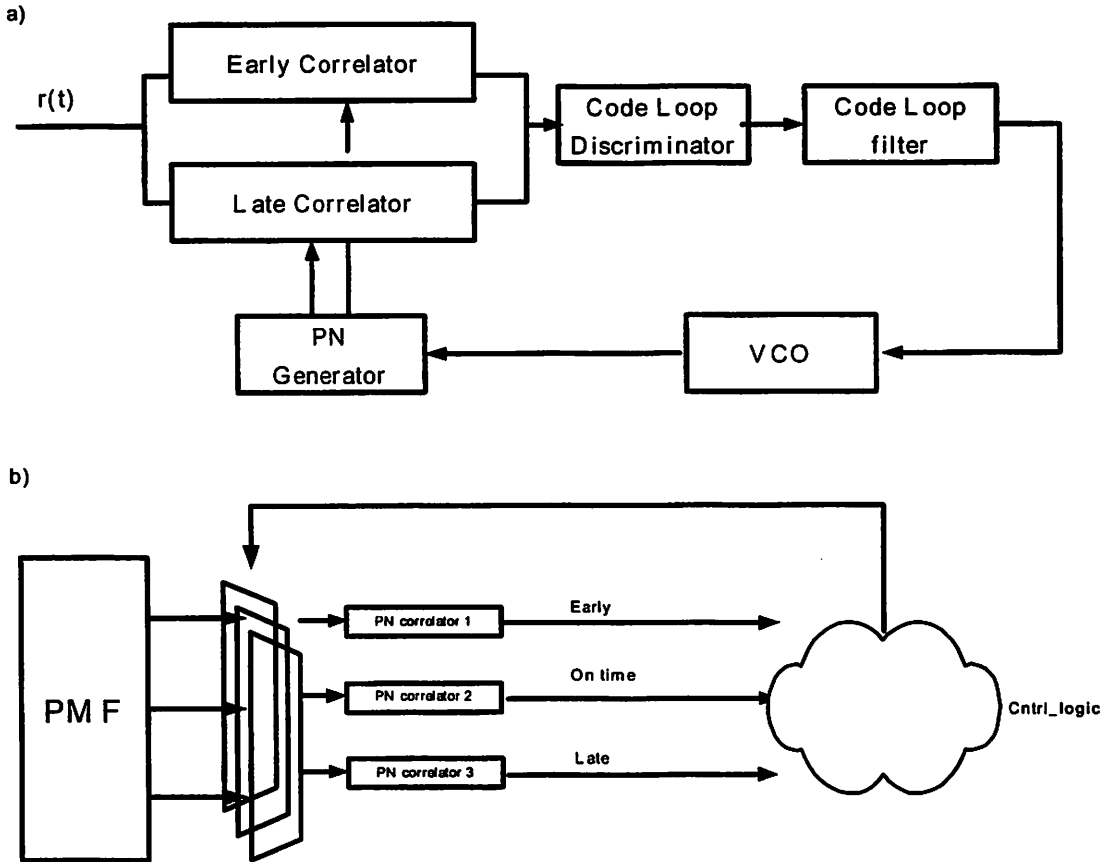


Figure 4-9 a) DLL tracking loop b) UWB tracking loop

In UWB baseband, we don't drive the PN generator with a clock, since the PN sequence is stored in registers, not a shift register array. Instead of shifting the phase of PN

sequence by the VCO, we shift the reference signal by selecting a different set of outputs from PMF, because each output of PMF corresponds a specific timing phase. If the received signal comes before the on-time reference signal, the early correlation will be larger than late correlation. Note that the time duration of the UWB pulse autocorrelation function is at the nano-second range, which is quite small compared to a narrowband system. Given the same moving environment and oscillator instability, UWB system is more vulnerable to drifting. Since the time spacing between early and late correlators is much closer. As a consequence, if the control logic shifts the phase of an incoming signal whenever the early (/late) correlation goes beyond (/below) the on time correlation value, one should expect to see the selector goes back and forth frequently. Sometimes this may be caused by noise, inducing a false drifting alarm. A remedy to this problem is that the selector shift the phase of incoming signal when the early (/late) goes beyond the on-time value for N consecutive times. This way, the system moves the tracking phase only when it gains enough confidence. This method could reduce the tracking jitter; however, the tracking range will be decreased, since the tracking loop responds more slowly to the drifting, i.e. the loop bandwidth is narrower. Alternatively, we could do something like a discriminator. If we know the relationship between delay error and discriminator output, like for a narrowband system, we can use this information to determine how much the phase needs be shifted given a certain discriminator output. Since the tracking performance heavily depends on pulse shape and the channel response, we need to keep the tracking strategy flexible enough that we can play with it in the future testing. The method finally used in this project is to trigger the selector when early/late correlation goes beyond the on-time correlation for a consecutive N times, where N is a flexible parameter. Of course, N could be one, which gives the maximum loop bandwidth and worst jitter performance.

How do we choose the early and late samples? In narrowband system, the early/late points are usually chosen to be half of the chip time apart from the on-time point, so that the correlation value of early/late should be half of the on-time correlation value as in the perfect synchronized case. Indeed, it depends on the link budget of the system. The more SNR degradation is allowed for sampling, the farther it could be between the early and

late samples. In UWB, the spacing between early and late point depends on the pulse autocorrelation function. Since the pulse shape is still unknown, the spacing between the early / late will be another flexible system parameter that can be set externally.

Theoretically, in acquisition mode, the receiver could simply take one sample within a correlation period since the samples are all correlated within that lobe. However, the UWB pulse duration is so short that the autocorrelation profile is also very short and steep. If the receiver only looks at one or even two samples within that correlation lobe, they can be sitting at the bottom of profile. Certainly, we don't want to risk losing such a significant SNR. Therefore, we still look at all correlation samples at the Nyquist rate.

#### 4.5 UWB Baseband Components

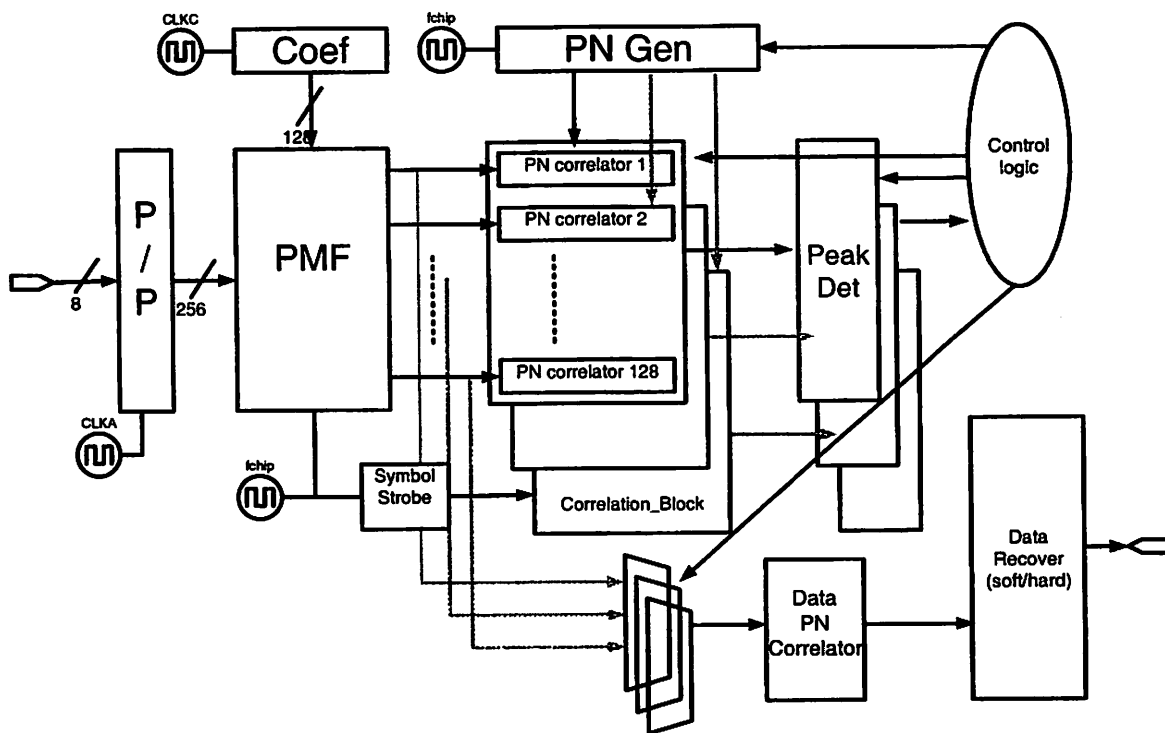


Figure 4-10 Baseband overview

Figure 4-10 is an overview of the whole system, which includes the pulse matched filter, correlation block, peak detector, control logic and data recovery block. In this section, we

will go through the design of each block. The parallel inputs from ADC are the samples within a sampling window of 256 samples (128 ns) after buffering in the analog front end. The baseband will then try to demodulate the sending data out of these inputs.

#### 4.5.1 Matched Filtering

If the transmitted signal,  $s(t)$ , is corrupted by additive white Gaussian noise,  $n(t)$ , the optimal detection will be passing the received signal through a matched filter, whose filter response is  $h(t)=s(T-t)$  [Proakis00]. Thus, the output SNR obtained from matched filter will be maximized. If the input noise has a power spectral density of  $N_0/2$  and transmitted signal has the energy of  $E_s$ , output SNR of matched filter is  $E_s^2/N_0$ . Alternatively, from a signal space perspective, matched filtering could also be viewed as a projection onto the signal dimension, which will then transform the waveform detection into a 1-D detection problem [Tse01]. The signal dimension is where all signal information is contained. Therefore, by projection onto this direction, the receiver already rejects most of the noise, which is orthogonal to the transmitted signal. In other words, the SNR is maximized.

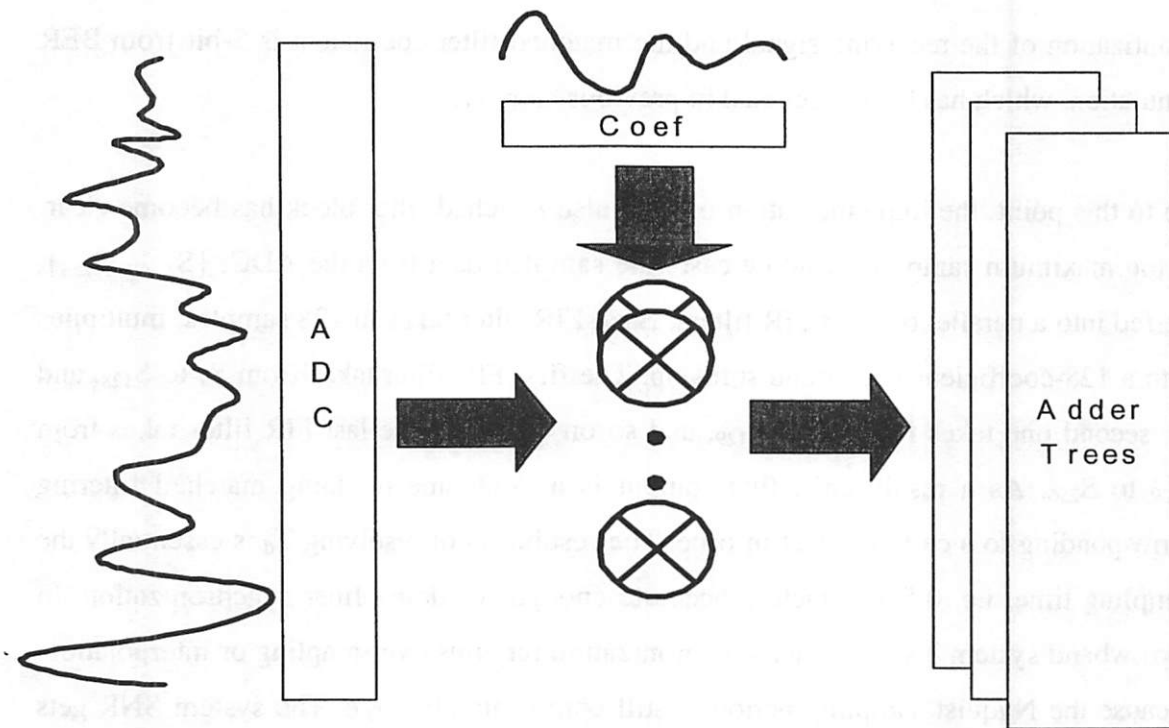
In UWB, the noise could be colored owing to narrowband interferers. The optimal receiving strategy is to whiten the noise first, and then apply the matched filtering. However, the whitening filter requires the prior knowledge of channel response, which complicates the current design. Therefore, we treat the noise to be white for now, and leave the whitening story in the future.

In this project, we did the matched filtering in the digital domain, therefore, the signal has to be sampled at least at the Nyquist rate, which preserves all the sufficient statistics for detection. If quantization could be infinite precision, i.e. no quantization noise, then the sampled version of matched filter should be exactly the same as analog matched filtering. Given the “undetectable UWB” scenario, the specification of the system is to do 1-bit

quantization of the receiving signal and the matched filter coefficient is 5-bit from BER simulation, which has been discussed in previous chapter.

Up to this point, the implementation of the pulse matched filter block has become clear. In the maximum sampling window case, the sampled data from the ADC,  $\{S_1 \dots S_{256}\}$ , are fed into a parallel bank of FIR filters. Each FIR filter takes in 128 samples, multiplies with a 128-coefficient vector and sums up. The first FIR filter takes from  $S_1$  to  $S_{128}$ , and the second one takes from  $S_2$  to  $S_{129}$ , and so on so forth. The last FIR filter takes from  $S_{129}$  to  $S_{256}$ . As a result, each filter output is an outcome of doing matched filtering corresponding to a certain offset in time. The resolution of resolving  $T_d$  is essentially the sampling time, i.e. 0.5ns, which is accurate enough for doing finer synchronization. In narrowband system, usually finer synchronization requires oversampling or interpolation, because the Nyquist sampling period is still comparatively large. The system SNR gets improved by doing so. However, in UWB, the Nyquist samples are already fast enough, there won't be too much improvement in an even finer resolution, like fractional of nano-second. Therefore, in PMF, there are a hundred and twenty-eight 128-tap FIR filters with a sampling time offset between any two consecutive ones. The basic idea of PMF is shown in Figure 4-11. From the derivation of matched filtering, the more deviation between matched filter response and received signal, the more SNR loss it would cause to the system. In a time-invariant channel, once we get an accurate enough signal shape for the filter coefficients, the system essentially becomes a Rake receiver, which collects all multipath energy within a 64 ns range. It is certainly possible to modify the current system with some adaptive algorithms. Some initial thought about making an adaptive receiver is done in [ChenShi01]. There, they use the LMS algorithm and a training sequence to implement Rake receiving. MMSE detection is also compared in that report. More detailed studies of adaptive reception will require more knowledge about UWB channel modeling, which could be one of the interesting topics for future research.





**Figure 4-11 Pulsed Matched Filter**

#### 4.5.2 Correlation Block

The PN correlators are used to offer enough processing gain in order to keep low BER. There are 128 outputs from the PMF, thus we need 128 correlators for one correlation block. Each correlation block is associated with one PN phase. During the acquisition mode, one correlation block will search one cell in Figure 4-4. In our hybrid acquisition mode, we need eleven correlation blocks so that we could search 11 cells per dwell time.

#### 4.5.3 Peak Detector

The peak detector (energy detector in Figure 4-3) goes after correlation block. The goal is to select the maximum value out of these 128 correlation values, and compare it to a predetermined threshold set externally (threshold calculation is referred to chapter 3). If the signal does exist in any of these eleven cells, one of the peak detectors should output

“high”. This is when the system ends acquisition mode. Besides selecting the maximum value, the peak detector needs to locate the correlator with the maximum value for data recovery.

Since the impulse response of an FIR filter is programmed to be the pulse shape convolved with channel response, all the signal information is contained in the peak value. However, one could also program the impulse response to be the pulse shape, thus the output profile of the correlation block becomes the channel response. The peak value is simply the strongest multipath. The better strategy is to collect all the multipaths, which is so called maximum ratio combining.

#### 4.5.4 Data Recovery

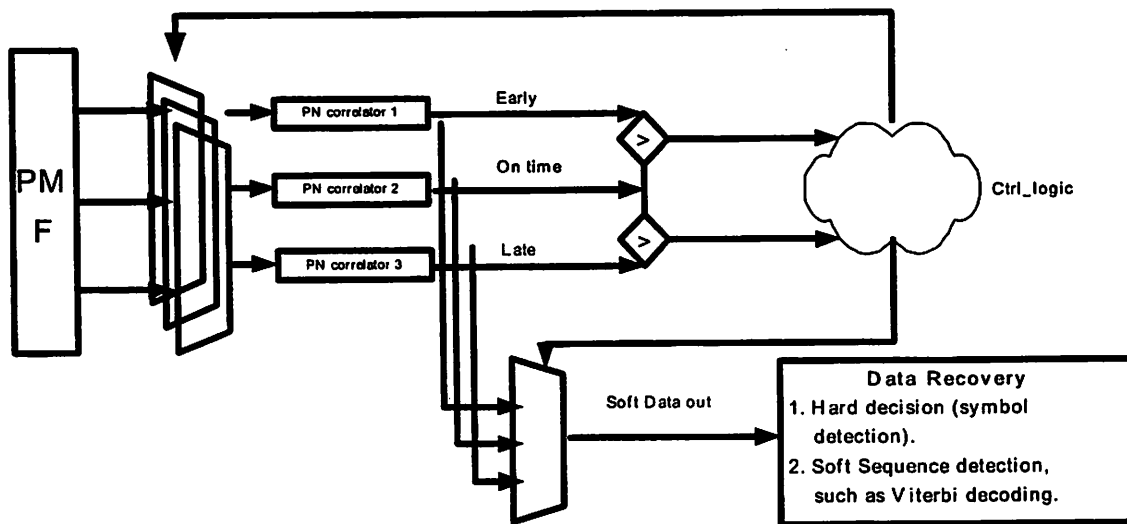


Figure 4-12 Data recovery diagram

After acquisition mode, the receiver needs to start data demodulation. The position of the pulse,  $T_d$ , within the sampling window has been solved by the PMF, and is reported from the peak detector. All the correlation blocks go into sleep mode for power saving during data recovery. Therefore, three muxes are placed to select the maximum value (on time), and two neighboring points (early/late) for code tracking purpose. Data recovery will

again take the maximum of the three values. Normally, the on time correlator should have the maximum value, but as drifting occurs, either early or late correlator will have the bigger value, this is the reason we still select the maximum of the three for the detection. The detection done on chip is simply hard decision, i.e. sign bit detection. For soft detection, we could output the values from the chip and do post processing.

To indicate if the transmitter stops transmitting or somehow the tracking loop loses lock, a tracking signal, named “at\_track”, needs to be generated by data recovery block. It simply compares the absolute value of on time signal with a certain value. If the absolute value goes lower than the threshold, the control logic will force the system to go back to acquisition mode.

#### **4.5.5 Symbol Clock Generation**

During acquisition, the transmitter shouldn't modulate any data on the pulse train. It only transmits the PN sequence repeatedly for a certain amount of time, i.e. a preamble time, Figure 4-13. The preamble packet should be long enough for the receiver to get synchronized with the incoming signal in the worst case scenario. After the preamble, a data packet will be sent.

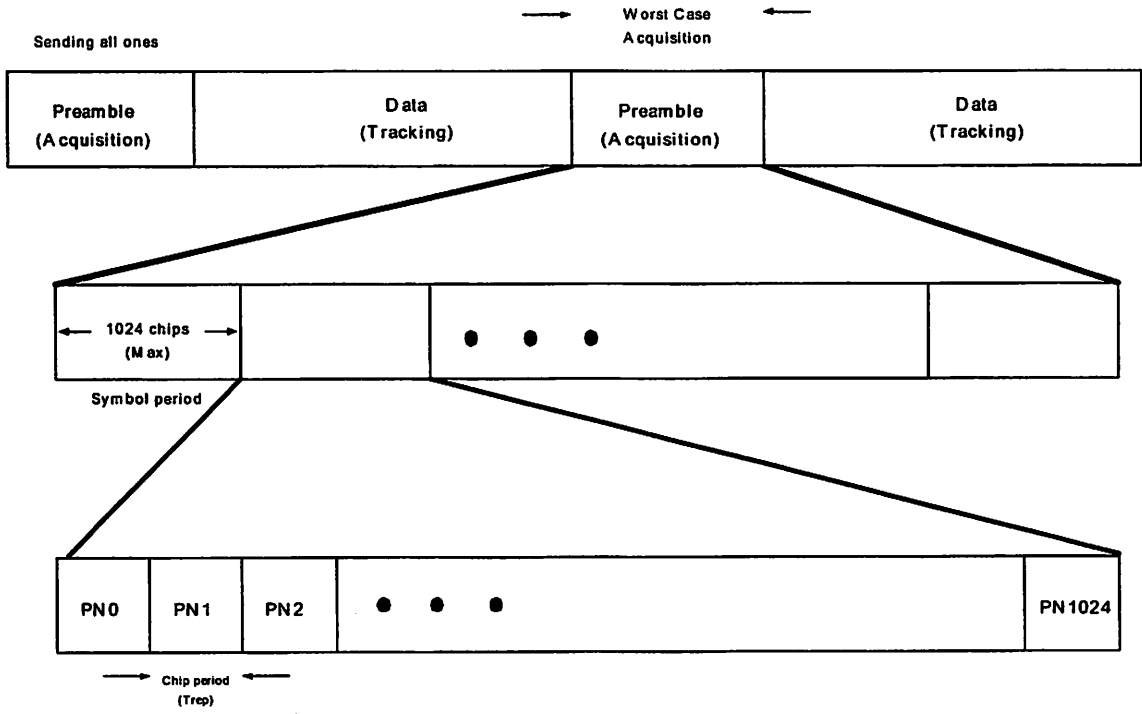


Figure 4-13 Proposed packet format

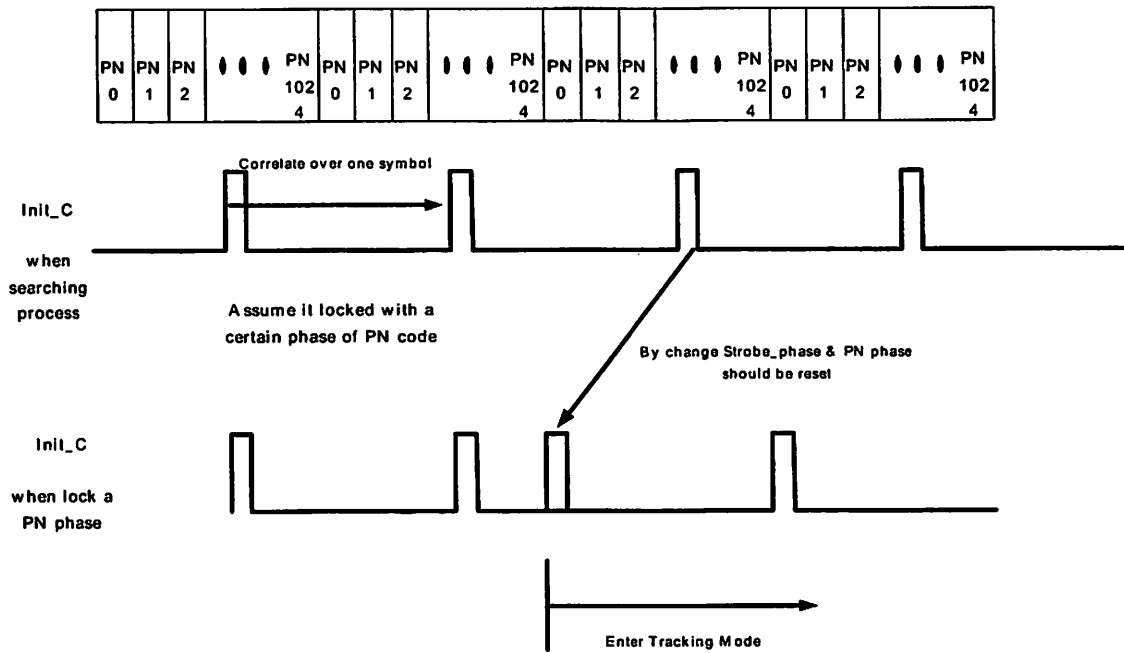
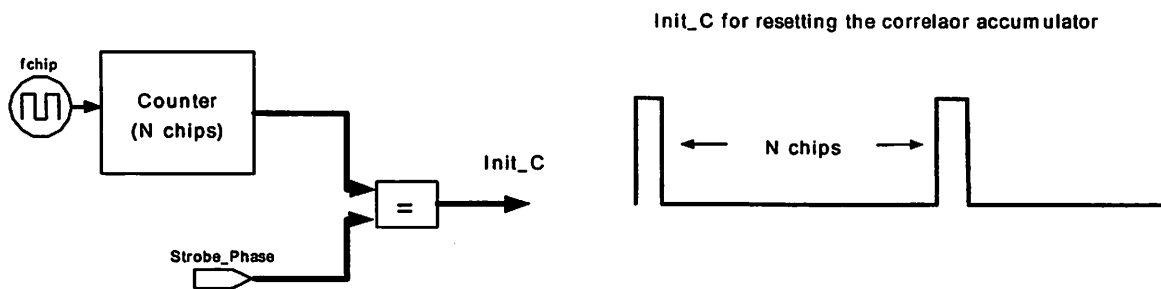


Figure 4-14 Symbol boundary

Shown in Figure 4-14 is the maximum length PN sequence, PN0 to PN1023. The sequence is sent periodically for the preamble period. The synchronization procedure not only detects the presence of the signal, but also finds out the symbol boundary, which helps the data bit demodulation. Assume we have a certain clock running at the symbol rate (number of chips \* chip period), and the data symbol always goes from PN0 to PN1023. Now, the incoming signal correlates with a certain PN phase over a symbol period. At the end of the current symbol period, the system will be able to tell if the signal matches the phase of PN sequence. If the answer is positive, the symbol clock should somehow be adjusted to align with the data symbol boundary (PN0~PN1023). Since the phase of the PN sequence being synchronized is known, the symbol clock should shift (number of chips – phase of PN) chip time. The implementation of the described action is done by relational logic following a free running counter, shown in Figure 4-15. The `strobe_phase` is generated from control logic, which is basically the number of chips needs to shift. By changing `strobe_phase`, the symbol clock changes its phase.



**Figure 4-15 Symbol clock generation**

### 4.5.6 Control Logic

Control logic is used to keep the system working in the correct mode and also for the purpose of power saving. This is done in a state machine (stateflow in Simulink). The main task of the control logic is to coordinate between the two operation modes: acquisition and tracking mode. In order to implement a flexible receiver, we integrate all

the system parameters into the control logic, such as the number of PN chips, correlation blocks, etc.

Given the chosen topology, we need to generate a programmable counter to read out the value out of PN register arrays. Each counter has a variable wrap-around value, initial value and the ability to turn off. Under these design constrains, stateflow is the simplest way to implement the logic function. Therefore, the two main functional blocks are done in stateflow. One is the main control block and the other is the readout block.

In acquisition mode, the main control block takes in the peak detection outcomes compared to a programmable threshold, while it outputs the initial phase to the readout block. If no signal has been detected after one symbol time, the initial phase for all readout blocks has to increase by one, and so on and so forth. If all the PN phases have been searched without the presence of a signal, “sh\_win” will go high for one chip cycle. “sh\_win” is the feedback signal to analog front end. If “sh\_win” is one, sampling window will be delayed by a certain amount of time and vice versa. “sh\_win” should stay zero when no shifting is required. After the signal is detected, the system will enter the transition mode.

During the transition from acquisition to tracking mode, the system has changed the symbol boundary clock. Plus, the transition of the state is always triggered at the rising edge of the symbol boundary clock. Therefore, a wait\_transition state has been created to wait at least one symbol time for the data recovery correlator to get the correct correlation value before the system gets into tracking mode.

In tracking mode, the main control block will do several things. One is to disable all the correlation blocks by connecting all the PN ports of the correlation block to ground. Another is to divert three of the outputs of the PMF to the data recovery block, which require the information of the peak value position from the correlation block, called “maxadr”. Plus, all the readout blocks have to reset to zero except the first one, which

will connect to the data recovery block. Also, the strobe\_phase signal is generated for the symbol boundary clock.

As long as "at\_track" from the data recovery block stays high, the system will remain in the tracking mode. However, the pulse position or offset may drift over time. There are early and late signals also fed from the data recovery block. If the late signal is high, meaning the signal drifts backwards, the control block increases the "maxadr" by one, and vice versa. Another programmable parameter, "Guard", is set up to detect when the signal hits "Guard" samples from the symbol boundary. If the signal hits the line, "sh\_win" will output 1 or -1 depending on which end it touches, and "maxadr" will do the necessary changes depending on the time offset of sampling window shifting.

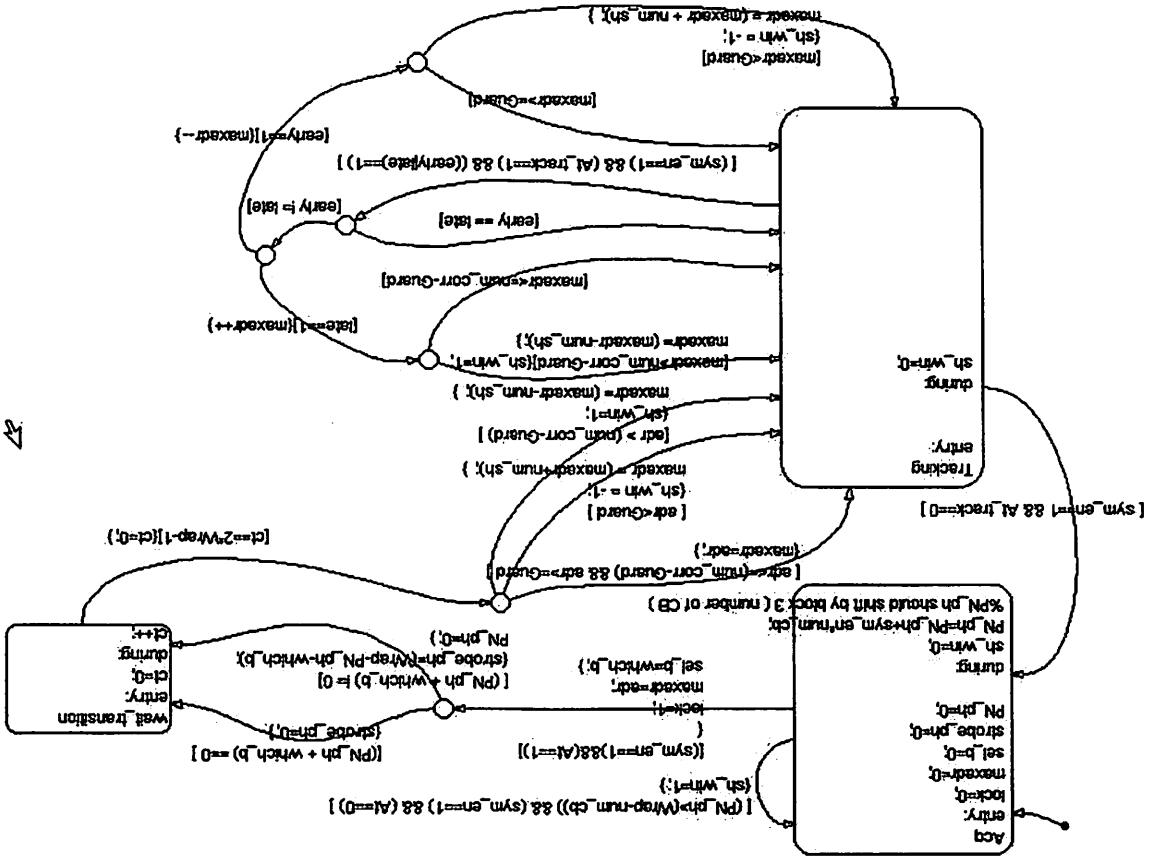


Figure 4-16 Example of main control logic

# Chapter 5 Implementation of ASIC

## Digital Backend

### 5.1 Pulse Matched Filter

Essentially, the operation of the pulse matched filter is to do a lot of parallel multiplies and adds. Given the 1-bit ADC scenario, the multiplier is reduced to an XOR gate, leaving the major components as the multi-operand adders. The functionality of one fundamental slice is described as the following equation:

$$Z_i = \sum_{i=1}^{N_{pulse}} C_i \cdot S_i$$

$N_{pulse}$ : number of filter taps, which is the pulse duration after channel;

$\{C_i\}$ : sequence of fixed coefficients of matched filter;

$\{S_i\}$ : sequence of samples from 1-bit ADC;

- DISTRIBUTED ARITHMETIC

Two different implementations are candidates in the design. One is to use distributed arithmetic (DA) [Lars99], which utilizes memory to augment the computation of serial/parallel multiplies. This technique is very useful for implementing the inner products of a fixed vector with a variable vector, i.e.  $\{C_i\}$  and  $\{S_i\}$ . It saves computation power in the sense of storing the pre-computed results in the memory. However, the memory size required goes up exponentially with the length of the vector, i.e.  $N_{pulse}$ . If we use only one memory, the number of words required is  $2^{N_{pulse}}$ . The capacitance on word lines will increase, which increases the power dissipation and time delay on reading out the stored values. Of course, one could use multiple memories to mitigate the problem. However, the more memories we use, the more adders that have to be used as well. Plus, the  $N_{pulse}$  we are targeting here is 128, which is definitely not a small number. In



summary, by pre-computing of the adds of several fixed-coefficients, one could save power as long as the input width to the look-up-table is in a reasonable range so that capacitance on the bit line is small enough not to cause too much cost by reading out the values. Besides the cost issue, in terms of a configurable baseband where we may need to change the pulse shape from time to time, it is more inconvenient to use DA, because we have to load the values into memory every time we want to change the testing criterion. Therefore, we seek to another implementation, which is adder tree or array.

- MULTI-OPERAND ADDER

There are several structures which could be adopted for adding 128 5-bit signed numbers. They can be sequential in structure, a linear array or a tree structure. These multi-operand adder structures are applied to dot products, convolution, multiplication, etc. In this project, this is used for FIR filtering. The following discussion will be based on summing up N K-bit numbers.

1) Sequential addder

This is the smallest design, with the cost of time latency (N cycles). The adder has to be (K + logN) bit wide, as shown in Figure 5-1. Time delay scales linearly with number of operands.

$$T_{delay} = O(N(\log(N + \log K)))$$

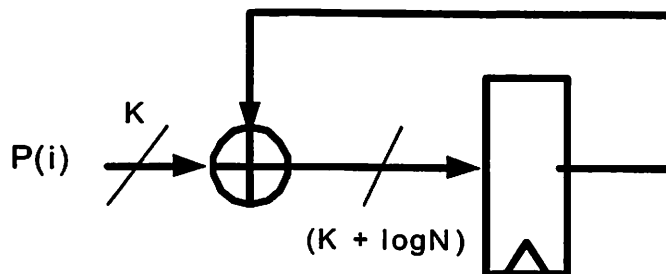
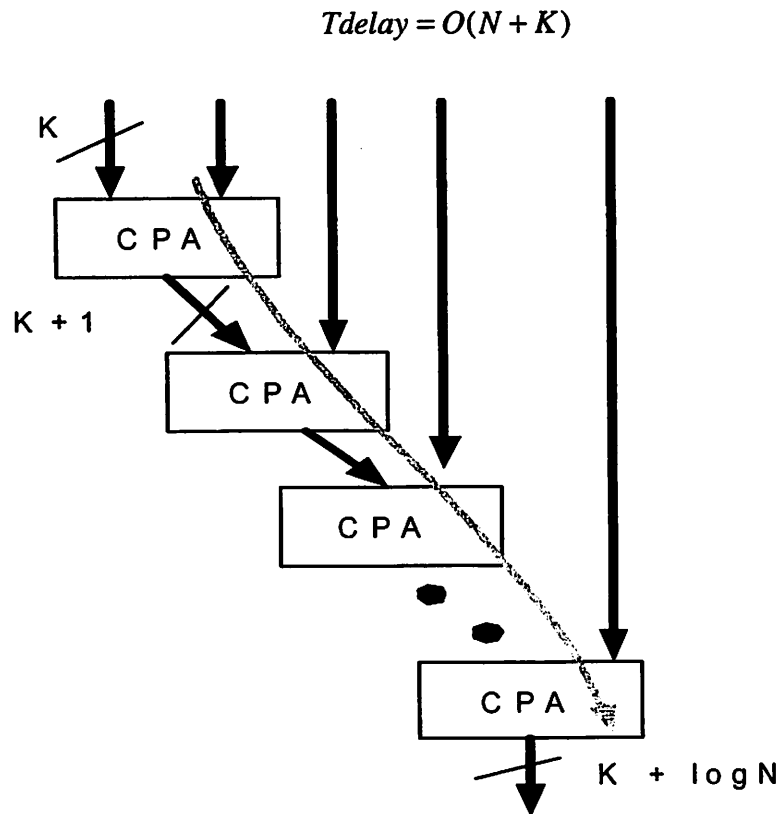


Figure 5-1 Sequential Addder

2) Linear adder array

To eliminate the computation latencies, while trading-off the chip size, one could use adder arrays. The operands are summed sequentially over each level, shown in Figure 5-2. It requires  $N$  propagation adders, each width incremented by 1 bit compared to the previous stage. Therefore, the critical path delay is still proportional to the number of operands and bit width.



**Figure 5-2 Linear adder array**

### 3) Propagation adder tree

Linear dependence on the number of operands will cause too much time delay for our design. Therefore, making a binary tree structure will reduce the number of levels to be logarithmic with  $N$ . The critical path depends linearly on number of tree levels and logarithmically on the number of bits for a fixed  $N$ , if each propagation adder module has logarithmic delay, such as carry-lookahead adder.

$$T_{delay} = O(\log K + \log(K+1) + \dots + \log(K + \lceil \log N \rceil - 1))$$

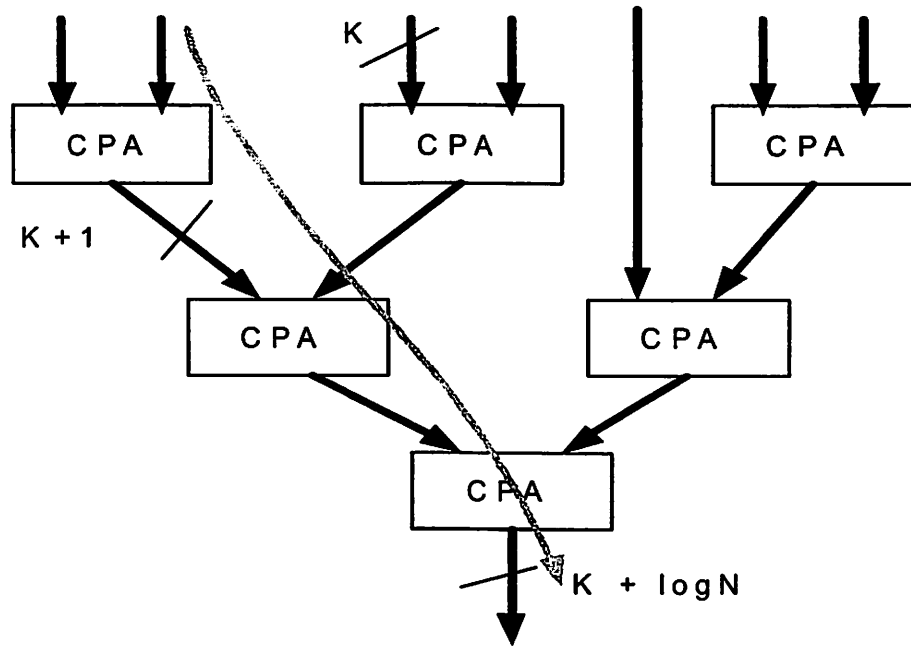
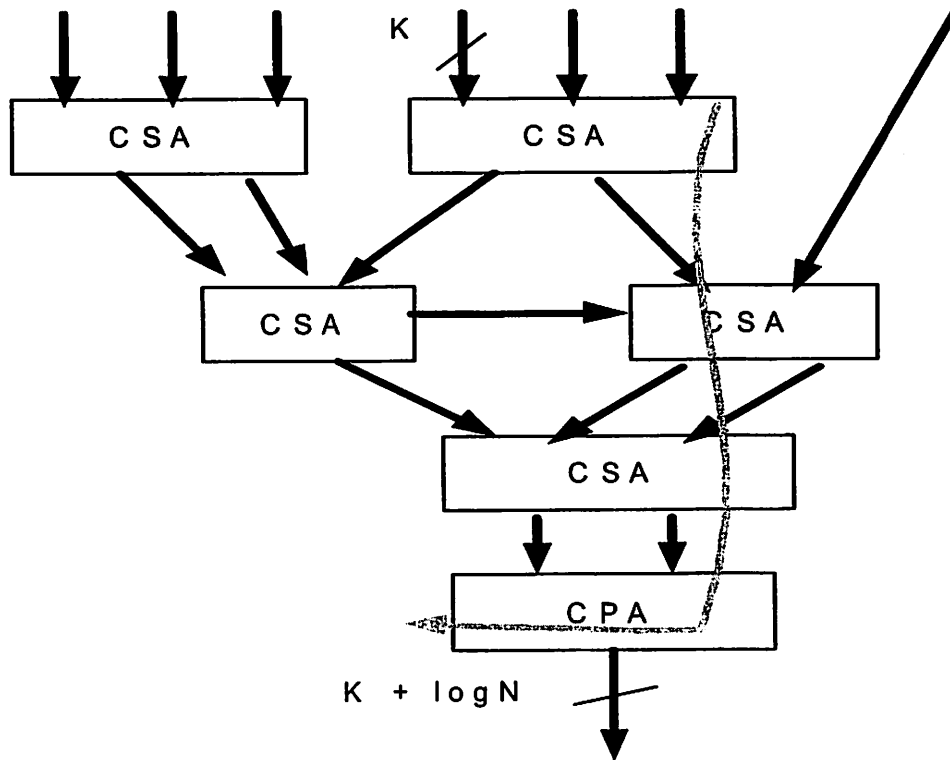


Figure 5-3 Binary adder tree

#### 4) Carrysave adder tree

The fast implementation of multi-operand adder is built with the carrysave adder tree. Shown in Figure 5-4, each carrysave adder cell takes in three operands and outputs carry and sum vector, without merging them. CSA cell is also known as (3, 2) compressor. Carries and sums propagate through the whole tree, and are merged only at the last stage. Usually, at the final stage, people use a fast propagation adder, where one optimizes for speed. Compared to the propagation adder tree, where carry propagates at every level, the time delay depends entirely on the tree height and time delay of last stage merge adder. Thus it is faster.

$$T_{delay} = O(\log(K + \lceil \log N \rceil - 1) + \log N)$$



**Figure 5-4 Carrysave adder tree**

Well-known structures, i.e. the Wallace tree and Dadda tree, are based on this carrysave idea. The difference is that the Wallace tree [Wallace64] reduces the number of operands at the earliest possible opportunity, and minimize the overall delay by making final merge adder as short as possible. While Dadda tree [Dadda66] tries to use the fewest number of FA's and HA's without increasing the tree height.

The Wallace tree takes in  $N$   $K$ -bit operands and outputs two  $(K + \log_2 N - 1)$  bit vectors. At the internal stages of the tree, it is essentially a partial products pattern reshaping. A (3,2) counter suppresses three operands into two. Modifications, like (7,3), (10,4) or (5,5;4) counter, are doing more aggressive reduction or reshaping. The problem with carrysave adder is the irregularity and complex routing problem.

The adder structures are investigated with Module Compiler, a good tool to do the datapath design and architecture explorations. Table 5-1 is the summary of the

architecture comparisons from MC. It compares adder trees with 128 operands for different types of propagation adder and carrysave adder. Inputs are 1-bit samples from the ADC, multiplied with 5-bit tap coefficient. It is equivalent to sum up 128 partial products with a 12-bit wide final output, if no internal rounding occurs.

Not too surprisingly, for the design without carrysave, both area and delay get increased. MC is good at reducing the area and delay for carrysave adder [MC00]. And given the bit width of the output, the ripple carry adder for the final merging stage is the best one to use.

1-bit multiplication was shown to be the most efficient in implementation when using the `sgnmult()` built-in function of MC. The PMF is the largest block of the whole system, therefore, optimizing the size is critical for the whole design.

Area ( $\mu\text{m}^2$ )	Delay (ns)	Parameters
87284	6.32	fat=ripple,cs=on
87284	6.32	Fat=clsa, cs=on
87284	6.32	Fat=csa, cs=on
87648	6.87	Fat=cla, cs=on
88044	6.71	fat=fastcla,cs=on
128062	7.03	fat=ripple, cs=off
128062	6.89	Fat=clsa, cs=off
128062	7.03	Fat=csa, cs=off
131344	7.32	Fat=cla, cs=off
136352	7.38	fat=fastcla, cs=off

**Table 5-1 Architecture comparison of adder tree**

The size of the PMF depends on the input vector width, which is two times of the pulse spreading length, i.e.  $N_{\text{pulse}}$ . The area grows as the square function of  $N_{\text{pulse}}$  as shown in Table 5-2. The number of PMF output signals is also  $N_{\text{pulse}}$ . Each output corresponds to sampled inputs of different time offsets going through the matched filter. We made the

maximum input vector width or the length of filter taps to be the power of two, so that we could fully make use of the output bits.

As mentioned in the system specifications, the UWB baseband should allow flexibility to vary the pulse spreading length for our future testing requirements. In addition, the entire block runs at the pulse repetition rate, which is the highest clock rate in the baseband; therefore, we need to disable some portions of the block via a control vector  $\{g_i\}$  to save power. Since there is no pipelining inside the block, the way to disable the adder trees, is to use control vector  $\{g_i\}$  to mux between GND and the input vector  $\{C_i\}$ . Via the control vector  $\{g_i\}$ , one can do power saving in the tracking mode and simultaneously achieve the flexibility of varying pulse spreading length, i.e.  $N_{pulse}$ . Adding this extra circuitry increases the PMF area 18% according to MC report. The generation of control vector will be described in control logic section.

Another control signal, AL, is enabled when turning into always-on mode, where the receiver samples continuously. In this mode, the length of the input sample vector from the ADC will be exactly one pulse repetition period. The enabled AL signal will mux the input vectors into registers. These delayed samples together with the current incoming samples can be fed into the same PMF adder tree structure to do matched filtering. Of course, one has to change the pattern, programmed into the coefficient register array,  $C\{i\}$ , so that the filter taps are matched to the corresponding incoming signals. Through the AL control signal and appropriate coefficients, one could continuously capture UWB pulses without any information loss in the always-on mode.

<b>Number of FIR taps</b>	<b>Coefficient bit width</b>	<b>Area (<math>\mu\text{m}^2</math>)</b>	<b>Delay (ns)</b>	<b>Output bitwidth</b>
128	1	1430428	6.16	8
128	3	3551232	6.82	10
128	5	5788436	7.46	12
128	7	7946344	7.71	14
64	1	355656	5.31	7

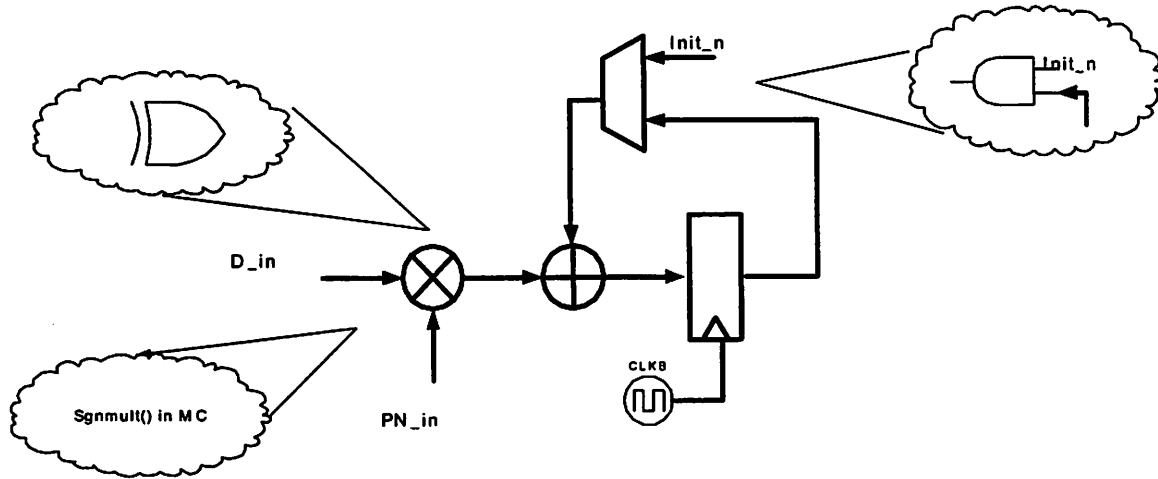
64	3	889668	6.43	9
64	5	1446620	6.66	11
64	7	1986366	6.77	13
32	1	88804	4.74	6
32	3	220516	5.49	8
32	5	357296	5.79	10
32	7	489448	5.76	12

**Table 5-2 Area/delay versus size and bit width of PMF**

<b>Input</b>	<b>Attributes</b>	<b>Output</b>	<b>Attributes</b>
S1 ~ S127	1-bit samples from ADC	Z0 ~ Z127	12-bit outputs
C1 ~ C127	5-bit filter tap coefficients		
g0 ~ g127	Enable/disable cntrl signals		
AL	Always On mode		

**Table 5-3 I/O of PMF**

## 5.2 PN Correlator and Downsampler



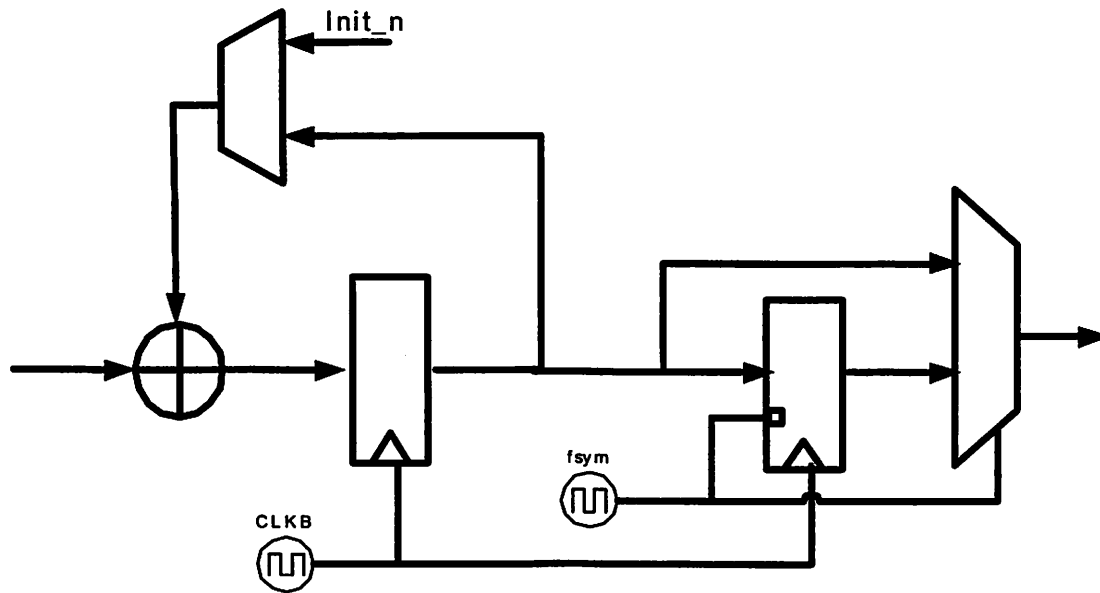
**Figure 5-5 Architecture exploration for correlator**

The PN correlator is composed of an accumulator and downsampler. The accumulator is running at the chip rate, and the downsampler registers the final value in symbol rate. Several accumulator and downsampler architectures are compared in MC.

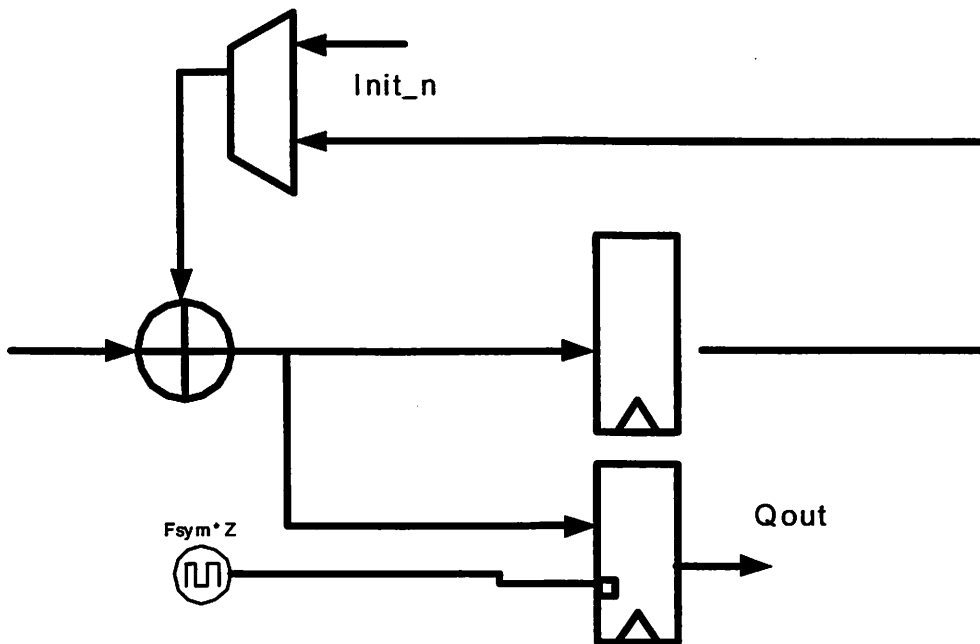
For the accumulator, architecture one and two utilize the `sgnmult()` built-in MC 1-bit multiplication function, and use a mux to feedback the register output. Architecture three uses a bitwise XOR on the incoming signals `D_in` and `PN_in`. The adder adds feedback value, XOR value and `PN_in`. Architecture four uses a mux to choose between the negated and the original value. The negation is done by inverting each bit of the incoming signal `D_in` and adds one at the following adder as the carry in. This is two's complement negation. Architecture five again makes use of built-in function in MC, `mac()`, turns out this is the most efficient implementation in terms of area. However, `mac()` takes in at least 2-bit wide signal, while in our case, `PN_in` is only 1-bit wide. Therefore, we cannot use the `mac()` function. Table 5-4 demonstrates the area and timing performance of each architecture. Architecture one was finally adopted.



a)



b)



**Figure 5-6 Downsampler architectures**

For the downsampler part, two different architectures could be implemented, shown in Figure 5-6. Architecture one, shown in Figure 5-6a, makes use of an enabled register and mux. The symbol-boundary clock,  $f_{sym}$ , drives the enable port. The use of the mux

ensures the correlation value is ready at the first cycle of the next symbol. If the correlation over the previous symbol period cannot be ready at the first cycle of the next symbol, there will be a penalty of losing at least one symbol time, which increases the cost of the whole system. For example, during tracking mode, this increases the possibility of losing track of the UWB pulse. The second implementation, Figure 5-6b, directly connects the node before the accumulation register to an enabled register, whose enable port is driven by a early-symbol clock, which is a shifted version of  $f_{sym}$  with exactly one chip period offset. The first architecture is vulnerable to glitching because of the propagation delay of the mux and  $f_{sym}$ . While in the 2<sup>nd</sup> implementation, the output is synchronous without glitches, although a one-chip earlier clock has to be generated. In order to avoid the misdetection caused by the temporary glitch, 2<sup>nd</sup> architecture is the one to be used.

Arch	Area	Delay	Bit width
1	1030	2.31	7 -> 11 (16 chips)
2	1188	2.61	7 -> 11 (16 chips)
3	1186	2.4	7 -> 11 (16 chips)
4	1190	2.31	7 -> 11 (16 chips)
5	960	2.38	7 -> 11 (16 chips)
5	1248	3.54	(32 in)10 -> 14 (16 chips)
5	1500	3.51	(32 in)10 -> 17 (128 chips)
5	1754	3.54	(32 in)10 -> 20 (1024 chips)

**Table 5-4 Area/Delay of PN correlator**

From the previous chapter, we've decided the whole system contains 11 PN correlation blocks, which amounts to roughly 5 mm<sup>2</sup> chip size. Each correlation block has 128 correlators.

A control signal, named lock, will connect either PN register output or GND to PN\_in. When the system turns into data recovery mode, this lock signal will push all PN

correlators into sleep mode for power saving purpose. In sleep mode, the value on D\_in holds and PN\_in connects to ground, therefore no further charge or discharge actions will take place on all nodes inside the block. The signal isolation is very crucial in the design, because most portion of the system is used for faster synchronization or testing purpose. If the whole system keeps running, it will inevitably consume too much power, which lost the original motivation of doing low power design.

<b>Input</b>	<b>Attributes</b>	<b>Output</b>	<b>Attributes</b>
D_in	12-bit input from PMF	Q_out	22-bit outputs after dump register
PN_in	1-bit Barker code		
Init_n	Reset signal from relational output of counter		
Init_early	Enable signal for dump register		
Lock			

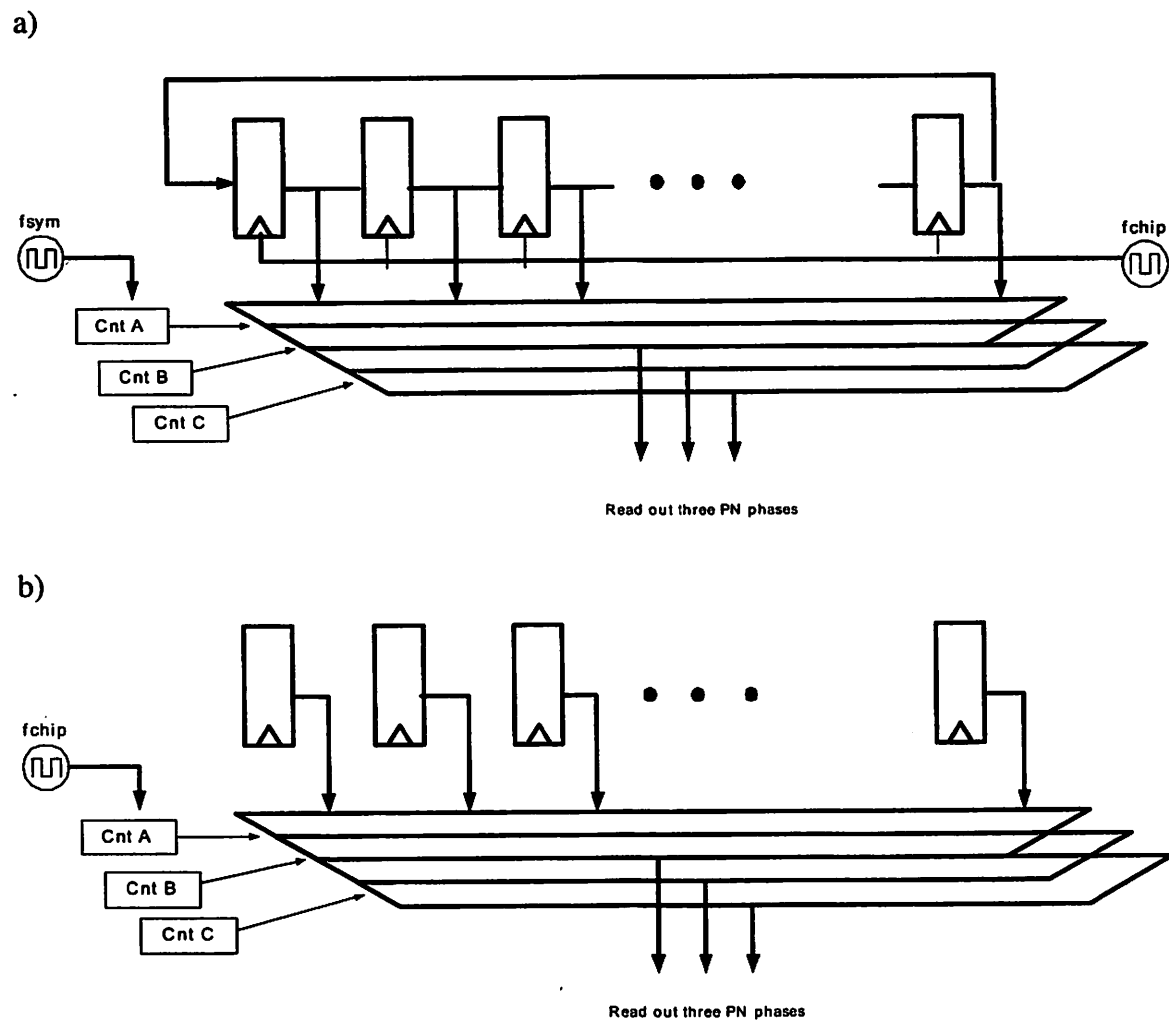
**Table 5-5 I/O of PN correlator**

### **5.3 Coefficients Registers**

The storage of filter coefficients can be programmed from off-chip. Several ways are possible to store the values. A scan chain register is one possibility that is already built-in. The most convenient way to implement these registers given the current design flow is simply to build a shift register chain with enable ports. All the clock ports are connected to the global clock. During the initialization of the chip we enable the flops and clock in the values to be stored in these registers. After programming, the scan functionality is disabled and the registers will hold the values until scan is enabled again. Another possible implementation is to use an addressable register file. However, it requires additional address input and data input pins. In the current design, the coefficient width is

5-bits, while address input is 7-bits, which amounts to more than 50% overhead in order to scan in 5-bit data. As long as we could accurately control the clock cycle along with the corresponding data input, the shift enable registers should be the easiest way to implement the configurable baseband.

### 5.4 PN Generator



**Figure 5-7 Architectures of PN generator**

The PN generator is composed of registers, which are accessed via a mux. It shifts the PN phase according to the input control signal. Two architectures are explored to store PN

sequence. Architecture one, Figure 5-7a, is just shift-registers. It's easier to split out the different PN phases (each internal node represents one PN phase). PN chips rotate around the registers at the chip rate. Therefore, power is constantly dissipated on the ring once the system is on, no matter which operation mode is active. The alternative, Figure 5-7b, makes use of a register array along with a mux to read out the stored values. In this architecture, no power is consumed on these registers. Instead, power dissipates on the internal nodes of mux array, the same as for architecture one. However, this topology requires a counter running at the chip rate in order to read out the PN chips, compared to a counter running at the symbol rate in first topology. When system gets into tracking mode, there is only one counter still running, and the others stay zero. To determine which architecture consumes more power, one has to compare the load of the mux control bus and internal nodes of shift register ring. Load on mux control bus is roughly  $\text{num\_mux} * \text{PN\_chips} * C_{\text{gate}}$ , while load on shift registers is roughly  $\text{PN\_chips} * (\text{num\_mux} * C_d + C_{\text{reg}})$ . If diffusion capacitance is comparable to gate capacitance, the power consumption is larger in architecture one owing to larger load and continuously running feature. Architecture two was selected.

In the acquisition mode, each mux will read out one particular PN phase, same as rotating PN shift registers. In tracking mode, only one PN phase is needed.

## 5.5 Peak Detector

The peak detector finds out the maximum correlation value and its position. In MC, there is a  $\text{max2}()$  function which compares two numbers and output the maximum one. If the first input is greater than or equal to the second input, it will set the output to 1. In order to pick the maximum value out of 128 signals, a binary tree of  $\text{max}()$  functions is used. The comparison signals are concatenated from each level of the tree (LSB to MSB), which will give the position of the maximum signal.

The following block is a comparator, whose programmable threshold can be calculated given the information of input waveform and input SNR, as mentioned in chapter three.

Finally, all outputs, named {Ati}<sup>1</sup>, from each comparator are fed into an OR gate, which forms the lock signal. Assuming there is at most one correlation block above the detection threshold, the lock signal goes high whenever the system gets synchronized. And the location of synchronized correlation block should be identifiable by decoding {Ati} signals. The “which\_b” signal tells which block is the synchronized one.

## 5.6 Control Logic

The Control logic is composed of two main components: Main\_Cntrl and Readout. The Main\_Cntrl block takes in “lock” and “which\_b” signals from the processing elements, including PMF, correlation block and detection blocks. It then outputs all the control signals that required in acquisition mode, tracking mode and the transition between the two. The role of this block is very crucial for the whole system. It determines when and how to change operation modes. The operation modes and transition issues are addressed in the previous chapter. The I/O ports of Main\_Cntrl are summarized in Table 5-6.

While Readout block is a set of configurable counters that control the readout mux of PN registers. Each configurable counter, “readouti”, is described using an individual stateflow block, instead of simply using “Mod” operator. The reason is that “Mod” operator cannot have a variable operand. In order to make a synthesizable VHDL description, the operand has to be a constant power of two, which is absolutely not allowed in our flexible UWB baseband. Therefore, input signal “Wrap” is a parameter to set number of chips. The readout signal should reset to zero whenever it hits “Wrap”. Also, if the beginning phase of the readout signal exceeds “Wrap”, it will reset to zero since it’s already in the overflow state. The stateflow handles those issues. The I/O ports of Readout block are summarized in Table 5-7.

Control logic is implemented with stateflow in Simulink. SF2VHD is then used to translate stateflow into VHDL code [Camera01].

---

<sup>1</sup> Ati is the output of comparator that follows correlation block i.

<b>Input</b>	<b>Attributes</b>	<b>Output</b>	<b>Attributes</b>
At	Goes high if any of the correlation block get synch	lock	Output high when system in synch
Adr	Which correlator within the correlation block that gives the peak value, i.e. on-time instance	maxadr	Index of on-time correlator
Sym_en	A sequence of pulses running at symbol rate	Sel_b	
Wrap	Number of PN chips being tested	Strobe_ph	Control signal for defining the symbol boundary
Which_b	Which correlation block that locks the UWB pulse	PN_ph	The PN phase offset, output to Readout block
Early	High if "Early" samples give larger correlation value than the "On-time" samples'	sh_win	Control signal feedback to analog front end for shifting sampling window
Late	High if "Late" samples give larger correlation value than the "On-time" samples'		
Guard	"Guard" margin to move the sampling window		
Num_corr	Number of correlators within a correlation block		
Num_b	Number of correlation blocks in the system		

Input	Attributes	Output	Attributes
At_track	“At” equivalent signal during tracking mode		
SFRESET	Reset signal for control logic		
num_sh	Time offset of shifting window		

**Table 5-6 I/O of main control block**

Input	Attributes	Output	Attributes
Wrap	Number of PN chips being tested	Pnread{i}	Control signal to drive PN readout mux for correlation block {i}
PN_ph	The PN phase offset, input from Main_Cntrl block		
Sym_en	A sequence of pulses running at symbol rate		
Lock	Lock signal from Main_Cntrl block		
At_track	At equivalent signal during tracking mode		
SFRESET	Reset signal for control logic		

**Table 5-7 I/O of readout block**

## 5.7 PMF control signal decoder

In acquisition mode, the decoder has to block out those unused slices in PMF, which will also effectively shut off the corresponding ones in the PN correlation block. A thermometer decoder is used in this mode.



In tracking mode, one needs to decode the peak address that is locked from acquisition mode plus the neighbor address, and spit out 128 control signals to disable unused correlators. In this way, one could make sure only three slices are running at the tracking mode in order to save power. Otherwise, the PMF is running at chip rate, which is the highest clock in the system. When optimizing for speed, in MC, the delay is 1.72ns with an area of 23488  $\mu\text{m}^2$ .

## 5.8 Top level I/O

Here is the summary of the final I/O pins in the baseband.

Input	bits	Attributes	Output	bits	Attributes
S0~S255	1	Inputs from Analog frontend	Data_out	22	Detection output
al	1	Always on mode activate	sh_win	2	
cliff	7	Determine the window size			
spc	7	Spacing between the early/late samples			
C_in	5	Input to coefficients reg. array			
coef_en	1	Enable coefficients reg.			
PN_in	1	Input to pn reg. array			
pn_en	1	Enable PN reg.			
Wrap	10	Length of PN sequence			

<b>Input</b>	<b>bits</b>	<b>Attributes</b>	<b>Output</b>	<b>bits</b>	<b>Attributes</b>
num_corr	7	Number of correlators within one block			
num_cb	4	Number of correlation blocks			
SFRESET	1	Reset signal to stateflow			
RESET	1	Reset signal to all registers in the system			
Guard	7	The margin to toggle the sh_win			
Threshold_T	22	Threshold for tracking block			
Threshold_S	22	Threshold for pci block (synch)			
Num_sh		Number of shifting window once sh_win==1			

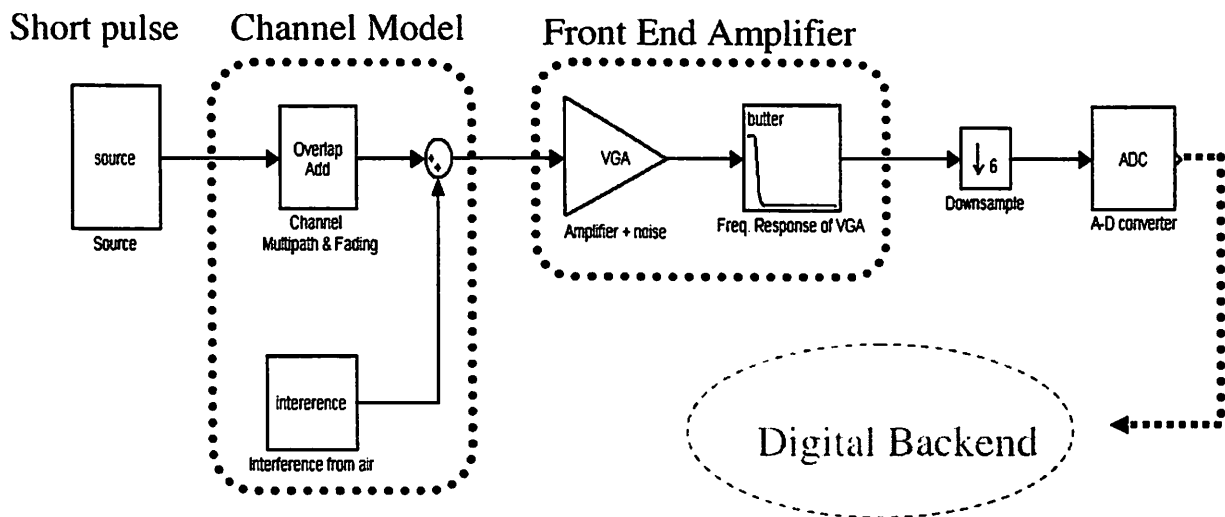
**Table 5-8 Top level I/O**

# Chapter 6 System Simulation

## 6.1 Introduction

A system simulation environment has been built in Simulink from the transmitter through the multipath channel and analog frontend to digital backend. One can include the impairment from the channel, antenna, and RF frontend into this model and probe the system performance or check the functionality of digital backend. The Simulink model has a conceptually clear block diagram, where designer could study how each block affect system performance. And the digital backend could make use of fixed-point blocks or Xilinx blockset (for Xilinx FPGA), which are consistent with hardware implementation. However, the slow simulation speed prohibits us from running long-run simulations, such as BER testing. Implementation on BWRC emulation engine (BEE) has been done.

## 6.2 Communication Chain Modeling

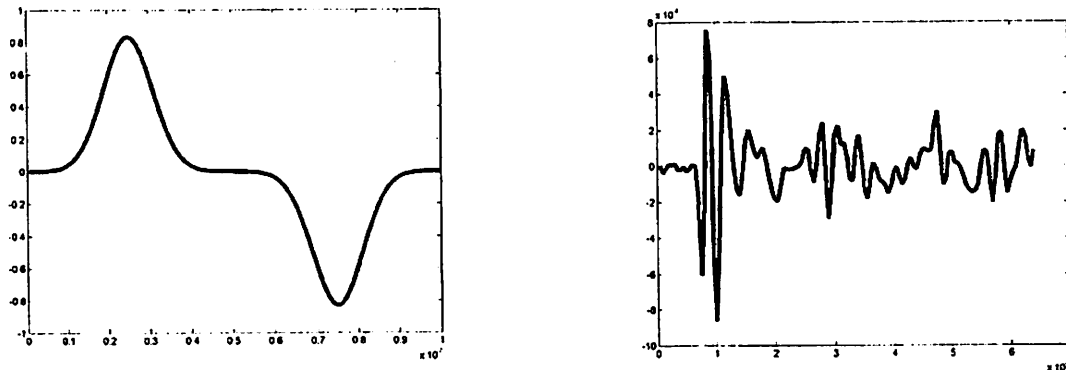


**Figure 6-1 Communication chain overview**

The target of using Simulink is to build a hardware consistent description for digital backend. In order to test the performance of UWB baseband and impacts from analog or channel impairments, we include the whole communication chain from transmitter to the final detector in Simulink environment. This section will briefly describe the chain before digital backend.

• TRANSMITTER MODELING

The transmitter is basically a pulser and antenna. The pulse shape will depend on the type of Tx and Rx antenna and receiving angle. The modeling of the antenna from a circuit perspective is ignored in the simulation. Because it is the final pulse shape seen by the receiver front end that will affect the system performance. Here, we separate the wireless channel (multipaths) and antenna effects for convenience, although both can be lumped into one channel distortion. The pulse shape measured in the lab is saved in matlab workspace, and called into Simulink. Two examples of pulse shape are shown in Figure 6-2.

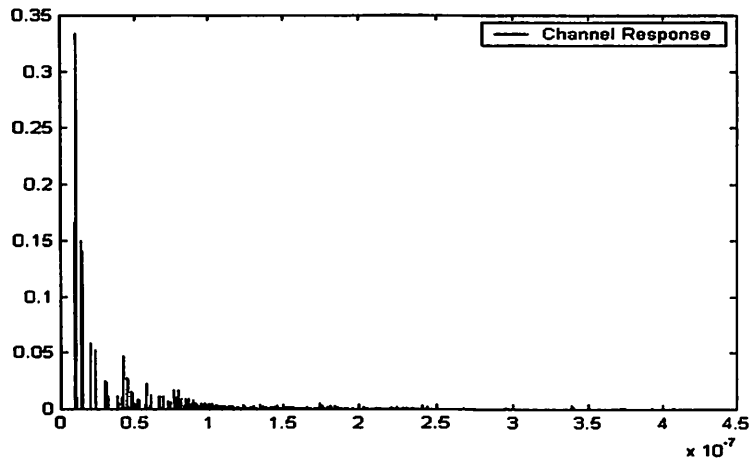


**Figure 6-2 a) Ideal Doublet pulse by current loop antenna b) Monocycle pulse by monopole antenna**

• CHANNEL

To begin with, the system assumes a time-invariant channel. The channel model is making use of ray-tracing algorithm [Tang02] given a certain floor plan of the

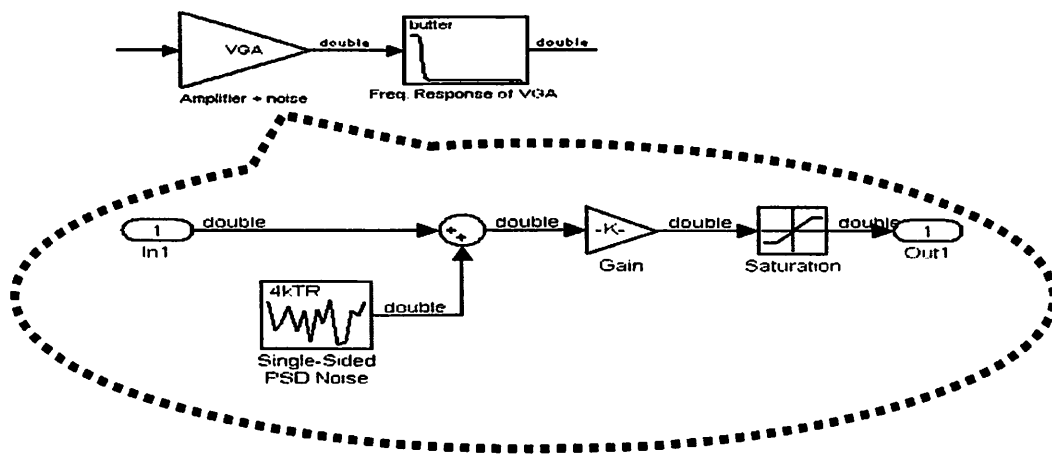
environment and position of Tx and Rx antennas. Figure 6-3 is a channel model example for Rx and Tx antenna three meters apart.



**Figure 6-3 Multi-path channel model at Cory 2nd floor**

Another impairment from wireless channel is the in-band (MHz to 1 GHz) narrowband interference. The interference is modeled in Simulink by capturing five to ten biggest interferers based on the measurement in BWRC lab and assigning each a random initial phase.

• FRONT END AMPLIFIER



**Figure 6-4 Front end amplifier modeling**

The amplifier is simply modeled by injecting 50-ohm thermal noise (assuming the LNA is doing 50-ohm impedance matching), and the aggregate noise figure of the receiver chain, amplifier saturation modeling, and gain. The frequency response and the internal filter stages are modeled by a Butterworth IIR filter in Simulink, where one can specify filter types, orders and passband edge frequencies. The linearity issue of the amplifier is ignored in the simulation because it can be lumped into part of pulse shape change.

• S/H AND ADC

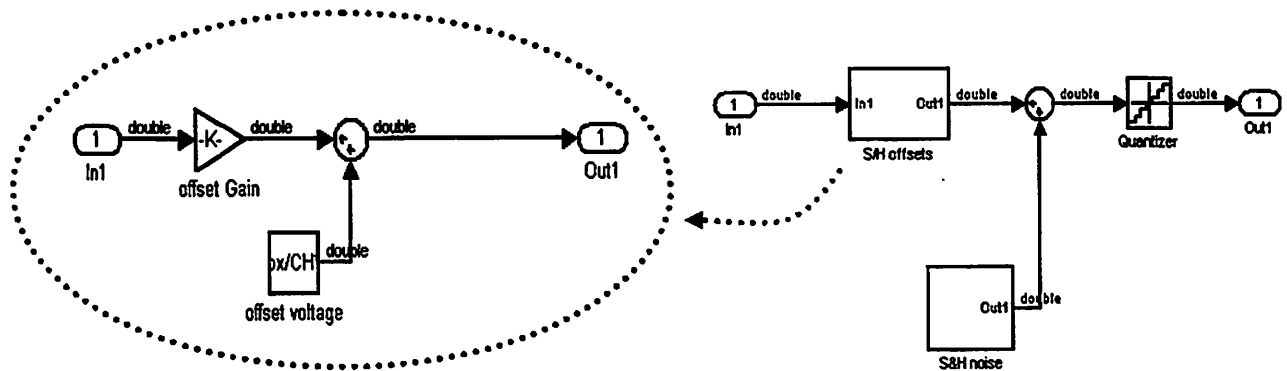


Figure 6-5 Sample and hold circuit and Quantizer

The pulse is being sampled and quantized. Sample and hold circuit models the thermal noise by  $KT/C$ , 1 pole approximation, plus charge injection and clock feedthrough induced offset voltage and gain error. For the ADC part, a Simulink quantizer is adopted or slicer if it is a 1-bit ADC. The thermal noise contributed by the circuits is substantially smaller than the interference from wireless channel. Therefore, the modeling of thermal noise is simply a first order approximation, which should be accurate enough for the system simulation.

- A SIMULINK SIMULATION RESULT

Signal is probed through the whole communication chain from input to the output of ADC. The figure shown is a pure signal transmitted without noise. Otherwise, one couldn't see any signal on scope (all hidden in the noise level).

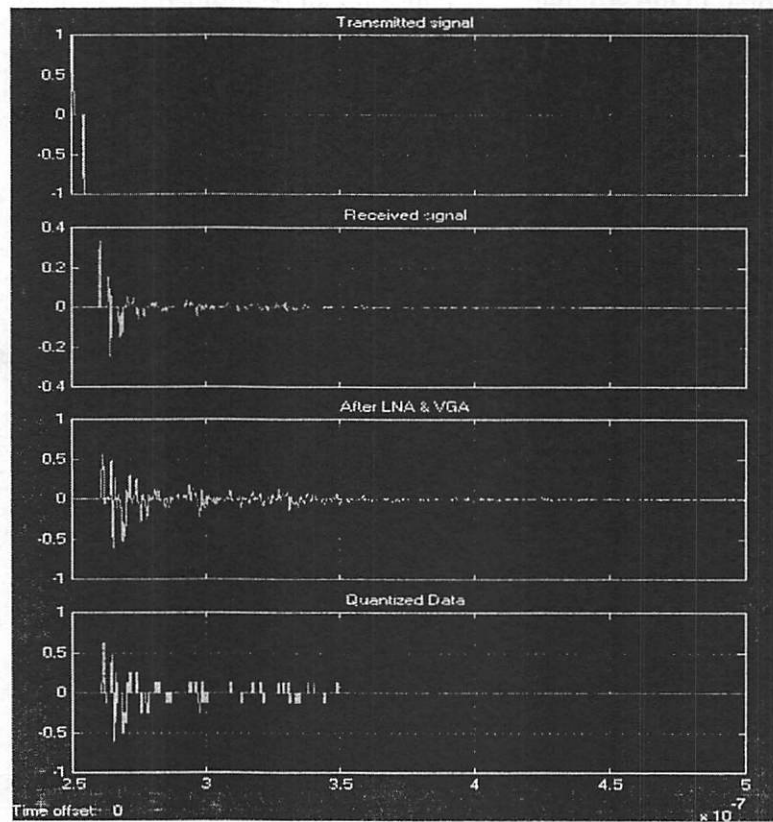
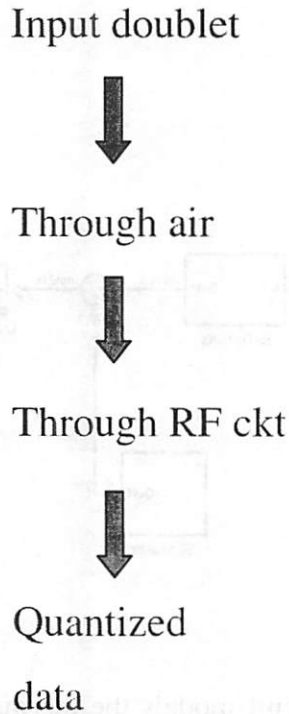


Figure 6-6 Signal waveform along the chain

### 6.3 Xilinx Implementation of Baseband

A Simulink implementation using the Xilinx Blockset and Stateflow has been done. It has been compiled through system generator, generating the VHDL code and cores for all of the Xilinx blocks. Then XST or Synplify Pro (Synplicity) were used for synthesis. Together with the Xilinx CORE Generator and LogiCORE IP libraries, the FPGA implementation could be translated from Simulink blocks.

One version of the baseband system with smaller scale has been implemented on BEE (Biggascale Emulation Engine) in order to fit on one FPGA. It uses a 16-tap FIR filter for the PMF, three correlation blocks, and eleven chips of Barker code. Inputs from ADC are made up from Matlab. It is transmitting a doublet sequence at a certain pulse rate. The preamble packet send ten ones, and data packet sends another ten information bits for testing the functionality of the baseband. Figure 6-7 shows the top-level view of the whole system. Some of the components and method of building the system will be described as follows.

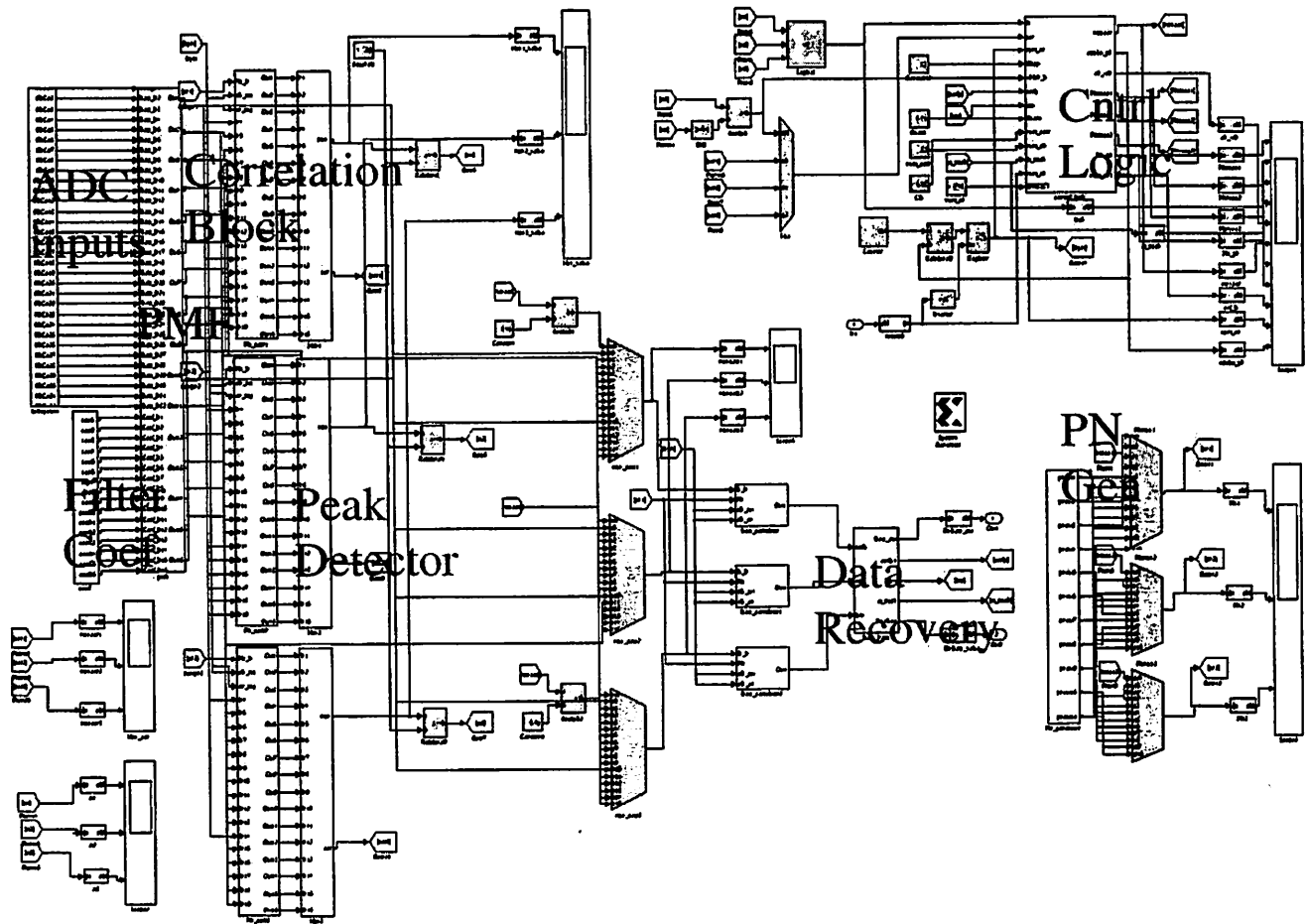
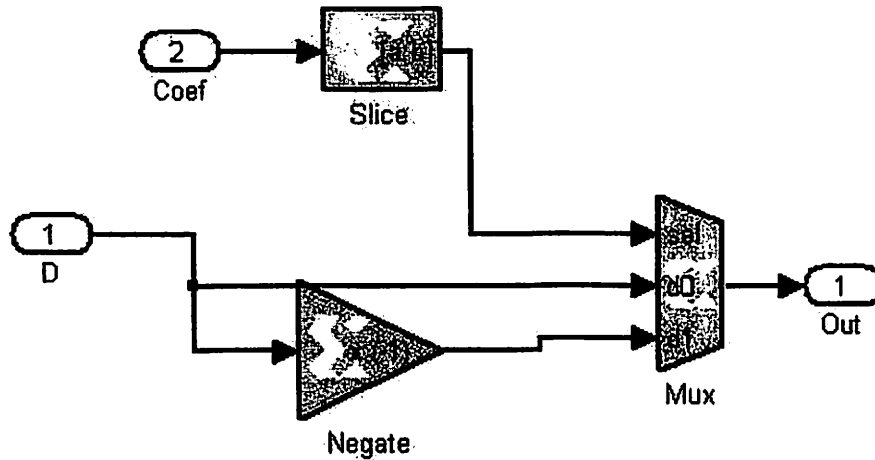


Figure 6-7 Xilinx Implementation Overview

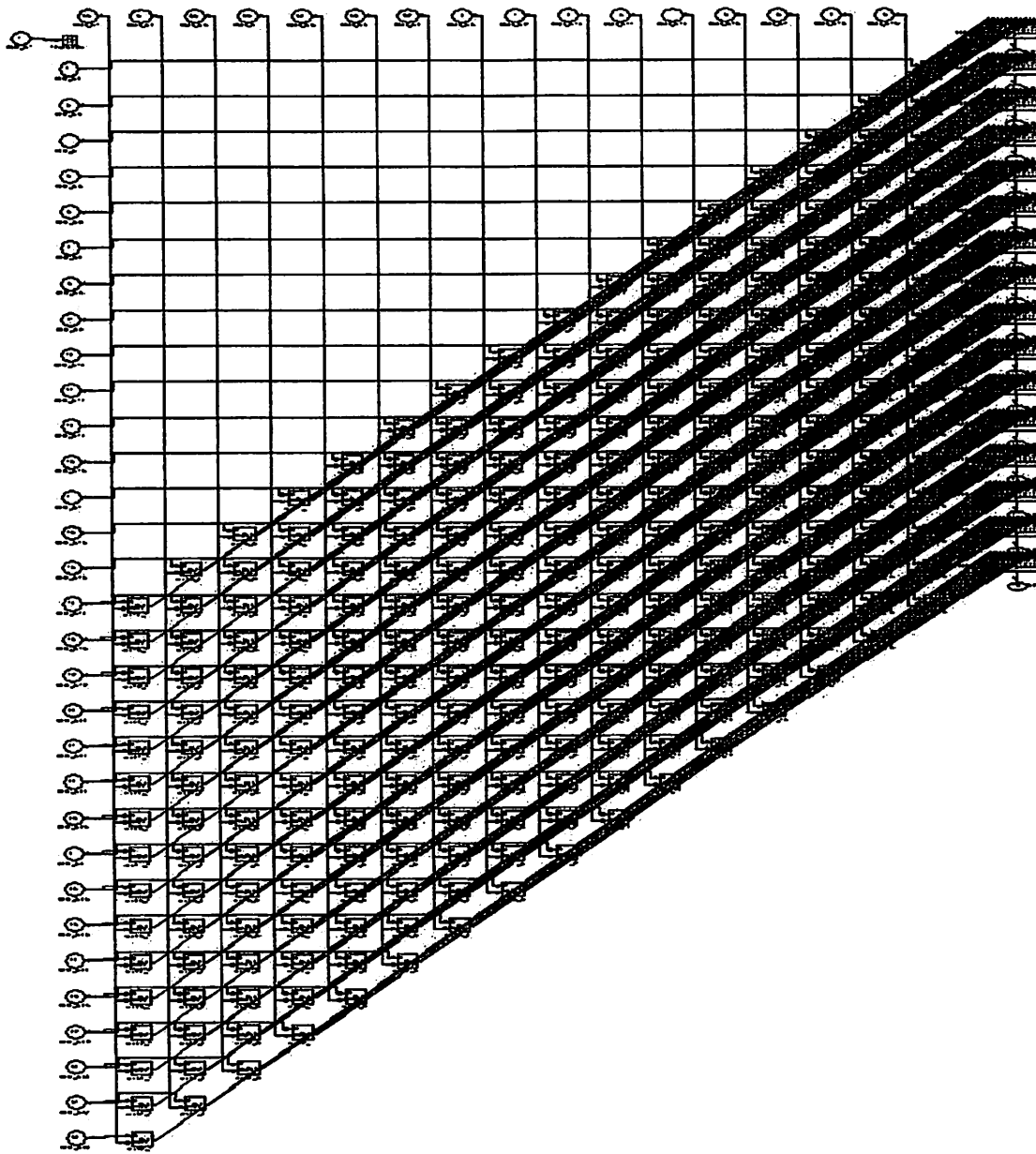


One basic element in the system is doing 1-bit multiplication in PMF and correlator. It is done using a negate block and mux as shown in Figure 6-8. The output is chosen to be positive or negative one depending on the sign bit.



**Figure 6-8 1-bit multiplication**

In the PMF, the parallel binary adder trees are rather difficult to layout manually. Therefore, a script written in Matlab is used to generate this block. As a matter of fact, in order to quickly change the scale or architecture of the baseband, writing a Matlab script is much more convenient than placing those blocks in Simulink especially for large design. Shown in Figure 6-9, a lot of interconnects goes from 1-bit multiplication to binary adder tree.



**Figure 6-9 PMF block**

The correlator is composed of an accumulator and downsampler, shown in Figure 6-10. The accumulator is used to sum the input over one symbol period and reset at the very end. The downsampler registers the accumulated value at the very end of the chip cycle of previous symbol period before it gets reset, and the value should be ready at the beginning cycle of next symbol period.

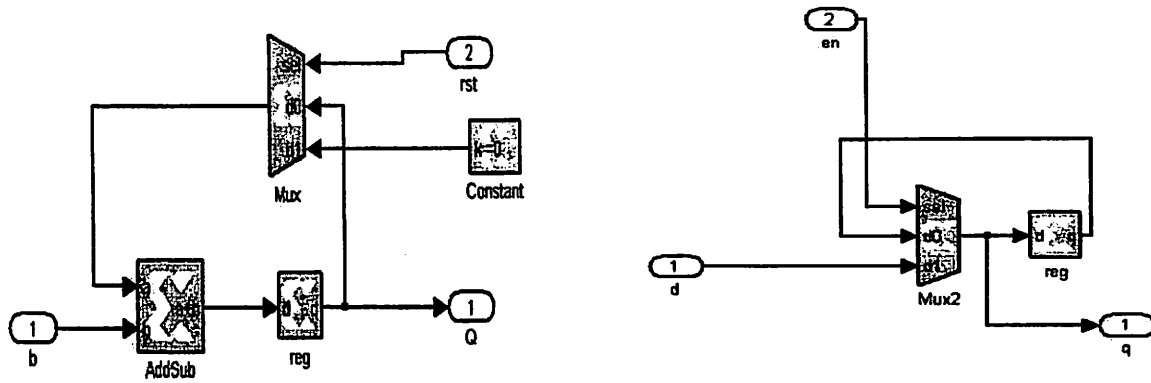


Figure 6-10 a) Accumulator b) Downsampler

The data recovery block keep tracking three correlator outputs: early, late and on-time. And it feeds back the drifting information to the control logic. Before we compare the early/late sample with the on-time sample, an absolute value block is inserted, so that the system always compares the signal energy regardless of the data modulation.

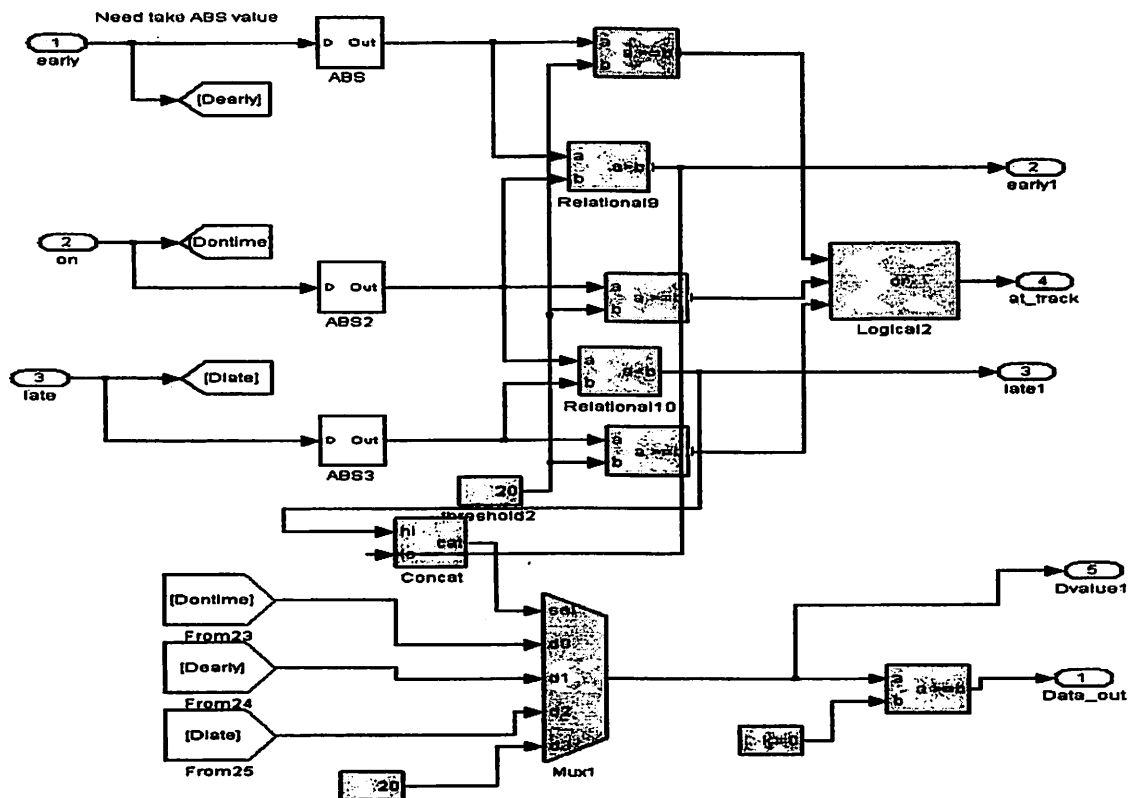


Figure 6-11 Xilinx Data recovery block

Several storage components, including the filter coefficients and PN sequences, are implemented in the system using the Xilinx block “Constant”, and value is the variable from Matlab workspace. This allows one to change the PN sequence or pulse shape easily from Matlab.

The peak detector is made out of a tree of max cells. Each max cell outputs the maximum of the two inputs with a logic bit, which goes “zero” if 1<sup>st</sup> entry is larger than 2<sup>nd</sup>. Thus, by concatenating these bits, the final output of the tree will indicate the position of maximum entry.

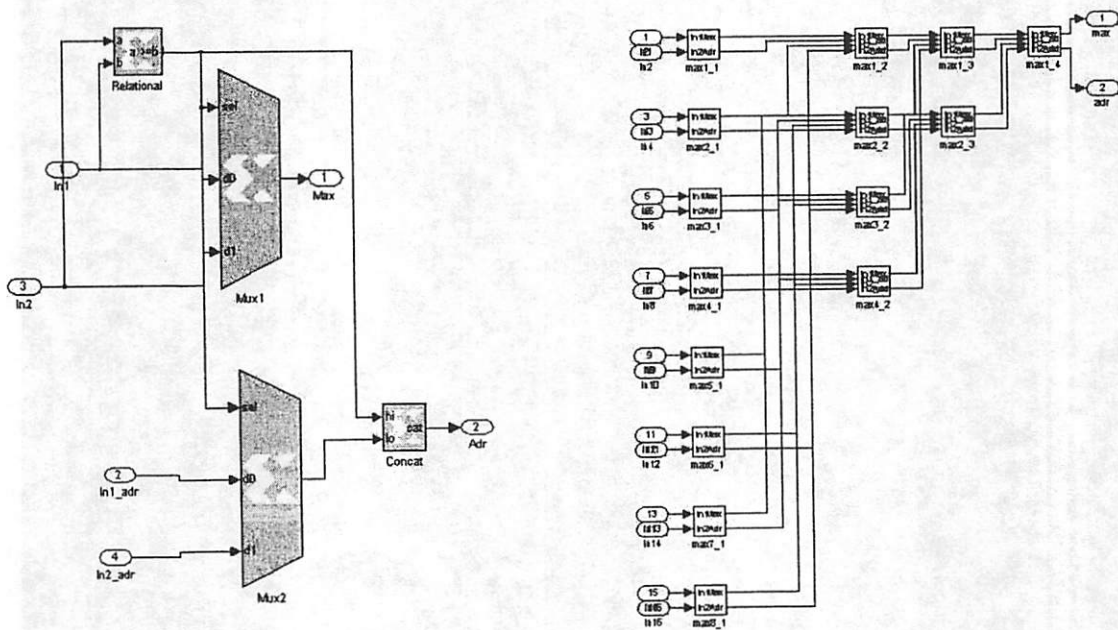


Figure 6-12 a) Max cell b) Max cell tree (Peak detector)

### 6.3 Functionality Simulation

The system just described has been implemented on an FPGA. The simulation results were probed by ChipScope and were shown to match with Simulink outputs. The test vectors are stored in a block ROM on FPGA, and repeatedly being read out, which is essentially the same as real transmitter, sending the pulse sequence periodically.

However, until the input can be fed externally for real time emulation, the system will be constrained to the number of input patterns that could be stored on a single chip.

The simulation results shown here are for both the Simulink simulation and FPGA outputs. Figure 6-13 shows the system starts from the acquisition mode where PNread1 ~ PNread3 are counting. After a while, the “lock” signal goes high and pushes the system into tracking mode. In the right window are shown the early/late/on-time correlation values. As one could tell from the scope, the packet has been demodulated.

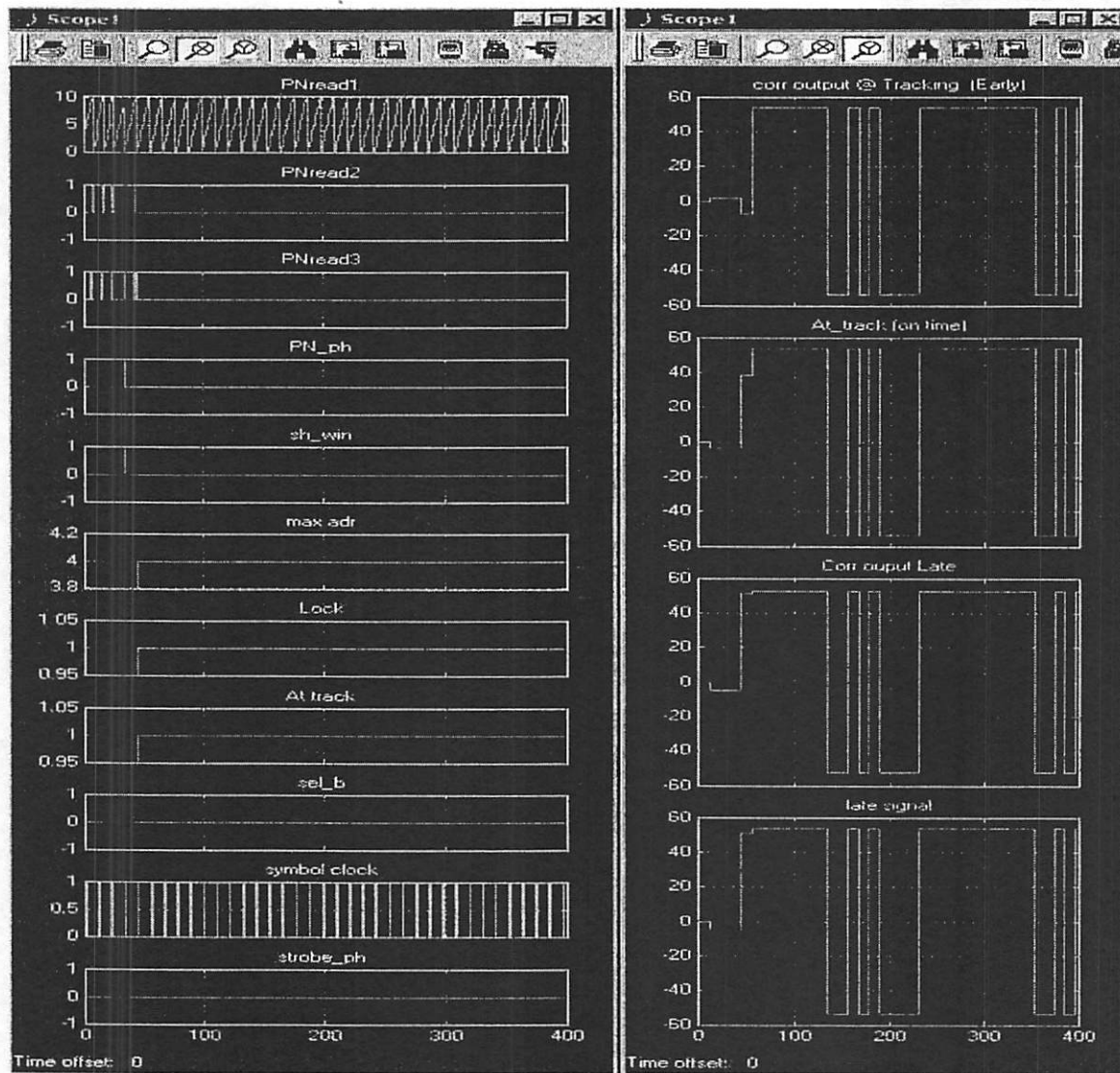


Figure 6-13 Simulink Simulation without drifting

The next simulation has an input pattern that is drifting forwards. The late correlation is always larger than the on-time one, and the “maxadr” signal keeps increasing, which indicates the peak value keep moving toward the boundary of sampling window.

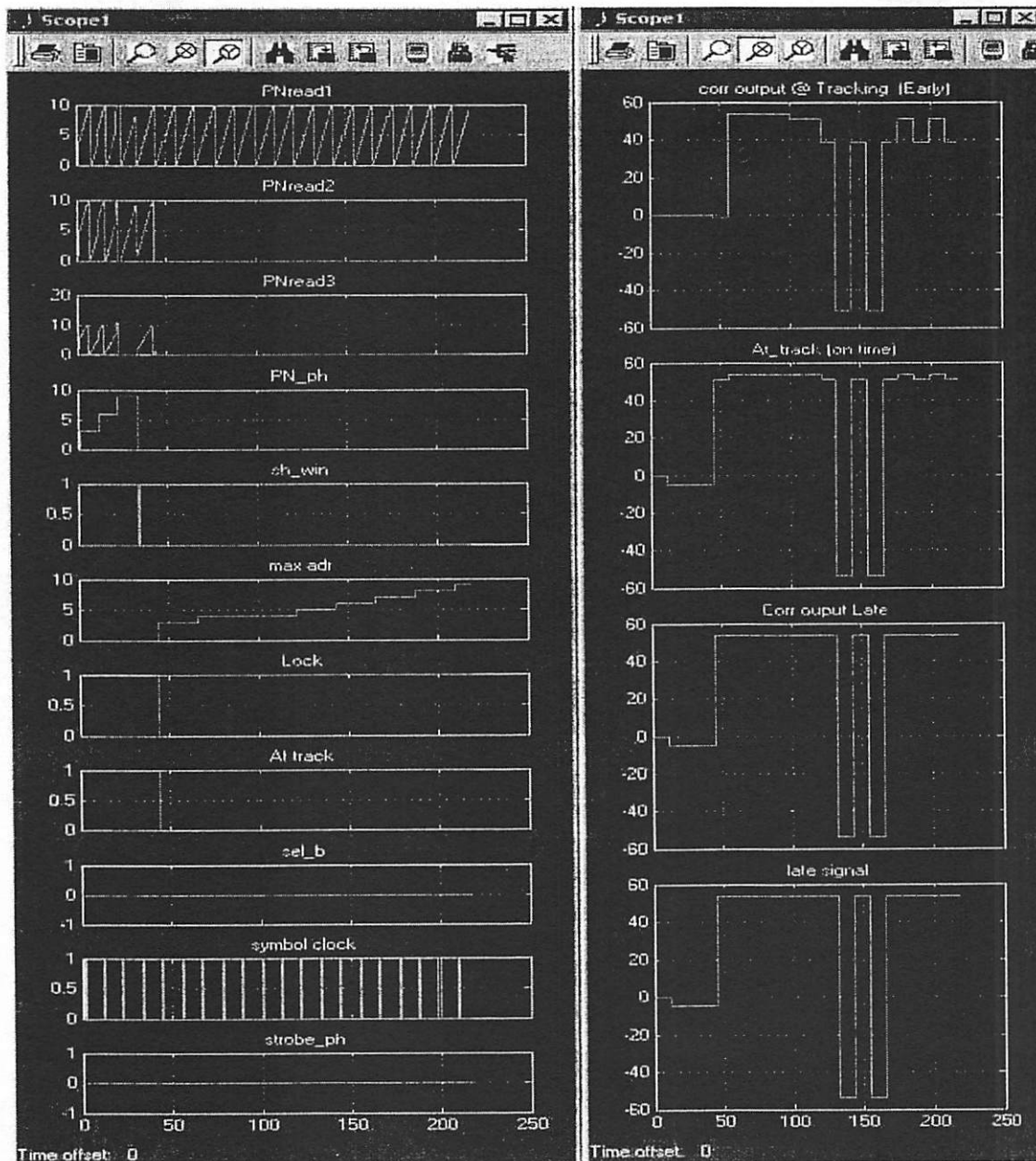


Figure 6-14 Simulink simulation with drifting

Finally, the waveform captured by ChipScope is shown in Figure 6-15. Because the system is continuously running on FPGA, and queues the outputs into a 4K deep buffer,

we could not see the system getting into tracking mode from acquisition mode. What it could show is already in tracking mode.

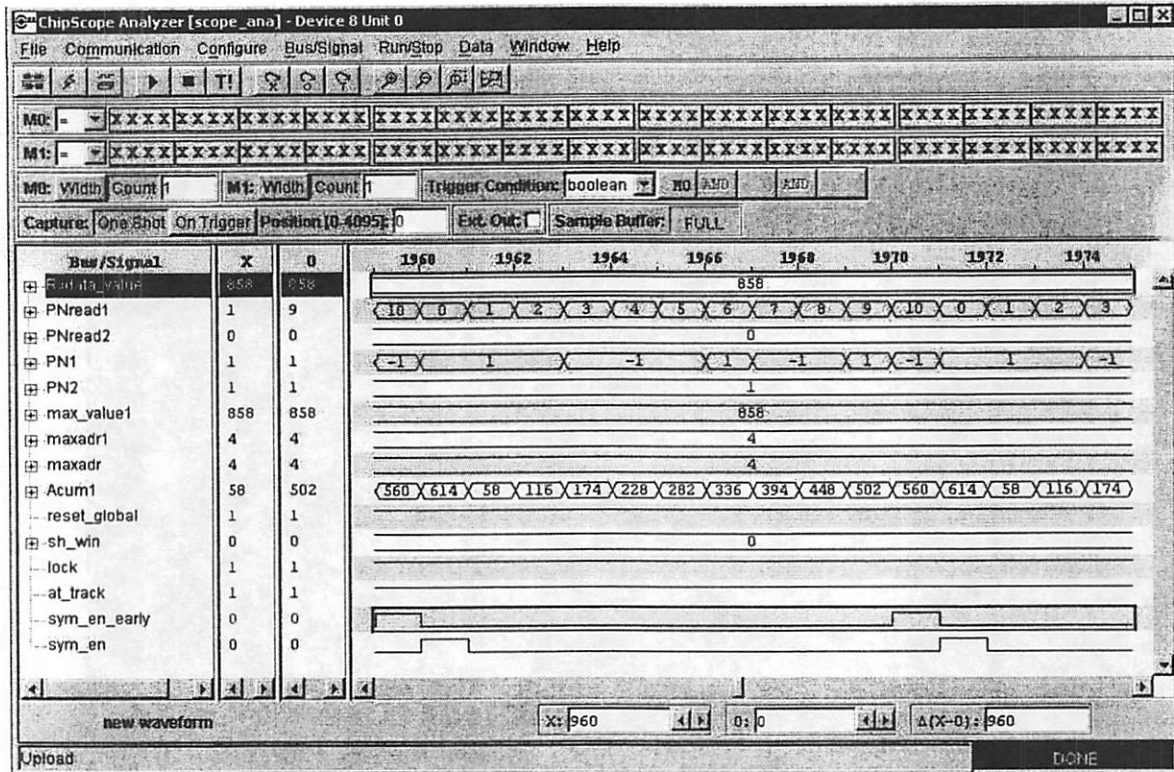


Figure 6-15 Results from FPGA simulation

# Chapter 7 Conclusions

This report presents the background and design issues of the first version of an impulse UWB radio's digital baseband. It extends from the basic detection problem to modulation schemes and channel capacity issues. There, we illustrated the limit of UWB throughput. The required system parameters were also calculated. Then the details of designing each blocks of the system were explored. Finally, a full version of the system implementation was built in Simulink and verified on BEE (Biggascale Emulation Engine) in BWRC. The goal of the current project is a low power ASIC implementation of flexible UWB baseband. Therefore, some architecture explorations in Module Compiler have been done as well.

The work shown in this report is by no means complete. Several improvements can be done for future UWB systems. As the first version, we simply use 2-PAM due to its simplicity and robustness. However, it wastes the time degree of freedom between the pulses. For example, using PPM will carry more information per symbol, which will help us to get closer to the channel capacity. Therefore, modulation scheme can be improved in the future.

Interference is definitely a main concern for system performance. In this first version, we simply treat all interferers as white Gaussian noise. However, one could do better detection if we could somehow whiten the color noise before it goes through the matched filter. The interference may also come from other UWB devices, creating a multi-user detection problem. The system we are currently building is considering single user peer-to-peer communication. After some technique of canceling the interference, 1-bit quantization may not be enough to improve the system performance. Also, from chapter two, we know that in order to increase data rate, we should send higher pulse amplitude.



As the pulse amplitude goes higher while the power spectrum is still under FCC emission regulation, the system may require more ADC resolution.

The transmission frequency band is from MHz range to 1GHz in this project. FCC has regulated the new commercial UWB band to 3GHz to 10GHz. The channel will be quite different at that band and also Nyquist sampling at such a wide bandwidth implies 14 GHz sampling rate, which is exorbitantly high. As mentioned in chapter four, some algorithms have been brought up to do sub-Nyquist rate sampling, which may seem appealing to such a ultra-wide bandwidth data transmission.

How to implement a UWB system in a wider and higher bandwidth while maintaining low power consumption will be the challenging task and main topic for the future. The first version of UWB will definitely help us to understand and learn more about UWB technology.

# References

[FCC02] [http://hraunfoss.fcc.gov/edocs\\_public/attachmatch/FCC-02-48A1.pdf](http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-02-48A1.pdf)

[Aetherwire] <http://www.aetherwire.com>

[XtremeSpectrum] <http://www.xtremespectrum.com>

[Razavi98] B. Razavi, *RF Microelectronics*, Upper Saddle River, NJ: Prentice Hall 1998.

[Cover91] T.M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc, 1991.

[Proakis00] J. G. Proakis, *Digital Communications, 4<sup>th</sup> Ed*, McGraw-Hill Companies, Inc, 2000.

[Scholtz93] R. A. Scholtz, "Multipil access with time-hopping impulse modulation," in *Proc. IEEE Military Comm. Conf*, Boston, MA, October 1993.

[Tse01] D. Tse, Class notes of EE 226A, 2001 Fall.

[Shi02] C. Shi, "Statistical Method for Floating-point to Fixed-point Conversion," M.S. Thesis, University of California at Berkeley, May 2002.

[Scholtz98] M. Z. Win and R. A. Scholtz, "On the Robustness of Ultra-Wide Bandwidth Signals in Dense Multipath Environments," *IEEE Comm. Letters*, Vol. 2, No.2, Feb. 1998.

[Foerster01] J. R. Foerster, "The effects of multipath interference on the performance of UWB systems in an indoor wireless channel," *Proc. Of the IEEE 53<sup>rd</sup> Vehicular Technology Conference (VTC2001 Spring)*, pp. 1176-1180, Greece, May 2001.

[Rusch01] Leslie Rusch, at BWRC lunch seminar, Dec. 2001.

<http://bwrc.eecs.berkeley.edu/Seminars/Lrusch-12.27.01/L%20Rusch%20Seminar%20UWB%20Channel%20Measurements.pdf>

[Taylor00] J. D. Taylor, *Ultra-Wideband Radar Technology*, CRC Press LLC, 2000.

[Vetterli02] J. Kusuma, A. Ridolfi, and M. Vetterli, "Sampling of communication systems with bandwidth expansion," in *Proc. of the IEEE Conference on Communication*, pp. 1601-1605, April 2002.

[Kaplan96] E. D. Kaplan, *Understanding GPS Principles and Applications*, Artech House, Inc, 1996.

[Holmes77] J.K. Holmes and C.C. Chen, "Acquisition Time Performance of PN Spread-Spectrum Systems", *IEEE Tran. On Comms., Vol. COM-25, No. 8*, pp. 778-783, 1977.

[Simon94] M.K. Simon, J.K. Omura, R.A. Scholtz and B.K. Levitt, *Spread Spectrum Communications Handbook: Revised Edition*, McGraw-Hill Inc., 1994

[Zeimer85] R.E. Ziemer and R.L. Peterson, *Digital Communications and Spread Spectrum Systems*, Macmillan Publishing Company, 1985.

[ChenShi01] M. Chen, C. Shi., EE290s Project Report – "Adaptive receiver for UWB data Recovery," Fall 2001, EECS, UC. Berkeley.

[Lars99] W. Lars, *DSP Integrated Circuits*, San Diego, Calif. : Academic Press, 1999.

[Wallace64] C.S. Wallace, "A suggestion for a fast multiplier", IEEE Trans. Electronic Computers, vol. EC-13, pp. 14-17, 1964.

[Dadda66] L. Dadda, "Some schemes for parallel multipliers", Acta Freuenza, vol. 45, pp. 574-580, 1966.

[MC00] Module Compiler User Guide, version 2000.05, May 2000.

[Molisch02]. A. Molisch, M. Win, and D. Cassioli, "The Ultra-Wide Bandwidth Indoor Channel: from Statistical Model to Simulations," IEEE P802.15-02/284-SG3a and IEEE P802.15-02/285-SG3a.

[Camera01] K. Camera, "SF2VHD: A Stateflow to VHDL Translator," M.S., May 2001.

[Tang02] H. Tang, PhD thesis, UC Berkeley, expected 2002.