IC DESIGN, UWB SYNCHRONIZATION CIRCUIT

TOH WEI DA

(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF

ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

ABSTRACT

In conventional radio frequency (RF) transceivers, clock drift problem exists between transmitter and receiver due to the different system clock employed. Therefore, clock and data recovery (CDR) circuitry will be required to recover the received data correctly. Phase locked loop (PLL) and delay locked loop (DLL) are very commonly used analog building blocks for clock and data recovery (CDR) applications. Another popular timing recovery technique involves the use of analog-to-digital converter (ADC), coupled with digital signal processing algorithm. However, these techniques are very power consuming and not easily scalable with Complementary Metal–Oxide–Semiconductor (CMOS) technology. A novel low power, all digital timing recovery technique to be used together with the ultra-wideband (UWB) radio frequency front-end is therefore proposed in this work, which achieves low power consumption and is easily scalable. In addition, the voltage controlled oscillator (VCO) employed in UWB transmitter is susceptible to process, supply voltage and temperature (PVT) variations which can lead to spectral shift and variations which violate the UWB Federal Communications Commission (FCC) spectral mask. A digital calibration technique is also proposed and implemented in this work to tackle the problem to achieve accurate transmitting frequency and pulse width calibration.

ACKNOWLEDGMENTS

My sincere thanks go to all National University of Singapore's staffs who have shared their knowledge with me.

The one who enhanced my learning by answering my doubts, giving me information and all the necessary materials is my NUS project's supervisor, Dr Heng Chun Huat. I want to thank Dr Heng Chun Huat for his patience and care that he has shown throughout the entire duration of this project.

Last but not least, I would like to thank Dr. Zheng Yuanjin, from Institute of Microelectronics, Singapore, who has helped me to reserve all the necessary area and equipments, which I needed, to do functional verification and debugging.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

"*ADC*"	« Analog-to-Digital Converter »
"*BER*"	« Bit Error Rate »
"*CDR*"	« Clock and Data Recovery »
"*CML*"	« Current Mode Logic »
"CMOS"	« Complementary Metal–Oxide–Semiconductor »
"CP"	« Charge Pump »
"*DC*"	« Direct Current »
"*DLL*"	« Delay Locked Loop »
"*FCC*"	« Federal Communications Commission »
"*FSM*"	« Finite State Machine »
"HDL"	« Hardware Description Language »
"*IDA*"	« Infocomm Development Authority »
"*IME*"	« Institute of Microelectronics »
"*IR*"	« Impulse Radio »
"*LFSR*"	« Linear Feedback Shift Register »
"*LNA*"	« Low Noise Amplifier »
"*LPF*"	« Low Pass Filter »
"*MAC*"	« Media Access Control »
"*MCU*"	« Microcontroller Unit »
"*NRZ*"	« Non-Return-to-Zero »
"*OOK*"	« On-Off Keying »
"*PD*"	« Phase Detector »

“*PLL*”	« Phase Locked Loop »
“*PN*”	« Pseudorandom Number »
“*PVT*”	« Process, Voltage, Temperature »
“*RF*”	« Radio Frequency »
“*RTL*”	« Register Transfer Level »
“*RZ*”	« Return-to-Zero »
“*SPI*”	« Serial Peripheral Interface »
“*TSPC*”	« True Single Phase Clock »
“*UFZ*”	« UWB Friendly Zone »
“*UWB*”	« Ultra Wideband »
“*VCO*”	« Voltage-Controlled Oscillator »
“*VGA*”	« Variable Gain Amplifier »

CHAPTER 1

INTRODUCTION

1.1 Background

Ultra Wideband (UWB) is a radio technology [1-2] that transmits signals with much wider bandwidth than conventional narrow-band radio technology. It is defined as any transmission that has a fractional bandwidth of more than 20% or -10 dB bandwidth of more than 500 MHz. In year 2002, the U.S. Federal Communications Commission (FCC) regulated the spectrum and transmission power for UWB radio transmission. According to the regulation, UWB signal occupies the 3.1 GHz to 10.6 GHz frequency band and should limit the output power level smaller than -41.3 dBm/MHz. In Singapore, Infocomm Development Authority (IDA) has issued trial permits for controlled UWB emissions within Science Park II which is known as UWB Friendly Zone (UFZ) to facilitate research and development of UWB technology. The allowable frequency band in UFZ is slightly wider than the specified FCC spectral mask (2.2 GHz to 10.6 GHz). The UWB output power in UFZ is also relaxed further by 6 dB. The UWB FCC spectral mask and UFZ emission limit are shown in Fig. 1-1.

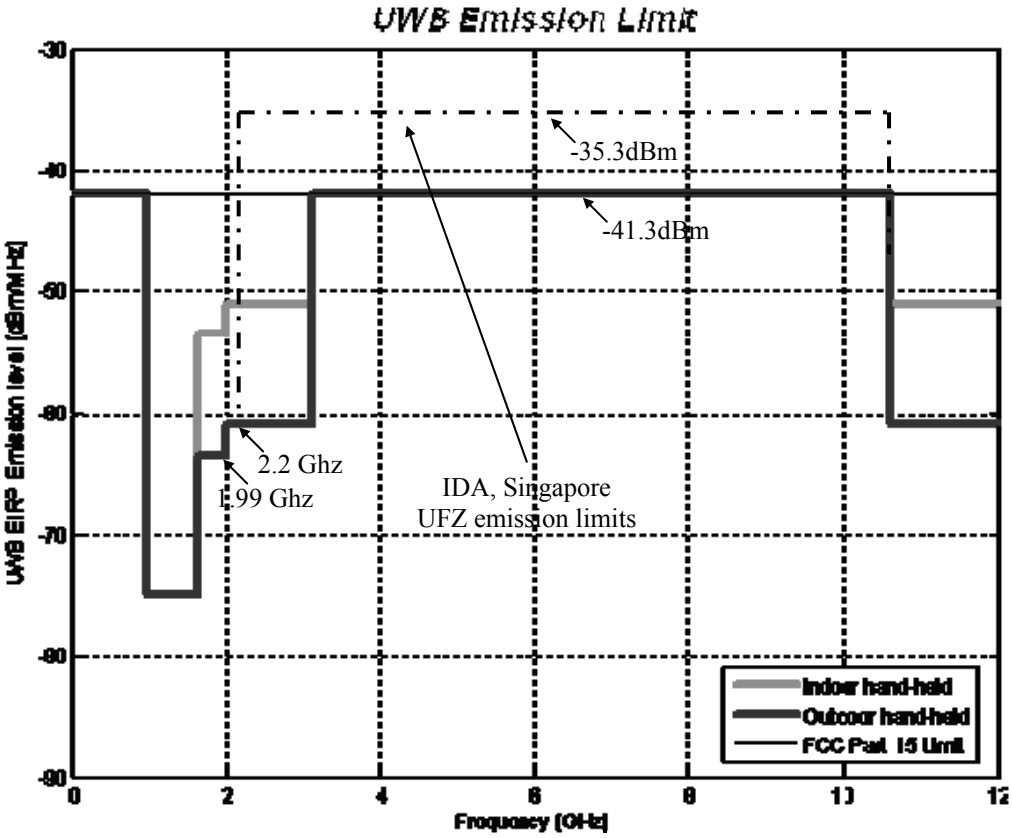


Fig. 1-1. UWB FCC spectral mask

There are several advantages of UWB technology namely, low complexity and low power transceiver architecture, potentially high data rate, and etc. Unlike conventional narrowband radio frequency (RF) transceiver, impulse radio UWB (IR-UWB) eliminates power hungry RF building blocks such as mixers and synthesizer. This low architecture complexity coupled with burst mode operation makes very low power consumption possible. In addition, the large bandwidth employed by the UWB transmission also facilitates high data rate transmission.

However, there are a few design challenges that need to be tackled in order to realize the above mentioned advantages for IR-UWB transceiver, which will be discussed next.

1.2 Motivation and Objective

Clock and data recovery (CDR) is required for all transceivers to recover the transmitted data correctly. For UWB transceiver, this is even more important as clock synchronization will enable burst mode operation to further minimize the power consumption. However, most of the UWB researches only center on the RF front-end implementation [3-5], that is responsible for recovering the transmitted pulse into either return-to-zero (RZ) or non-return-to-zero (NRZ) formats. They usually exclude the CDR function and leave it to the back-end processing. Conventionally, phase locked loop (PLL) or delay locked loop (DLL) which consists of mainly analog building blocks is usually employed for CDR. However, they are not amenable for integration with UWB transceiver as one of the main attractions for UWB transceiver being its simple architecture without any need of mixer and PLL. Another popular method involves the use of analog-to-digital converter (ADC) coupled with digital signal processing (DSP) algorithm to achieve clock synchronization and data recovery. Neither is this solution attractive due to the narrow pulse nature of IR UWB signals which might require very high sampling rate and high power ADC. These prompt us to look for an alternative simple way of timing and data recovery which works well with conventional IR-UWB RF front-end. The successful synchronization will overcome the clock drift issues between different transceivers and enable burst mode operation to achieve

low power operation. In addition, although UWB pulse is relatively easy to generate, due to process, voltage and temperature (PVT) variations, the resultant power spectrum might violate the specified FCC spectral mask. For mass production, automatic calibration or tuning is required to ensure high yield. This motivates us to look for various calibration techniques to ensure proper shaping and centering of the UWB power spectrum. Besides these key issues, other baseband functions, such as encoder, decoder, pseudorandom sequencer, BER tester and etc are also implemented in this project to enhance the transceiver performance.

1.3 Thesis Overview

This thesis primarily focuses on the design of the digital baseband with emphasis on the proposed CDR algorithm and calibration of the transmitting frequency and pulse width of the IR-UWB transmitter. Chapter 2 provides the literature reviews on some of the existing techniques for tackling both issues. Chapter 3 explains the proposed techniques in details. The overall digital baseband architecture will also be discussed together with the post layout simulation to validate the functionality of the proposed design. Experimental measurement and test settings will be discussed in chapter 4. Finally, conclusion and possible future improvement will be given in chapter 5.

CHAPTER 2

REVIEW of EXISTING TECHNIQUES

In this chapter, we will review some of the available existing techniques for both CDR as well as calibration. The better understanding on these techniques will lead us to a better solution, to overcome the issues, which is more amenable to the integration with IR-UWB transceiver.

2.1 Clock and Data Recovery

2.1.1 Phase lock loop (PLL)

Phase lock loop is one of the popular analog building blocks used in clock and data recovery applications [6-11]. A typical PLL, which is shown in Fig. 2-1, consists of a phase detector (PD), a charge pump (CP), a low-pass filter (LPF), and a voltage controlled oscillator (VCO).

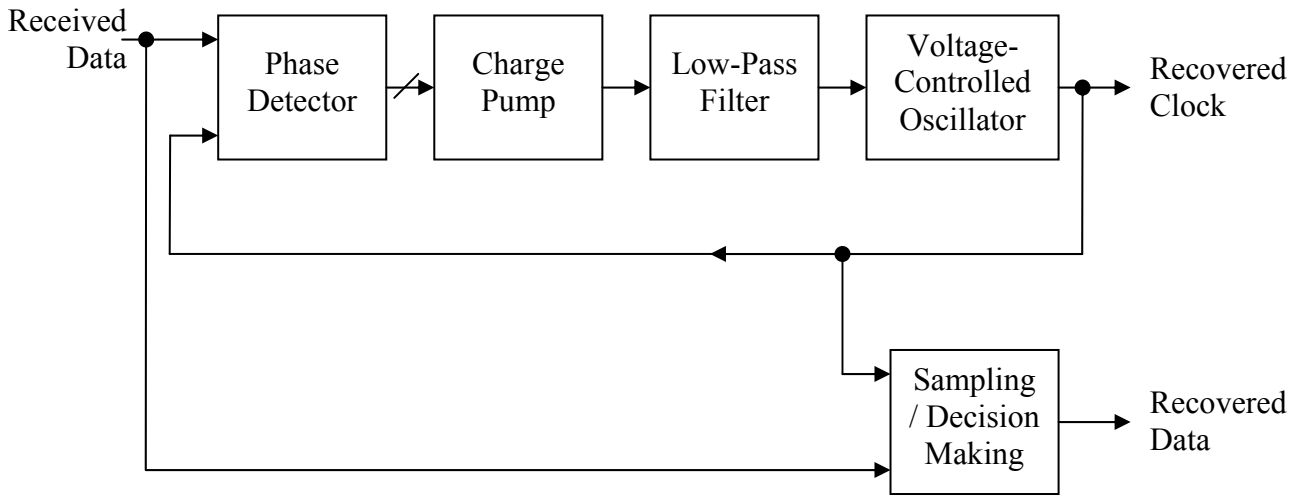


Fig. 2-1. Typical PLL Block Diagram

The incoming random data in return-to-zero (RZ) format will be sent to the phase detector and compared with the local VCO output. Any direct current

(DC) phase error will be extracted and smoothed out by the LPF to generate the desired control for the VCO to obtain the synchronized clock frequency. The synchronized clock can then be used to sample the incoming data at the maximum eye opening to obtain non-return-to-zero (NRZ) data.

There are a few issues associated with PLL solution for CDR. First of all, it is mainly an analog technique which not only increases the complexity of IR-UWB transceiver but also the overall power consumption. The loop filter, if integrated on chip, could incur significant area penalty. In addition, the analog building block, such as VCO and CP, are not easily scalable with complementary metal–oxide–semiconductor (CMOS) technology especially at low voltage supply used in deep sub-micron CMOS technology.

2.1.2 Delay lock loop (DLL)

Delay lock loop is another popular building block for CDR application and is shown in Fig. 2-2 and Fig. 2-3 [12-13]. The two figures mainly differ in the selection of delayed signals for phase comparison. In Fig. 2-2, the incoming data is compared with the delayed reference clock, whereas the reference clock is compared with delayed incoming data in Fig. 2-3. The architecture looks very similar to PLL and replaces the VCO in the PLL with the voltage controlled delay line (VCDL). The phase information is generated by the phase detector through a comparison between delayed reference clock and incoming data, or delayed incoming data and reference clock. Any DC phase error will be extracted and smoothed out by LPF to generate a controlled voltage. The controlled voltage will

then tune the delay cell in such a way that the delayed reference clock or delayed incoming data will now synchronize with the incoming data or reference clock. This synchronized clock can then be used to sample the incoming data at maximum eye opening. Compared to PLL, DLL achieves much lower timing jitter because the phase errors in the delay cells are not accumulated over time. However, it suffers from the same problem as PLL mentioned earlier. It should be mentioned that there exists digital delay cell which is implemented with simple inverter using binary controlled load and driving strength. This delay cell is easily scalable with deep sub-micron CMOS technology compared to the analog delay cell. However, fine timing resolution can only be achieved with advanced 90nm CMOS technology or smaller.

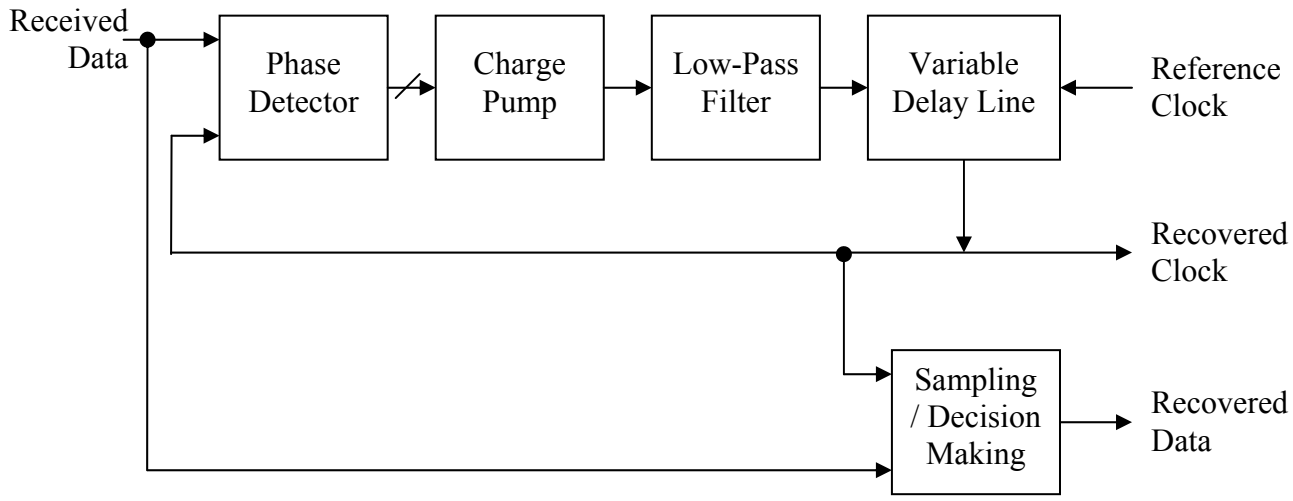


Fig. 2-2. Delayed Clock CDR Block Diagram

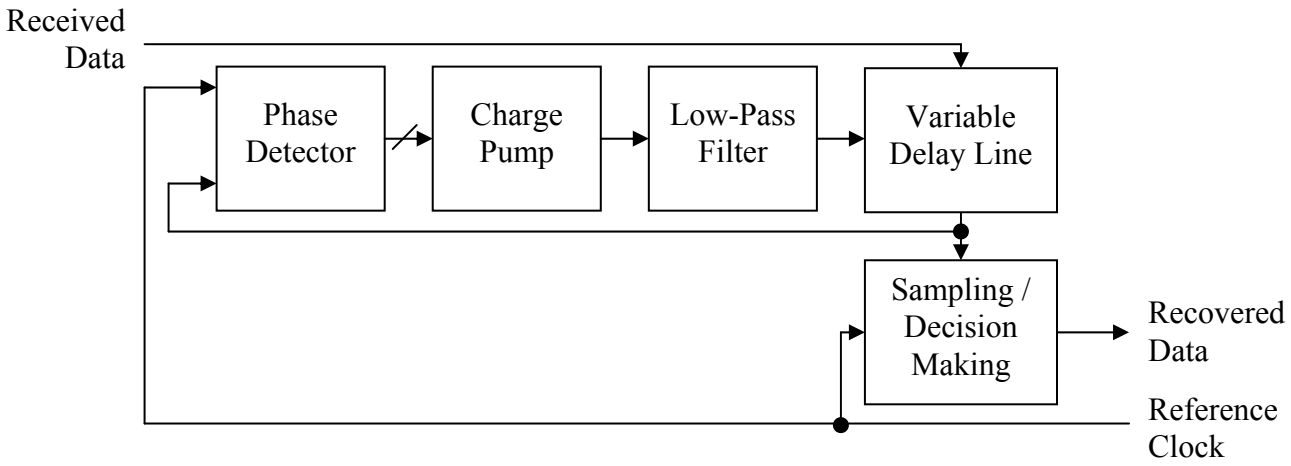


Fig. 2-3. Delayed Data CDR Block Diagram

2.1.3 Analog to Digital Converter (ADC)

Another popular timing recovery technique involves the use of ADC coupled with DSP algorithm. This architecture requires the partition of analog and digital boundary at the very early stage of the RF front-end, usually right after the low noise amplifier (LNA) and variable gain amplifier (VGA) as demonstrated in Fig. 2-4, thus allowing the backend signal processing to be done fully in digital domain. However, due to the narrow pulse width or wide bandwidth of the IR-UWB signal, the required sampling rate of the ADC (based on Nyquist criterion) will be in the range of Giga samples per second. The design and implementation of this ADC are very critical as the performance of this receiver will eventually be underscored by the capability of the ADC. In addition, complicated DSP algorithm is needed and will result in large digital circuitry and consume significant dynamic power.

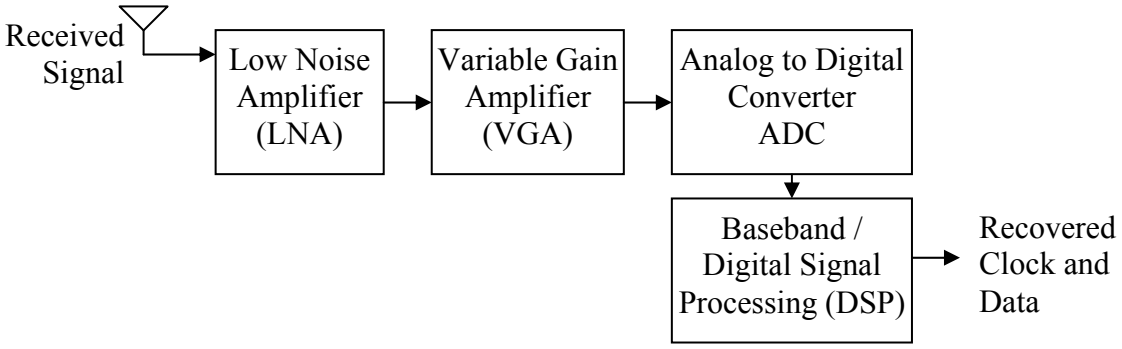


Fig. 2-4. Typical UWB Receiver Using ADC

To relax the requirement of high sampling rate ADC, time-interleaved architecture as shown in Fig. 2-5 can be used [14-15]. This architecture consists of a bank of parallel ADCs and a multi-phase clock generated through a delay chain. The sampling frequency of each ADC is relaxed by N times where N is the number of parallel ADC within the time-interleaved architecture. Although the power of ADC can be significantly reduced due to the lowering of the sampling rate, significant area penalty as well as matching issues will plague this implementation.

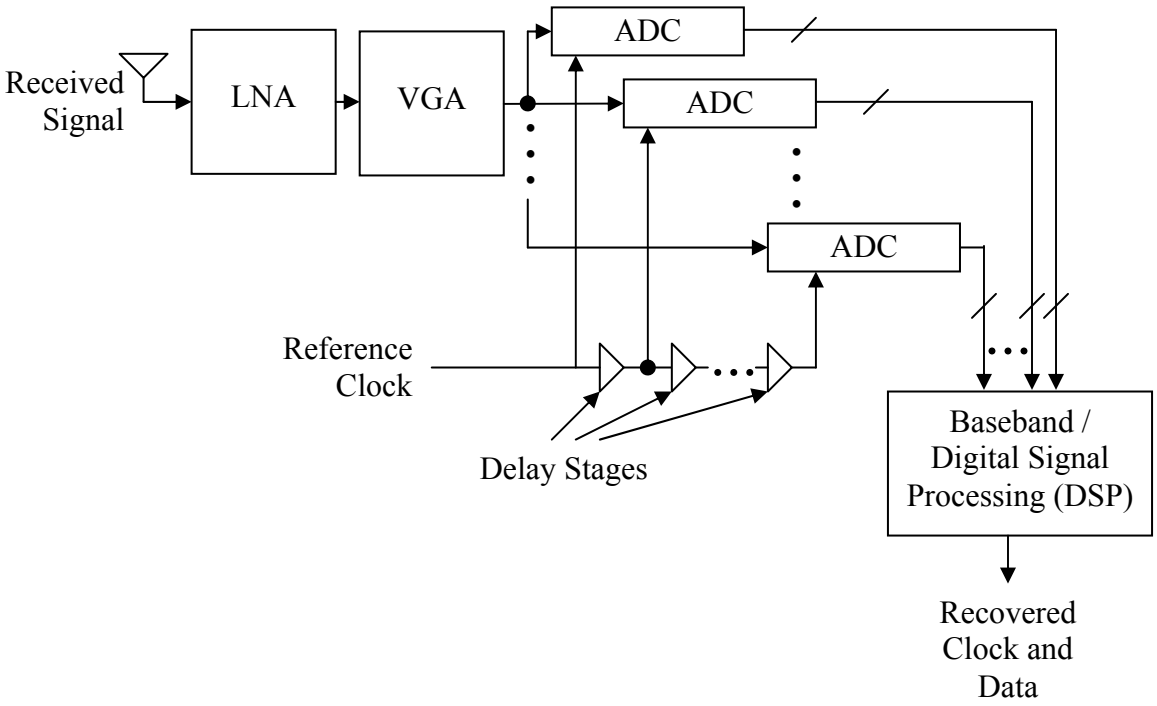


Fig. 2-5. Typical UWB Receiver Using Time-Interleaved ADC

2.2 Transmitter VCO Calibration

The transmitted UWB power spectrum should meet the regulated FCC spectral mask. For IR-UWB, this implies the accurate control of the carrier frequency within the short pulse as well as the pulse width. The carrier frequency will determine the spectrum shift whereas the pulse width will determine the shape of the UWB spectrum. Without tuning or calibration, the PVT variations will cause the shift in carrier frequency as well as pulse width. The resulting spectrum might therefore violate the specified FCC spectral mask and could not be used for UWB applications.

Some of the UWB transmitters use a precise VCO frequency calibration technique. This precise VCO frequency calibration technique involves the use of a PLL [16-17] as shown in Fig. 2-6. As the PLL cannot be turned on and off quickly, it usually remains on during the burst mode operation for the UWB pulse transmission. This technique therefore incurs additional area and power penalty.

As the UWB specification only specifies the desired frequency band and output power level without any specific details about the modulation and implementation, the VCO carrier frequency and UWB pulse width can be loosely defined as long as the resulting spectrum meets the FCC spectral mask. Therefore, offline calibration method can be employed between bursts, and the calibration accuracy requirement could also be relaxed so that accurate PLL techniques might not be needed.

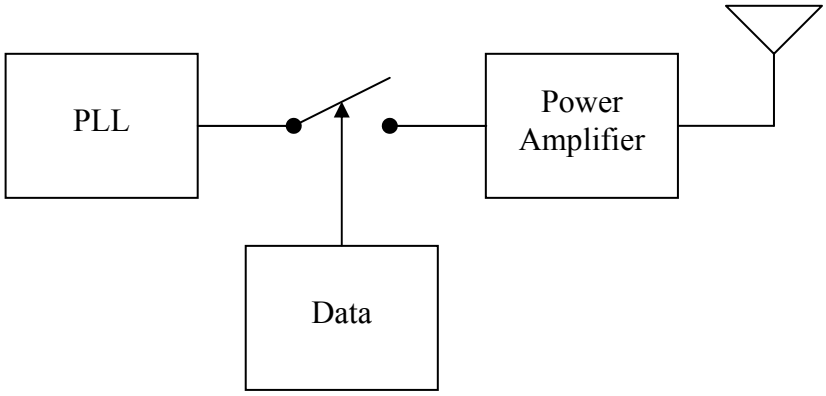


Fig. 2-6. Typical PLL Based VCO Frequency Calibration

2.3 Conclusion

In this chapter, we examine existing solutions for CDR applications and technique for overcoming PVT impacts on UWB spectrum. For CDR applications, both PLL and DLL do not scale well with technology and incur additional power as well as area penalty. ADC solution is too expensive for IR-UWB due to the wideband/narrow pulse nature of IR-UWB signal. For calibration, due to the loosely defined UWB specification, calibration accuracy can be relaxed and simple calibration without PLL might be feasible.

CHAPTER 3

PROPOSED DIGITAL BASEBAND ARCHITECTURE

3.1 Design Considerations

As mentioned in the earlier chapters, this work mainly focuses on novel techniques that can provide CDR as well as calibration for the UWB transmitter. The proposed techniques are purely digital implementation so that it can integrate well with IR-UWB RF front-end without incurring too much power and area penalty. The implemented digital baseband will be combined with the IR-UWB front-end developed in Institute of Microelectronics (IME) [5], [18]. On-off keying (OOK) modulation scheme is employed for the IR-UWB transceiver. The digital baseband will receive the recovered rail-to-rail RZ data from the IR-UWB RF front-end.

The proposed digital baseband will act as an interfacing circuitry between the RF front-end and the micro-controller unit (MCU). For transmission, MCU will assemble the medium access control (MAC) layer data and send to the digital baseband for transmission. The digital baseband will then create the transmitted data in the desired format required by the UWB transmitter. Similarly, when RZ data is received by the RF front-end, the digital baseband will demodulate the data into NRZ format and recovered the synchronized data clock before sending it back to MCU for packet disassembling. In addition, MCU can issue commands to the digital baseband to control the different RF front-end setting to achieve the optimum RF transceiver performance.

The full digital baseband is shown in Fig. 3-1. It consists of the functional blocks such as clock and data recovery, VCO calibration, pulse width calibration, encoder, decoder/correlator, pseudorandom number (PN) sequence generator, BER estimator and serial peripheral interface (SPI).

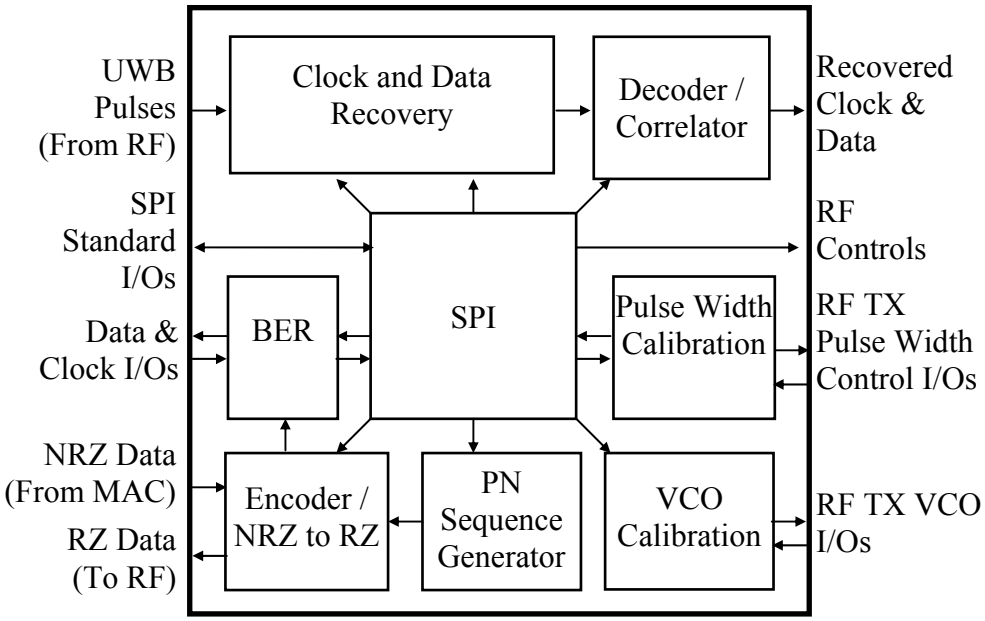


Fig. 3-1. Digital baseband block diagram

A SPI bus is employed to interface with the MCU for all the data and control signals communication. This will minimize the valuable I/O pin resources required by the digital baseband. To enhance the sensitivity of the transceiver, additional encoder and decoder/correlator blocks are also included. The encoder will encode the transmitted bit sequence with either Barker code or Gold code before sending it to the transmitter. This can improve the sensitivity of the transceiver by providing additional coding gain if needed. However, it will lower the overall throughput of the transceiver due to the additional coding bits.

The PN sequence generator is included for testing and debugging purpose. It can generate random bit sequence with length of 2^9-1 based on linear feedback shift register (LFSR). The generated bit sequence can either serve as the input signal to the transmitter for transmitter testing, or input signal to the receiver path within the digital baseband for receiver testing.

The BER estimator is included for the optimization of the RF transceiver settings. During the calibration mode, the digital baseband will cycle through the different RF transceiver settings and obtain the corresponding BER for each setting. The setting which gives rise to the lowest BER will be chosen as the optimum settings for the RF transceiver after calibration.

The CDR will generate the desired synchronized data clock and convert the received RZ data to NRZ format. Both the pulse width and VCO calibration will tune the UWB transmitter so that the overall transmitting spectrum will meet the FCC mask requirement.

In the following sections, various blocks will be discussed in detailed with particular emphasis on CDR and calibration based on novel algorithms.

3.2 Clock and Data Recovery Design

The clock and data recovery function is designed with two assumptions being made. Firstly, the digital baseband requires a pilot signal consists of 16 consecutive transmitted 1s at the start of each data packet for the identification of the received UWB pulse location. Secondly, due to the chosen OOK modulation, the continuous tracking of pulse location can only occur for received 1s. Therefore, long consecutive received 0s are prohibited in the transmitted sequence. This should not pose any severe problem as a well designed MAC protocol should ensure sufficient randomization of the transmitted bit sequence and avoid the above mentioned scenario.

The clock and data recovery design is separated into two parts namely the pulse searcher and the pulse tracker. The pulse searcher and pulse tracker are used together to mimic the function of clock and data recovery. Given a sequence of pilot signals, the pulse searcher will identify the location of the received UWB pulse within a specific time frame. However, due to the clock drift issue between the transmitter and receiver, this identified location will also drift with time and become inaccurate. To circumvent this problem, the pulse tracker module will be invoked once the pulse searcher has successfully identified the received UWB pulse location. It will continuously track the location of the UWB pulse even in the presence of clock drift. It is also responsible for recovering the received data in NRZ format and the corresponding data clock. The period and duty cycle of the data clock will be adjusted continuously due to the presence of clock drift. In addition, after synchronization, the pulse tracker module will also generate the

enable signal for the RF transceiver's burst mode operation. The design of the pulse searcher and pulse tracker will be explained in detail in the following sections. The design of the whole CDR is implemented using Verilog hardware description language (HDL) and post layout simulation will also be shown.

3.2.1 Pulse Searcher Design

This UWB pulse searcher consists of four D-flip-flops detectors, a sampling controller, ten negative edge-triggered 4-bits registers (NR0~NR9), ten positive edge-triggered 4-bits registers (PR0~PR9) and a decision finite state machine (FSM) as shown in Fig. 3-2. The UWB pulses are being detected using the D-flip-flop detector as shown in Fig. 3-3, instead of using ADC.

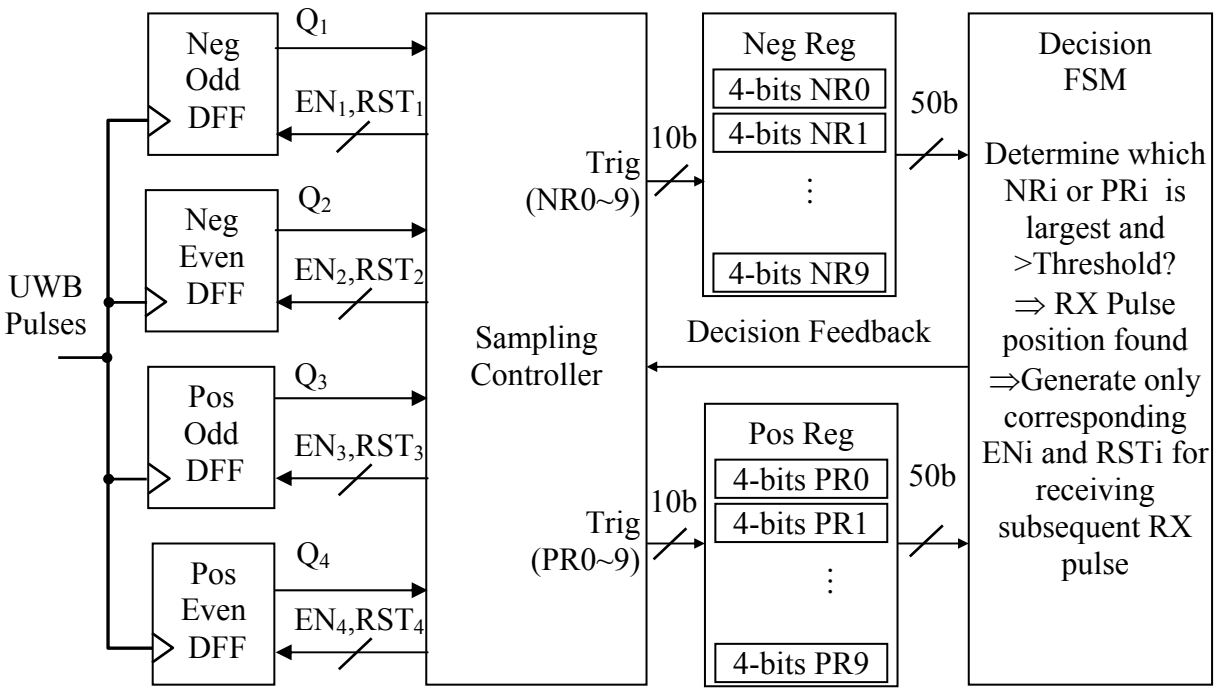


Fig. 3-2. UWB pulses searching block diagram

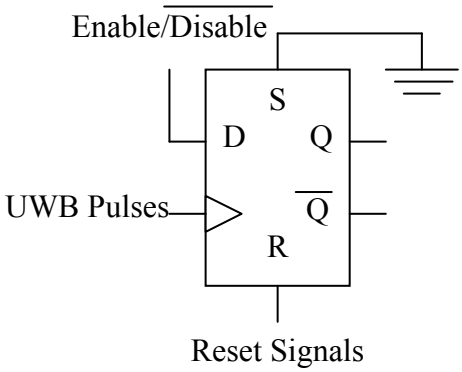


Fig. 3-3. D-flip-flop detector

For any given data rate, the system clock is first set to 10 times of the data rate through an internal frequency divider. Each symbol period is then divided into 10 smaller duration windows by the sampling controller, i.e. one duration window (T_{dur}) is equivalent to one system clock interval (T_{clk}), and window is being tracked by either positive counter or negative counter as shown in Fig. 3-4. In addition, each duration window will have its corresponding 4-bits register to store the number of UWB pulse detected during that duration window. As an example, NR1 would be the 4-bits register corresponds to the duration window when the negative counter equals to 1. The sampling controller will be responsible for generating the control signals (ENi and RSTi) for each of the D-flip-flops as shown in Fig. 3-4.

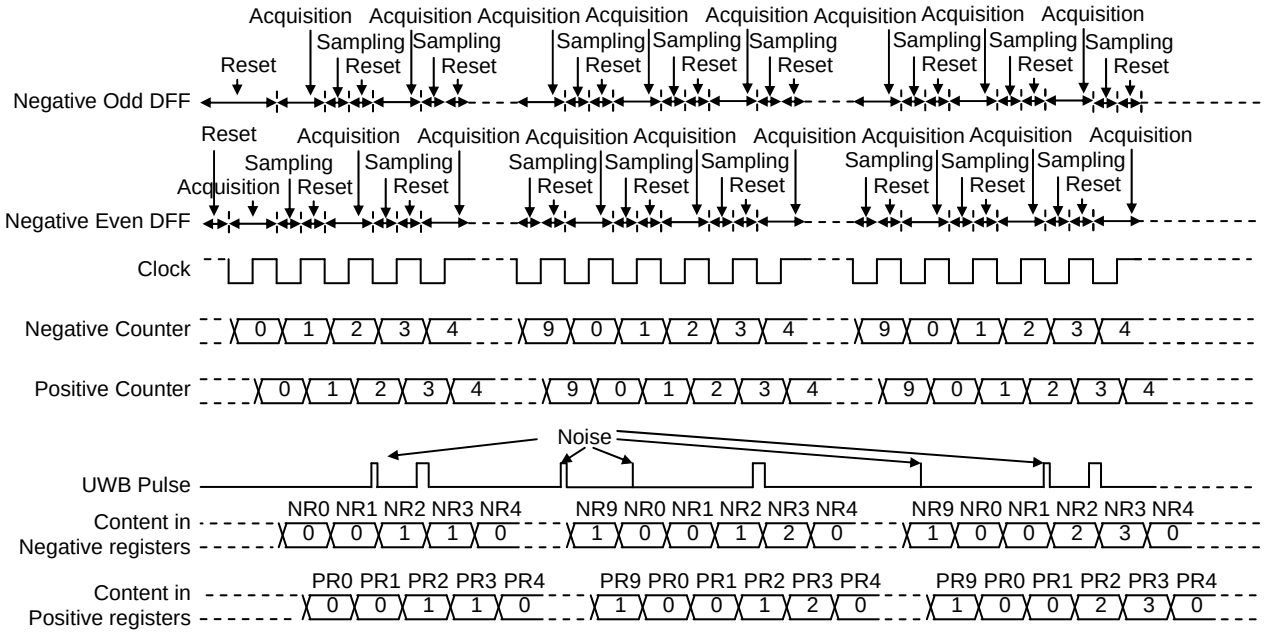


Fig. 3-4. UWB pulse searching timing diagram

There are three operation phases for the D-flip-flop detector. The D-flip-flop is first reset during the reset phase (RST=1). After which, the D-flip-flop will enter acquisition phase (EN=1/RST=0) to detect any incoming UWB pulse for one full duration window. Its output (Q) will then be determined during the sampling phase by the controller. A successful detection of 1s will trigger the count of the corresponding 4-bits register (NRi or PRi) through Trig. As the three phases of operation span two duration windows ($2\times T_{clk}$), both odd and even D-flip-flops are needed to cover the whole 10 duration windows alternatively. The sampling controller will generate the proper ENi and RSTi for the 4 flip-flop detectors as well as the Trig signals for NRi and PRi based on the Q outputs from the flip-flop detectors. At the end of the 16 pilot symbols of transmitted 1s, both the negative and positive edge-triggered registers (NRi and PRi) will have the

statistics which indicate the most likely duration window of the incoming UWB pulses. The decision FSM will evaluate the collected data from the registers and determine the duration window where the incoming UWB pulses occur. Its decision will be feedback to the sampling controller for the proper selection of D-flip-flop detector as well as control signals for the subsequent detection of incoming UWB pulses. As illustrated by the example shown in Fig. 3-4, the incoming UWB pulses occur at duration window corresponding to NR3 and PR3, which have the highest number of count. During the detection, noise and glitches could randomly increase the register counts, such as NR2/PR2 and NR9/PR9. By setting proper threshold in the decision FSM (NRi or PRi $\geq$ 12), we should be able to mask these detection errors and find the correct duration window for the incoming UWB pulses. The setting of threshold depends on two factors, i.e. the receiver sensitivity and the channel condition. Poorer channel condition might result in a lot of detection errors and require higher threshold to mask these detection errors. On the other hand, poorer sensitivity with good channel condition might require lower threshold to enable the detection of the desired UWB received pulses. From measurement, the proposed algorithm performs satisfactorily with a threshold of more than or equals to 12.

Most of the time, positive and negative edge-triggered registers will record the same number of UWB pulse detection. Therefore, either one can be chosen for subsequent UWB pulse detection. In the proposed algorithm, we choose the duration window corresponding to NRi if NRi=PRi. However, if the incoming pulse is occurring right at the transition edge, the corresponding edge-triggered

registers might fail to detect the pulses due to meta-stability issue. Therefore, both positive and negative edge-triggered flip-flop and registers are included to overcome the problem. If positive(negative) edge-triggered flip-flop encounters the meta-stability issues and fails to detect the incoming signal, the negative(positive) edge-triggered flip-flop should still be able to detect the incoming pulse and results in NRi>PRi (or PRi>NRi). The flowchart for the pulse detect algorithm is also illustrated in Fig. 3-5.

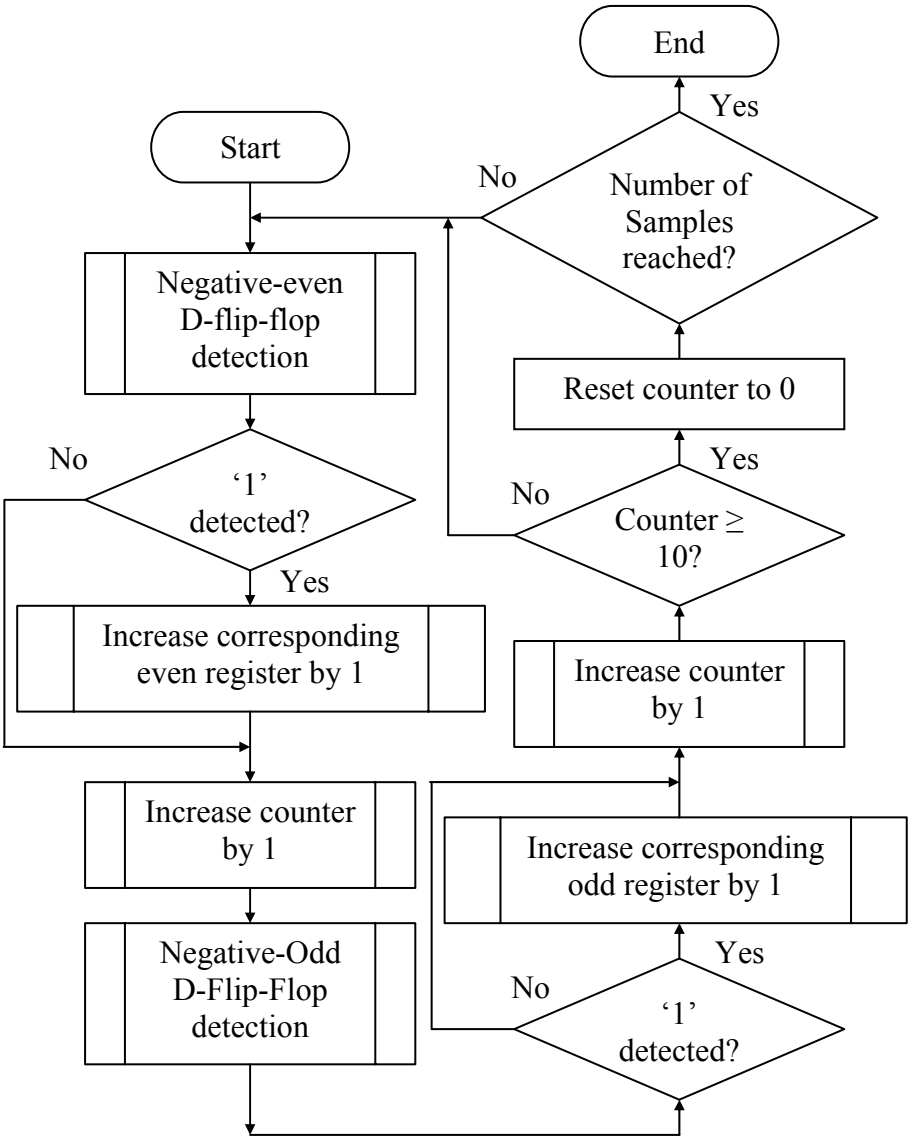


Fig. 3-5. UWB pulse detect algorithm flowchart

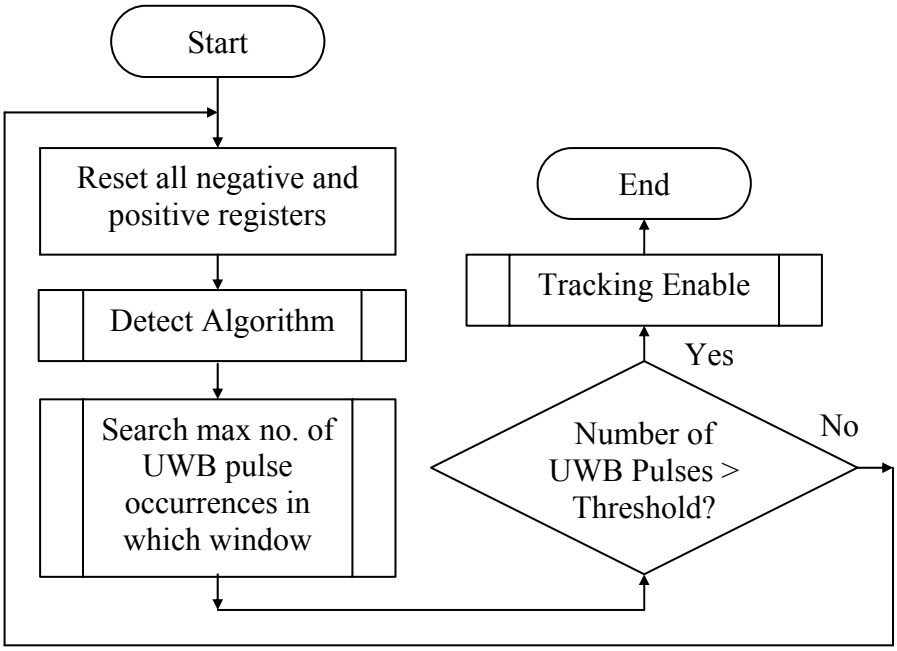


Fig. 3-6. UWB pulse searcher flowchart

Figure 3-6 shows the complete pulse searching algorithm flowchart. At the beginning detection of each data packet, the count for both the positive and negative edge-triggered registers are reset to zeros. After which, the UWB pulse detection algorithm discussed earlier will be invoked. At the end of the 16 pilot symbols, the FSM will determine the duration window with the maximum number of detected UWB pulses. If the number also exceeds the preset threshold (>12), the whole pulse searching algorithm completes and the pulse tracker will be invoked. Otherwise, the whole searching algorithm will be repeated. It should be pointed out that once the pulse tracker is invoked, the sampling controller will only produce ENi and RSTi corresponding to the duration window that UWB pulses occur. As an example, once NR3 is determined to be the correct duration window from previous discussion, only ENi and RSTi correspond to that duration window and its adjacent windows will be generated by the sampling controller.

3.2.2 Pulse Tracker Design

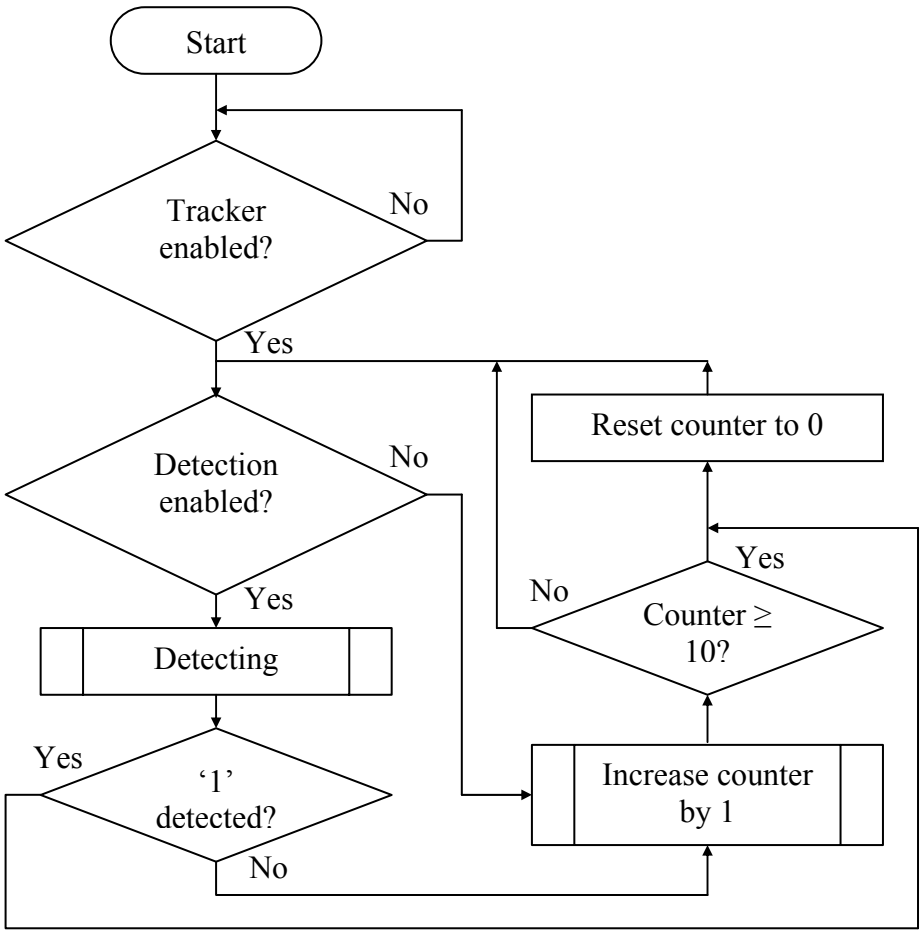


Fig. 3-7. UWB pulses tracker flowchart

Once the UWB pulse searching algorithm completes, the exact duration window where UWB pulses occur will be known. The UWB pulse tracking algorithm as shown in Fig. 3-7 will begin. The pulse detection will span three consecutive duration windows with the center duration window obtained from the pulse searching algorithm. A tracking counter will be used to track the UWB pulse. As the tracking algorithm only corrects the drift upon the detection of 1s, slow clock drift coupled with short consecutive transmission of 0s are needed to

ensure the functioning of the algorithm. This is a reasonable assumption given that typical crystal oscillator exhibits ±50ppm frequency stability. Assuming a worst case scenario of 100ppm between the RX and TX system clocks, this will give rise to a maximum frequency difference of 10kHz for system clock of 100MHz. For the algorithm to fail, the RX and TX system clocks have to be out of synchronization by more than one system clock interval (10 ns) between the detection of 1s. This corresponds to the transmission of 1000 consecutive 0s at 10Mbps (100 μs), which is unlikely to happen for a well designed MAC.

When the tracking counter value is zero, it indicates the position of the duration window where the UWB pulse should occur. Under normal condition with no pulse detected, the tracking counter would repetitively count to 10 cycles and reset, which matches the symbol period exactly. The algorithm will reset the tracking counter to zero once the UWB pulse is detected. If the position of the UWB pulse does not change over time, it will get detected when the tracking counter value is zero. So it would not incur any change on the tracking counter as well as the duration window. However, if the clock drift causes the UWB pulse position to change to the adjacent duration window, the received UWB pulse will reset the tracking counter to zero instantaneously and the position of the center duration window will now be updated. The detailed timing diagram of the pulse tracking algorithm is shown in Fig. 3-8. As illustrated, the initial UWB pulse occurs when the negative counter equals to 3, where the tracking counter is set to zero. Once the UWB pulse drift to the duration window where the negative counter equals to 2, the tracking counter will be reset right away to reflect the

changes. The corresponding NRZ data and data clock can then be easily generated based on the tracking counter and the detecting D-flip-flop output.

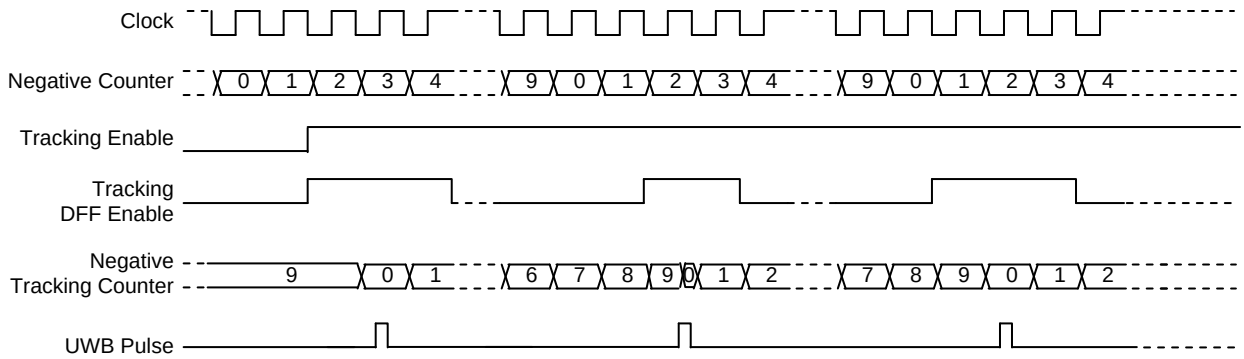


Fig. 3-8. UWB pulse tracker timing diagram

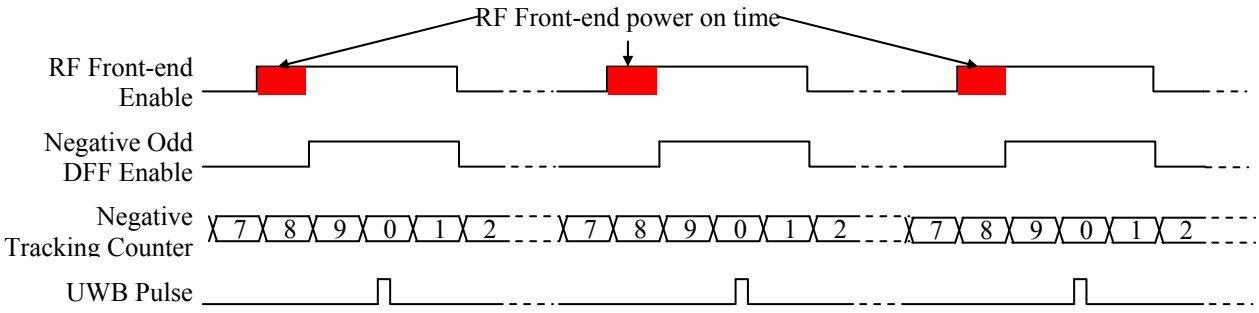


Fig. 3-9. RF receiver front-end enable timing diagram

Through the pulse searcher and the pulse tracker algorithm, the digital baseband is able to determine the duration window when the incoming UWB pulses occur. This information could then be used to turn on RF front-end periodically to conserve power. As illustrated in Fig. 3-9, when the digital baseband determines that the UWB incoming pulses occur during the window spanning three duration windows (Negative tracking counter = 9, 0, 1), an Enable signal will be sent to the RF front-end one cycle earlier to allow some turn-on

time for the RF front-end. For the rest of the duration windows, RF front-end can then be turned off to conserve energy.

3.2.3 Post Layout Simulation

The post-layout simulation of the clock and data recovery function is shown in Fig. 3-10. A test bench, is created, which incorporates two digital basebands to mimic the transmitting-receiving scenario. Unfortunately, the digital simulation does not allow different system clocks for both transmitter and receiver to model the clock drift issue. This will give rise to meta-stability issues and cause the simulator to display undefined states. However, in our proposed algorithm, we have specially employed both positive-edge and negative-edge counters to circumvent the meta-stability issue as mentioned earlier. The actual measurement shows the feasibility of the idea. Nevertheless, the post-layout simulations are still useful to verify the full function of the pulse searcher and partial function of the pulse tracker.

.

As shown in Fig. 3-10, the received data from RF front-end is in RZ format. Through the digital baseband, both the clock and data are being recovered. This validates the feasibility of the proposed algorithms.

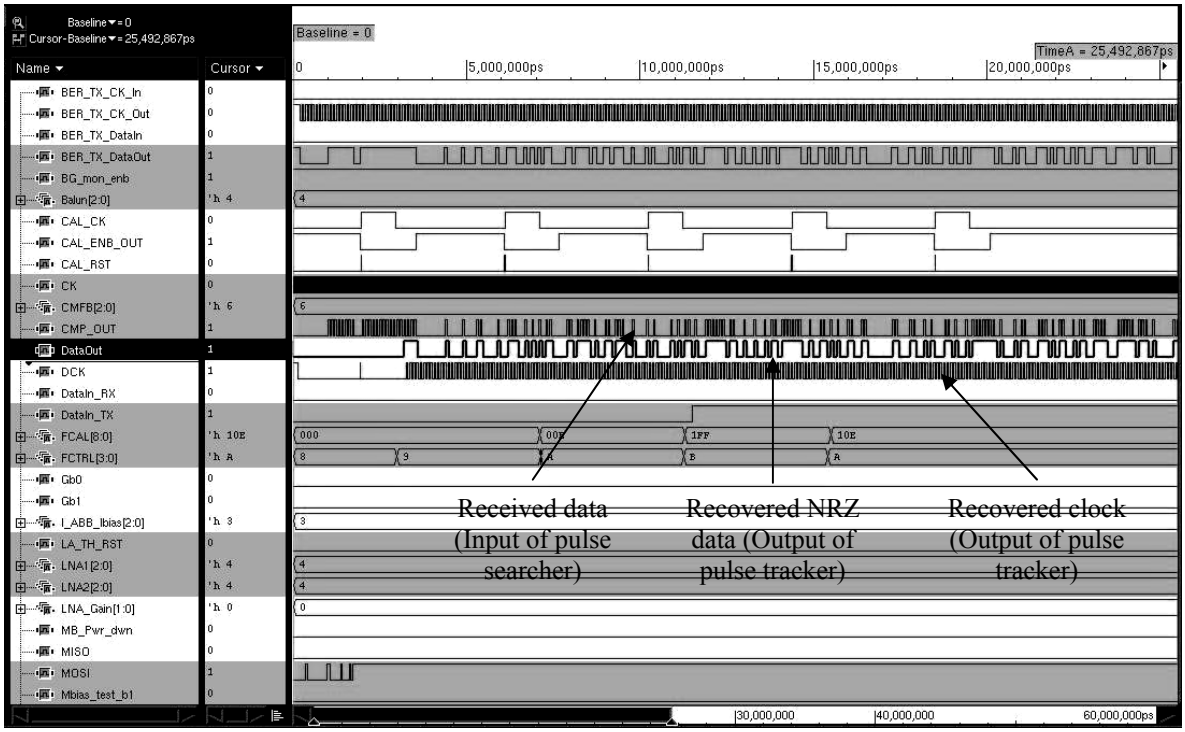


Fig. 3-10. Clock and data recovery post-layout simulation waveforms

3.3 Voltage-Controlled Oscillator Calibration Design

The VCO calibration will work with the transmitter block in the RF front-end to calibrate the centre frequency of the transmitted UWB pulse. The frequency of the VCO is first being divided down by current mode logic (CML) D-Flip-Flops to about 1 GHz. Dynamic true single phase clock (TSPC) D-Flip-Flops is then used to further divide down the frequency of the VCO. This divided down VCO output is then used to trigger a counter.

The flowchart in Fig. 3-11 shows the VCO calibration algorithm. Once the VCO calibration cycle is enabled, the digital baseband will reset the counter within the RF transmitter block and kick start the frequency division. The counter is allowed to count for 1μs which is accurately set by the system clock. At the end of the 1μs interval, the counter is disabled and the output of the counter is read by the digital baseband. The resulting counter output is then compared with predefined thresholds. If the counter output falls within the predefined thresholds, the whole VCO calibration cycle is completed. Otherwise the digital baseband will modify the frequency of the VCO by controlling the binary switched capacitor bank within the RF transmitter block according to the counter output. The VCO frequency will be increased/decreased if the counter output is smaller/larger than the predefined thresholds.

The proposed calibration method is off line calibration which can only be applied when the transmitter is not in used. This does not pose a severe problem as the UWB transmission is usually operating at burst mode to minimize the

power consumption. In addition, the calibration accuracy will be limited by the binary switched capacitor bank as well as the counter resolution. However, due to the loosely defined specification for UWB transmission, accurate tuning of centre frequency is not needed and the proposed calibration technique will function satisfactorily.

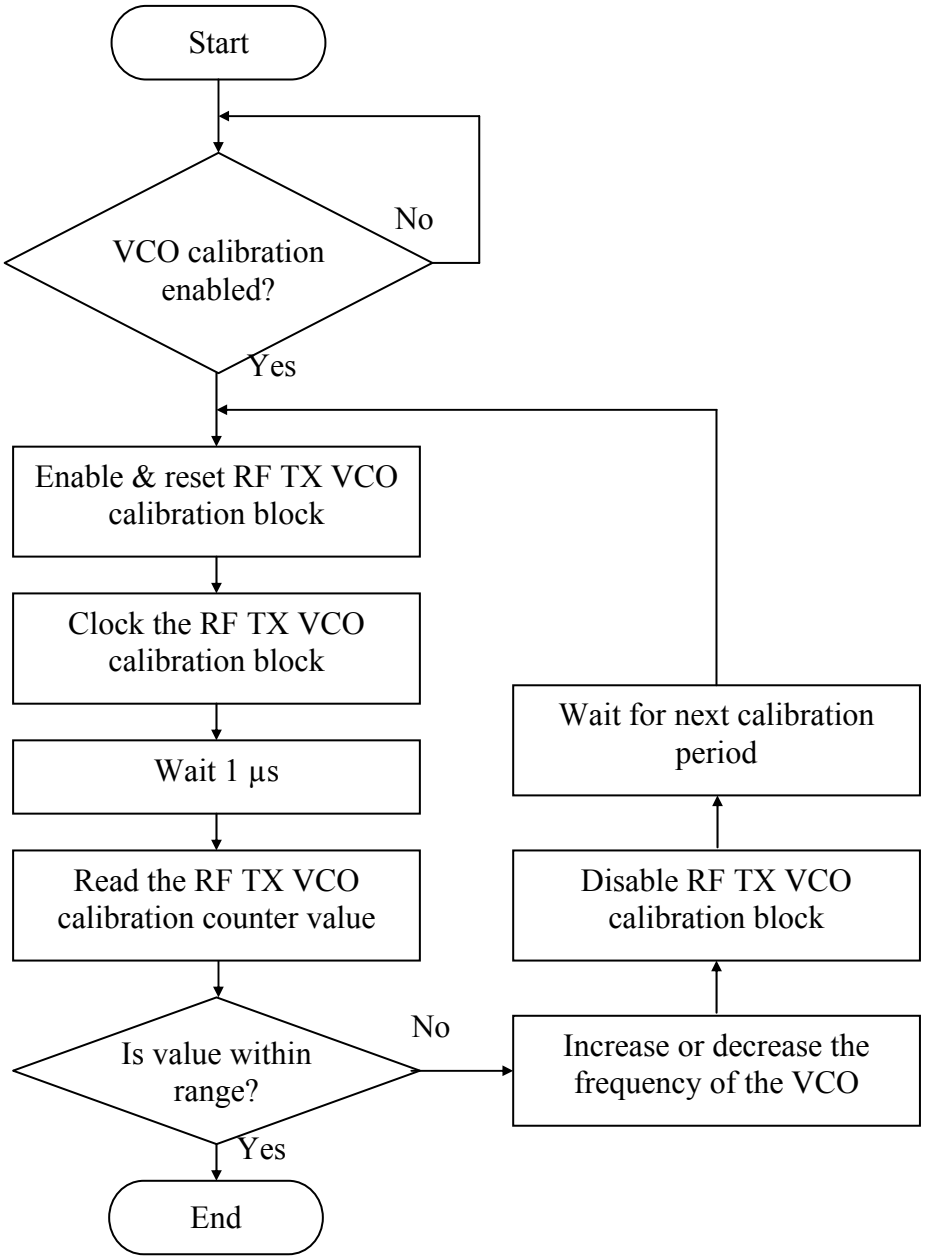


Fig. 3-11. VCO frequency calibration flowchart

The post-layout simulation is shown in Fig. 3-12. The VCO calibration cycle is first enabled through the SPI interface. The algorithm will then reset the counter in the RF transmitter. Following the reset, the counter will start counting for 1 μs interval based on the divided down VCO output. At the end of 1μs interval, the counter value within the RF counter will be read into the digital baseband. As illustrated in Fig. 3-12, the read counter value changed from 000_{16}, $00E_{16}$, $1FF_{16}$ to $10E_{16}$ during the calibration cycle. At the same time, the algorithm also output the control signal to the transmitter to adjust the frequency (8_{16}, 9_{16}, A_{16} to B_{16}) corresponding to the read counter value.

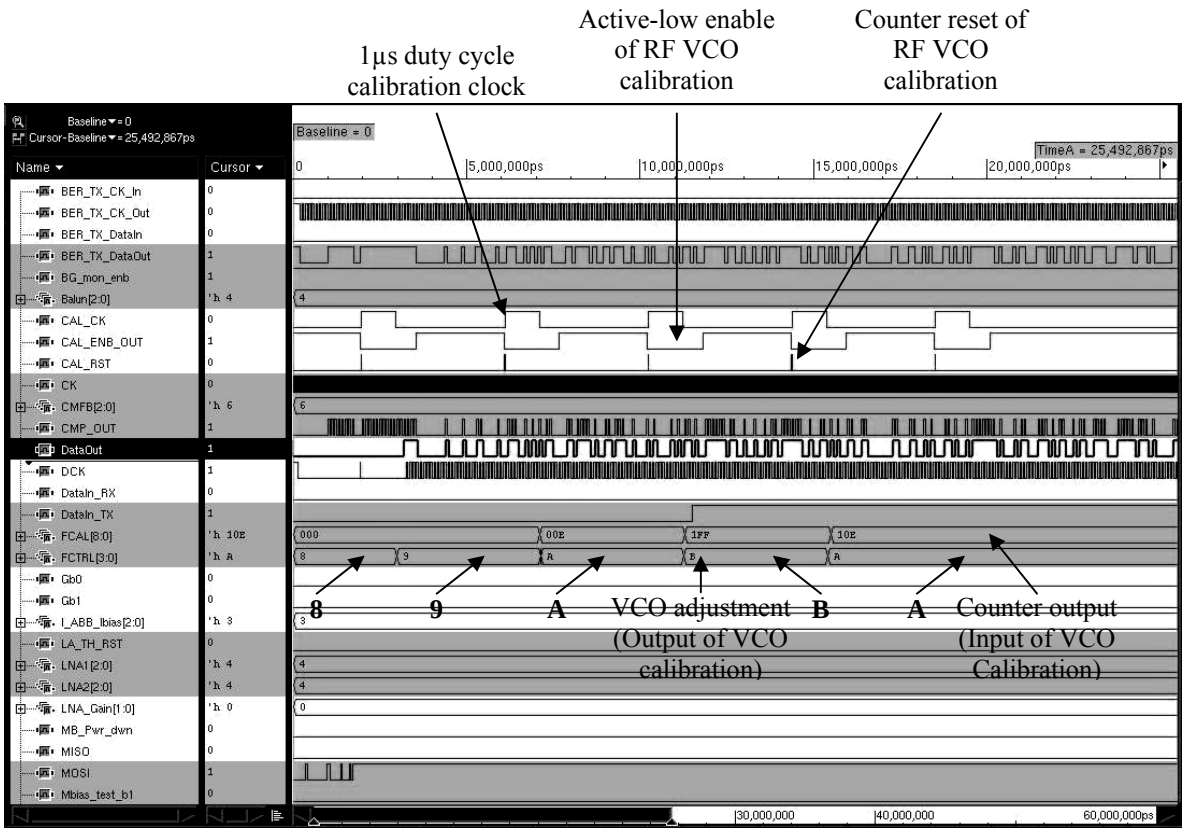


Fig. 3-12. Voltage-controlled oscillator calibration post-layout simulation waveforms

3.4 Pulse Width Calibration Design

The pulse width calibration will also work with the transmitter block in the RF front-end to calibrate the pulse width of the transmitted UWB pulse. CML shift register within the transmitter block is used to detect the pulse as shown in the Fig. 3-13. The shift-registers are being clocked by the rising edge of the transmitter UWB pulse.

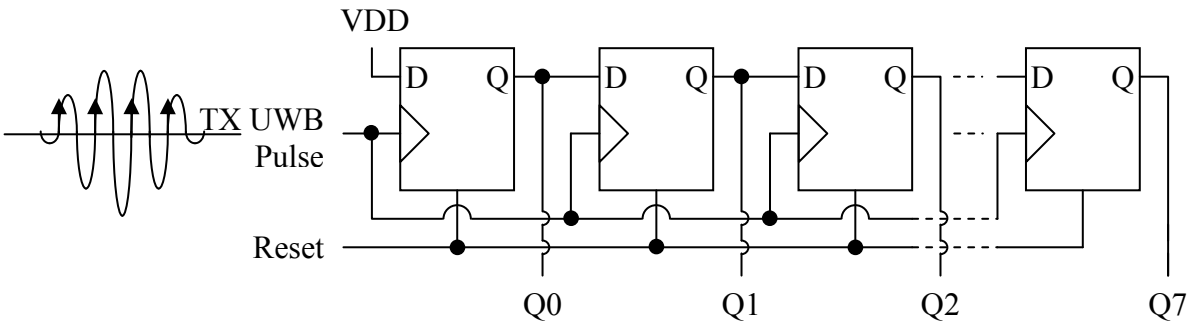


Fig. 3-13. Current mode logic shift register

The flowchart in Fig. 3-14 illustrates the pulse width calibration algorithm. Once the pulse width calibration is enabled, the digital baseband will reset the CML shift register within the RF transmitter block. The digital baseband will then signal the RF transmitter to transmit a UWB pulse. The UWB pulse will trigger the shift-register according to the number of significant peaks within the UWB pulse. This number will be recorded down by the shift-register output and read by the digital baseband. The value is then compared with the predefined thresholds. If the shift register output value is within the threshold, the pulse width calibration cycle is completed. Otherwise, the digital baseband will increase/decrease the pulse width if the shift register output value is smaller/larger than the threshold.

Similarly, off line calibration is adopted for pulse width calibration. Narrow pulse width will lead to wide bandwidth spectrum whereas wider pulse width will lead to narrow bandwidth spectrum. Accurate pulse width control is not needed as long as the resulting spectrum meets the FCC mask specification.

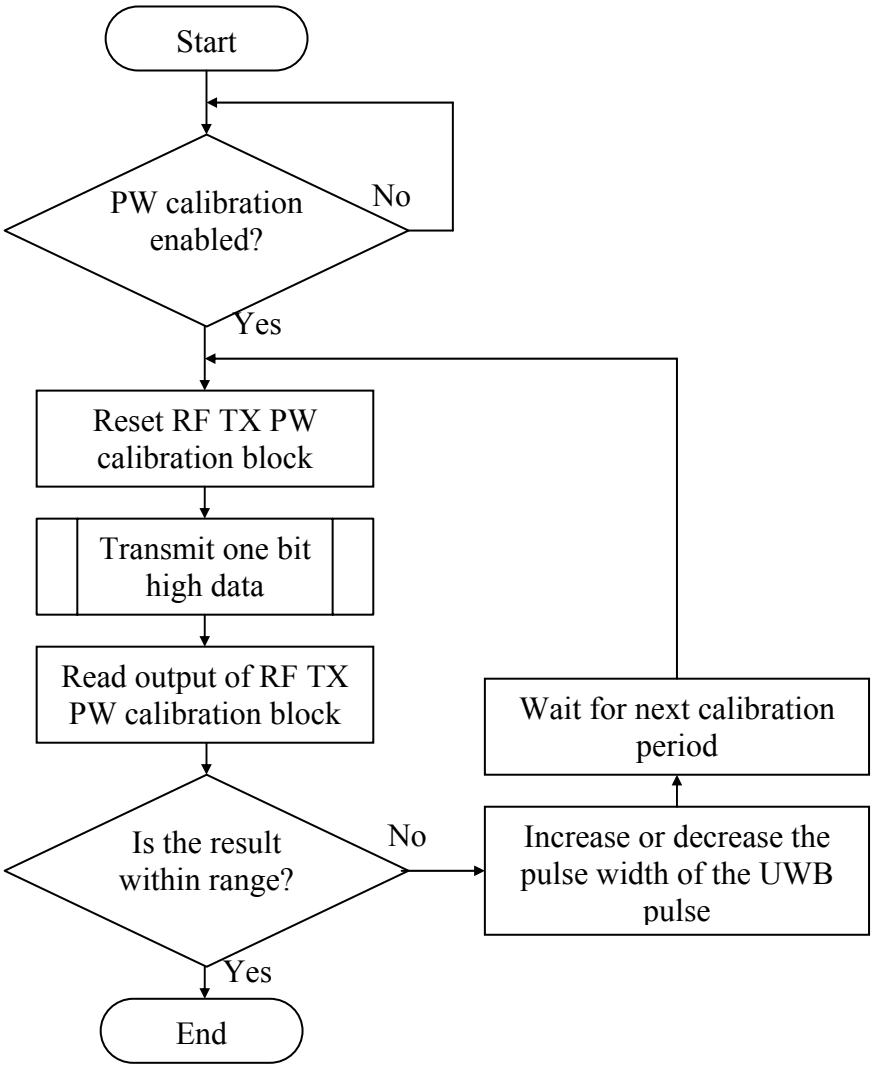


Fig. 3-14. Pulse width calibration flowchart

The post-layout simulation is shown in Fig. 3-15. The pulse width calibration is enabled through the SPI interface. The algorithm then signals the

transmitter to transmit a UWB pulse through TX_DataIn. At the end of the transmission, the shift register output, o, will be read into the digital baseband and the pulse width is then adjusted accordingly.

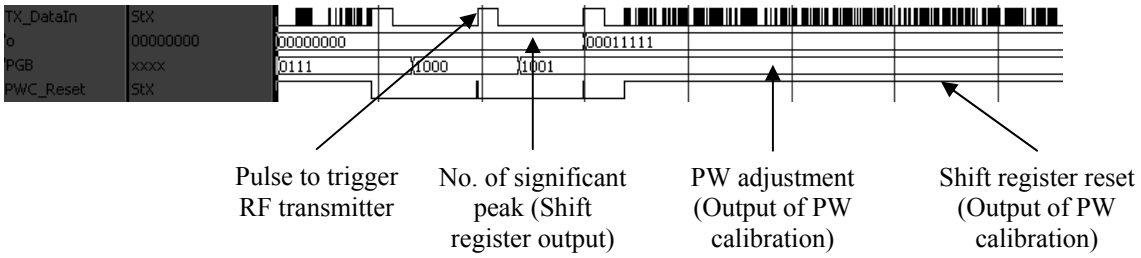


Fig. 3-15. Pulse width calibration post-layout simulation waveforms

3.5 Encoder Design

The encoder is used to convert the NRZ data from the MCU to RZ data for the RF transmitter and to enhance the sensitivity of the transceiver. The conversion of the NRZ data to the RZ data is achieved by simply logically AND the NRZ data with the data clock. The encoder can also encode the transmitted bit with Barker code or Gold code before transmitting. This can improve the sensitivity of the transceiver by providing additional coding gain. However, it will lower the overall throughput of the transceiver due to the additional coding bits.

The encoder contains three different coding scheme, namely the 11-bits Barker code, the 13-bits Barker code and the 31-bits Gold code. For the 11-bits and 13-bits Barker codes, the registers are preloaded with the standard codes, 11'b10110111000 and 13'b0101001100000 respectively. For the 31-bits Gold code, two polynomials are implemented using linear feedback shift registers

(LFSRs). One of the polynomials used is $z_1^5 + z_1^2 + 1$ with an initial state of [0 0 0 0 1] in the shift register. The other polynomial used is $z_2^5 + z_2^4 + z_2^3 + z_2^2 + 1$ with an initial state of [0 0 0 1 0] in the shift register. Both the outputs of the LFSRs are then logically XORed as shown in Fig. 3-16. The code generated by the XOR gate is 31'b1000100111111000010100111100011.

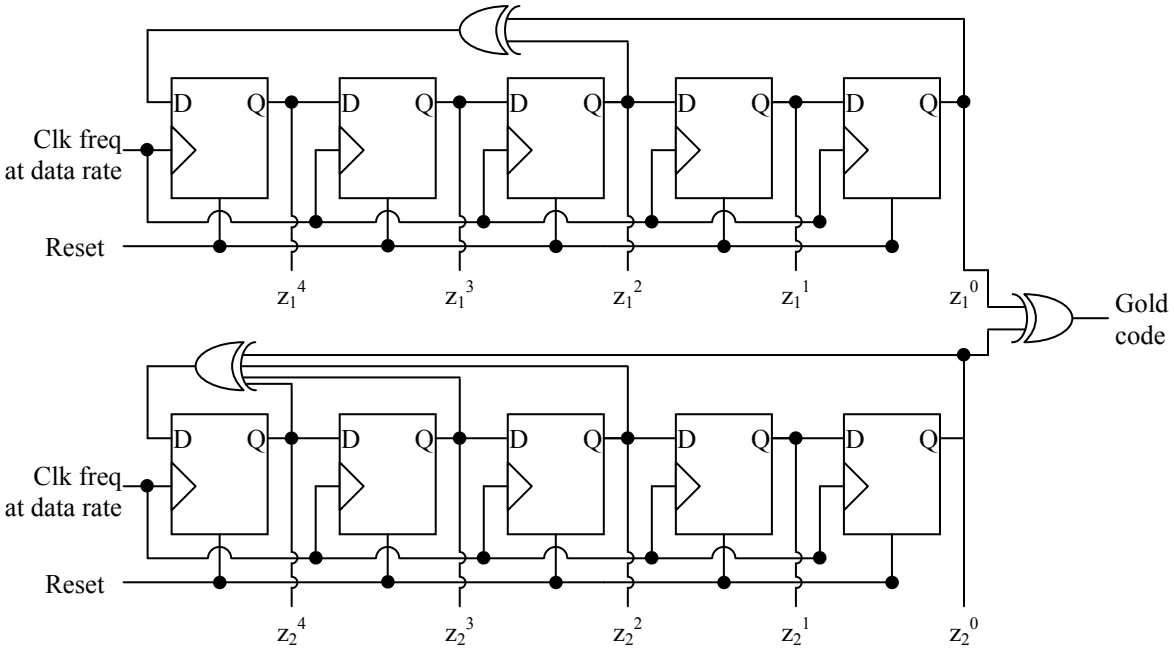


Fig. 3-16. Gold code generator

The post-layout simulation will be shown together with the decoding functions in the later section. Besides the post-layout simulation, MATLAB Simulink simulations are also studied to compare the performance between the 13-bits Barker code and the 31-bits Gold code. The received data stream is corrupted by a PN sequence generator to introduce bit error rate ranging from 0% to 10%.

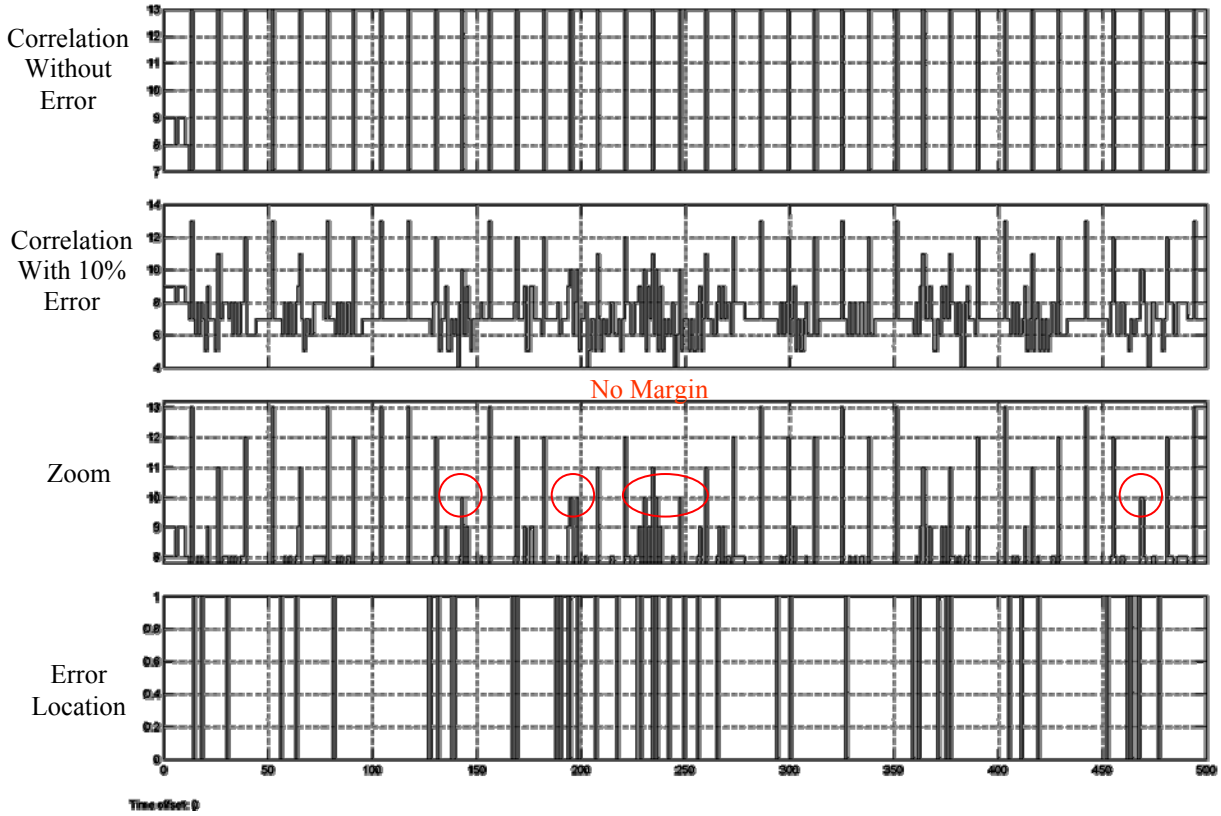


Fig. 3-17. 13-bits Barker code with 10% error waveform

The 13-bits Barker code 10% BER is shown in Fig. 3-17. The first waveform shows the simulated correlation output for the error free 13-bits Barker code. The second waveform illustrates the simulated correlation output of 13-bits Barker code with 10% BER. The zoom-in version is shown in the third waveform and the last waveform indicates the exact error location. With 10% BER, the 13-bits Barker code failed to achieve error free transmission due to insufficient margin. Here, the margin is defined as the difference between the minimum correlation output of correct data and the background noise.

Similarly, the 31-bits Gold code with 10% BER is shown in Fig. 3-18. With 10% BER, there is still a margin of three between the minimum correlation output of correct data and background noise. In this example, the correlator threshold can be set to 23, 24 or 25 to recover the data correctly.

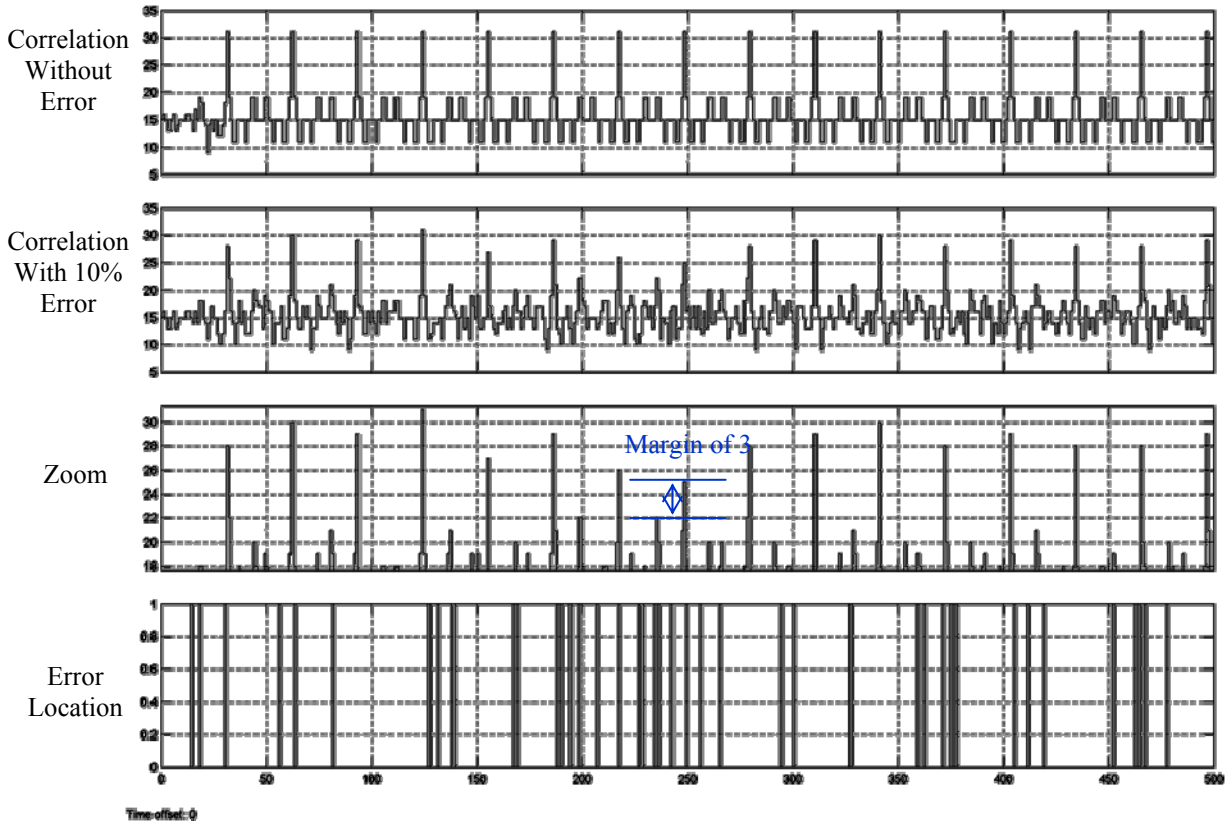


Fig. 3-18. 31-bits Gold code with 10% error waveform

The margin with various BER for 31-bits Gold code and 13-bits Barker code is shown in Fig. 3-19. As illustrated, when the raw data experienced BER up to 10%, 31-bits Gold code can be employed to achieve error free transmission. On the other hand, 13-bits Barker code can only be employed when raw data experienced BER up to 4%.

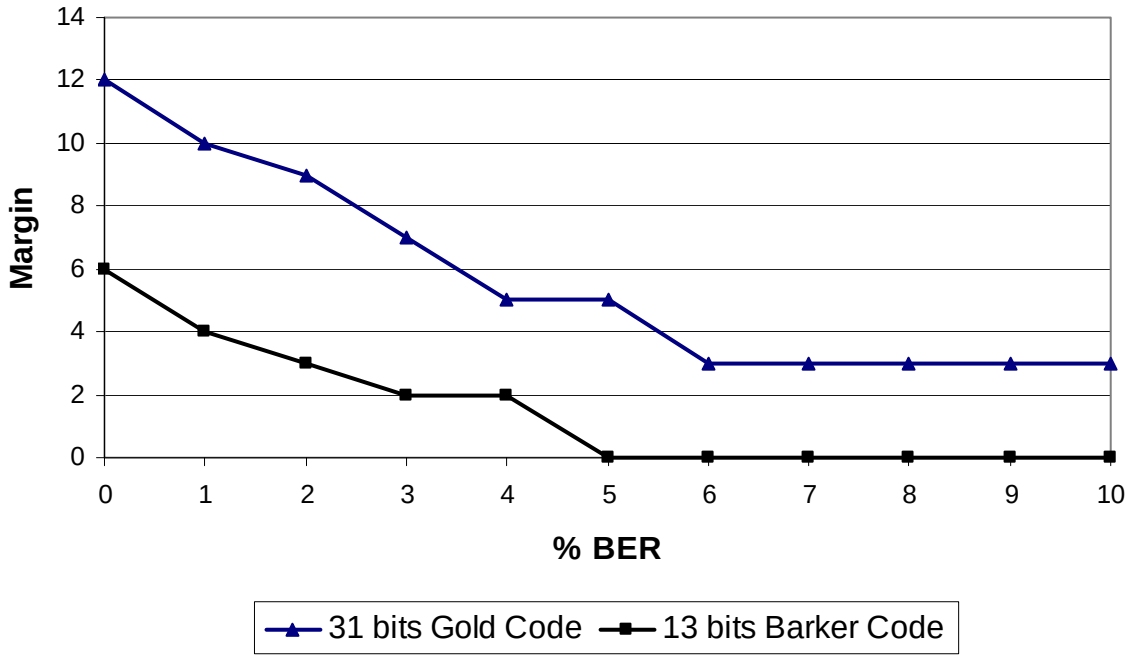


Fig. 3-19. 31-bits Gold code and 13-bits Barker Code BER graph

3.6 Decoder/Correlator Design

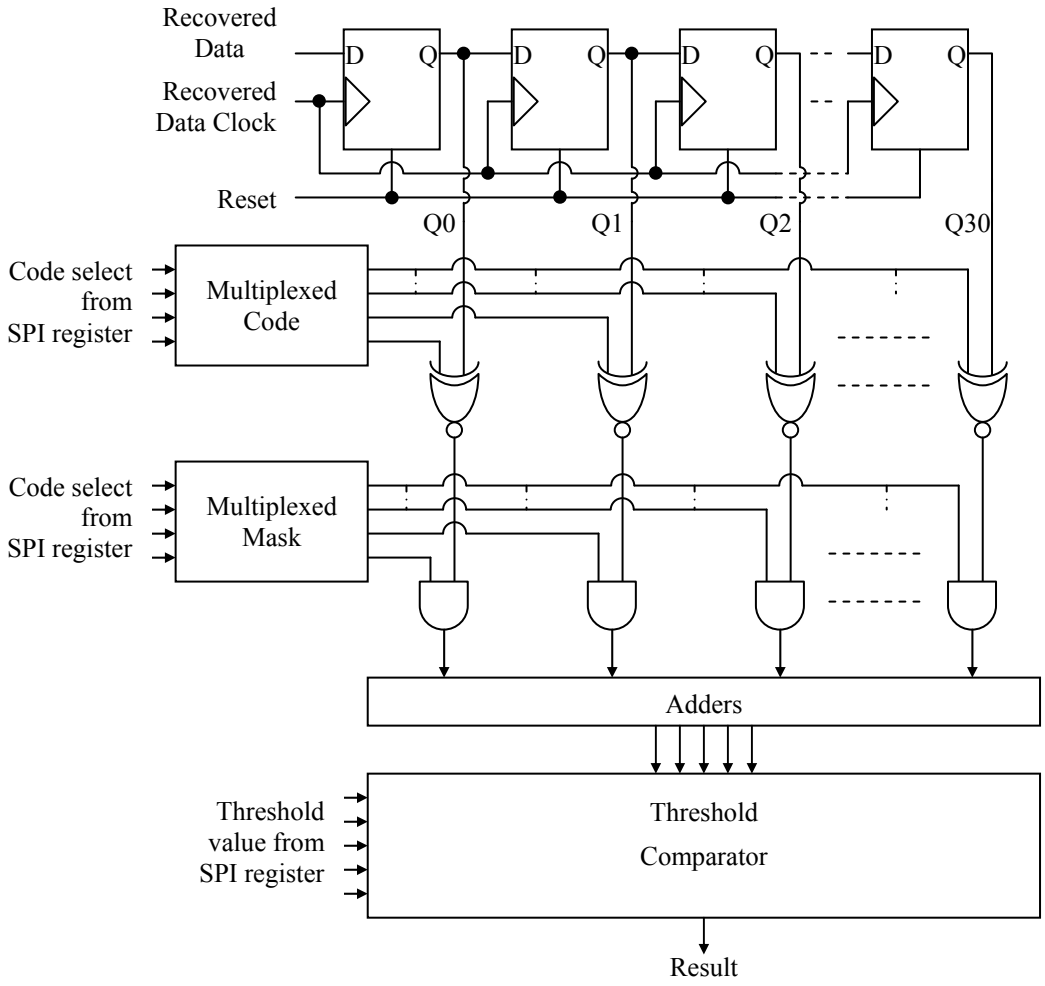


Fig. 3-20. Variable correlator

The decoder/correlator block will correlate the received bit sequence with the given Barker or Gold code to recover back the original transmitted data and is shown in Fig. 3-20. The decoder/correlator will shift in the received bit sequence using the recovered clock and data from the digital CDR into the shift register of the decoder/correlator block. The shift-in data will be compared with the preloaded the 11-bits/13-bits Barker codes or the 31-bits Gold code during each shift through XNOR gates. The output is then summed to obtain the final

correlation value. The obtained correlation value is then compared with the preset threshold to determine whether a bit 1 had been transmitted.

The post layout simulation for both the encoder and decoder/correlator is shown in Fig. 3-21.

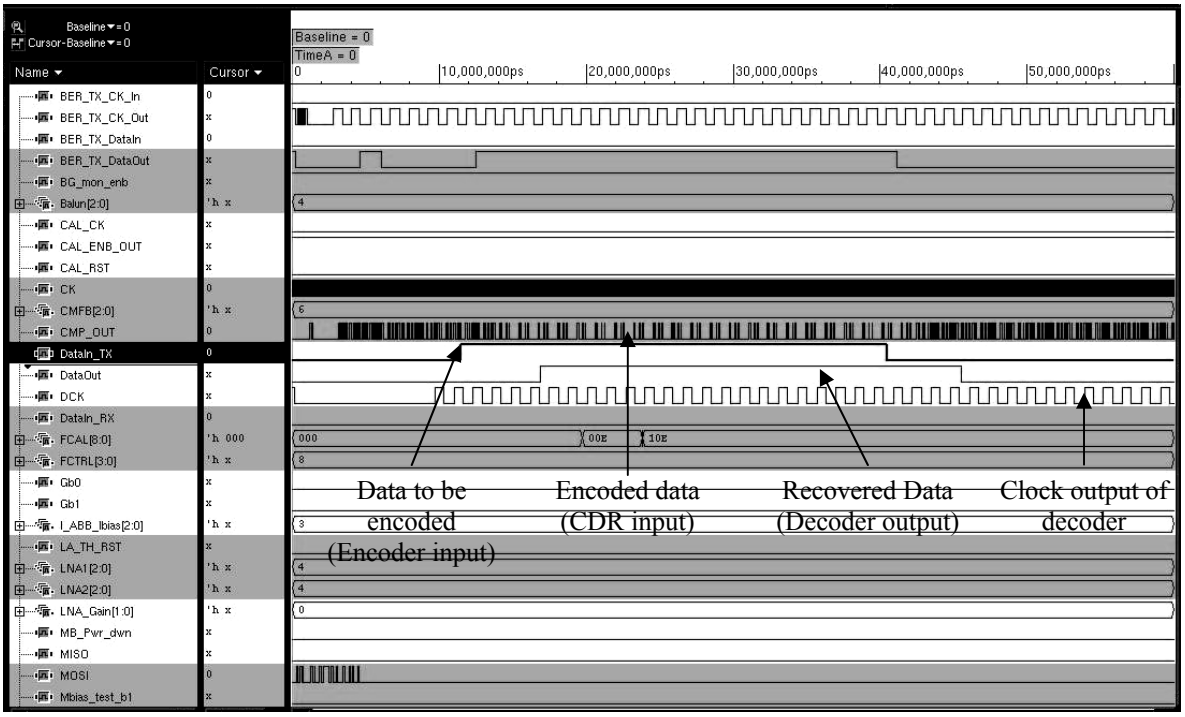


Fig. 3-21. Encoder and decoder/correlator post-layout simulation waveforms

3.7 Pseudorandom Number Sequence Generator Design

The PN sequence generator is included for testing and debugging purposes. It can generate random bit sequence with length of 2^9-1 based on LFSR. The generated bit sequence can either serve as the input signal to the transmitter for transmitter testing, or input signal to the receiver path within the digital baseband for receiver testing. The post layout simulation of the CDR block presented earlier in Fig. 3-10 employs the data generated from this PN sequence generator.

3.8 Bit Error Rate Estimator Design

The BER estimator is included for testing and debugging purposes as well as the optimization of the RF transceiver settings. During the calibration mode, the digital baseband will cycle through the different RF transceiver settings and obtain the corresponding BER for each setting. The setting which gives rise to the lowest BER will be chosen as the optimum settings for the RF transceiver after calibration. This calibration can be done by an external device such as a MCU or a computer through the parallel port using the SPI interface of the digital baseband.

The BER estimator is designed to enable the user to setup a physical link between the transmitter and the receiver. The BER estimator in the transmitter will transmit the predefined data through this physical link to the receiver. The BER estimator in the receiver will compare the recovered data with the predefined data. The number of erroneous bits for the whole transmission will then be stored in the SPI registers. This information can then be used by the external MCU to achieve optimization of RF front-end setting.

3.9 Serial Peripheral Interface Bus Design

The implemented SPI is a standard synchronous serial data link. The SPI is designed to have both read and write capabilities. The serial data format consists of 1-bit read/write command, 7-bits internal registers address and 8-bits data as illustrated in Table 3-1. This 16-bits command word is used to access the internal registers. The command word can be sent as single 16-bits transfer or two 8-bits transfers. The MSB is always transmitted first. The SPI registers definition of the UWB transceiver is listed in Appendix A.

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	R/W~	Address[6:0]						

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Data[7:0]							

Table 3-1. Command Word

3.10 Design Summary

The final digital baseband comprises nine modules as shown in Fig. 3-22. The CDR function is implemented through both pulse searcher and pulse tracker modules. In this chapter, each implemented block has been described with particular emphasis on CDR and calibration module. Post layout simulations have also been shown to validate the feasibility of the whole architecture.

The whole register transfer level (RTL) design is implemented in Verilog HDL language. The RTL design is then synthesized with Synopsys, and place and

route through Cadence Encounter. The final chip layout fabricated in 0.18-μm CMOS process occupies a die area of 0.7 mm×0.8 mm and is shown in Fig. 3-23.

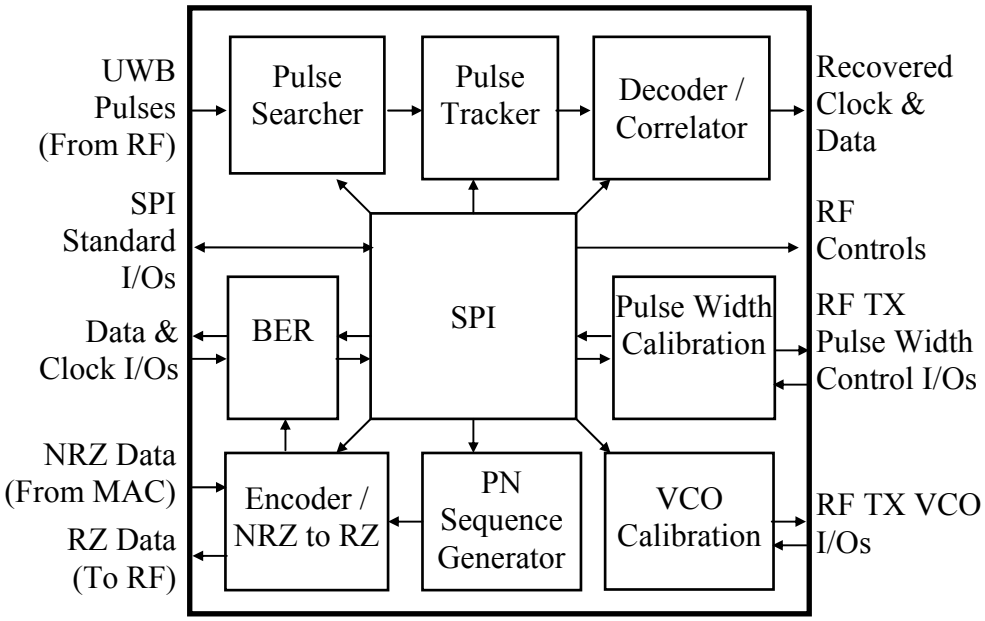


Fig. 3-22. Nine modules of digital baseband

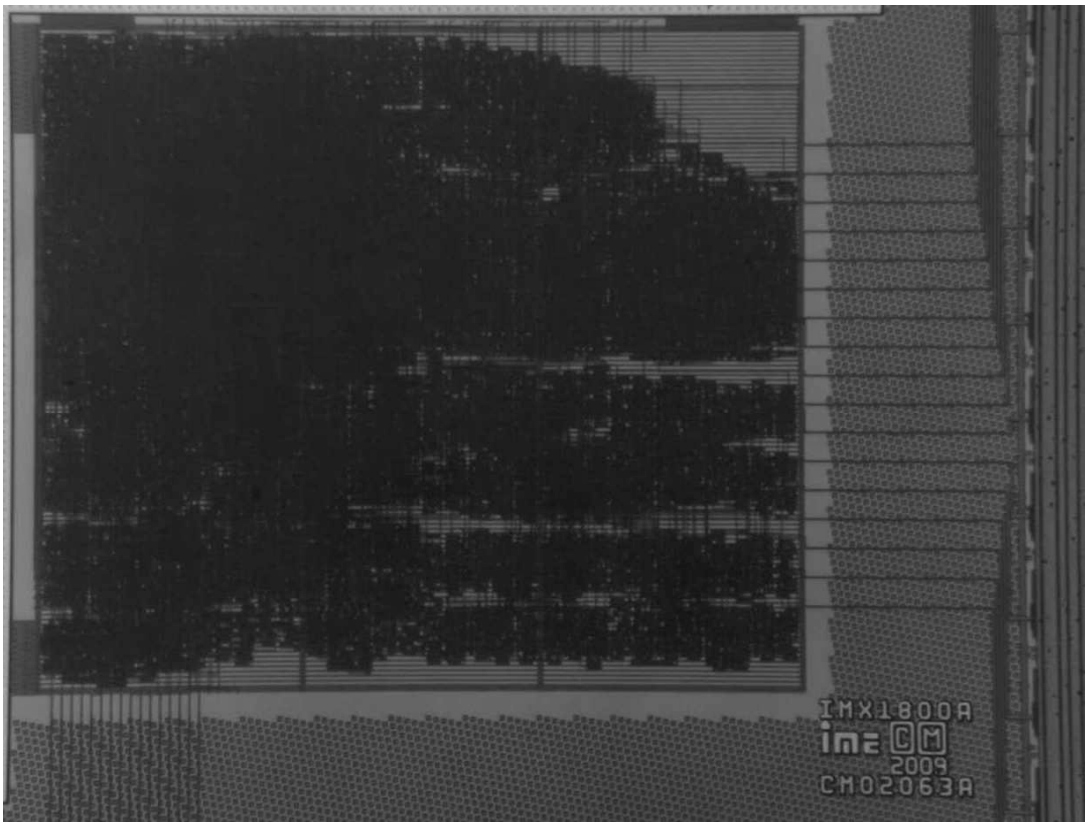


Fig. 3-23. Die photo

CHAPTER 4

MEASUREMENT RESULTS

The fabricated digital baseband operates at 1.8 V and consumes only 5 mW under 1.8 V supply. From testing, it can handle data rate up to 20 Mbps. This test is done using the 100 MHz system clock and changing the number of duration windows from 10 to 5 through an internal divider to achieve data rate of 10 Mbps or 20 Mbps. The performance of the digital baseband is summarized in Table 4-1. As the main focus of this thesis is on CDR and calibration, only measurement results related to those modules will be shown.

Table 4-1. Digital Baseband Specification

System Clock	100 MHz
Data Rate	10 Mbps/20Mbps
Area	800 μm × 700 μm
Operating Voltage	1.8 V
Power Consumption	5 mW
Technology	0.18-μm CMOS

4.1 Clock and Data Recovery Measurement Results

To show the functioning of the clock and data recovery, the following setup is used. A 100 MHz crystal oscillator is employed as the system clock and the targeted data rate is 10 Mbps. A PN data sequence with bit rate of 10.5 Mbps is being transmitted through a UWB transmitter and received by a corresponding UWB receiver front-end. The recovered RZ data from the receiver front-end is then sent to the proposed digital baseband for proper clock and data recovery. The PN data is purposely set to 10.5 Mbps to introduce the clock drifting phenomenon. Otherwise, it will take too many cycles to verify the function of CDR and cannot be displayed within one measurement window. The recovered NRZ data and corresponding data clock are shown in Fig. 4-1. As illustrated, the digital baseband correctly recovers the transmitted data in NRZ format and eliminates the clock drift issue. As the pulse tracking algorithm can only update the pulse position when 1s are detected, as shown in Fig. 4-1, the data clock duty cycle is only updated when 1s are received. In addition, when longer consecutive 0s are observed in the transmitted sequence, more drift error is accumulated, which result in larger change in data clock duty cycle as illustrated. It should be pointed out that under normal operation, the changes in the duty cycle would not be very drastic due to the small difference in the system clock frequencies.

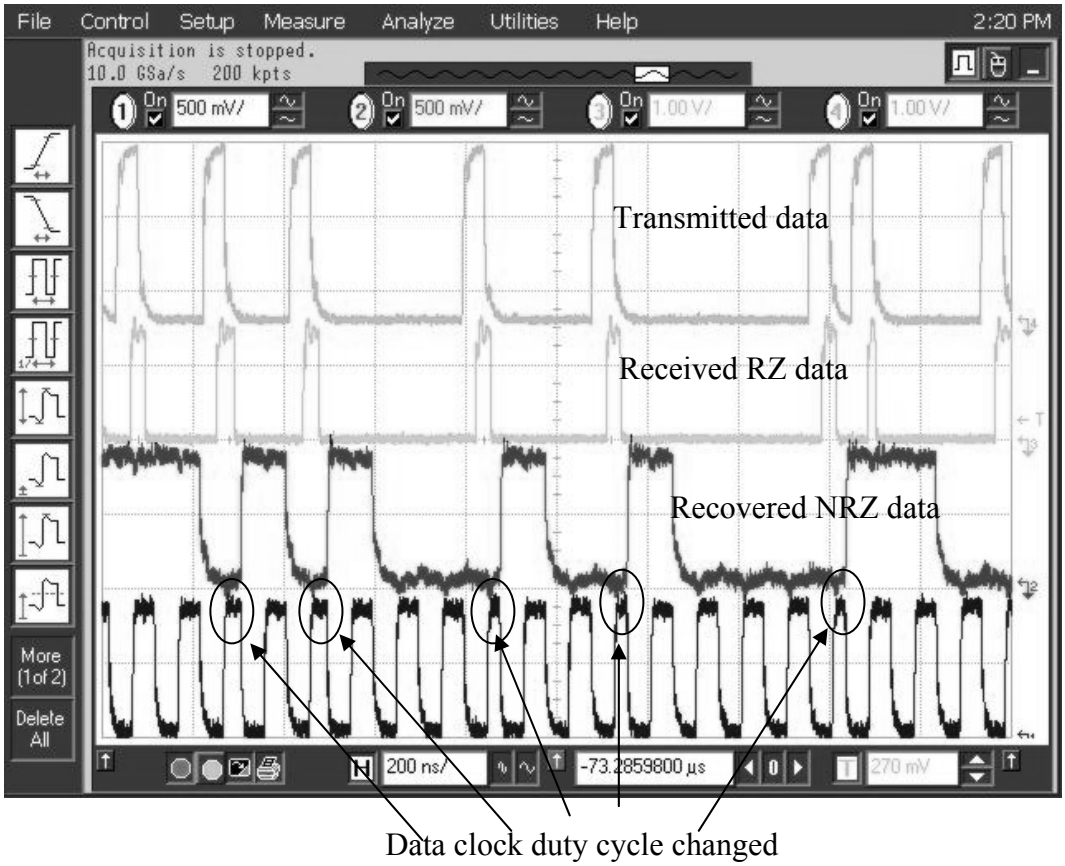


Fig. 4-1. CDR and PN sequence generator measurement result

4.2 VCO Calibration and Pulse Width Calibration Measurement Results

Under default settings, the VCO frequency and pulse width controls are already optimum in the original design (4 GHz and 1 ns) and require no further tuning. In order to verify the VCO calibration algorithm, the VCO frequency is first tuned to frequency larger or smaller than 4 GHz before the invocation of the algorithm. It is observed that the algorithm is able to bring the VCO frequency back to the default setting of 4 GHz. Similarly, the pulse width settings are first adjusted to give rise to power spectrum that violates the FCC mask as shown in Fig. 4-2. After the pulse width calibration is done, the pulse width is back to about 1ns and its spectrum is shown in Fig. 4-3 which meets the FCC mask.

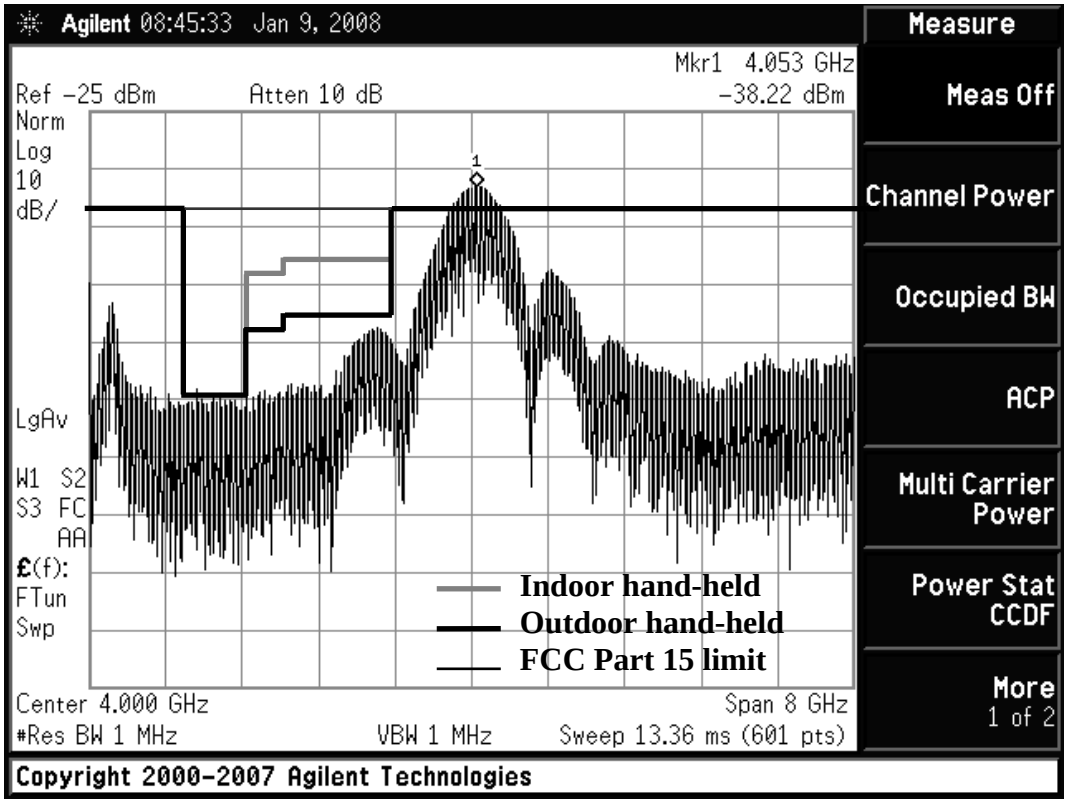


Fig. 4-2. Long pulse width spectrum

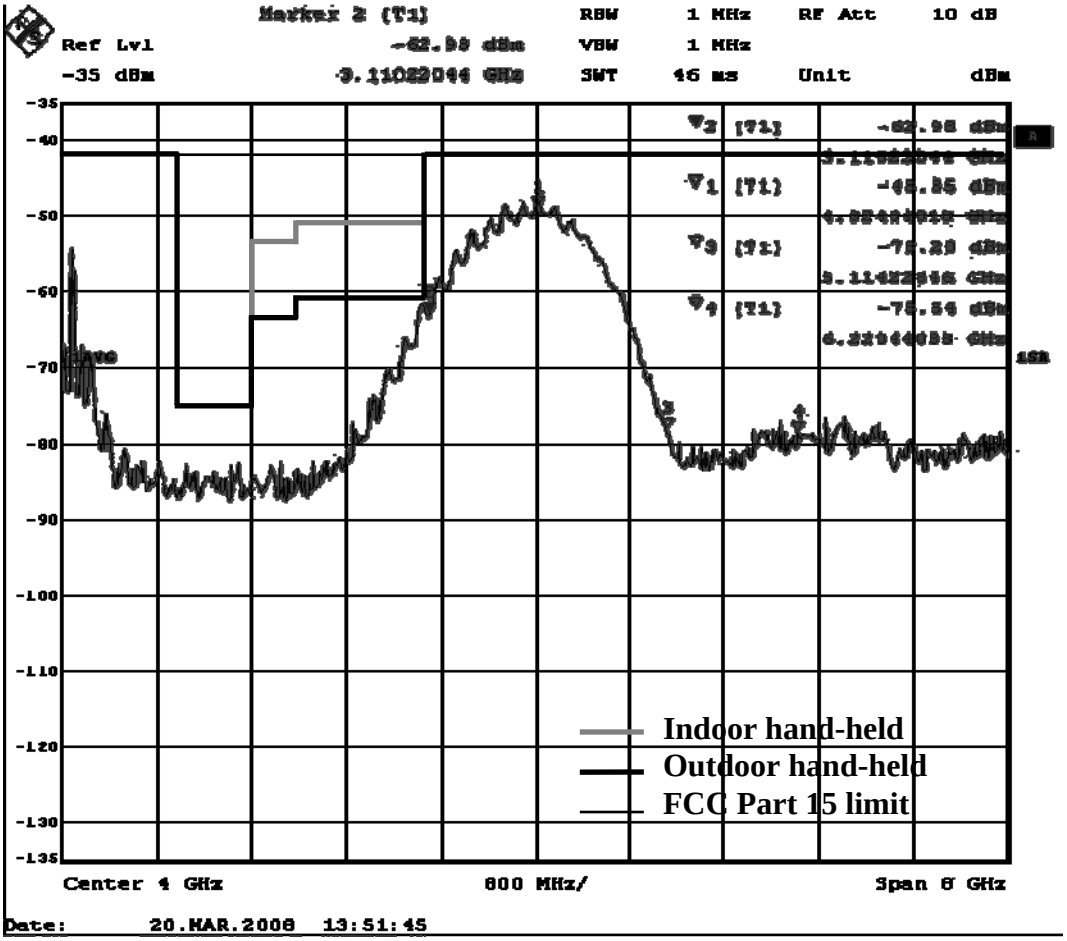


Fig. 4-3. UWB signal spectrum

CHAPTER 5

CONCLUSION

A reconfigurable digital baseband for UWB transceiver is implemented in this work using 0.18-μm CMOS technology. The clock and data recovery function is fully implemented in digital domain without any need for PLL/DLL or ADC, and is thus very amenable to the integration with UWB transceiver. The successful synchronization of received data also allows burst mode operation to minimize the power of the receiver front-end. In addition, the proposed VCO frequency calibration and pulse width calibration will also improve the yield of the IR-UWB transceiver under PVT variations.

One of the possible future improvements about this work is on enhancing the synchronized clock resolution. Currently, the duration windows are limited to one tenth of the data rate to simplify the Verilog coding. This severely limits the resolution attainable for the synchronized clock. In addition, the on-time spans at least 3 duration windows once the incoming UWB pulse position is determined through the proposed pulse searcher and pulse tracker. This restricts the achievable on-time duty cycle to 30% or higher and limits the achievable power saving. If the duration windows can be further reduced, the synchronized clock resolution, duty cycle ratio and RF front-end power consumption can be further improved.

REFERENCES

[1] Ian Oppermann, Matti Hämäla¨inen and Jari Iinatti, "UWB Theory and Applications", John Wiley & Sons Ltd, (2004).

[2] Huseyin Arslan, Zhi Ning Chen and Maria-Gabriella Di Benedetto, "Ultra wideband wireless communication", John Wiley & Sons Ltd, (2006).

[3] F. S. Lee, A. P. Chandrakasan, "A 2.5nJ/bit 0.65V pulsed UWB receiver in 90nm CMOS," IEEE J. Solid-State Circuits, vol. 42, pp. 2851-2859, Dec. 2007.

[4] T. Terada, S. Yoshizumi, M. Muqsith, Y. Sanada, and T. Kuroda, "A CMOS ultra-wideband impulse radio transceiver for 1-Mb/s data communications and ±2.5-cm range finding," IEEE J. Solid-State Circuits, vol. 41, pp. 891-898, Apr. 2006.

[5] Y. Gao, Y. Zheng, and C. H. Heng, "Low-Power CMOS RF front-end for non-coherent IR-UWB receiver" in Proc. ESSCIRC, pp. 386-389, Sept. 2008.

[6] S.B. Anand, B. Razavi, "A CMOS clock recovery circuit for 2.5-Gb/s NRZ data", IEEE J. Solid-State Circuits, vol. 36, pp. 432-439, Mar. 2001.

[7] W. Rhee, H. Ainspan, S. Rylov, A. Rylyakov, M. Beakes, D. Friedman, S. Gowda, and M. Soyuer, “A 10-Gb/s CMOS Clock and Data Recovery Circuit Using a Secondary Delay-Locked Loop,” IEEE Custom Integrated Circuits Conference, pp. 81 – 84, Sep. 2003.

[8] M. Rau et al., “Clock/Data Recovery PLL Using Half-Frequency Clock”, IEEE J. Solid-State Circuits, vol. 32, pp. 1156 – 1159, Jul. 1997.

[9] N. Sasaki, M. Fukuda, M. Nitta, K. Kimoto, and T. Kikkawa, “A Single-Chip Ultra-Wideband Receiver Using Silicon Integrated Antennas for Inter-Chip Wireless Interconnection”, Proc. Int'l Conf. Solid State Devices and Materials, pp. 70 - 71, Sept. 2006.

[10] J. Savoj and B. Razavi, "A 10-Gb/s CMOS Clock and Data Recovery Circuit with a Half-Rate Linear Phase Detector", IEEE J. Solid-State Circuits, vol. 36, no. 5, pp. 761 - 767, May 2001.

[11] P. K. Hanumolu, G. Y. Wei and U. K. Moon, “A wide-tracking range clock and data recovery circuit”, IEEE J. Solid-State Circuits, vol. 43, pp. 425-439 Feb. 2008.

[12] M. Johnson and E. Hudson, "A variable delay line PLL for CPU-coprocessor synchronization”, IEEE J. Solid-State Circuits, vol. 23, pp. 1218 – 1223, Oct. 1988.

[13] Fu, Wei et al., “A digital delay lock loop (DLL) circuit for Applied Micro Circuits Corporation”, United States Patent 7130367, Oct. 2006.

[14] I. O’Donnell, S. Chen, S. Wang, and R. Brodersen, “An integrated, low power, ultra-wideband transceiver architecture for low-rate, indoor wireless systems”, in Proc. IEEE CAS Workshop on Wireless Communications and Networking, pp. 1623-1631, Sept. 2002.

[15] R. Blazquez, P. Newaskar, F. Lee, and A. Chandrakasan, “A baseband processor for pulsed ultra-wideband signals”, in Proc. CICC, pp. 587-590, Oct. 2004.

[16] Ryckaert, J. et al, “Ultra-WideBand Transmitter for Wireless Body Area Networks”, Proc. 14th IST Mobile & Wireless Communications Summit, June 2005.

[17] M. Fischer, A. Adalan, A.L. Scholtz, C. Mecklenbräuker “Architecture of a Modular IEEE 802.15.4a Ultra-Wideband Transmitter”, Proc. of the Microelectronics Conference (Informationstagung. Mikroelektronik ME 2008), pg 284 - 288, Oct. 2008.

[18] S. Diao, Y. Zheng, and C. H. Heng, “A CMOS ultra low-power and highly efficient UWB-IR transmitter for WPAN applications”, IEEE Trans. on Circuits and Systems II, vol. 56, pp. 200-204, Mar. 2009.

APPENDIX A

SPI Registers Definition

Table A-1. SPI Registers Definition

ADDR [6:0]	7	6	5	4	3	2	1	0
00	**DR[7:0]**							
	8'h0A							
01	**Data_Rate**	**DBB_ENB**	**DR[13:8]**					
	1	0	6'h0					
02	**Gb1**	**Gb0**	**CAL_EN**	**MODE**	**PN_EN**	**CNT_VAL[8]**	**FMIN[8]**	**FMAX[8]**
	0	0	0	0	1	1	1	1
03	**FMAX[7:0]**							
	8'h17							
04	**FMIN[7:0]**							
	8'h0E							
05	**CNT_VAL[7:0]**							
	8'h9D							
06	**LNA_EN_val[7:0]**							
	8'h02							
07	**SQ_EN_val[7:0]**							
	8'h02							

ADDR [6:0]	7	6	5	4	3	2	1	0
08	**SEL[1:0]**		**Length_val[5:0]**					
	2'b10		6'h1					
09	**Pul_Gen[3:0]**				**FCTRL_VAL[3:0]**			
	4'h7				4'h8			
0A	**LNA_Gain[1:0]**		**LNA1[2:0]**			**LNA2[2:0]**		
	2'h0		3'h4			3'h4		
0B						**Balun[2:0]**		
						3'h4		
0C					**numpartition[3:0]**			
					3'hA			
0D	**PULWIDTH[7:0]**							
	8'h64							
0E	**code_select[3:0]**				**WinTH[3:0]**			
	4'h0				4'h3			
0F	**numcycle[2:0]**			**Thresh0[4:0]**				
	3'h5			5'h1C				

Table A-1. SPI Registers Definition (Con't)

ADDR [6:0]	7	6	5	4	3	2	1	0
10			**SPI_IO_SEL**	**Thresh1[4:0]**				
			1'b1	5'h1C				
11				**dck_delay[4:0]**				
				5'hE				
12	**VGF**	**VG**	**VG1**	**VG2**	**SW3**	**SW2**	**SW1**	**Vctr**
	1'b1	1'b1	1'b0	1'b0	1'b0	1'b0	1'b1	1'b0
13							**error_latch**	**rxfifo_flush**
							1'b0	1'b0
14	**latch_delay[7:0]**							
	8'h32							
15	**BER_ERRORS_No[15:8]**							
	8'h0 (read)							
16	**BER_ERRORS_No[7:0]**							
	8'h0 (read)							
17	**BER_Sample_No[15:8]**							
	8'h27							

ADDR [6:0]	7	6	5	4	3	2	1	0
18	**BER_Sample_No[7:0]**							
	8'h10							
19	**EXT**	**RXEN**	**TXEN**	**code_bypass**	**rz_bypass**	**BER_Done**	**BER_En**	**BER_Reset**
	1'b1	1'b1	1'b0	1'b1	1'b0	1'b0 (read)	1'b0	1'b1
1A	**freqcnt[7:0]**							
	8'hA							
1B	**PGB_C**	**PW_CAL_EN**	**PW_CAL_PERIOD[13:8]**					
	1'b1	1'b0	6'h2					
1C	**PW_CAL_PERIOD[7:0]**							
	8'h00							
1D			**PW_CAL_SAMPLE[13:8]**					
			6'h0					
1E	**PW_CAL_SAMPLE[7:0]**							
	8'hCA							
1F			**PW_CAL_RST_DC [13:8]**					
			6'h0					

Table A-1. SPI Registers Definition (Con't)

ADDR [6:0]	7	6	5	4	3	2	1	0
20	**PW_CAL_RST_DC[7:0]**							
	8'h02							
21			**PW_CAL_test_data_DC [13:8]**					
			6'h0					
22	**PW_CAL_test_data_DC [7:0]**							
	8'h66							
23	**Test_data_delay [7:0]**							
	8'h00							
24	**o_max [7:0]**							
	8'b00011111							
25	**o_min [7:0]**							
	8'b00001111							
26					**PGB_default [3:0]**			
					4'h3			
27	**V_CMFB_F_end[2:0]**			**V_LA_threshold[4:0]**				
	3'h3			5'h0F				

ADDR [6:0]	7	6	5	4	3	2	1	0
28	**MB_Pwr_dwn**	**BG_mon_enb**	**V_RSSI[2:0]**			**V_ABB_bias[2:0]**		
	1'b0	1'b1	3'h3			3'h7		
29	**Thresh_control[1:0]**		**PW_control[2:0]**			**I_ABB_Ibias[2:0]**		
	2'h2		3'h6			3'h3		
2A	**V_CC_mon_enb**	**V_CV_mon_enb**	**SQR2[2:0]**			**SQR1[2:0]**		
	1'b1	1'b1	3'h6			3'h6		
2B		**Mbias_test_b1**	**CMFB[2:0]**			**SQR3[2:0]**		
		1'b0	3'h6			3'h6		
2C								**LA_TH_RST**
								1'b0
2D			**SELOUT[5:0]**					
			6'h16					
2E			**initial_no_of_window[13:8]**					
			6'h0					
2F	**initial_no_of_window[7:0]**							
	8'h01							

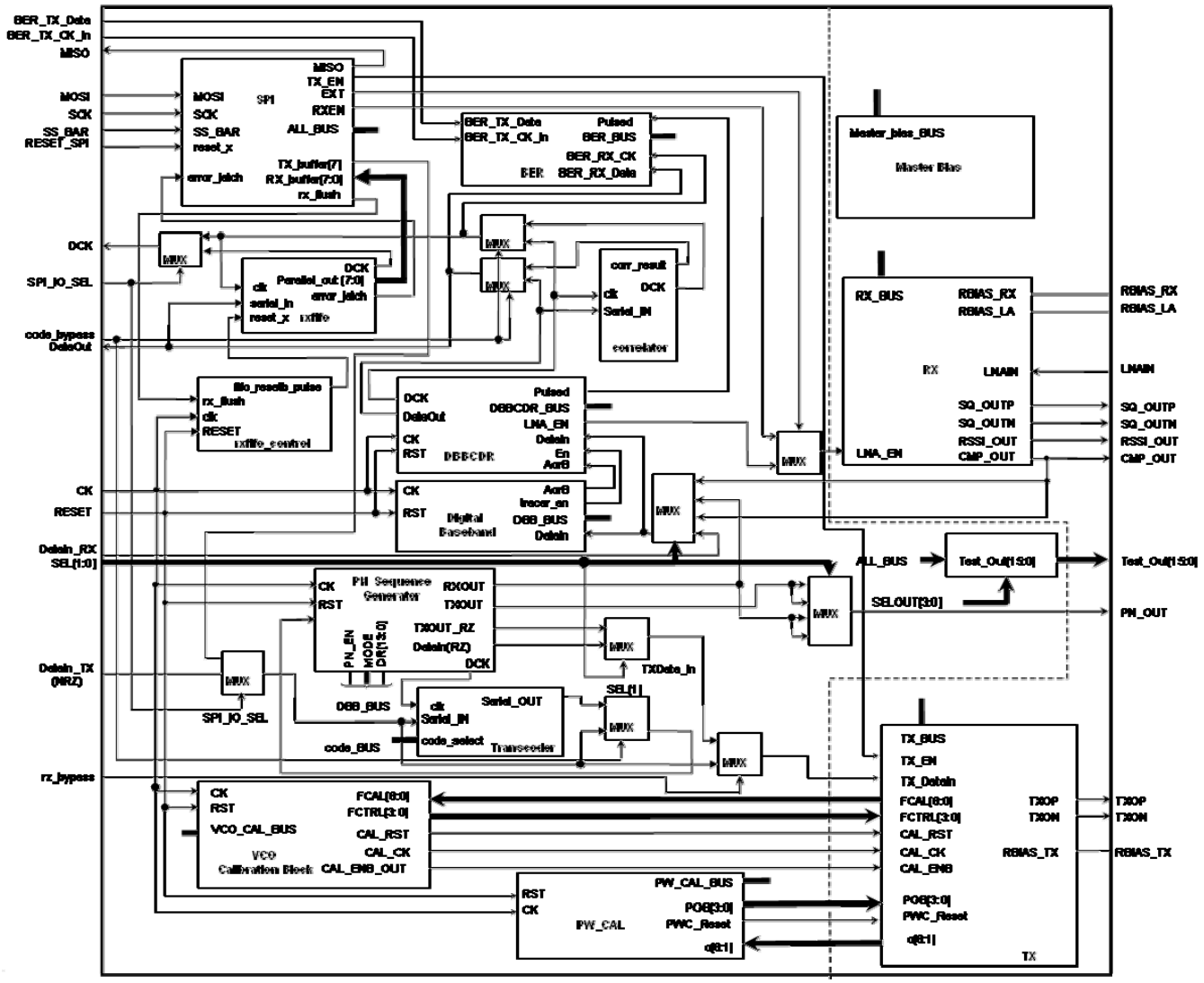
Table A-1. SPI Registers Definition (Con't)

ADDR [6:0]	7	6	5	4	3	2	1	0
30			**final_stop_sample_window[13:8]**					
			6'h0					
31	**final_stop_sample_window[7:0]**							
	8'h01							
32			**final_before_start_window[13:8]**					
			6'h0					
33	**final_before_start_window[7:0]**							
	8'h01							
34			**RFRX_before_start_window[13:8]**					
			6'h0					
35	**RFRX_before_start_window[7:0]**							
	8'h01							
36			**RFRX_disable_window[13:8]**					
			6'h0					
37	**RFRX_disable_window[7:0]**							
	8'h01							

ADDR [6:0]	7	6	5	4	3	2	1	0
38			**initial_before_start_window[13:8]**					
			6'h0					
39	**initial_before_start_window[7:0]**							
	8'h01							
7E	**rx_buffer[7:0]**							
	8'h0							
7F	**TX_buffer[7:0]**							
	8'h0							

APPENDIX B

Top Module of Digital Baseband



APPENDIX C

Design Compiler Digital Synthesis Script

```
/* Synopsys design compiler dc_shell script for WT */


/* read in the entire design */

/*read -f verilog {../ForSyn_Final/UWBTXRX_final.v}*/
read_verilog {../ForSyn_Final/UWBTXRX_final.v}


current_design UWBTXRX
link

check_design

/* resolve multiple references */
uniquify

/* apply constraints and attributes */
/*include defaults.con */
/* if you have constraints file separately */

set_fix_multiple_port_nets -all

/* transfor_csa -decompose_multiply all -duplicate -area */

set_dont_use {typical/AND3*}
set_dont_use {typical/AND4*}
set_dont_use typical/AOI*
set_dont_use typical/NAND3*
set_dont_use (typical/NAND4*)
set_dont_use (typical/NOR3*)
set_dont_use (typical/NOR4*)
set_dont_use (typical/CMPR*)
set_dont_use (typical/XOR3*)
set_dont_use (typical/XNOR3*)
set_dont_use (typical/OAI*)
set_dont_use (typical/OR3*)
set_dont_use (typical/OR4*)
set_dont_use (typical/SDFF*)
set_dont_use (typical/SEDFF*)

set_dont_use (typical/MXI2X4)
create_clock CK -period 5 -waveform {"0" "2.5"}
create_clock SCK -period 100 -waveform {"0" "50"}

create_clock CMP_OUT -period 25 -waveform {"0" "5"}
create_clock BER_TX_CK_In -period 25 -waveform {"0" "12.5"}

set_drive 0 [get_ports {CK}]
set_drive 0 [get_ports {SCK}]
set_drive 0 [get_ports {CMP_OUT}]
set_drive 0 [get_ports {BER_TX_CK_In}]
```

```
/*set_drive 0 find(port, CK)*/
/*set_drive 0 find(port, SCK)*/

/*set_drive 0 find(port, CMP_OUT)*/
/*set_drive 0 find(port, BER_TX_CK_In)*/

/*set_drive 0 {RST_FSM RST_SPI}

set_dont_touch_network [get_ports {CK}]
set_dont_touch_network [get_ports {SCK}]
set_dont_touch_network [get_ports {CMP_OUT}]
set_dont_touch_network [get_ports {BER_TX_CK_In}]

/*set_dont_touch_network find(clock,CK)*/
/*set_dont_touch_network find(clock, SCK)*/

/*set_dont_touch_network find(clock,CMP_OUT)*/
/*set_dont_touch_network find(clock,BER_TX_CK_In)*/

/*set_dont_touch_network {RST_FSM RST_SPI}

set_clock_uncertainty 0.1 CK
set_clock_uncertainty 0.1 SCK

set_clock_uncertainty 0.1 CMP_OUT
set_clock_uncertainty 0.1 BER_TX_CK_In

/* set load at output ports */

set_load 5 {DataOut DCK MISO PN_OUT TX_DataIn RF_RX_EN RF_TX_EN
Test_Out BER_TX_DataOut BER_TX_CK_Out}
set_load 1 {LNA1 LNA2 LNA_Gain Balun SQR2 SQR1 SQR3 CMFB VGF VG
VG1 VG2 SW3 SW2 SW1 Vctr PW_control Thresh_control LA_TH_RST}
set_load 1 {MB_Pwr_dwn BG_mon_enb V_CMFB_F_end V_LA_threshold
V_RSSI V_ABB_bias I_ABB_Ibias V_CC_mon_enb V_CV_mon_enb
Mbias_test_b1}
set_load 2 {CAL_RST CAL_CK CAL_ENB_OUT Gb1 Gb0 PGB PWC_Reset PGB_C
FCTRL}


set_input_delay 1 -clock SCK {MOSI SS_BAR}
set_input_delay 1 -clock CK {DataIn_TX DataIn_RX o FCAL}

set_input_delay 1 -clock BER_TX_CK_In {BER_TX_DataIn}


set_output_delay 1 -clock SCK {MISO}
set_output_delay 1 -clock CK {DataOut DCK PN_OUT}
set_output_delay 1 -clock CK {BER_TX_DataOut BER_TX_CK_Out}
set_output_delay 1 -clock CK {PGB PWC_Reset CAL_RST CAL_CK
CAL_ENB_OUT FCTRL}


set_max_fanout 20.0 UWBTXRX


set_min_delay 3 -from UnitPN/delay/in -to UnitPN/delay/out

/* override auto wire load selection */
```

```
set_wire_load_model -name "chartered18_wl10"

set_wire_load_mode top

check_design

/* compile the design */


compile_ultra

check_design
report_area > area.report;
report_net  > net.report;
report_timing > timing.report;
report_cell > cell.report;
report_power -cell -net -hier -verbose > power_net.report;
report_resources -hierarchy
redirect report_hold.rpt { report_constraint -all_violators -
verbose -min_delay }
redirect report_setup.rpt { report_constraint -all_violators -
verbose -max_delay }
redirect report_timing.rpt { report_timing }

/* Define verilog output setting */
/*verilogout_no_tri = true*/
set verilogout_no_tri true
/*verilogout_show_unconnected_pins = true*/
set verilogout_show_unconnected_pins true
change_names -hier -rules verilog
define_name_rules TOP -allowed "A-Za-z0-9_[]" -replacement_char ""

/* Export verilog */
write -f verilog -hier -o syn_UWBTXRX_Final.v

/* Export db file */
/*write -f db -hier -o syn_UWBTXRX.db */
write -format ddc -hier -output syn_UWBTXRX_Final.ddc

write_sdf -version 2.1 syn_UWBTXRX_Final.sdf -context verilog

write_sdc syn_UWBTXRX_Final.sdc

exit
```

APPENDIX D

Cadence Encounter Place and Route Script

```
#### INPUT FILE #######
loadConfig /proj/uwbsc/DIGITALIC_TWD/enc_spi/Default.conf 0
commitConfig
# fit
#### FLOORPLAN & IO #######
setDrawMode fplan
loadFPlan /proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRX_Finalfp.fp
# setDrawMode fplan
#### BLOCK PLACE ##########
# placeInstance I_ehs_top/I_TX_FIFO 420.42 419.44 R180
# placeInstance I_ehs_top/I_RX_FIFO 704.74 419.44 R180
####  ADD Halo TO BLOCK RAM ########
# selectInst I_ehs_top/I_TX_FIFO
# addHaloToBlock 4.6 0 4.6 4.6 I_ehs_top/I_TX_FIFO
# selectInst I_ehs_top/I_RX_FIFO
# addHaloToBlock 4.6 0 4.6 4.6 I_ehs_top/I_RX_FIFO
#### POWER #################
fit

globalNetConnect VDD -type pgpin -pin VDD -inst * -module {}
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {}
globalNetConnect VDD -type tiehi -pin VDD -inst * -module {}
globalNetConnect VSS -type tielo -pin VSS -inst * -module {}
globalNetConnect VDD -type net -net VDD -module {}
globalNetConnect VSS -type net -net VSS -module {}
#globalNetConnect VSS -type net -net GND! -module {}

addRing -spacing_bottom 1 -spacing_top 1 -spacing_right 1.0 -
spacing_left 1.0 \
        -width_left 2 -width_bottom 2 -width_top 2 -width_right 2
\
        -layer_bottom METAL1 -layer_top METAL1 -layer_right METAL2
\
      -layer_left METAL2 -around core  -jog_distance 0.66 -
threshold 0.66 \
        -offset_bottom 2.0   -offset_left 2.0  -offset_right 2.00
\
      -offset_top 2.0  -nets {VSS VDD } -stacked_via_bottom_layer
METAL1 \
      -stacked_via_top_layer METAL2

addStripe -block_ring_top_layer_limit METAL2 -
max_same_layer_jog_length 1.2 \
          -padcore_ring_bottom_layer_limit METAL2 -
set_to_set_distance 170 \
          -stacked_via_top_layer METAL2 -
padcore_ring_top_layer_limit METAL2 \
          -spacing 1.0 -merge_stripes_value 1 \
          -layer METAL2 -block_ring_bottom_layer_limit METAL2 \
          -width 2 -nets {VSS VDD } -stacked_via_bottom_layer
METAL1
```

```
sroute -jogControl { preferWithChanges differentLayer } -
layerChangeTopLayer 2

# selectInst I_ehs_top/I_TX_FIFO
# addHaloToBlock -4 -4 -4 -4  I_ehs_top/I_TX_FIFO
# selectInst I_ehs_top/I_RX_FIFO
# addHaloToBlock -4 -4 -4 -4  I_ehs_top/I_RX_FIFO



#### CELL PLACEMENT ##########
placeDesign -prePlaceOpt

#opti
#### TRAIL ROUTE #####
trialRoute -maxRouteLayer 5
setDrawMode place

#### SAVE DATABASE ####
saveDesign
/proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRX_Final_b4ct.enc

#### CLOCK TREE SYNTHESIS ####
#create
#ClockTreeSpec -output pad_ehs_dig.ctstch -bufFootprint A+YA  \
#               -invFootprint A+OE+Y!(A) \
#                -bufferList CLKBUFXL CLKBUFX1 CLKBUFX2 CLKBUFX3 \
#                CLKBUFX4 CLKBUFX8 CLKBUFX12 CLKBUFX16 CLKBUFX2
#

specifyClockTree  -clkfile mod.ctstch

ckSynthesis       -rguide mod_cts/mod_cts.guide \
                  -report mod_cts/mod_cts.ctsrpt \
                  -forceReconvergent

saveDesign /proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRXACT_Final.enc

saveDesign
/proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRXACTop_Final.enc
#TIEHI TIELO

#### Route #####

getNanoRouteMode -quiet
getNanoRouteMode -quiet envSuperthreading
getNanoRouteMode -quiet drouteFixAntenna
getNanoRouteMode -quiet routeInsertAntennaDiode
getNanoRouteMode -quiet routeAntennaCellName
getNanoRouteMode -quiet timingEngine
getNanoRouteMode -quiet routeWithTimingDriven
setNanoRouteMode -quiet routeWithTimingDriven true
getNanoRouteMode -quiet routeWithEco
getNanoRouteMode -quiet routeWithSiDriven
getNanoRouteMode -quiet routeTdrEffort
getNanoRouteMode -quiet routeWithSiPostRouteFix
getNanoRouteMode -quiet drouteAutoStop
getNanoRouteMode -quiet routeSelectedNetOnly
getNanoRouteMode -quiet drouteStartIteration
```

```
getNanoRouteMode -quiet envNumberProcessor
getNanoRouteMode -quiet envSuperthreading
setNanoRouteMode -quiet routeTopRoutingLayer 5
setNanoRouteMode -quiet routeBottomRoutingLayer 1
getNanoRouteMode -quiet drouteEndIteration
getNanoRouteMode -quiet routeEcoOnlyInLayers
trialRoute -handlePreroute
setCteReport
writeDesignTiming .timing_file.tif
freeTimingGraph
#globalDetailRoute
wroute
saveDesign
/proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRX_Final_route_b4op.enc

saveDesign
/proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRX_Final_route_atop.enc

##### TIMING ANALYSIS & OPTIMIZATION #####


##### ADD FILLER CELL ########
addFiller -cell FILL64 -prefix FILL64 -markFixed
addFiller -cell FILL32 -prefix FILL32 -markFixed
addFiller -cell FILL16 -prefix FILL16 -markFixed
addFiller -cell FILL8 -prefix FILL8 -markFixed
addFiller -cell FILL4 -prefix FILL4 -markFixed
addFiller -cell FILL2 -prefix FILL2 -markFixed
addFiller -cell FILL1 -prefix FILL1 -markFixed


#### METAL FILL ########
#setDrawMode place
#setMetalFill -layer 1 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 2 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 3 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 4 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 5 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 6 -minDensity 30.00 -maxDensity 40.00
#setMetalFill -layer 6 -minLength 10 -gapSpacing 2 -minWidth 2 \
#             -minDensity 30.00 -maxDensity 40.00 -decrement 2 \
#             -maxLength 20 -activeSpacing 2
#addMetalFill -area 235 235 1180 1180 -layer { 1 2 3 4 5 6 } -nets
{ VSS VDD }
#addIoFiller -cell PFILL1 -prefix FILLER

#### OUTPUT NETLIST ####
saveNetlist pnr_UWBTXRX_Final.v
delayCal -sdf pnr_UWBTXRX_Final.sdf

streamOut UWBTXRX_Final_pnr.gds -mapFile stream.map -libName
DesignLib  \
          -structureName UWBTXRX \
          -stripes 1 -units 1000 -mode ALL

saveDesign
/proj/uwbsc/DIGITALIC_TWD/enc_spi/UWBTXRX_Final_routed_no_vio.enc
```