An Energy Efficient Sub-Threshold Baseband Processor Architecture for Pulsed Ultra-wideband Communications

by

Vivienne Sze

B.A.Sc. Electrical Engineering (2004)
University of Toronto

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Science and Engineering

at the

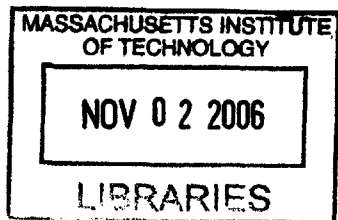MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

© Massachusetts Institute of Technology 2006. All rights reserved.

Author ....
Department of Electrical Engineering and Computer Science
May 12, 2006

Certified by...
Anantha P. Chandrakasan
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Arthur C. Smith

Chairman, Department Committee on Graduate Students

BARKER

An Energy Efficient Sub-Threshold Baseband Processor Architecture for Pulsed Ultra-wideband Communications

by

Vivienne Sze

Abstract

Ultra-wideband (UWB) communications is currently being explored as a medium for high-data-rate last-meter wireless links. Accordingly, there has been much interest in integrating UWB radios onto battery-operated devices, creating a strong demand for energy efficient UWB systems. The objective of this work is to describe how operating the digital baseband processor in the sub-threshold region and increasing the degree of parallelism can translate into energy savings across the entire UWB receiver. While sub-threshold operation is traditionally used for low energy, low performance applications such as wrist-watches, this work examines how sub-threshold operation can be applied to low energy, high performance applications. Simulation results for a 100-Mbps UWB baseband processor using the digital logic cell library of a 90-nm process are presented.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to thank Anantha Chandrakasan for giving me the opportunity to work on a project that challenged me at so many different levels as well as for all his encouragement and guidance over the past two years. I greatly appreciated your support particularly during the stressful times.

I would like to thank Raul who has been a wonderful yoda-like mentor since the beginning. Thank you for introducing me to the world of baseband processing, digital design and anna's taqueria. I would also like to thank Manish for being a constant source of communications and cinematic knowledge.

Life would be easy without having to deal with those darn tools. Thanks to Daniel, Joyce, Nathan, Denis, Frank and Mike McIlrath for all their help in the struggle to tame the tools. In particular, I would like to thank Daniel for all the long hours he spent helping me with what seemed at the time to be a never-ending battle.

Thanks to Dave and Fred Lee for answering all my questions about UWB frontends. I would also like to thank the other members of the ananthagroup for making the lab an incredibly enjoyable, fun and interesting place to work. For all the hours that we need to spend in lab, I'm glad it's with you folks.

Thanks to Jeremy and Fred Chen for taking the time to proof read my thesis and ICASSP papers respectively. Thanks to Margaret for making sure all the paper work runs smoothly.

Finally, I'd like to thank those who provided invaluable support outside the lab. Rumi, thanks for keeping me in shape. Maryam, thanks for being there for the good times and the bad. Thanks for being my family in Boston. And to my family in Toronto, Mom, Jen and TeeTee, thank you for helping through the most challenging and rewarding time thus far. Your unconditional love and support mean the world to me.

I also want to acknowledge STMicroelectronics for providing process and cell library access, and DARPA and NSERC for funding and fellowship, respectively.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Ultra-wideband (UWB) is a wireless communications technology that can achieve high data rates of up to 1 Gbps over short distances using very low power signals [4, 5, 6]. Traditional UWB transmission consists of sending pulses on the order of a few nanoseconds or less that occupy large bandwidths. The short pulse duration gives UWB communications several unique advantages over narrowband communications. First, UWB pulses have high multipath resolution and thus immunity to fading. Second, due to its wide bandwidth, UWB signals can be transmitted at power levels significantly below that of narrowband signals. As a result, UWB signals have minimal interference with other radio frequencies as well as low interception and detection properties [7] .

In 2002, the Federal Communications Commission (FCC) in the United States authorized Ultra-wideband wireless communications in the 3.1-GHz to 10.6-GHz band with a minimum bandwidth of 500 MHz. UWB wireless communications can overlay existing narrow-band services and the FCC spectral mask regulates this coexistence by placing strict limits on the power spectrum with a maximum equivalent isotropic radiated power (EIRP) spectral density of -41.3 dBm/MHz [1]. The spectral mask approved by the FCC is shown in Figure 1-1. UWB has attractive potential applications in short-range high-data-rate wireless communications such as wireless personal
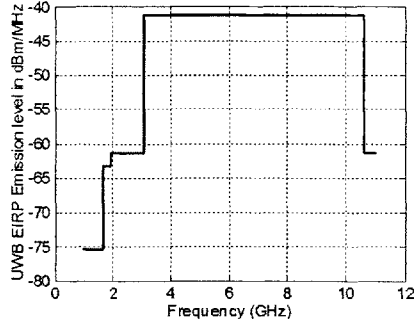
Figure 1-1: EIRP mask approved by the FCC [1].

area networks, localization at centimeter-level accuracy, and high resolution ground penetrating radar (GPR) [8]. In addition, the low power levels and wide spectrum make UWB ideal for stealth communications used by the military, while its high spatial resolution makes it ideal for applications such as medical imaging.

1.2 Motivation

The consumer electronics industry is exploring the use of UWB communications to complement longer range radio technologies such as Wi-Fi, WiMAX, and cellular wide area communications [9]. UWB communications can be used to relay data from a host device to other devices in the immediate area and thus eliminate the need for wires and increase mobility. The use of UWB as a medium for high-data-rate last-meter wireless links requires that UWB radios be integrated onto battery-operated devices such as mobile phones, handheld devices and sensor nodes. Consequently, there is a need for an energy efficient UWB system. This thesis will describe how parallelism in the digital baseband processor can reduce the energy required to receive UWB packets.

1.3 Objective

An energy efficient baseband processor can be achieved by exploiting two forms of parallelism. First, the supply voltage of the baseband can be lowered so that the

16

correlator[1] operates near its minimum energy point. The supply voltage at this minimum energy point occurs below the threshold voltage, placing the circuit in the sub-threshold region [10, 11]. The correlator and the rest of the baseband must then be parallelized to maintain throughput at this reduced voltage. Second, the correlators can be further parallelized for a significant reduction in the synchronization time. The reduced synchronization time allows the baseband and the rest of the receiver to be turned off earlier, resulting in a system-wide reduction in energy.

Therefore, the first form of parallelism reduces the energy per operation of the baseband processor, while second form of parallelism reduces the energy per packet consumed by the receiver. This thesis will describe how to select the optimum degrees of parallelism for each form in order to achieve an energy efficient baseband processor.

1.4 Previous Work

This work will focus primarily on the energy efficiency of the baseband processor architecture rather than on the robustness of the baseband algorithm. It will use a previously designed baseband algorithm [12, 13] as a basis for implementing energy saving techniques. In fact, several simplifications were made to the algorithm so that the focus is placed on those areas where parallelism can be utilized to reduce energy consumption. These modifications will be discussed in Chapter 3.

It is important to note that other works have discussed the use of parallelism to reduce the *power* consumption for a baseband that uses both autocorrelation and cross-correlation [14]; however, the goal of this work is to reduce the *energy* consumption for a baseband that uses only cross-correlation.

1.5 Thesis Outline

This thesis will discuss the design and implementation of an energy efficient digital baseband processor for a UWB receiver. Chapter 2 will provide an overview of the en-

---

[1]A block in the baseband processor used for synchronization and channel estimation. Details provided in Chapter 3.

tire Ultra-wideband system, within which the baseband processor will be integrated. A detailed discussion of the baseband algorithm and the architecture used to implement it will be provided in Chapter 3. The prototyping platform that was used to verify the baseband algorithm will also be presented.

Chapter 4 will describe how energy can be reduced by operating at the minimum energy point via voltage scaling. Topics in this chapter include how the optimum voltage was selected as well as the implications of sub-threshold operation.

Chapter 5 will explain how parallelism in the baseband processor can be used to reduce the synchronization time in order to reduce the energy required to receive a packet. Modifications to the baseband algorithm based on the results of the optimum degree of parallelism will be described.

Chapter 6 will discuss issues pertaining to the implementation of the parallelized baseband processor. Specifically, it will address challenges such as leakage power, interconnect capacitance and peak current all of which are magnified for highly parallelized designs. Power saving techniques that can be used for the design of highly parallelized low energy systems such as clock gating and power gating will be presented. The issue of process variations will also be addressed as their effects become more pronounced at lower voltages. A description of the tools used for implementation will be provided in Appendix A. The floorplan and the bonding diagram of the chip is shown in Appendix B.

Finally, Chapter 7 will conclude with a summary of the work performed for this thesis as well as discuss possible extensions.

Chapter 2

Receiver Architecture

There are two competing methods of performing UWB communications: Multi-Band Orthogonal Frequency Division Multiplexing (MB-OFDM) UWB and Direct Sequence-UWB (DS-UWB) transmission. Figure 2-1(a) and Figure 2-1(b) show block diagrams of the two approaches [15, 16].

Although the OFDM approach uses a simpler and more direct approach to adapting for multipath effects[1] than DS-UWB, the major drawback of OFDM is that it requires a higher-complexity and higher-power transmitter than DS-UWB that must include an Inverse Fast Fourier Transform (IFFT) operation, a high-speed digital-to-analog conversion (DAC) and increased linearity in the radio frequency (RF) path [18]. An additional advantage of DS-UWB over MB-OFDM is that it is scalable based on the quality of the channel. For instance, in a DS-UWB, the number of bits in the ADC can be reduced if high SNR is attained [4, 19]. Consequently, for a low energy approach, the pulse-based DS-UWB system was investigated.

Numerous architectures have been proposed for DS-UWB receivers including fully

---

[1]Multipath is a typical characteristic of wireless communications channels. Multipath is due to the existence of reflectors and obstacles between the transmitter and receiver that cause the signal to travel via multiple paths of varying distances. When the signal arrives at the receiver, the energy is spread across various echoes with different amplitudes, phase and delays. In order to maximize the performance of the system, the receiver must be able to collect as much energy as possible from the echoes. Although UWB doesn't suffer from Rayleigh fading as in the case of narrowband communications [17], the challenge for the UWB receiver is to recover the energy that is distributed across closely spaced short duration echoes. In other words, the high resolution multipath requires a high resolution receiver.
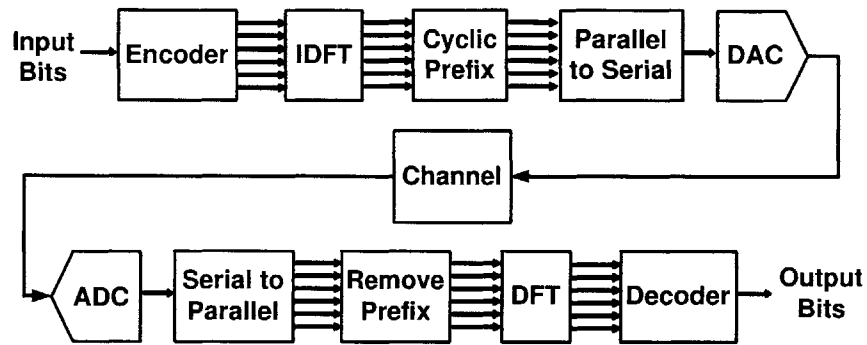
(a) OFDM Block Diagram.
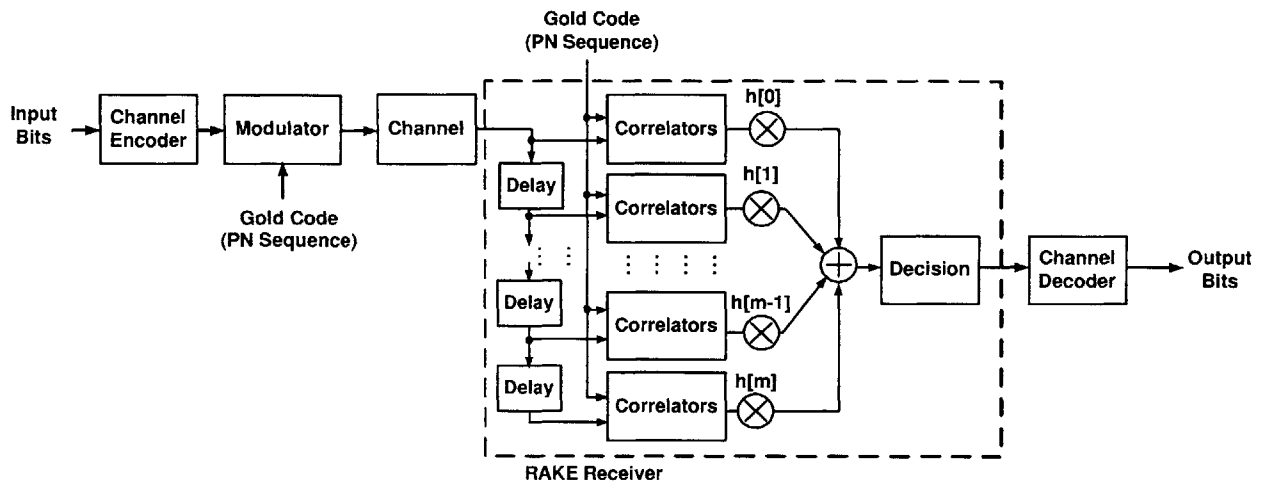
(b) Direct Sequence Block Diagram.

Figure 2-1: Two competing approaches for UWB communications.

14 Channel Frequency Plan
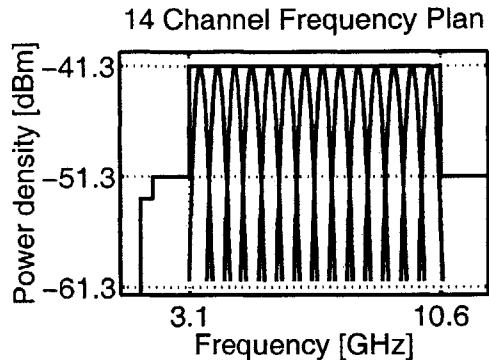
Figure 2-2: Channel Frequency Plan.

digital implementations with a channelized or sub-sampling analog-to-digital converter (ADC) as well as partially analog architectures [20]. A fully digital architecture using down conversion prior to the ADC was selected for this work. The fully digital implementation allows for scalability, and down conversion before ADC reduces the required sampling rate. A homodyne (i.e. direct conversion) architecture was selected for the front end rather than heterodyne since the latter would require a very sharp image rejection filter due to the large bandwidth of the signal [21].

2.1 Transceiver Specifications

The UWB packets are built from a sequence of binary phase-shift keying (BPSK) pulses with a 500-MHz bandwidth. The frequency plan of the system is shown in Figure 2-2. The transmitter generates packets comprised of approximate Gaussian pulses $p(t)$ and upconverts the packet to one of 14 channels in the 3.1-GHz to 10.6-GHz band. An upconverted pulse is shown in Figure 2-3.

The receiver, shown in Figure 2-4, uses a direct conversion architecture in the front-end and the in-phase and quadrature components are sampled at 500-MSPS by two 5-bit ADCs [3, 2]. For real-time demodulation of the UWB packet, the digital baseband processor must perform the signal processing with a throughput of 500 MSPS. Synchronization is performed entirely in the digital domain. Only the automatic gain control (AGC) is fed back to the analog domain so that the digital baseband is scalable. The focus of this thesis will be to design an energy efficient
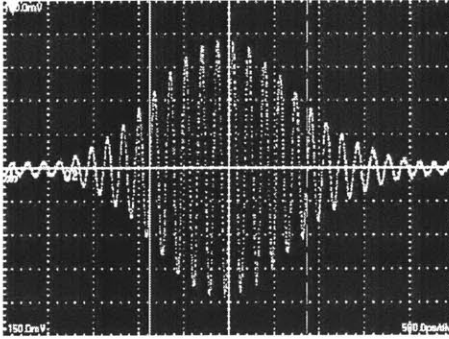
Figure 2-3: A 500-MHz approximate Gaussian pulse upconverted with a carrier of 5 GHz.
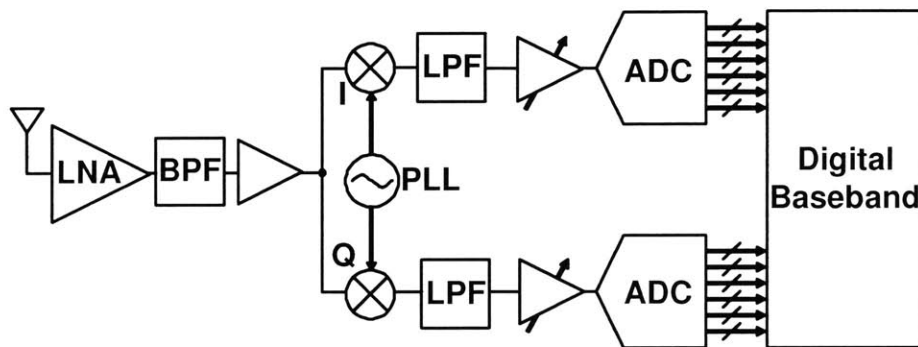
Figure 2-4: Block diagram of UWB receiver.

digital baseband processor that will operate within this system architecture.

Each UWB packet, shown in Figure 2-5, is divided into two sections: preamble and payload. The preamble is used by the baseband for synchronization and channel estimation. It contains multiple repetitions of a $N_c = 31$-bit Gold code sent at a pulse repetition frequency (PRF) of 25 MHz, or $T_{pre} = 40$ ns.

Gold codes are a type of pseudo noise (PN) sequence[2] that are generated from the modulo-2 addition of a preferred pair of maximum length sequences which are created using linear feedback shift-registers [15]. Gold codes have desirable periodic autocorrelation properties (i.e. they have a single narrow autocorrelation peak and low sidelobes) which are important for their use in synchronization[3]. The preamble

---

[2]A pseudo noise sequence is a code that exhibits random noise like properties (i.e. appears uniformly distributed and statistically independent). The applications of these sequences include multiple-access spread spectrum communication systems, ranging, synchronization, and data scrambling. This work will focus on the use of these codes for synchronization.

[3]Note that Gold codes also have desirable cross-correlation properties (better than an ordinary maximum length sequence); however, this not required for synchronization.
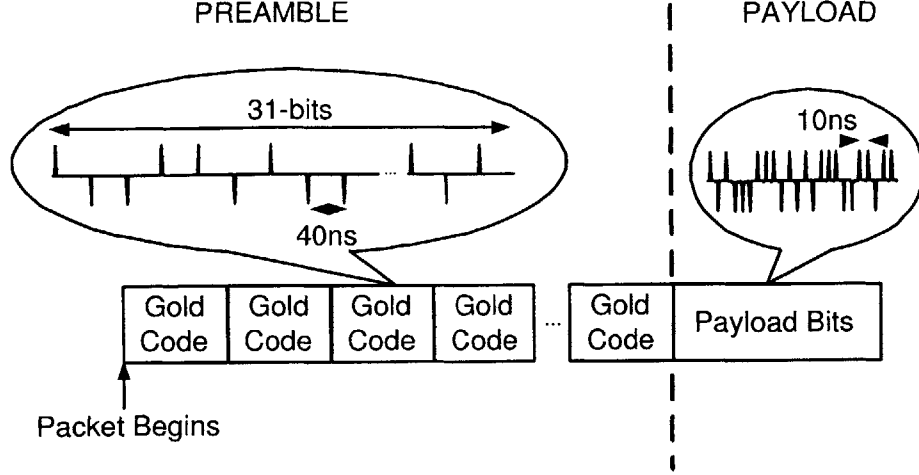
Figure 2-5: UWB packet format.

sequence can be mathematically described by the following equation:

$$s_{pre}(t) = A \sum_{j=0}^{R(M)-1} \sum_{i=0}^{N_c-1} r_j c_i p(t - jN_c T_{pre} - iT_{pre}) \qquad (2.1)$$

where A is the amplitude of the pulse, c_i is the binary Gold code sequence (+1 and -1) of length N_c, T_{pre} is the time between consecutive pulses, and $R(M)$ is the number of times the code is repeated. r_j is a binary sequence (+1 and -1) of length $R(M)$ that inverts the last repetition of the code. (e.g. if $R(M) = 4$, then $r_0 = 1$, $r_1 = 1$, $r_2 = 1$ and $r_3 = -1$)

The payload contains the actual data and is sent at a PRF of 100 MHz, or $T_{pay} = 10$ ns, for a 100-Mbps data rate with no channel coding. This payload sequence can be described by

$$s_{pay}(t) = A \sum_{k=0}^{N_{payload}-1} b_k p(t - kT_{pay}) \qquad (2.2)$$

where b_i is a binary sequence (+1 and -1) of data bits and T_{pay} is the time between two consecutive pulses in the payload. $N_{payload}$ is the number of bits transmitted in

23

the payload section of the packet. The entire packet can then be described as

$$s_{packet}(t) = s_{pre}(t) + s_{pay}(t - N_c R(M) T_{pre}) \qquad (2.3)$$

where the payload is delayed by $N_c R(M) T_{pre}$[4] since it occurs after the preamble. The purpose of the digital baseband processor will be to efficiently demodulate this UWB packet.

2.2 Power Consumption of Various Blocks in the UWB Receiver

The power breakdown of the receiver shown in Figure 2-4 is listed in Table 2.1. Note that the transmitter consumes 23 mW and the phase-locked loop (PLL) for the local oscillator, which is shared by the transmitter and receiver, has a power dissipation of approximately 34 mW [22]. This thesis will focus on reducing the energy of the UWB receiver, and therefore the transmitter and PLL power[5] will not be included in the calculations.

Block	Power Consumption
ADC	2×7.8 mW
Baseband Amplifiers (estimated)	~ 10 mW
Receiver Front End	54 mW

Table 2.1: Power dissipation for various blocks. Note that ADC and baseband amplifier power accounts for both the I and Q channels [2, 3].

---

[4]The time between the last pulse in the preamble and the first pulse of the payload is 40-ns.

[5]Note: In order for the system to know when to turn on and off, the PLL must always remain on even when the rest of the transceiver is powered off.

Chapter 3

Baseband Algorithm and Architecture

The digital baseband processor performs packet detection, acquisition, delay correction and channel estimation using the preamble, followed by demodulation of the payload. Additional repetitions in the preamble are required for AGC, but will not be included in the discussion.

One of the primary purposes of the baseband processor is to determine when to begin demodulation. This requires that the baseband and the incoming signal be synchronized. To achieve synchronization, the baseband processor must determine the delay of the incoming signal by computing the cross-correlation between the preamble of the packet and a locally stored template of the Gold code.

In addition, the baseband must correct for any non-idealities introduced by the channel and the transceiver. Degradation due to multipath effects in the wireless channel can be mitigated by estimating the channel, and weighting the echoes of the signal accordingly with a Rake receiver.

Non-idealities introduced by the transceiver include the mismatch of phase and frequency between local oscillators (LO) in the transmitter and the receiver. A phase offset will cause the received signal to be found on both the I and Q channels following direct conversion. Figure 3-1(a) shows the constellation with phase offset. A frequency offset will cause the constellation to rotate as shown in Figure 3-1(b). If

the clocks used by the pulse generator in the transmitter, and by the ADCs to sample the incoming signal at the receiver are independently generated, rather than being derived from the local oscillator, an additional form of non-ideality is introduced. The baseband algorithm designed in this thesis will correct for the phase offset between the local oscillators using a matched filter, and the mismatch between the pulse generator clock and ADC sampling clock using a delay locked loop with early/late tracking. It will not correct for frequency offsets between the local oscillators.

The first section will begin with a description of the baseband algorithm. It will discuss the original algorithm that was previously developed in [12, 13, 23], and the modifications that were made to achieve the algorithm that was implemented for this thesis. The second section will discuss how the baseband algorithm is mapped to a parallel architecture. The third section will provide a detailed discussion on the architecture of each block used to implement the baseband algorithm. The final section in this chapter will discuss the prototyping platform that was used to perform the verification of the algorithm.

3.1   Baseband Algorithm

The original algorithm presented in [23] contained four states of operation. Figure 3-2 outlines these four states of operation with respect to the packet. Figure 3-3 shows the state diagram of this algorithm.

In State 0, the acquisition phase, the baseband detects the presence of a packet and provides an initial estimate of its delay. This is accomplished by performing a correlation of the input with an unknown delay against a 31-bit Gold code. Each correlation takes place over $T_{code} = N_c \times T_{pre} = 1240$ ns. The delay must be resolved up to 2-ns accuracy; therefore, there are a total of 620 possible delays and corresponding correlations: 20 to match the pulse position[1], and 31 to match the Gold code. Until acquisition is achieved, the baseband remains in State 0 and performs these correlations. The threshold detector determines when a correlation exceeds a predefined

---

[1]recall that the duty cycle in the preamble is 5%
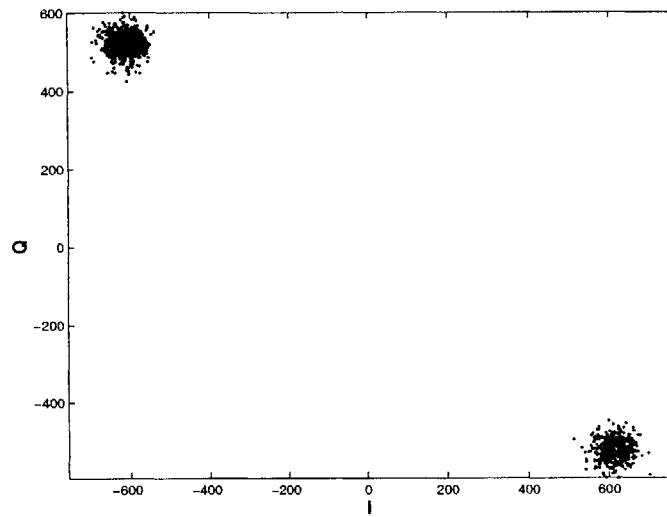
(a) With phase offset of -39.1 deg.
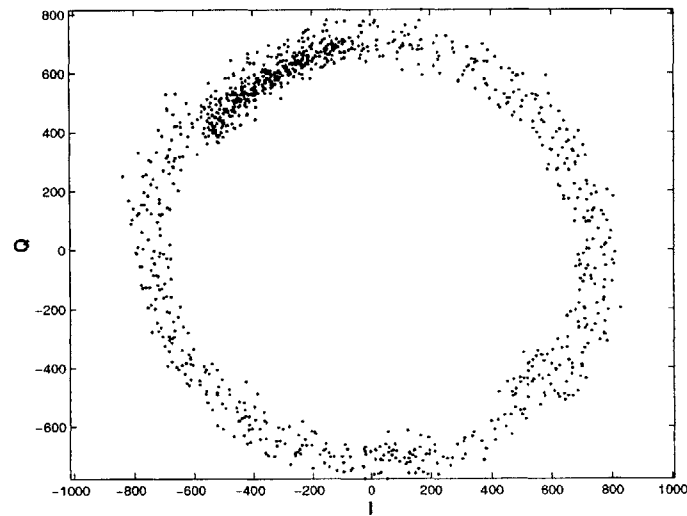
(b) With frequency offset of 10 ppm for a center frequency of 3.5-GHz over a 30.2 μs time period.

Figure 3-1: Constellation of signal with 20 dB SNR.
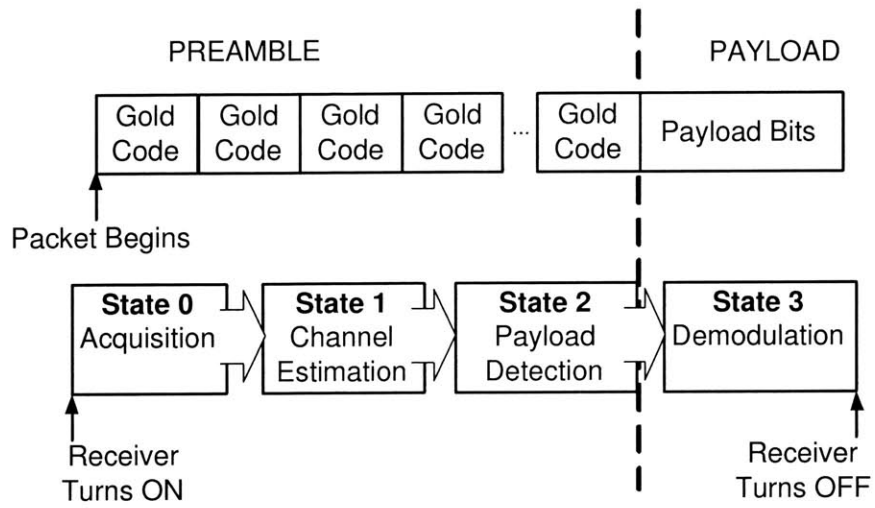
Figure 3-2: UWB packet with respect to states in baseband algorithm.
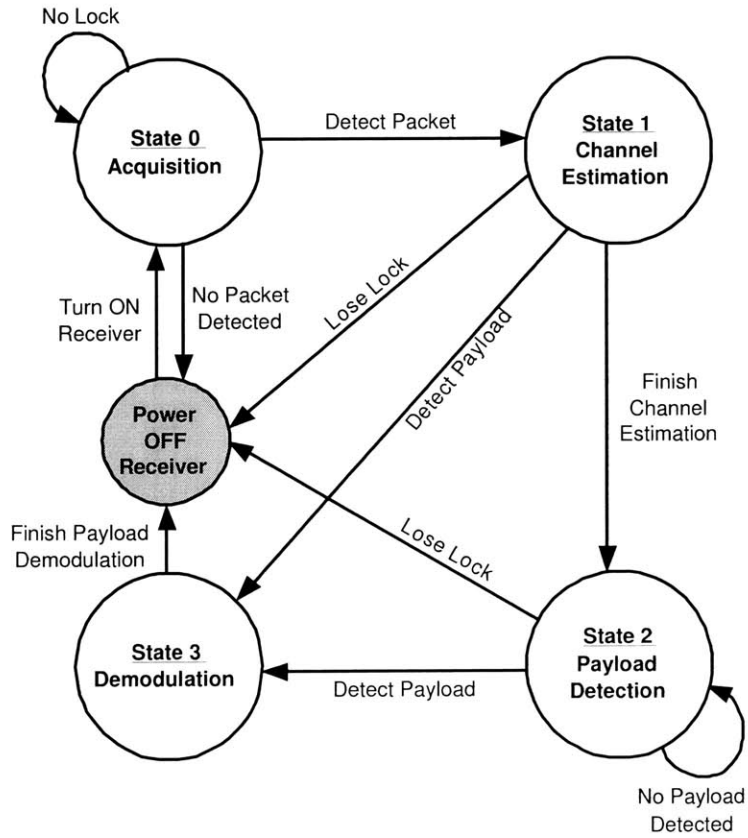
Figure 3-3: State diagram of original algorithm.
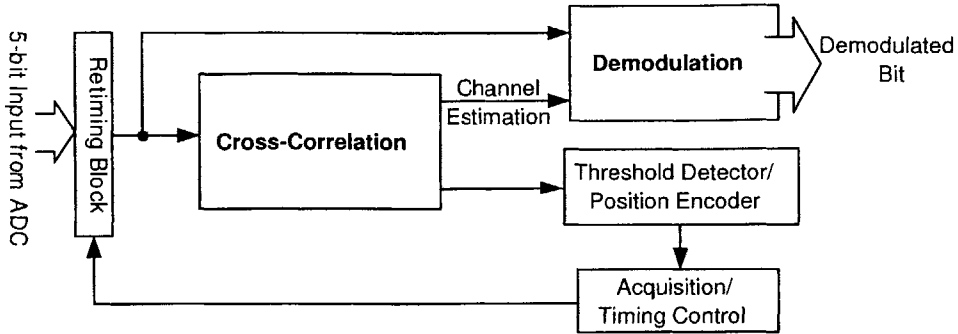
Figure 3-4: Original UWB digital baseband.

threshold at which point acquisition is declared (i.e. lock is detected). The position encoder determines the delay and the baseband retimes the input so that it is aligned and synchronized before moving on to State 1. If all 620 delays are checked and the baseband does not detect lock, the UWB receiver turns off.

In State 1, the channel estimation phase, the baseband acquires the channel estimates from the maximum of the cross-correlation function and its four adjacent neighbours (two on either side). This provides a least square error estimate of the multipath in the channel by averaging over the 31-bit Gold code. This must be done before demodulation in order to compensate for the detrimental effects in the UWB channel [25]. The channel estimates are used to construct a five tap finite impulse response (FIR) matched filter that takes both the pulse shape and channel impulse response into account.

In State 2, the payload detection phase, the baseband waits for the end of the preamble which is indicated by an inverted replication of the Gold code. During State 1 and 2, the baseband continuously performs correlations to check that the baseband remains locked. If a threshold is not met, the packet is assumed to be lost or to have been a false packet lock, and the baseband and the rest of the UWB receiver turns off. In addition, the baseband performs delay correction with the use of a delay locked loop which is part of the retiming block. Finally, in State 3, the demodulation phase, each pulse of the payload is filtered by the matched filter derived from the channel estimates and then passed through a decoder that resolves the bit. A block diagram of the original baseband is shown in Figure 3-4.
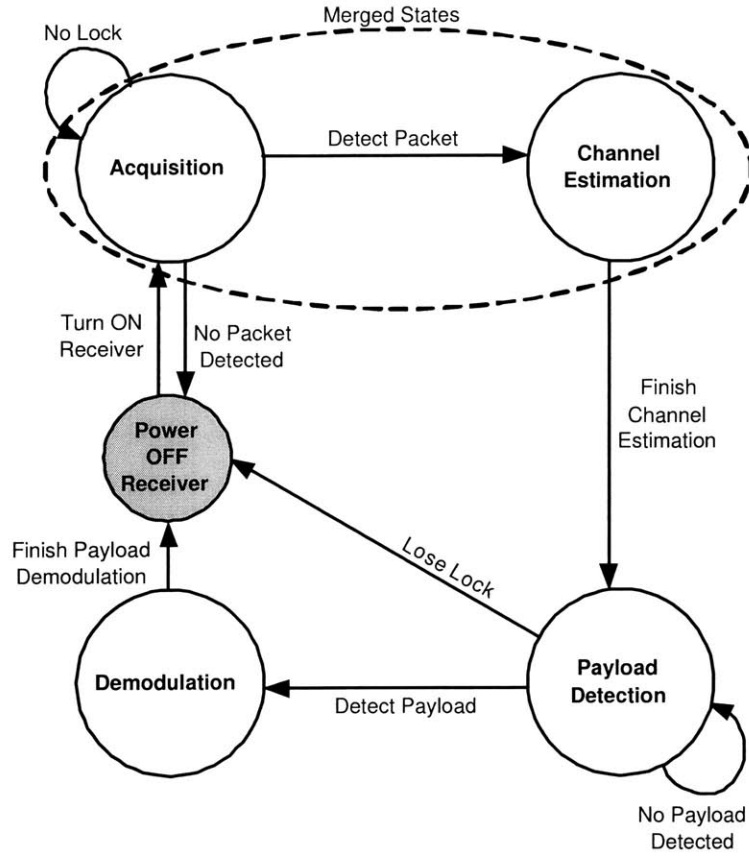
Figure 3-5: State diagram of modified algorithm.

The original algorithm presented in [12, 13, 23] underwent several modifications. First, the Viterbi-based maximum likelihood sequence estimator (MLSE) equalizer for intersymbol interference (ISI) and the Costas Loop for frequency offset correction were removed[2].

Second, the separate channel estimation state was removed. Rather than obtaining the channel estimation after retiming the baseband processor, the channel estimation was performed immediately after acquisition. This requires the use of an additional multiplexer but reduces the on-time of the entire receiver by one T_{code} unit. Figure 3-5 shows the state digram of this modified algorithm.

---

[2]These blocks were unnecessary for the purpose of showing parallelism as a method for achieving low energy. Therefore, the baseband algorithm designed in this thesis will not address issues such as ISI and frequency offset. It will however address phase offset in the local oscillators and offset between the ADC and pulse generator clocks.

3.2 Mapping Algorithm to Parallel Architecture

The computation of the cross-correlation can be modeled using a matched filter. The length of the Gold code, the sampling rate and the pulse repetition frequency of the preamble dictate that a 620-tap matched filter is required. The drawback of actually implementing a matched filter however is that it would have to operate at an energy-inefficient clock frequency of 500 MHz due to the 500 MSPS sampling rate of the ADC. It would be desirable to map the cross-correlation operation to a parallelized architecture in order to be able to reduce the clock frequency. Figure 3-6 illustrates how a correlator bank architecture that utilizes a parallel architecture can be derived from a matched filter [37]. Parameter L in the figure denotes the spacing of non-zero coefficients in the matched filter. For the system presented in the thesis, $L = T_{pre}/T_{sample} = 20$, since the pulses in the preamble are spaced by $T_{pre} = 40$ ns and the signal is sampled at every $T_{sample} = 2$ ns. The correlator bank implementation can achieve the same performance of a matched filter[3] with the flexibility of adjusting the frequency to reduce energy. The energy efficient baseband utilizes this parallelized implementation.

Figure 3-7 illustrates the block diagram of the baseband processor including the parallelized correlator bank architecture. The energy efficient baseband exploits two forms of parallelism, N and M, to reduce the energy required to compute each point in the cross-correlation function and the time required to compute the entire function respectively. Specifically, N defines the degree of parallelism required to operate the digital baseband at the frequency of the minimum energy point[4] while maintaining a throughput of 500 MSPS. M is defined as the number of Gold code correlations performed simultaneously. Each sub-bank, composed of N correlators, checks for one Gold code delay. The trade-offs involved in the specification of N and M will be discussed in Chapter 4 and Chapter 5 respectively.

The cross-correlation of between the Gold code and the input (i.e. incoming signal)

---

[3]Note that this matched filter uses rectangular pulses rather than perfect replicas that match the pulse shape.

[4]The minimum energy point will be discussed in Chapter 4

31
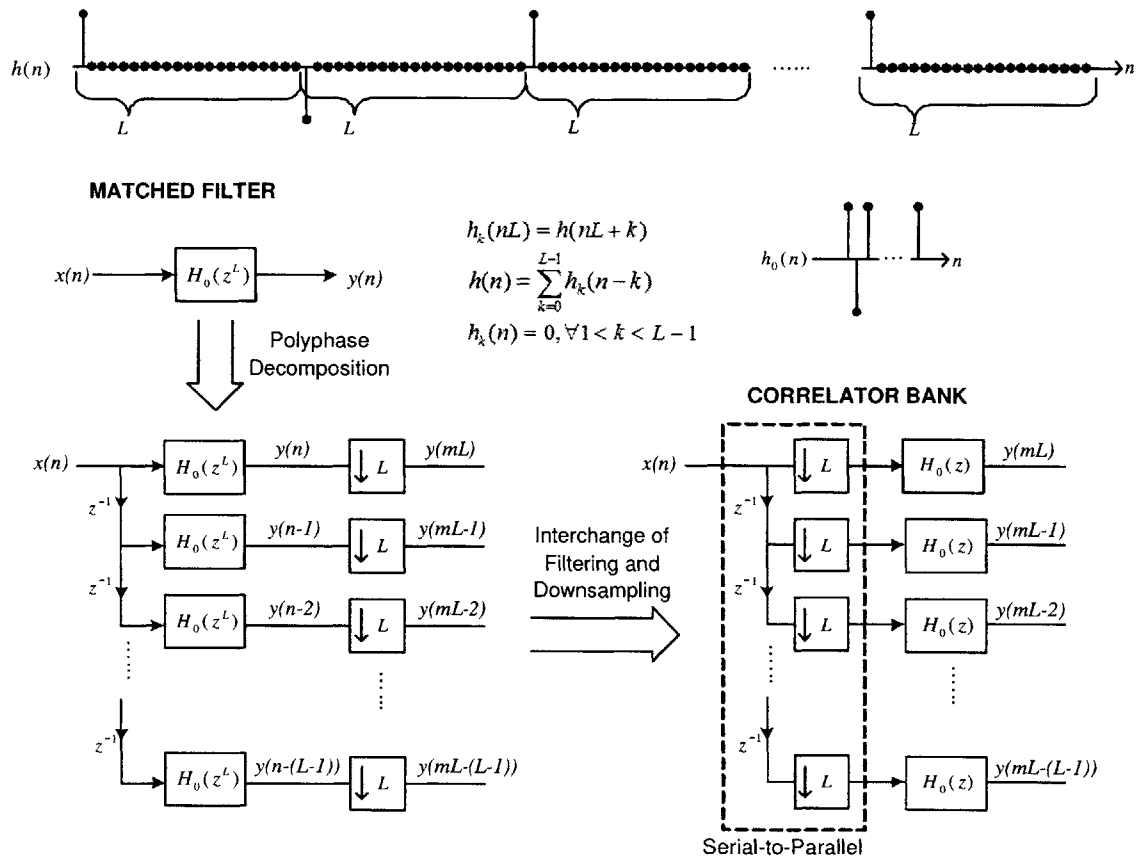
Figure 3-6: Mapping of the matched filter to a parallelized correlator architecture.
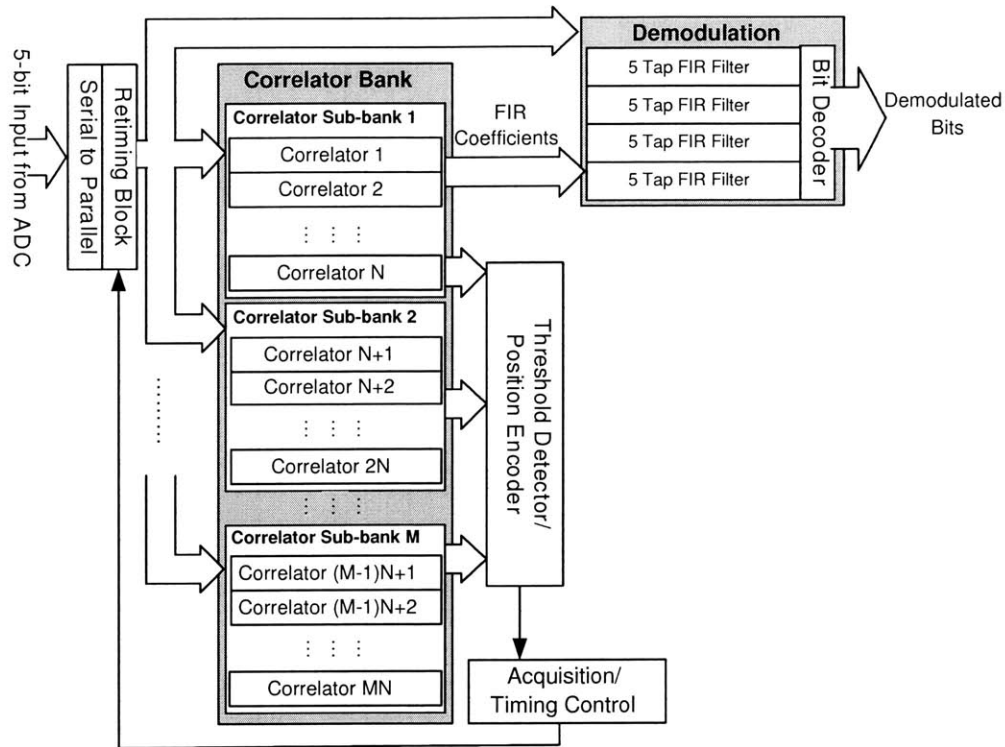
Figure 3-7: Energy efficient UWB digital baseband.

can be computed by a parallelized correlator bank in two ways: either the same input goes to all sub-banks and each sub-bank contains a template of the Gold code with a different delay, or all sub-bank contain the same Gold Code template and each sub-bank receives the input with a different delay. In the energy efficient baseband, the correlator bank was parallelized such that the same Gold Code is compared to delayed versions of the input. This is mathematically equivalent to the previously discussed 620-tap matched filter and therefore has equivalent performance[5].

Another advantage of delaying the input rather than delaying the Gold code is that each correlator sub-bank produces the outputs at different clock cycles rather than all sub-banks producing the outputs at once. Figure 3-8 illustrates the difference between the two implementations. In Figure 3-8(a), if the Gold code is delayed rather than the input, the $M = 8$ correlator sub-banks produce outputs at the same clock cycle, while in Figure 3-8(b), if the input is delayed rather than the Gold code, the

---

[5]In the implementation, the matched filter will use rectangular pulses rather than perfect replicas that match the pulse shape. For Gaussian pulse, 1.7 dB of SNR is lost compared to the ideal matched filter[13].
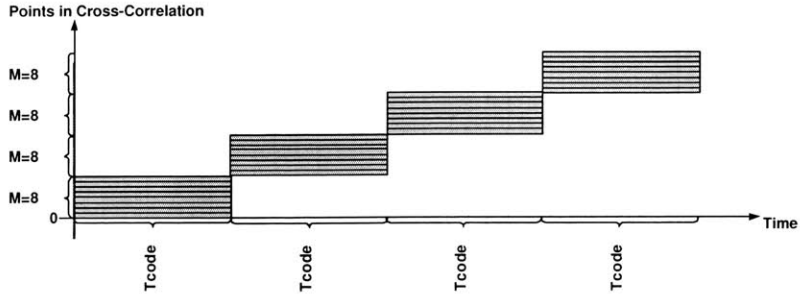
(a) Delay Gold code in each sub-bank
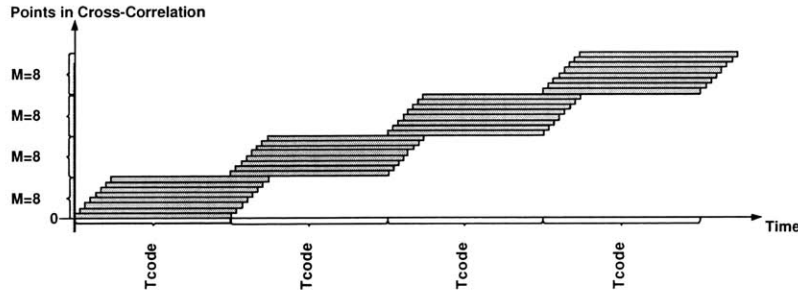
(b) Delay input to each sub-bank

Figure 3-8: Representation of time to compute cross-correlation points for the two possible methods of parallelizing the correlator bank. Each grey bar represents the time to check one Gold code shift.

$M = 8$ correlator sub-banks produce outputs at staggered clock cycles. If each sub-bank of correlators produce outputs at different times, they can share the threshold detector and position encoder. Therefore, the size of the threshold detector and position encoder do not scale with M.

3.3 Block Architecture

This section will provide a description of the architecture of each block in the baseband processor and how the blocks interact with one another. The design of the architecture is critical to ensuring a low energy implementation.

The baseband processor can be broken down into three main components: Synchronization Block, Demodulation Block and Control/Retiming Block. The synchronization block contains a correlator bank, the threshold detector and the position encoder. The demodulation block contains a bank of FIR filters followed by a bit de-

coder. The control/retiming block contains a serial-to-parallel block, retiming logic, and a finite state machine for the control. Peripheral test logic is also included in the design and will be briefly discussed.

3.3.1 Synchronization

Correlators and Correlator Bank

As previously discussed, the cross-correlation function is used to detect the presence of the packet and estimate the delay of the incoming signal. Let $x[n]$ and $y[n]$ be the input and output of the correlators, and $h[n]$ be the Gold code sequence.

$$\begin{aligned} y[n] &= x[n] * h[-n] \\ &= \sum_{k=-\infty}^{\infty} x[k] \times h[k-n] \end{aligned} \tag{3.1}$$

This cross-correlation function is computed using a bank of correlators. Each correlator, shown in Figure 3-9, computes one point of the cross-correlation function at a time. The correlator is comprised of two adders and two multiplexers. The two adders perform sign inversion (2's complement) and accumulation, while the two multiplexers are used to perform multiplication with Gold Code[6], and to clear the correlator after it has computed one cross-correlation point so that it can be used to compute the next point[7]. When the correlator bank is parallelized by M, M times more points of the cross-correlation function are computed approximately every T_{code}. This continues until all points of the cross-correlation function are generated. The trade-off of time to complete the cross-correlation versus parallelism M is shown in Figure 3-10. Notice that as M increases, the time it takes to finish computing all the points in the cross-correlation function reduces.

---

[6]This is possible since the Gold code only takes on values of +1 and -1

[7]The length of the Gold code is 31-bits. Therefore, at most, the correlator accumulates for 31 cycles.
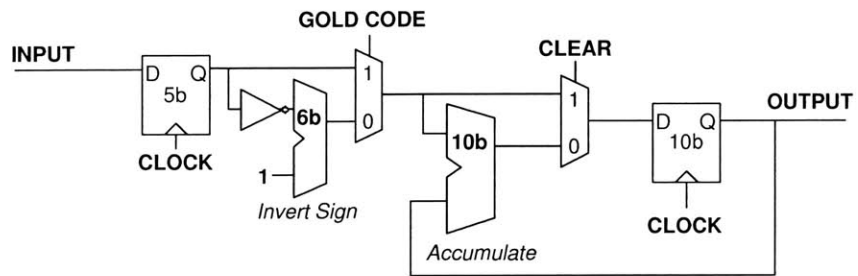
Figure 3-9: Correlator in digital baseband.
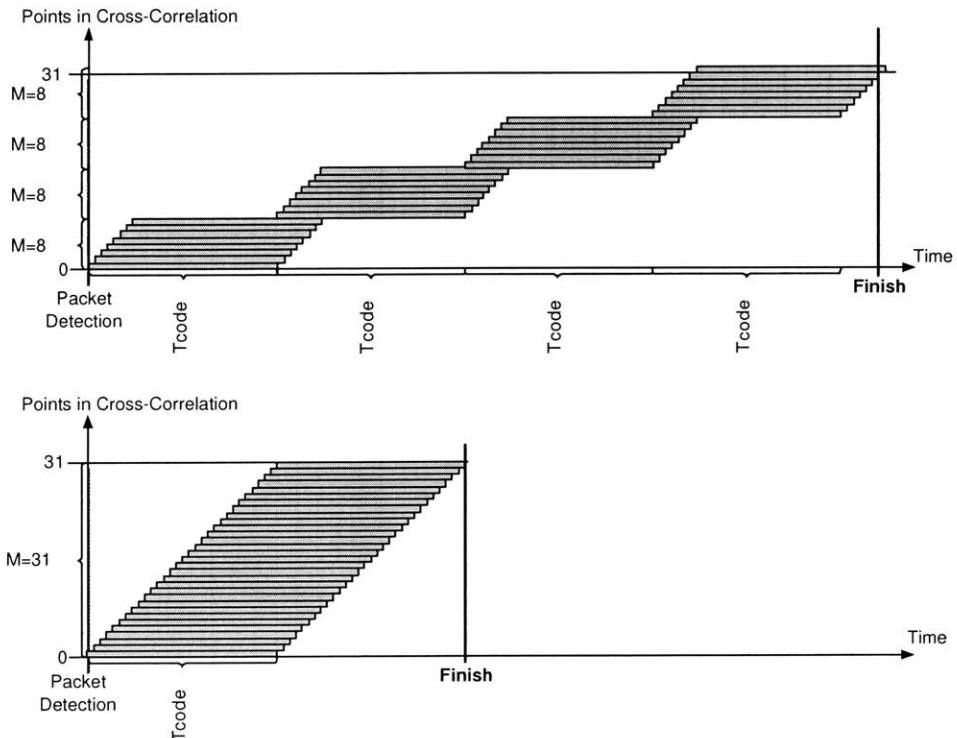
Figure 3-10: Time to complete cross-correlation function for M=8 and M=31.

Threshold Detector and Position Encoder

The purpose of the threshold detector is to determine when the cross-correlation function provided by the correlator bank exceeds a pre-defined threshold. When this occurs, lock is achieved. The position encoder then measures the delay of the packet with respect to the baseband and provides this information to the retiming block in order to synchronize the baseband.

As a result of the direct conversion in the front end, the input signal to the baseband processor is complex with I (real) and Q (imaginary) components. Therefore, a complex cross-correlation is required. The I and Q correlations can be computed by two separate real correlator banks. However, in order to perform the threshold comparison with the complex cross-correlation, I and Q components are combined by taking the sum of the absolute value of the real and imaginary parts of the cross-correlation function.

For a Minkowski distance of order p=1 (i.e. taxicab norm or Manhattan distance [26, 27, 28]), if $x = a + jb$, then $||x|| = |a| + |b|$. Let $x[n]$ be the complex input into the correlator, $h[n]$ the Gold code sequence, and $g[n]$ the input to the comparator in the threshold detector. Accordingly,

$$x[n] = x_I[n] + jx_Q[n] \tag{3.2}$$

$$g[n] = x_I[n] * h[-n] + jx_Q[n] * h[-n] \tag{3.3}$$

$$||g[n]|| = |x_I[n] * h[-n]| + |x_Q[n] * h[-n]| \tag{3.4}$$

A block diagram of the threshold detector can be seen in Figure 3-11. Note that in the implementation, the threshold detector is pipelined such that the absolute values and the output of the comparator are registered. This ensures that the threshold detector is not a critical path in the processor. However, increased pipelining increases the latency of the threshold detector, and therefore decreases the maximum number of accumulation cycles that the correlator can perform. In this system, there is one cycle

37
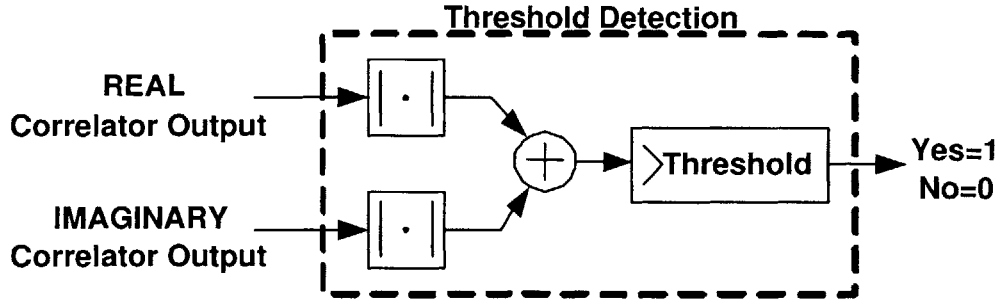
Figure 3-11: Block diagram of threshold detector

of latency through the output of the correlator[8], two cycles of latency through the threshold detector followed by another cycle of latency in the position encoder. The retiming and control circuitry, which will be discussed in the next section, required an additional two cycles. Therefore, the number of accumulation cycles is reduced from 31 to 25, which lowers the processing gain. However, MATLAB simulations and verification on the prototyping platform showed that this reduction can be tolerated.

For simplicity of design, the threshold comparisons are done in chronological order (i.e. threshold detection will detect lock in earlier positions than later ones). Therefore, the threshold detector will select the first correlation output that exceeds the threshold and not necessarily the maximum position. However, the threshold can be selected such that there is a high probability that only the maximum of the cross-correlation function will be detected[9]. The low sidelobes and narrow peak of the autocorrelation of the Gold code are crucial for this to be possible.

Following the threshold detection, the position encoder must generate control bits for the retiming block to synchronize the baseband to the input. In addition, the correlator outputs surrounding the maximum of the cross-correlation function must be selected as the coefficients for the FIR filter using a multiplexer (Figure 3-12).

---

[8]The output of the correlator is registered so that it doesn't toggle every cycle and only toggles when output is ready (i.e. once every 31 cycles). This was done to prevent the output nets of all the correlators from switching at the same time and at every clock cycle.

[9]The threshold can be determined from the Receiver Operating Characteristic using the Neyman-Pearson test to achieve a specific probability of detection (P_d) and probability of false alarm (P_{fa}) [29].
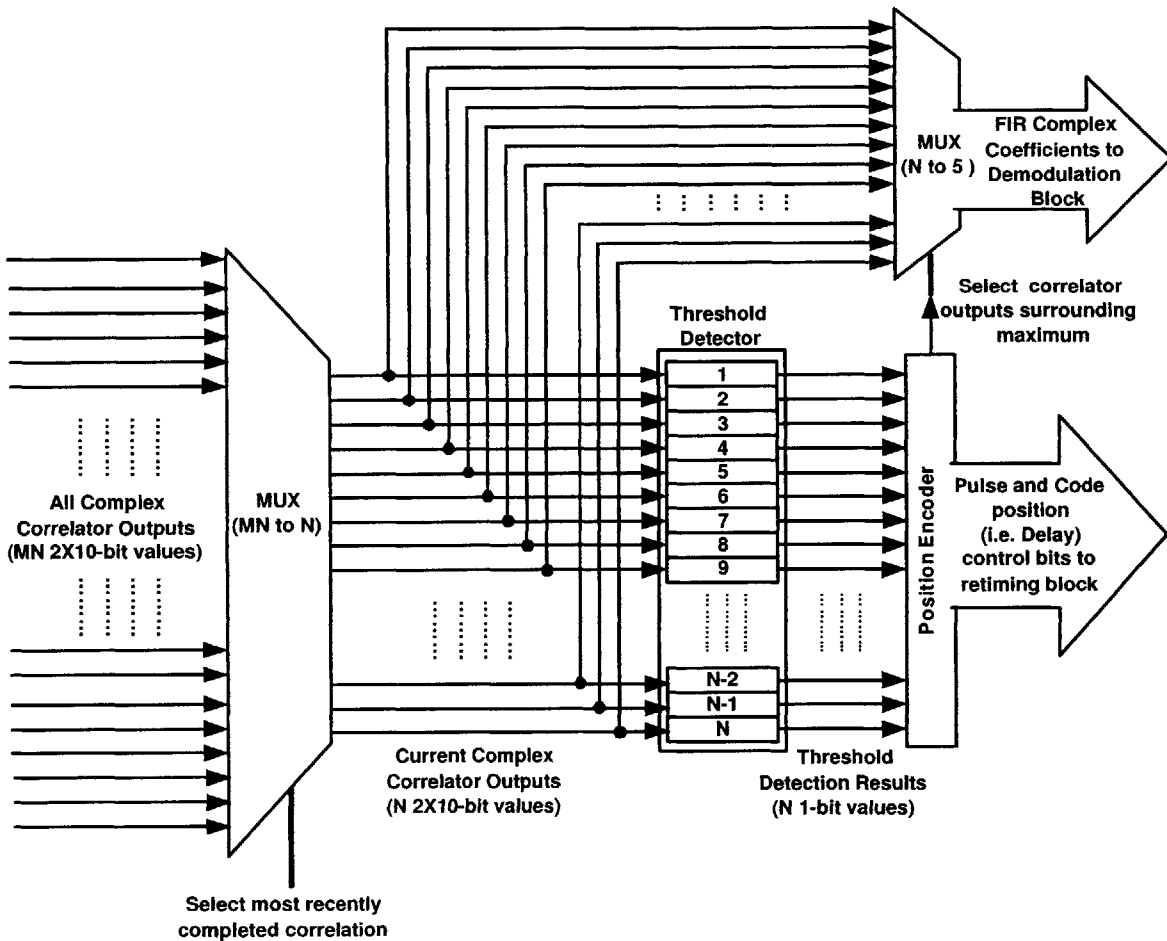
Figure 3-12: Data flow between correlator bank, threshold detector, position encoder and demodulation block.

3.3.2 Demodulation

FIR Filters and Bit Decoder

For demodulation, the retimed input signal must be filtered by a matched filter to correct for the multipath effects in the UWB channel. Note that both the input signal and the matched filter coefficients are complex. The matched filter also removes any constant phase offset between the transmitter and receiver local oscillators. Following the correction of phase, the signal should only have a real component. Thus the matched filter only needs to compute the real part of the product between the complex input signal and complex coefficients. The real and imaginary parts of the input signal can be multiplied with the real and imaginary parts of the coefficients, and the products can be summed to form the real component of the complex product. Let $b[n]$ be the complex input to the matched filter and $c(k)$ the channel estimation provided by the correlator bank. The complex conjugate of the channel estimates are the coefficients for the matched filter. $c^*(k)$ has a length of five for a 5-tap FIR filter.

$$b[n] \quad = \quad b_I[n] + jb_Q[n] \tag{3.5}$$

$$c^*(k) \quad = \quad c_I(k) - jc_Q(k) \tag{3.6}$$

$$Real(b[n] \times c^*(k)) \quad = \quad b_I[n] \times c_I(k) + b_Q[n] \times c_Q(k) \tag{3.7}$$

Figure 3-13 shows this matched filter operation. The demodulation block is also somewhat parallelized by having a bank of FIR filters. This allows the demodulation block to process the parallelized input at the same clock frequency as the synchronization block. Finally, the bit decoder looks at the sign bit of the output of the FIR matched filters to resolve the bit.

3.3.3 Retiming and Control

The retiming block consists of two sub-blocks: a serial-to-parallel (S2P) block and the retiming logic. The S2P was needed since the chip was pad limited and it was
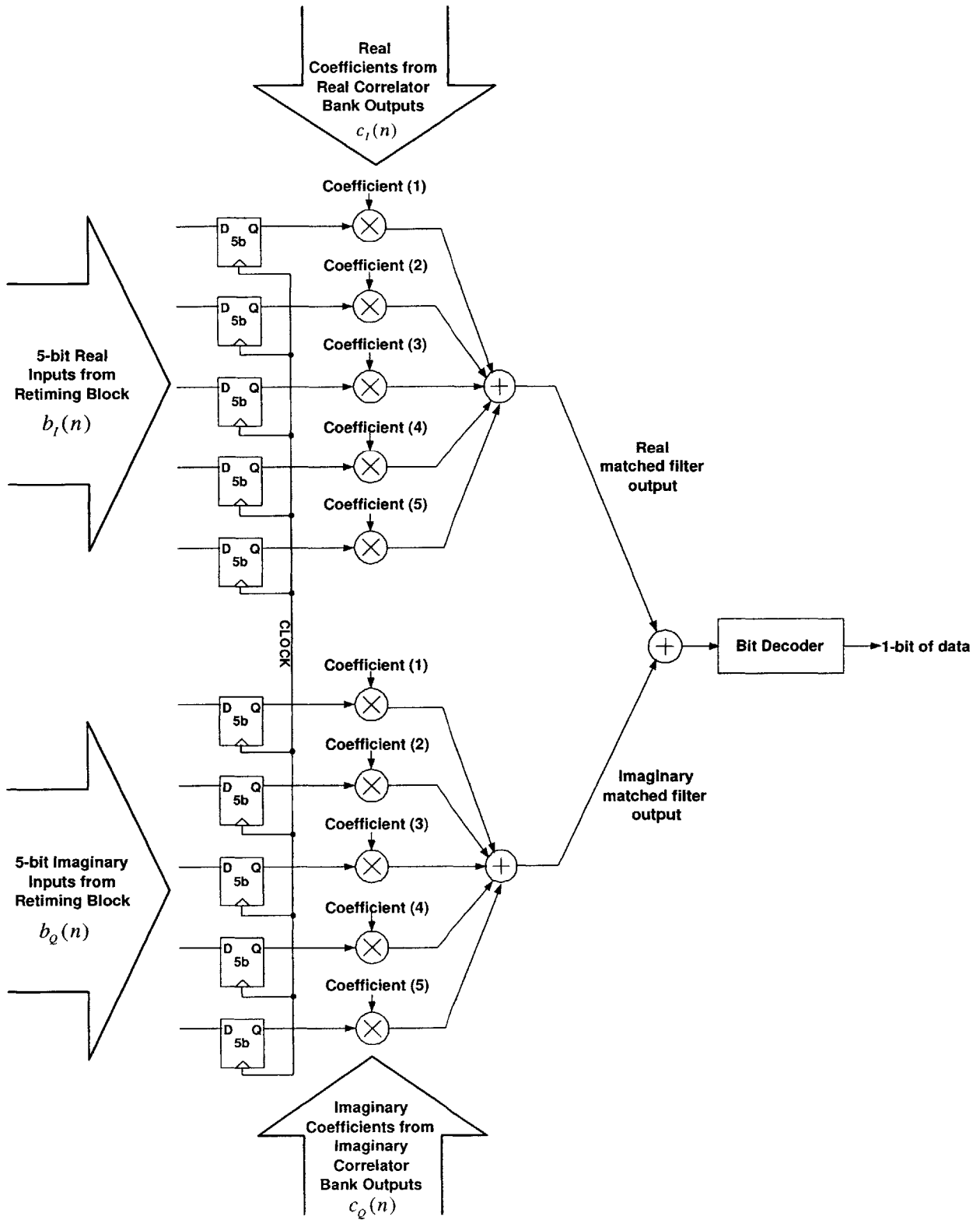
Figure 3-13: Block diagram of demodulation block

not possible to have all N complex inputs enter the chip at the same time. Instead, the inputs enter semi-serially and are parallelized by the S2P. The retiming logic is used immediately after acquisition, when the position of the pulse is determined by the threshold detector and position encoder. Based on the information provided by the position encoder, the retiming logic adjusts the incoming signal such that the maximum of the pulse lands in a specific position at the output of the retiming block. A DLL involving early/late tracking was also implemented to account for drifts in the clocks and can also control the retiming logic.

The control logic determines when each block is reset, enabled, or disabled depending on the state of the system. In addition, the control logic determines when each correlator in the bank is cleared and when each correlator output is registered. The control logic was implemented as a finite state machine where certain parameters could be set prior to entering the acquisition state, when the rest of the baseband processor is in reset. These parameters include the number of bits in the payload and the maximum number of correlations before shut down. In addition to these control logic parameters, other parameters that could be programmed include threshold, Gold code, and filter coefficients of the loop filter in the DLL. All these programmable bits are read in serially when the baseband is in reset.

The retiming logic is placed after the S2P. As a result, it operates at the same clock frequency as the rest of the system and the control block only needs to operate in one clock domain. The retiming logic also consumes less energy when it is placed after the S2P, since it operates at a lower frequency.

The control logic for the correlator bank enables each sub-bank at different clock cycles. This was done in order to have delayed inputs rather than delayed Gold codes. Accordingly, each sub-bank was also cleared[10] at different clock cycles.

---

[10]The correlator has to be cleared every 31 cycles of accumulation so that it can be used to compute the next cross-correlation point.

Test Mode	Description	Code
0	Normal operation	000
1	S2P: Feed in known input, look at parallel outputs (positions 1, 2 and 3)	001
2	DLL: Feed known input and look at DLL output	010
3	Correlators: Feed known input into correlator 3	011
4	Retiming Logic, Threshold Detector and Position Encoder: Feed known input to correlator 7, check that Position Encoder output	100
5	Channel Estimation: Feed known input on correlator 3, stream out Channel Estimation after acquisition	101
6	Demodulation (Matched Filter): Feed known input to correlator 3, look at 10 MSB of matched filter output	110
7	Look at known input	111

Table 3.1: Available test modes

3.3.4   Testing

Peripheral testing circuitry was added to enable testing of various blocks in the processor. This included a block that generated known input values, a replica correlator, and a multiplexer at the output that selects between internal values and output values of the processor. A test plan consisting of 8 modes was implemented. These modes are listed in Table 3.1. Three input I/O pads are dedicated to the test mode setting.

3.4   Verification on Prototyping Platform

3.4.1   Prototyping Platform

A modular prototyping platform for pulsed-UWB was developed to allow for rapid prototyping and performance characterization [30]. Using this platform, different blocks of the UWB transceiver could be tested within a complete system environment. The platform contained prototypes of future chip implementations built from discrete components and several FPGAs. The main advantage of this platform was that it offered high programmability and flexibly that could not be achieved from a chipset implementation. The platform was used to test and debug the baseband algorithm.
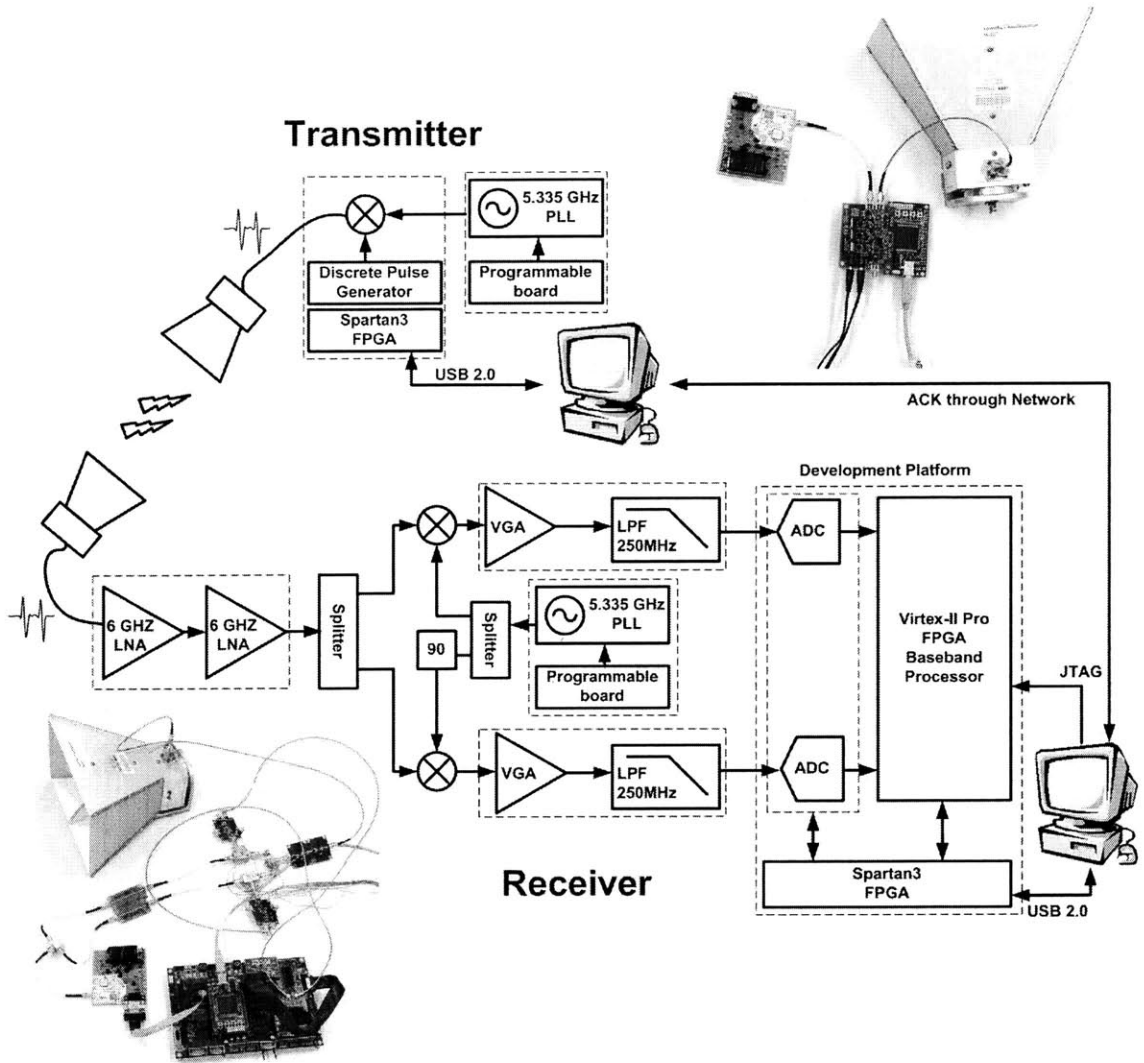
Figure 3-14: Prototyping platform setup.

Figure 3-14 shows the setup of the prototyping platform. The ADCs and the baseband processor were integrated onto the same board, which will be referred to as the development platform, while the RF front end of the transmitter and receiver occupied several separate PCBs.

The Verilog code for the baseband algorithm was loaded onto a Virtex-II Pro (VP30-6) FPGA using a JTAG connection from the PC. A control FPGA called Spartan3 (XEM3001 provided by OpalKelly) was used to interface between the development platform and a PC via USB 2.0. Therefore, all control and data signals to and from the PC travelled through the Spartan3.

The development platform contained a dual channel 8-bit ADC from ATMEL that operated at 500 MSPS to sample the inputs from the RF front end. However, the baseband processor utilized only the 4 most significant bits in order to match the performance of the chip to be designed as well as to reduce the size of the baseband code so that it would fit on the FPGA[11].

Finally, since there was only one prototyping platform, only a one-way UWB communication channel could be established. In order to complete the loop, an ACK was sent through the network from the receiver to the transmitter when the packet was correctly received[12].

3.4.2   Algorithm Verification

The baseband algorithm described in the previous section was tested on a prototyping platform with $N = 20$ and $M = 1$. This verification was necessary as several blocks of the previous implementation [31, 13] had been removed or modified. The blocks that were removed included the Viterbi-based maximum likelihood sequence estimator (MLSE) equalizer and the Costas Loop. The programmable Rake receiver [31] was replaced with a fixed 5-tap matched filter. These blocks were removed as they were not required in the illustration of how parallelism can reduce energy. It was important to verify that although the robustness of the baseband was reduced, it would still be able to operate in regions with low-to-medium multipath effects. Testing the algorithm on the FPGA of the prototyping platform provided this verification assurance.

In the prototyping platform, the local oscillators and the FPGAs/ADC ran off different clocks. Therefore, there can be an additional source of delay offset during sampling. This was corrected using a DLL in the baseband algorithm. The two PLLs used for the local oscillators of the receiver and transmitter had a very low frequency deviation of less than 2 ppm [32]; therefore, the baseband processor could operate without the Costas Loop[13]. For this implementation, the packet had a payload size of

---

[11]Note: In the end, 5-bit input was actually used for the chip design.

[12]Before sending the ACK, a CRC check was done on the payload bits to ensure that they were correctly demodulated.

[13]A Costas Loop is a PLL for BPSK modulation that is used to correct for frequency offset.

approximately 16-kbits and 33 repetitions (~ 41 μs) of the Gold code in the preamble to ensure lock.

Chapter 4

Reducing energy per operation

The input from the ADC arrives at a rate of 500 MSPS. As a result, a serial implementation of the baseband processor would have to run at a frequency of 500 MHz if the input is to be processed in real time. In order that the critical paths, through the correlator in the synchronization block and through the matched filter in the demodulation block, meet the timing constraints, the digital circuitry must run at its maximum supply voltage. However, running at the maximum voltage is not energy-efficient. It is important to reduce the energy of the correlator (Figure 3-9) since it consumes the largest portion of energy in the baseband during synchronization. This chapter will discuss how the energy per operation in the baseband processor can be reduced by lowering the supply voltage and parallelizing by a factor of N to compensate for the reduction in speed. Using parallelism to increase throughput to allow for voltage scaling is an idea that was presented in [33]. In this work, the voltage will be scaled such that the correlator operates near its minimum energy point.

4.1   Minimum Energy Point

The energy per operation can be reduced by lowering the supply voltage (V_{DD}) [10]. At maximum V_{DD}, the transistors in the circuit operate in the saturation region. If V_{DD} is lowered below the threshold voltage (V_T) of the device, the circuit is said to be operating in the sub-threshold region. Lowering V_{DD} increases the latency per

operation (T_{period}) linearly in the saturation region, and exponentially in the sub-threshold region. This increases the leakage energy as it is linearly related to T_{period}. There is a minimum operating energy point since the dynamic energy and the leakage energy scale in an opposite manner with V_{DD} [11].

The energy per operation of the correlator can be described by the following set of equations [34]:

$$E_{total} = E_{dynamic} + E_{leakage} \tag{4.1}$$

The dynamic energy ($E_{dynamic}$), also known as the switching energy, is defined as the energy required for low-to-high transitions. This energy can be described by the equation

$$E_{dynamic} = C_{eff}V_{DD}^2 \tag{4.2}$$

where C_{eff} is the effective capacitance being switched and represents the average capacitance switched at every clock cycle.

The leakage energy ($E_{leakage}$) is defined as the energy dissipated when the system is idle (in the absence of any switching activity). This energy can be described by the equation

$$E_{leakage} = T_{period}V_{DD}I_{leak} \tag{4.3}$$

where I_{leak} is the leakage current that flows between the supplies through reverse-biased diode junctions in the transistors. I_{leak} is primarily due to the sub-threshold leakage current that occurs between the source and drain and which is defined as

$$I_{leak} = I_o e^{-V_T/nV_{th}} \tag{4.4}$$

where I_o is from the leakage current model, V_T is the threshold voltage, n is the

48

sub-threshold slope factor[1], and V_{th} is kT/q.

The latency T_{period} can be described by the following gate delay model [36, 11]:

$$T_{period} \propto \begin{cases} \frac{V_{DD}[1+(V_{DD}-V_T)/(E_{sat}L)]}{(V_{DD}-V_T)^2} & \text{saturation region} \\ \frac{V_{DD}}{\left(e^{\frac{(V_{DD}-V_T)}{nV_{th}}}\right)} & \text{sub-threshold region} \end{cases}$$

where E_{sat} accounts for velocity saturation, and L is the effective channel length. Therefore, the energy per operation of the correlator can be modeled by

$$E_{total} = E_{dynamic} + E_{leakage} \tag{4.5}$$

$$= C_{eff}V_{DD}^2 + T_{period}V_{DD}I_o e^{-V_T/nV_{th}} \tag{4.6}$$

It is important to note that minimum supply voltage, and therefore minimum power, does not imply minimum energy due to the impact of leakage energy.

4.2 Impact of N on Energy per Operation

The correlator was implemented in a 90-nm CMOS process with standard threshold devices. Spectre simulations were performed to determine the maximum frequency and energy per operation at various supply voltages. Figure 4-1 shows the results of these simulations for the correlator. It was determined that operating at the minimum energy point of 0.3 V rather than at the maximum V_{DD} of 1 V reduces the energy per operation of the correlator by 89% (Figure 4-1).

At the minimum energy point, the baseband processing must be parallelized to maintain a throughput of 500 MSPS. For ease of design, it is desirable that the PRF of the preamble be a multiple of the clock frequency of the baseband. Since this is not possible at 0.3 V, the baseband operates slightly above the minimum energy point at 0.4 V with a frequency of 25 MHz which requires $N = 20$ correlators to form a sub-bank of correlators. Due to the shallow nature of the energy curve near the minimum energy point, this slight increase in supply voltage does not greatly affect

---

[1]n is also known as the sub-threshold swing parameter in the BSIM4 model [35]
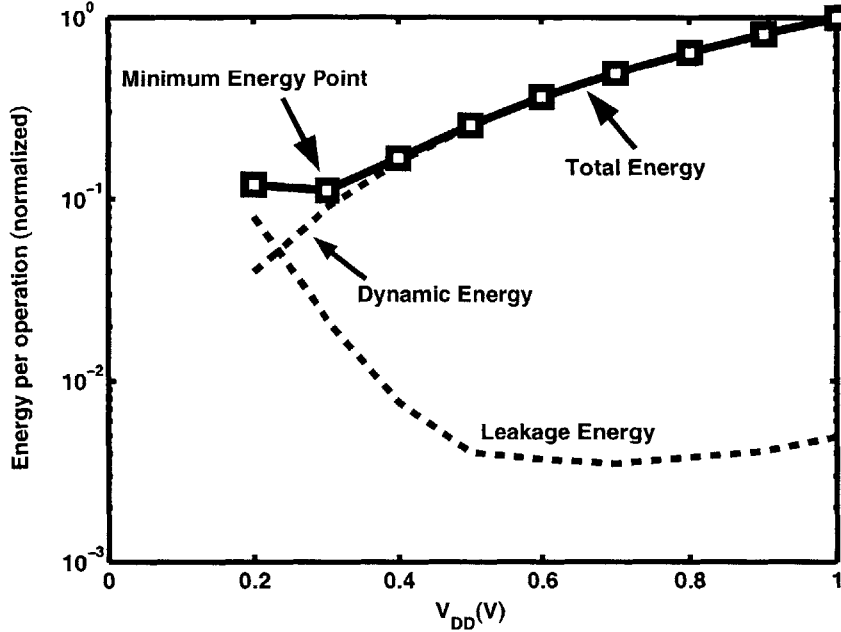
Figure 4-1: Simulated energy plot for the correlator.

the energy savings. Lowering the supply voltage from 1 V to 0.4 V (sub-threshold) results in an overall energy savings of 83% for the correlators and 68% for the entire baseband. The energy savings for the entire baseband is less since buffers are inserted in some paths to compensate for the increased transition time at 0.4 V.

4.3 Summary

Significant energy savings of 83% in the correlator bank and 68% across the entire baseband processor can be achieved by parallelizing by $N = 20$ and reducing the supply voltage to 0.4 V. While operating in the sub-threshold region provides large improvements in energy efficiency, it should be noted that one of the major drawbacks is the circuit's increased sensitivity to process variations (e.g. threshold voltage, channel length, doping, etc.) This issue will be addressed in Chapter 6 where implementation challenges are discussed.

Chapter 5

Reducing energy per packet

In order to achieve synchronization, the cross-correlation function must be computed to determine the delay. The faster the cross-correlation function can be computed, the less time spent on synchronization which reduces the on-time of the UWB receiver. A fast cross-correlation can be achieved by computing more points at the same time. Parallelism by M increases the number of points computed simultaneously by a factor of M. Specifically, it increases the number of Gold code shifts checked at the same time. Although this increases the power consumed by the baseband during acquisition, the time spent in acquisition decreases proportionally. In this chapter, a model is developed[1] to show that the baseband energy remains approximately the same for any M, while the energy spent by the rest of the receiver scales inversely with M. This results in an overall reduction in energy per packet.

5.1   Modeling Energy per Packet

The average time and amount of energy the baseband spends in each state must be determined. The total time the baseband spends in State 0 and 2 is set by the number of times the Gold code is repeated in the preamble, $R(M)$, which can be reduced by

---

[1]The model was developed based on a directly parallelized version of the original algorithm. However, it can be easily modified for the energy efficient algorithm by removing the channel estimation state.

increasing parallelism M.

$$R(M) = \left\lceil \frac{N_c}{M} \right\rceil \tag{5.1}$$

While the time spent in State 1^2 and State 3 is fixed, the distribution of time between State 0 and 2 is dictated by when the baseband detects lock. The maximum time the baseband will remain in State 0 is $R(M) \times T_{code}$. In this case, no time is spent in State 2. Let D be the number of code shifts between the Gold code in the preamble and the Gold code in the baseband. Assume that D is uniformly distributed over $[0, N_c - 1]$. Let $I(D, M)$ be the number of code durations (T_{code}) required to achieve acquisition in State 0. Let P_d be the probability that the baseband detects lock when the input and the Gold code are aligned, and P_{fa-M} be the probability that the baseband detects lock in one or more of M delays which are not aligned to the code. For small P_{fa} ($= P_{fa-1}$), $P_{fa-M} \cong M P_{fa}$ from the Union bound [29].

Assuming the detector is ideal ($P_d = 1$, $P_{fa-M} = 0$),

$$I(D, M) = \left\lceil \frac{D}{M} \right\rceil \tag{5.2}$$

The maximum number of code repetitions required to achieve acquisition in State 0 is $R(M)$ and each repetition requires T_{code} to process. As the baseband performs different operations during each state, the energy per T_{code} varies per state. E_0 and E_2 are the energies consumed over T_{code}, in State 0 and State 2, while E_1 and E_3 are the energies required to perform channel estimation and demodulation respectively[3]. It is important to note that E_0, to a first order, scales linearly with M. After acquisition, $M - 1$ of the correlator sub-banks can be turned off through the use of clock gating and power gating so that E_1, E_2 and E_3 are not dependent on M to a first order.

---

[2]Ignore energy for this state for energy efficient algorithm.

[3]The actual values for E_0, E_1, E_2, and E_3 were obtained by synthesizing and place and routing the baseband processor for various degrees of parallelism M. Analysis therefore includes leakage and interconnect capacitance.

The energy per packet is computed as follows,

$$
\begin{aligned}
Energy(D, M) &= \sum_{X=1}^{R(M)} Pr(X, D, M) \times Energy(X, D, M) \\
&= \alpha E_0 + \beta E_1 + \gamma E_2 + \epsilon E_3
\end{aligned}
\tag{5.3}
$$

$Pr(X, D, M)$ is the probability that the baseband will stay in acquisition for X units of T_{code} given a packet with delay D. $Energy(X, D, M)$ is the energy consumed by the baseband.

For all X$\neq$I(D,M),

$$
Pr(X, D, M) = P_{fa-M}(1 - P_{fa-M})^{X-1}
\tag{5.4}
$$

$$
Energy(X, D, M) = (X E_0 + E_1)
\tag{5.5}
$$

When X=I(D,M),

$$
\begin{aligned}
&Pr(X, D, M) \times Energy(X, D, M) \\
&= P_d(1 - P_{fa-M})^{X-1}\{X E_0 + E_1 + (R(M) - X)E_2 + E_3\} \\
&\quad + (1 - P_d)(1 - P_{fa-M})^{X-1}R(M)E_0
\end{aligned}
\tag{5.6}
$$

It is assumed that $P_d = 0.9$ and $P_{fa} = 10^{-5}$, which were derived from the 802.15.3a proposal [38]. The average energy required by the baseband to process a packet for given a degree of parallelism M is computed by taking the expected value of the energy per packet over all possible delays D, conditioned on M. If P_{fa} is small, this average baseband energy does not change significantly since, to a first order, the same number of operations occur for any M. In addition, for a small P_{fa}, the required preamble time and hence energy spent during acquisition by the rest of the receiver scale inversely with M.

5.2 Impact of M on Energy per Packet

The energy per packet can be broken down into the preamble energy and the payload energy. The payload energy is fixed by the number of bits transmitted per packet. However, the length of the preamble, and consequently the preamble energy, can be reduced based on the configuration of the baseband. A previous version of the UWB baseband checked one combination of the Gold code at a time [31, 13]. In order to check all shifted combinations of the 31-bit Gold code, the baseband must perform at least 31 correlations. The preamble must last for $N_c \times T_{pre} \times R(M = 1) = 34.880\mu s$.

By using multiple sub-banks of correlators that operate in parallel, the number of Gold code shifts that can be checked in one cycle is increased, which reduces the number of repetitions required in the preamble. In a fully parallelized baseband, with 31 sub-banks of correlators, all 31 shifted possibilities of the Gold code are checked simultaneously, and the Gold code only has to be repeated once in the preamble for acquisition. This results in a 31× reduction in the preamble length.

As previously stated, for varying degrees of M, the energy spent by the baseband on the acquisition is almost the same with a slight increase due to increased interconnect capacitance that results from parallelism[4]. The actual energy savings result from the other circuitry in the UWB receiver. Reduction in acquisition time implies that the entire receiver needs to be on for a much shorter period of time. The RF front end, ADCs and the baseband amplifiers can be turned off once the packet has been demodulated. The measured power of these blocks is approximately 79% of the receiver power [3, 2]; shutting them off earlier translates into significant energy savings. The reduction in preamble energy consumed by the entire receiver is shown in Figure 5-1. At full parallelism of $M = 31$, the preamble energy is reduced by 93%. Figure 5-2 shows the reduction in energy per packet, with a payload size of 500 bytes, for various degrees of parallelism. It can be concluded that faster synchronization, combined with duty-cycling, reduces the energy required to receive a UWB packet.

As increasing M only affects preamble energy, the impact of using parallelism to

---

[4]Chapter 6 will address how careful layout of the correlators can greatly reduce interconnect capacitance
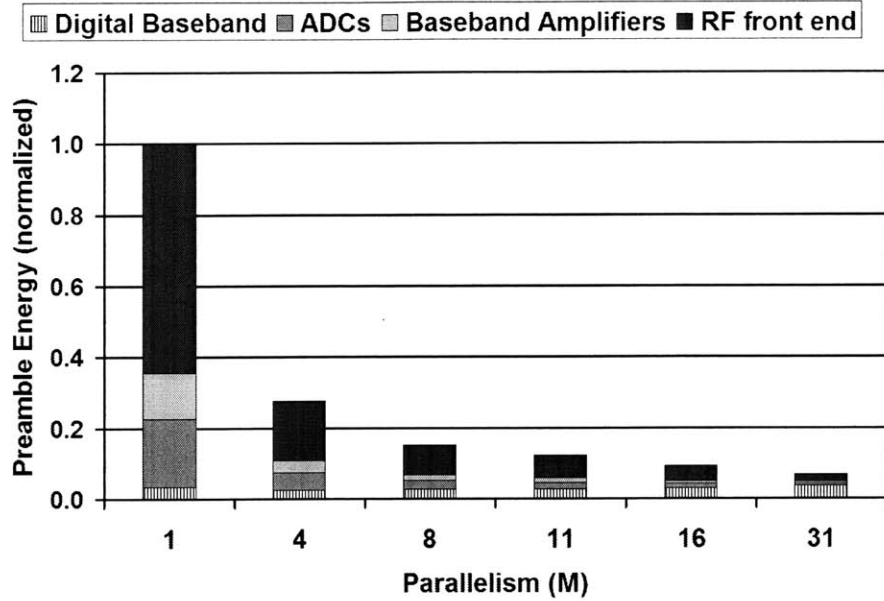
Figure 5-1: Average preamble energy consumption of the receiver subsystems for various degrees of parallelism.

reduce energy per packet varies with payload size. Figure 5-3 shows the reduction in total energy per packet consumed by the receiver for a fully parallelized baseband of $M = 31$ versus a non-parallelized baseband of $M = 1$. Therefore, parallelism by M has a significant impact for cases of short, bursty traffic where the payload is short.

5.3 Modification of Baseband Algorithm

For a fully parallelized design where $M = 31$, only one Gold Code repetition is required in the preamble. As a result, it can be assumed that as soon as the receiver achieves lock, it can move directly into the demodulation phase without having to wait for the end of the preamble. Consequently, the fully parallelized baseband can be designed such that it operates under only two states, the Synchronization State and the Demodulation State. It is important to note that during the demodulation state, all the correlators in the synchronization can be powered off[5]. Figure 5-4 shows a state diagram of the new algorithm.

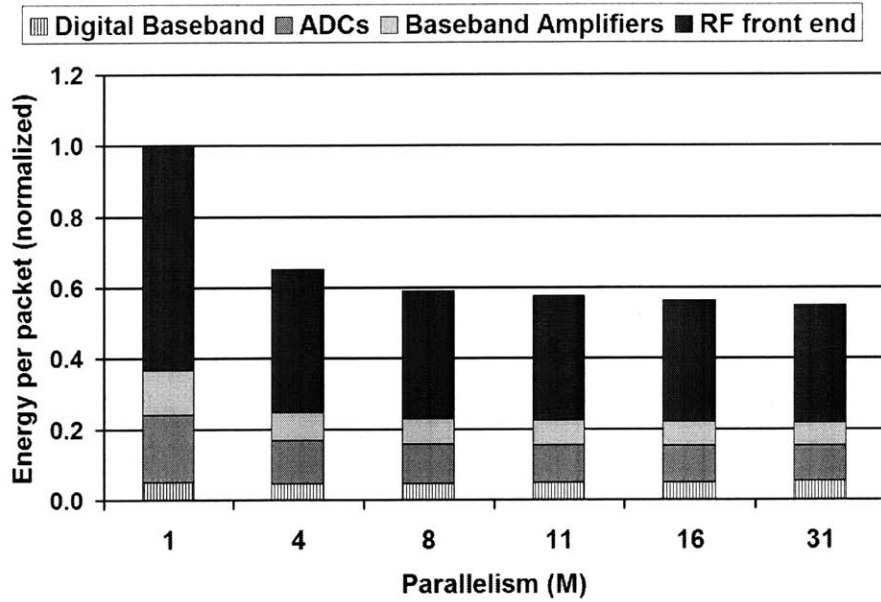[5]One set of correlators may remain on to perform early/late tracking if needed.

Figure 5-2: Average packet energy consumption of the receiver subsystems for various degrees of parallelism.
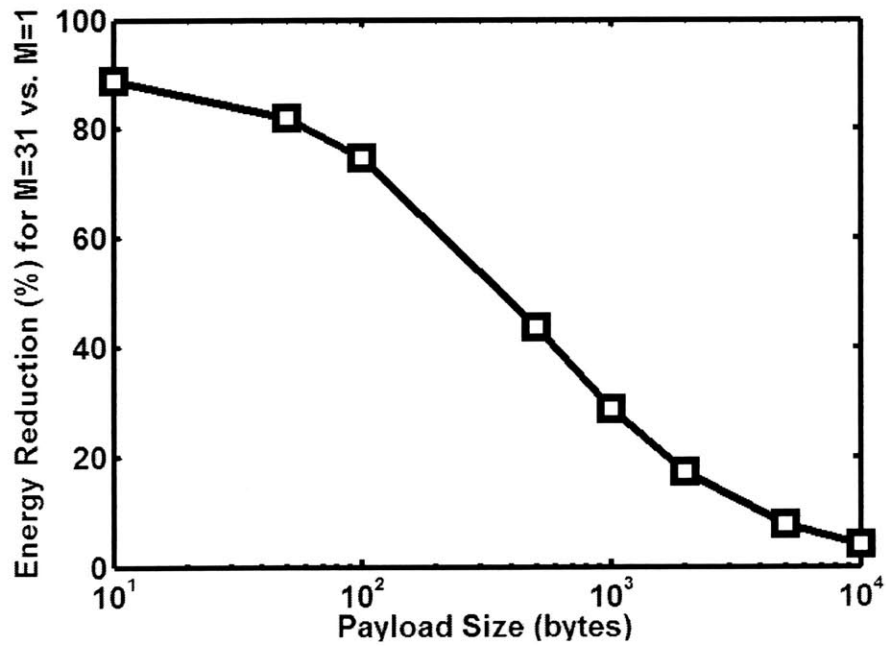

Figure 5-3: Energy reduction for various payload sizes.
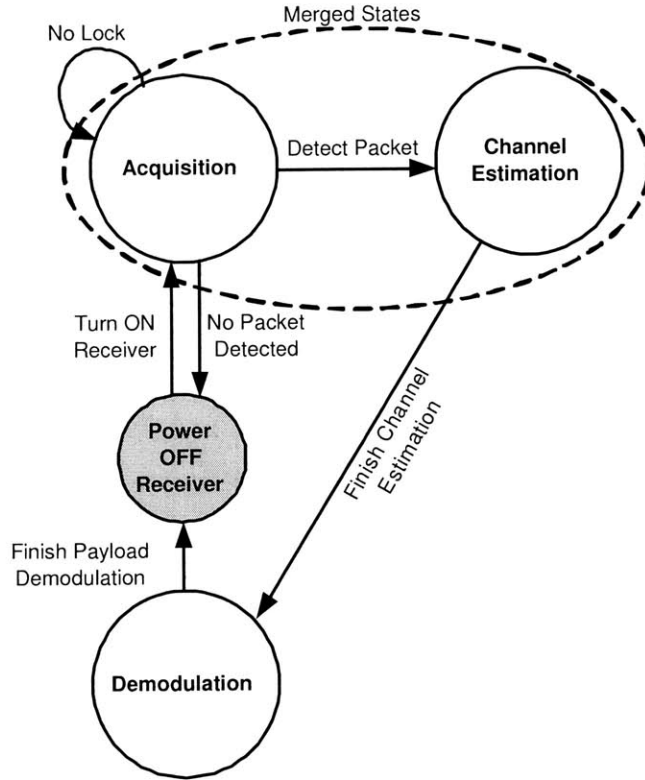
Figure 5-4: State diagram of fully parallelized energy efficient algorithm.

5.4 Summary

Parallelism by M reduces the synchronization time and therefore reduces the on-time of the UWB receiver[6]. A fully parallelized ($M = 31$) design of the baseband algorithm, where the entire cross-correlation function is computed simultaneously, achieves a 93% reduction, or approximately 14× reduction in preamble energy, and a 43% reduction in energy per 500-byte packet.

Note that reduction in preamble energy should not be made at the expense of the payload energy. Techniques such as clock gating and power gating are used to ensure that the power consumption of the baseband during demodulation does not increase with parallelization. During State 1, 2 and 3, either most or all correlators should be turned off and power gated to reduce leakage, and clock gating is used to reduce the

---

[6]An additional benefit of reducing the synchronization time is that it may reduce the need for finetracking with a Costas Loop and DLL, since reducing the number of preamble repetitions shortens the packet duration (without reducing the payload size) allowing less time for the clocks to drift. Therefore, it increases the maximum payload size in packets for which finetracking is not required.

impact of the increased interconnect capacitance due to parallelism. These techniques will be explored in more detail in the next chapter.

Chapter 6

Implementation

An energy efficient design requires attention to design of both the algorithm as well as the implementation. This chapter will address the challenges that were faced during the implementation of the baseband algorithm and power saving techniques to circumvent them. These challenges include process variations, particularly local variations, and challenges that arise for highly parallelized designs.

6.1 Challenges for Sub-Threshold Design

6.1.1 Process Variations

In sub-threshold operation, the effects of process variations are more pronounced. In addition to global variations from die to die, the local variations from transistor to transistor have a significant impact in the sub-threshold region [36]. This is due to the fact that the current is now exponentially related to the threshold voltage. In particular, local mismatch can cause significant changes in the delays of the path. As a result, Monte Carlo simulations had to be performed on the critical paths to ensure that hold time and setup time were met given local variations. These variations had to be taken into account during the selection of hold time slack for place and route with Astro (Please see Appendix A for further discussion of the Synopsys^{TM} flow). It is also important to note that as the number of gates in the path increases, the

some of the mismatch in the gates cancel each other out and the variation in delay is reduced[1].

6.2 Challenges for Highly Parallelized Designs

6.2.1 Interconnect Capacitance

Clock Gating

Extreme parallelism also leads to large interconnect capacitance which increases the energy consumption of the chip. Clock gating is used to reduce the switching capacitance on the clock network. During each cycle $20(\times 2$ for I and Q) correlators switch their outputs, while the remaining $600(\times 2)$ correlators outputs are disabled. Typically, when a register is disabled, a multiplexer is used to feedback the output to the input of the register to hold the state (Figure 6-1(a)). Although the internal nodes of the register do not change, the clock network leading to the register must be switched. This is an unnecessary waste of energy. Clock gating uses a latch in the clock path which "gates" the clock such that when a register is disabled, it is not clocked (Figure 6-1(b)). The clock gating can be placed earlier in the clock network so that the interconnect and the switching activity of the clock signal leading to the register is also reduced. This results in significant savings in the clock network of the chip.

Careful Layout

The arrangement of the correlators within the correlator bank is important to lower the interconnect capacitance of this highly parallelized block. Two possible arrangements of the correlators are to either group the correlators by code position (M sub-banks with N correlator per sub-banks) or group the correlators by pulse position (N sub-banks with M correlators per sub-bank). Although the former arrangement, shown in Figure 3-4, was used to explain the baseband algorithm in Chapter 3, the

---

[1]This is assuming that the mismatch of the gates are not highly correlated

(a) Enable register without clock gating

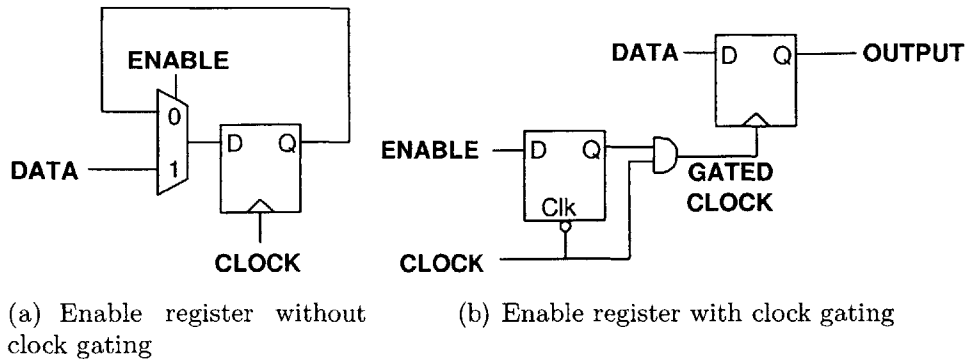(b) Enable register with clock gating

Figure 6-1: Various methods of implementing an enable for a register.

latter arrangement, shown in Figure 6-2 was actually implemented. In the former arrangement, N complex 5-bit input signals had to be distributed to M sub-banks. Roughly speaking, this implies routing $10MN$ nets around the chip. On the other hand, the latter arrangement required the routing of M real 1-bit Gold code signals to N sub-banks. This implies routing MN nets around the chip. Therefore, roughly speaking, choosing grouping by pulse position rather than code position results in a $10\times$ reduction in interconnect. An additional benefit is that if the time-interleaved ADCs were equally parallelized, a very simple architecture would result.

6.2.2 Leakage Current

Power Gating

During idle state, when there is no switching activity, the circuit continues to consume leakage power. When more transistors are used in a design, it often results in more leakage. Accordingly, leakage power is an important concern for highly parallelized designs.

When the baseband processor has completed demodulation, the entire baseband should be power gated. This implies placing a high threshold voltage device between the power supply and the chip. For this work, the gate will be placed off-chip between the power supply and the baseband processor. Figure 6-3 illustrates where the gating transistor is placed. The sizing of this transistor is crucial as it must be wide enough
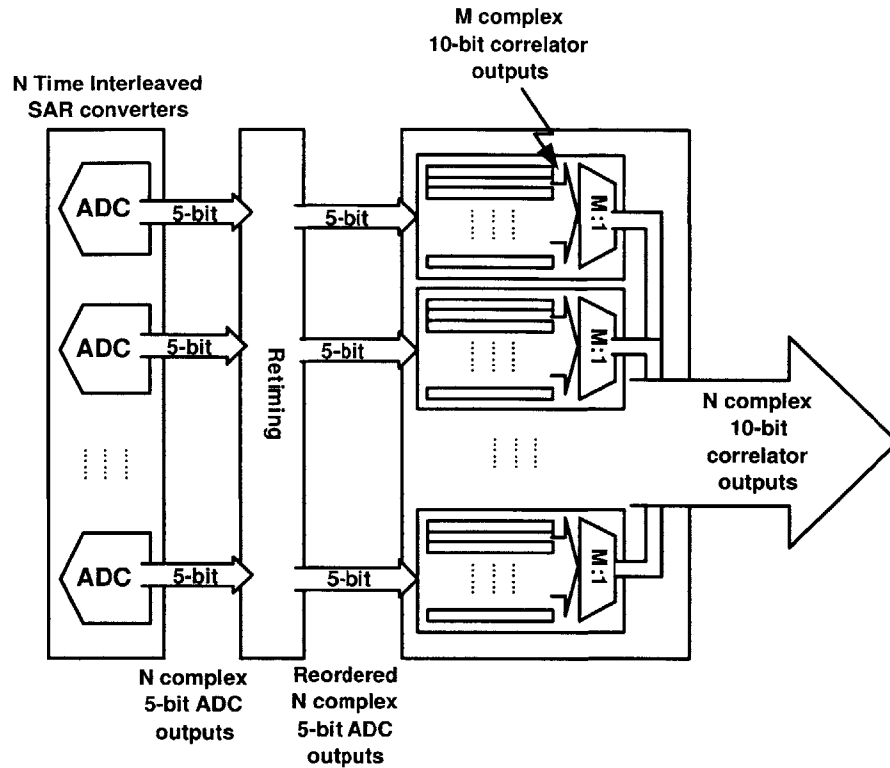
Figure 6-2: Correlator bank in digital baseband processor.

such that it doesn't hinder the performance of the baseband due to IR drops, but it must also be able to reduce the leakage current sufficiently. According to Design Compiler analysis, the idle leakage power accounts for 10 to 20% of the total power consumed by the processor. Power gating to reduce this power can translate into additional energy savings. Power gating should also be used on the other blocks in the receiver (ADC, RF front end, etc.) to minimize the system-wide leakage when the receiver is idle.

6.2.3 Peak Current

Due to the highly parallelized nature of the design, peak currents become a significant challenge when the design is operating at full voltage. Recall that the correlators have been parallelized by $N = 20$ for sub-threshold operation, followed by another level of parallelism of $M = 31$ for reduced synchronization time. Therefore, there are 620 correlators per channel, with two channels I and Q. The instantaneous current results
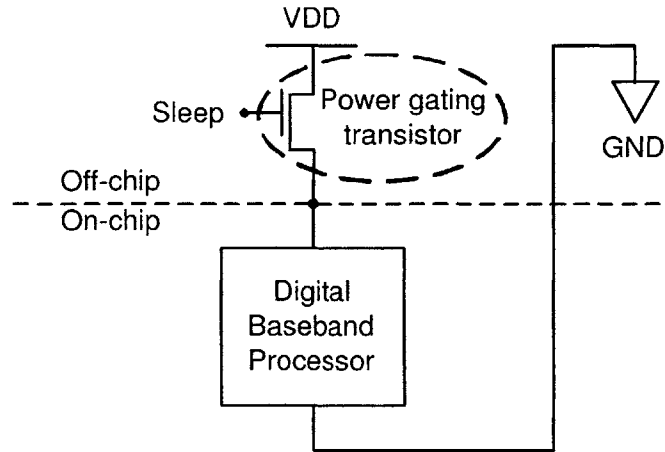
Figure 6-3: Architecture for power gating.

from Nanosim are shown in Figure 6-4 for a supply voltage of 0.4 V and 1 V. At 1 V, the peak currents may potentially lead to issues with voltage drops across the bond wires. Specifically, a voltage drop of Ldi/dt occurs due to the inductance of the bond wires and the sharp transitions in the current.

The impact of this instantaneous current was simulated using RLC circuit that modeled the bonding wire and the decoupling capacitance in the baseband processor. It was determined that with sufficient decoupling[2] > 12nF, the maximum supply voltage drop across the bond wire to the die would be around ± 100 mV for a 1 V supply and ± 5 mV for a 0.4 V supply. This translates into a worst case additional 10% change in delay.

6.3 Layout of Baseband Processor

The layout of the baseband processor is shown in Figure 6-5. The floorplan and the bonding diagram are provided in Appendix B. The 3.3-mm x 3.3-mm chip contains 152 pads including 24 power pads for the core, 16 power/ground pads for the output buffers, 6 I/O power pads, 29 ground pads for I/O and core and 77 signal pads. The power pads are spread out along the pad ring to ensure adequate delivery of power

---

[2]Since the design was pad limited, this on-chip decoupling capacitance was achieved simply by using MOS capacitors for filler cells.
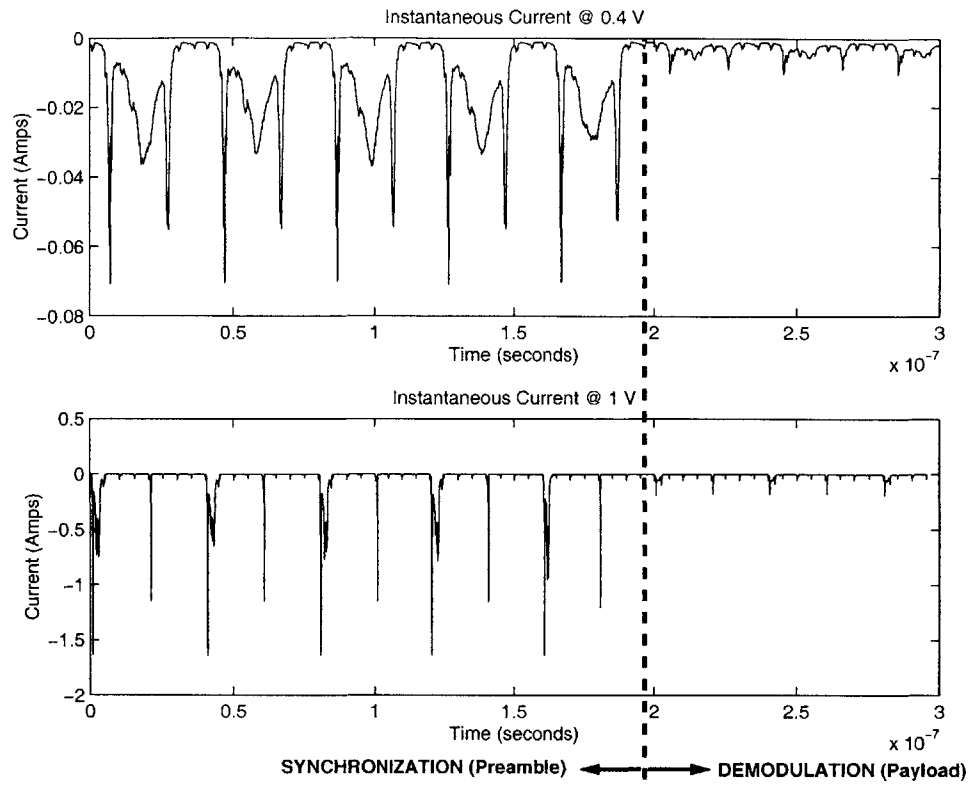
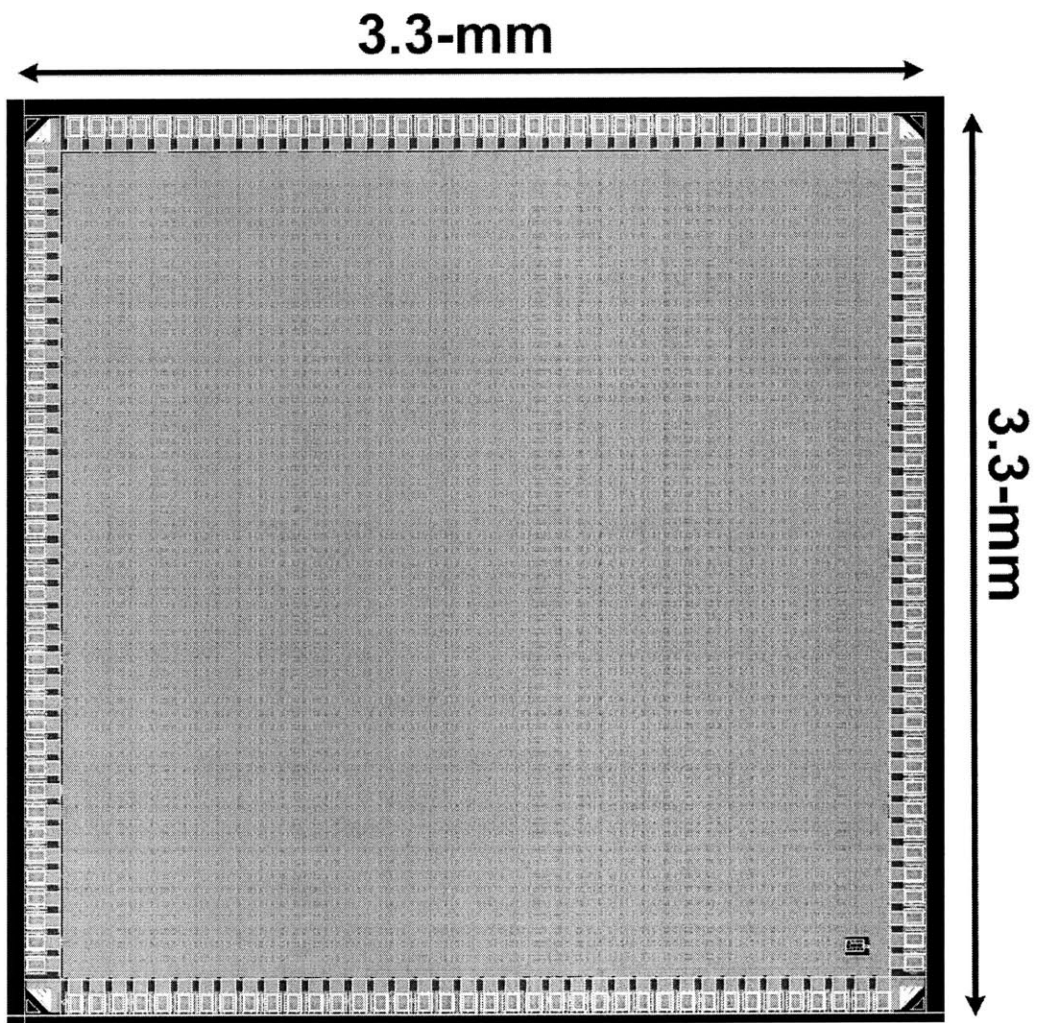Figure 6-4: Instantaneous current from Nanosim simulations.

across the chip.

Figure 6-5: Layout of baseband processor.

Chapter 7

Conclusion

7.1 Summary of Contributions

This thesis discussed how the baseband algorithm for a pulsed Ultra-wideband transceiver can be mapped to a parallel architecture in order to achieve an energy efficient baseband processor. Parallelism allows for voltage scaling and reduced acquisition time, which reduces the energy required to receive a UWB packet. Voltage scaling to sub-threshold allowed the correlator sub-banks to operate near the minimum energy point, resulting in an energy per operation reduction in the correlators of 83% and energy reduction of 68% across the entire baseband. Parallelism of $N = 20$ was required to maintain the throughput at 500-MSPS. The reduced acquisition time through further parallelization of the correlator sub-banks by $M = 31$ led to system-wide energy savings of 93% in preamble energy and 43% in energy per packet for a 500-byte packet.

Techniques such as power gating, clock gating and careful layout can help to mitigate issues of increased leakage power and interconnect capacitance for highly parallelized designs. Sufficient hold and setup time slack must be allocated to account for process variations which have increased impact in sub-threshold design. The baseband algorithm was verified using a prototyping platform and then implemented in a 90-nm CMOS process. The analysis presented in this thesis can be mapped to other high performance communication applications using sub-threshold operation and parallelism.

7.2 Future Work

Extensions of this work could include on-chip fine grained power gating for each of the states under which the baseband processor operates. Since this would require that the blocks of each state be on separate power domains, the supply voltage for each domain could also be scaled such that the processor operates at the minimum energy point in each state. While this thesis focused on parallelism in the correlator bank, parallelism in other portions of the algorithm could also be explored in future implementations.

Appendix A

Tools

This appendix is provided to give the reader further insight into the steps required to implement the baseband processor. This discussion will cover steps that come after the design and characterization of the baseband algorithm in MATLAB. The MATLAB code of the baseband algorithm must be manually translated into a bit true implementation in Verilog RTL code. MATLAB was also used to produce test vectors for any simulations that were necessary throughout the implementation. SignalStorm was used to characterize the digital CMOS libraries at 0.4-V [39].

A.1 Flow

The following is a list of steps required to implement the baseband via the Synopsys™ flow:

1. Verilog-XL and VCS (RTL Simulation)

2. Design Compiler (Synthesis)

3. Astro (Place and Route)

4. Calibre LVS/DRC (Layout Verification)

5. PrimeTime (Timing Verification)

6. NanoSim (Connectivity and Power Verification)

A.1.1 Verilog-XL and VCS

Verilog-XL and VCS were used to simulate the behavioural Verilog RTL. These simulators were also used following synthesis to verify gate level performance. Since the clock tree is not implemented until after place and route, it is not necessary to simulate with timing checks and delays. To verify only the functionality of the Verilog code, the VCS flag +*nospecify* was used for gate-level simulations (i.e. path delays and timing checks are ignored for a functional gate simulation) [40]. Simvision was used to view the output of the Verilog simulations.

A.1.2 Design Compiler

Design Compiler was used to translate the behavioural Verilog RTL to a synthesized gate level Verilog netlist [41]. Timing, area and power constraints were set during this step. Design Compiler reads in an SDC file that contains these constraints and maps the RTL to gates appropriately. Clock gating can be inserted during this stage using the following commands [42]

```
# Insert clock gating, using the integrated clock gating cells
set_clock_gating_style
-minimum_bitwidth 1
-num_stages 4
-positive_edge_logic {integrated}
-control_point before
insert_clock_gating -module
# Tie off test enable ports to logic 0.
hookup_testports -se_port test_en
```

Due to the I/O pad ring limitations, the input had to be read into the chip serially and parallelized on chip. As a result, an on chip clock divider was required. This divided clock was created by Design Compiler using the following command [42]

```
create_clock -period Clk100Period il_clock_100MHz
```

70

```
create_generated_clock -name GeneratedClk25 -edges {1 5 9} \
-source il_clock_100MHz [get_pins core1/s2p/fast/Clk25_reg/Q]
```

The timing constraints in the data path between the two clock domains were applied using *set_multipath_cycle* [42].

Due to the degree of parallelism in the design, the synthesis was performed hierarchically. From bottom to top, the modules that were synthesized included: a single correlator, a sub-bank of 31 correlators, and then the top-level.

A.1.3   Astro

Astro is a place and route tool that takes the synthesized Verilog netlist and builds the layout of the chip from standard cells. Clock tree synthesis is performed at this point. The Recommended Astro Script-Based Methodology was used where the entire place and route of the chip was scripted. The GUI was used to check the layout at intermediate steps. The inputs for Astro included [43, 44]:

1. Synthesized Verilog Netlist

2. Libraries for standard cells including layout, power, and timing information

3. Technology File containing design rules, parasitics, and other process specific information.

4. Timing Constraints (typically the same SDC file used by Design Compiler)

5. Floorplan file containing information about location of I/O pins and dimensions of chip

The main steps of the Astro flow included

1. *init_design*: Maps the netlist to the standard cells and outlines the chip based on information provided by the floorplan. The placement of I/O cells and routing

of the power grid is also performed. If the design is hierarchal, the placement of Macro blocks occurs at this stage[1]

2. *place_opt*: Initial placement of the standard cells based on timing constraints and routability.

3. *clock_opt*: Clock tree synthesis, minimize clock skew, gate and buffer sizing and delay insertion.

4. *route_design*: Signals are routed based on timing constraints, maximum capacitance and transition requirements. Minimize wire length and remove unnecessary jogs.

5. *post_route*: Fixes minor timing violations and clock skew due to optimization as well as checks for design rule constraints such as max fanout, transition and capacitance.

6. *ant*: Corrects for antenna rule violations by inserting diodes

7. *add_fill_1*: Fills empty space with decoupling caps and filler cells

8. *add_fill_2*: Fills empty tracks with metal

Following the place and route, Calibre DRC and LVS must be performed on the resulting layout to ensure that it meets all the design rules and has the correct connections. The Verilog netlist provided by Astro, that includes all additional buffering and power ports, must be converted to SPICE format by NetTran as well as some custom scripts in order to perform LVS.

A.1.4 PrimeTime

PrimeTime is used to verify timing of the layout including interconnect delays. Inputs to PrimeTime include the standard cell libraries with timing information for fast and

---

[1]Macro blocks were used in the initial implementation to allow for separate power domain that could be gated. However, due to the challenges faced with the peak currents, it was decided that the same power domain should be used for the entire chip so that we could reduce bond wire inductance to reduce Ldi/dt drops. Consequently, a flat structure of the layout was implemented.

slow corners, the SDC file of the timing constraints, the Verilog netlist and the SPEF (parasitics) file provided by Astro.

report_timing is used to check for timing violations [45]. In addition, histograms of the worst case setup and hold times were generated.

A.1.5 Nanosim

Nanosim verifies the circuit at the transistor-level and gives an estimate of power [46]. The circuit was verified for the following corners: 1-V (TYPICAL, FAST - 3 sigma), 0.4-V (TYPICAL), 0.5-V (SLOW - 3 sigma). At 1-V, the simulator is set to the default accuracy level of (2 2 2), however at 0.4-V, the accuracy setting of (4 4 4) was used.

Appendix B

Floorplan and Bonding Diagram

The floorplan of the pad ring for the baseband processor is shown in Figure B-1. The bonding diagram is shown in Figure B-2. Ground pads will be connected via downbonding to reduced inductance. A CQFP 144-pin package will be used.
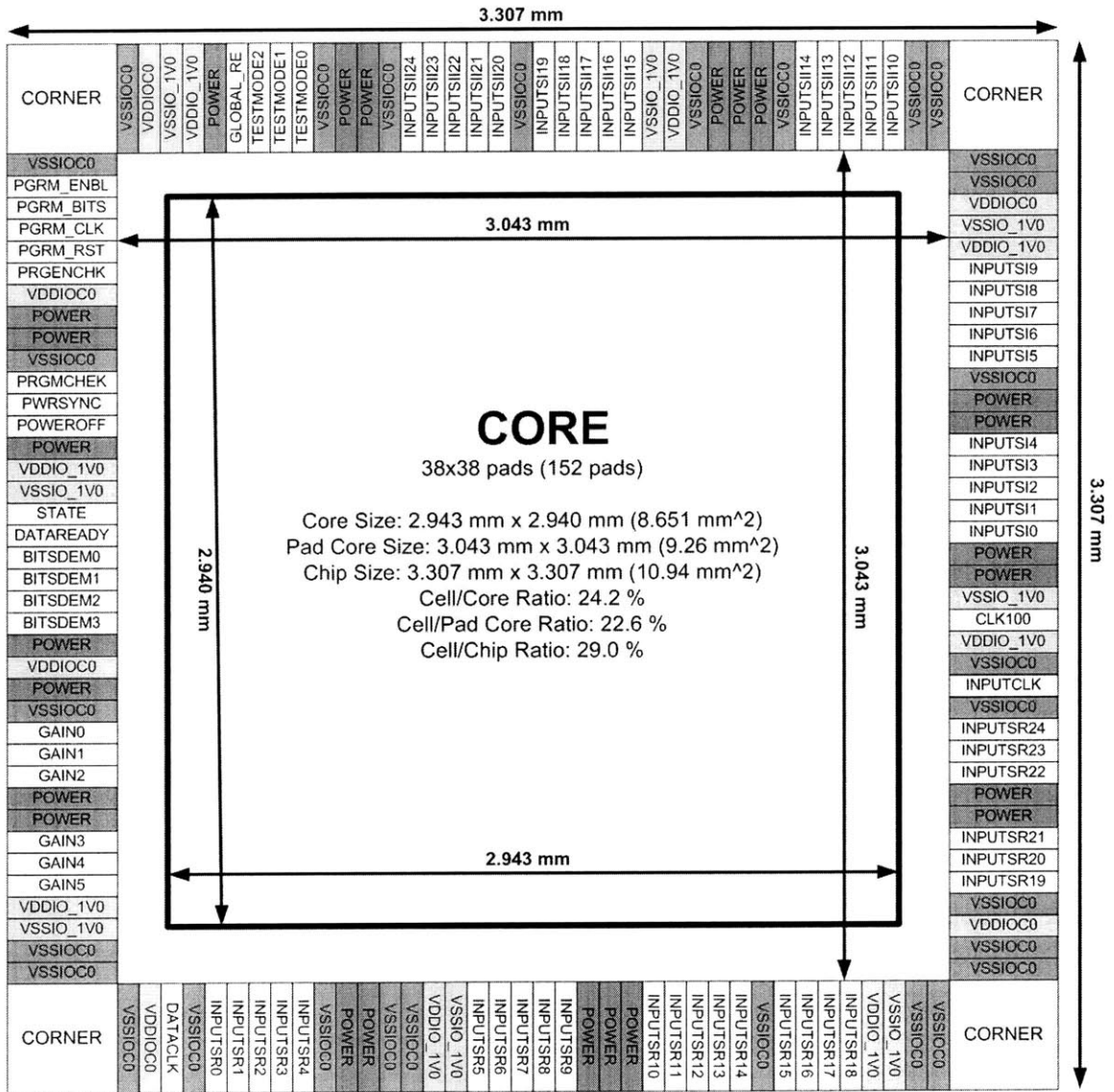
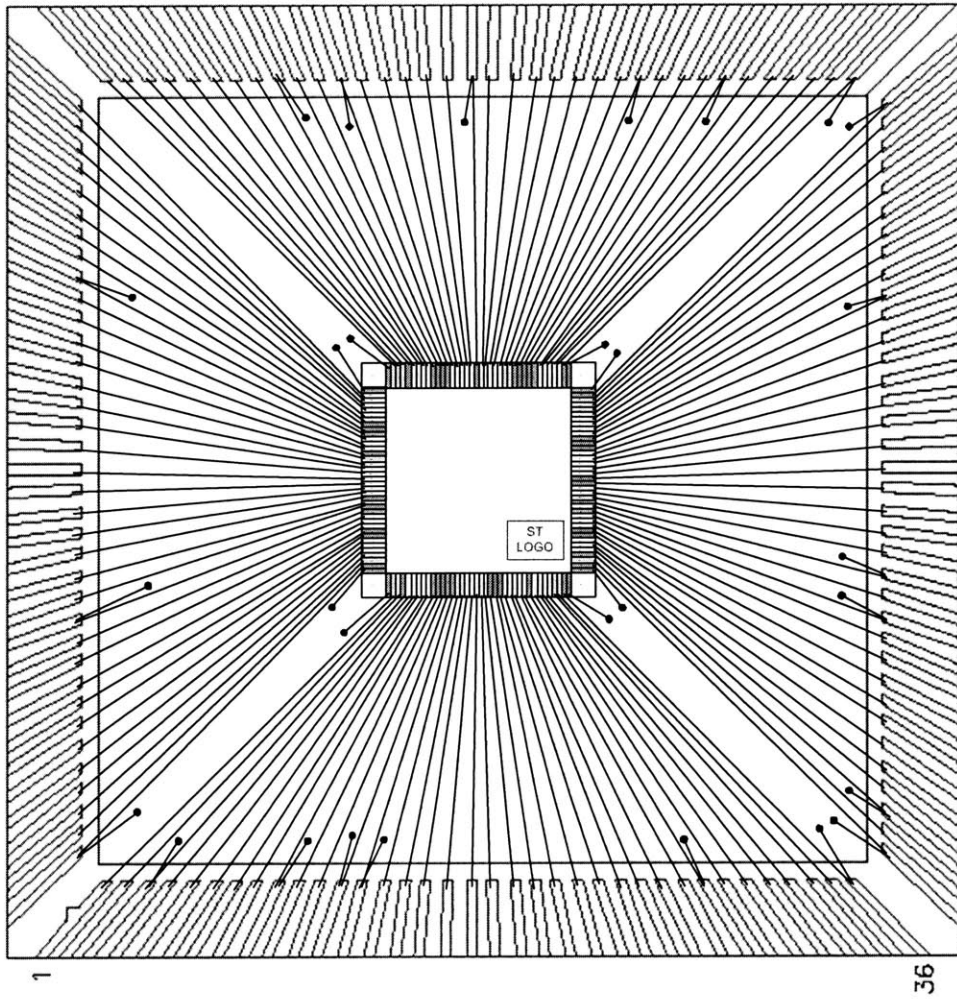Figure B-1: Floorplan for baseband processor.

Figure B-2: Bonding diagram for baseband processor.

Bibliography

[1] "Ultra-wideband First Report and Order," Federal Communications Commission, Tech. Rep., February 2002.

[2] B. P. Ginsburg and A. Chandrakasan, "Dual Scalable 500MS/s, 5b Time-Interleaved SAR ADCs for UWB Applications," in *IEEE Custom Integrated Circuits Conference*, September 2005.

[3] F. S. Lee and A. Chandrakasan, "A BiCMOS Ultra-Wideband 3.1-10.6GHz Front-End," in *IEEE Custom Integrated Circuits Conference*, September 2005.

[4] J. McCorkle, "Ultra wide bandwidth (UWB): gigabit wireless communications for battery operated consumer applications," in *2005 Symposium on VLSI Circuits Digest of Technical Papers*, June 2005, pp. 6–9.

[5] K. Siwiak, "Ultra-wide band radio: introducing a new technology," in *IEEE Vehicular Technology Conference*, vol. 2, May 2001, pp. 1088 – 1093.

[6] K. Siwiak, P. Withington, and S. Phelan, "Ultra-wide band radio: the emergence of an important new technology," in *IEEE Vehicular Technology Conference*, vol. 2, May 2001, pp. 1169 – 1172.

[7] K. Siwiak, "Impact of Ultra Wide Band Transmissions on a Generic Receiver," in *IEEE Vehicular Technology Conference*, vol. 2, May 2001, pp. 1181 – 1183.

[8] L. Yang and G. Giannakis, "Ultra-Wideband Communications An Idea Whose Time Has Come," *IEEE Signal Processing Magazine*, vol. 21, no. 6, pp. 26–54, November 2004.

[9] "Ultra-Wideband (UWB) Technology, Technology & Research at Intel." [Online]. Available: http://www.intel.com/technology/comms/uwb/

[10] A. Wang and A. Chandrakasan, "A 180mV FFT Processor Using Subthreshold Circuit Techniques," in *IEEE International Solid-State Circuits Conference*, February 2004.

[11] B. H. Calhoun and A. P. Chandrakasan, "Characterizing and Modeling Minimum Energy Operation for Subthreshold Circuits," in *International Symposium on Low Power Electronics and Design*, August 2004.

[12] R. Blazquez, P. Newaskar, F. Lee, and A. Chandrakasan, "A Baseband Processor for Impulse Ultra-Wideband Communications," *IEEE Journal Of Solid-State Circuits*, vol. 40, no. 9, pp. 1821–1828, September 2005.

[13] R. Blazquez-Fernandez, "Design of Synchronization Subsystem for an Ultra Wideband Radio," Master's thesis, Massachusetts Institute of Technology, 2003.

[14] M. Verhelst and W. Dehaene, "System Design of an Ultra-low Power, Low Data Rate, Pulse UWB Receiver in the 0-960MHz Band," in *IEEE International Conference on Communications*, May 2005.

[15] S. Haykin, *Communication Systems*, 4th ed. John Wiley & Sons, Inc., 2001.

[16] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[17] M. Win and R. Scholtz, "Characterization of Ultra-wide Bandwidth Wireless Indoor Channel: A Communication Theoretic View," *IEEE Journal on Selected Areas of Communication*, vol. 20, no. 9, pp. 1613–1627, December 2002.

[18] D. Wentzloff, R. Blazquez, F. Lee, B. P. Ginsburg, J. Powell, and A. Chandrakasan, "System Design Considerations for Ultra-Wideband Communication," *IEEE Communications Magazine*, vol. 43, no. 8, pp. 114–121, August 2005.

[19] R. Blazquez, F. Lee, D. Wentzloff, J. Newaskar, P.P.and Powell, and A. Chandrakasan, "Digital Architecture For An Ultra-Wideband Radio Receiver," in *Vehicular Technology Conference*, 2003.

[20] M. Verhelst, W. Vereecken, M. Steyaert, and W. Dehaene, "Architectures for Low Power Ultra-Wideband Radio Receivers in the 3.1-5GHz Band for Data Rates < 10Mbps," in *International Symposium on Low Power Electronics and Design*, 2004.

[21] B. Razavi, *RF microelectronics*. Prentice Hall, 1998.

[22] D. D. Wentzloff and A. P. Chandrakasan, "A 3.1-10.6 GHz Ultra-Wideband Pulse-Shaping Mixer," in *IEEE Radio Frequency IC Symposium*, June 2005.

[23] R. Blazquez, F. S. Lee, D. Wentzloff, B. Ginsburg, J. Powell, and C. A.P., "Direct Conversion Pulsed UWB Transceiver Architecture," in *Design, Automation and Test in Europe*, March 2005.

[24] H. Metr, G. Ascheid, *Synchronization in Digital Communications, Volume 1: Phase-Frequency-Locked Loops and Amplitude Control*. Wiley Interscience, 1990.

[25] J. Foerster, "Channel Modeling Sub-committee Report Final, IEEE P802.15-02/368r5-SG3a," IEEE P802.15 Working Group for WPAN, Tech. Rep., February 2002.

[26] "Taxicab Geometry," 2006. [Online]. Available: http://en.wikipedia.org/wiki/Taxicab_geometry

[27] Eric W. Weisstein et al., "Taxicab Metric From MathWorld–A Wolfram Web Resource." May 2006. [Online]. Available: http://mathworld.wolfram.com/TaxicabMetric.html

[28] "Distance," 2006. [Online]. Available: http://en.wikipedia.org/wiki/Distance

[29] A. S. Willsky, G. W. Wornell, and J. H. Shapiro, "6.432 Stochastic Processes, Dectection and Estimation Course Notes," Fall 2005.

[30] N. J. Ackerman, "A Platform for Ultra Wideband Communication Systems," Master's thesis, Massachusetts Institute of Technology, 2005.

[31] R. Blazquez and A. P. Chandrakasan, "Architectures for Energy-Aware Impulse UWB Communications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2005.

[32] FOXElectronics, "3.0V LOW PROFILE VCTCXO MODEL: FOX801BE," 2004. [Online]. Available: http://www.foxonline.com/pdfs/fox801be.pdf

[33] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal Of Solid-State Circuits*, vol. 27, no. 4, pp. 473 – 484, April 1992.

[34] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.

[35] "BSIM4.2.1 MOSFET Mode - User Manual." [Online]. Available: http://www-device.eecs.berkeley.edu/ bsim3/bsim4_arch_ftp.html

[36] Y. Cao and L. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach," in *Design Automation Conference*, 2005.

[37] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[38] IEEE P802.15 Working Group for WPAN, "IEEE 802.15.3a Updated DS-UWB Proposal Specification," IEEE, Tech. Rep., July 2004.

[39] Cadence, "SignalStorm Library Characterizer User Guide Product Version 3.2," February 2004.

[40] Synopsys, "VCS /VCSi User GuideVersion X-2005.06," August 2005. [Online]. Available: http://solvnet.synopsys.com

[41] ——, "Design Compiler User Guide W-2004.12," December 2004. [Online]. Available: http://solvnet.synopsys.com

[42] ——, "Design Compiler Reference Manual: Constraints and Timing Version X-2005.09," September 2005. [Online]. Available: http://solvnet.synopsys.com

[43] ——, "Astro User Guide Version W-2004.12," December 2004. [Online]. Available: http://solvnet.synopsys.com

[44] ——, "Recommended Astro Script-Based Methodology Application Note Version W-2004.12," December 2004. [Online]. Available: http://solvnet.synopsys.com

[45] ——, "PrimeTime User Guide: Advanced Timing Analysis Version X-2005.12," December 2005. [Online]. Available: http://solvnet.synopsys.com

[46] ——, "NanoSim User Guide Version W-2004.12," December 2004. [Online]. Available: http://solvnet.synopsys.com