Abstract
This paper describes an implementation for the digital backend of an ultra wide band receiver. The main task of the digital backend is to achieve synchronization between the receiver and the transmitter in order to enable data extraction. A new technique is presented, that aims to minimize power consumption, chip area and synchronization time. Furthermore, the design is flexible in the sense of handling different modulation schemes and PN code lengths. Some tests were carried out using FPGA kit showing that the system is working properly in its different modes of operation.

1. INTRODUCTION
UWB is a pulse based communication system that usually uses Gaussian pulses or one of its derivatives with small duty cycle (to save power) for data transmission. A pseudo noise (PN) sequence is XNORed with each data bit in the transmitter in order to spread the data spectrum, enhance the processing gain and to achieve multi-user access using a unique PN sequence for each user. The same PN sequence is generated in the receiver which is XNORed again with the coded data. The XNORing result is then accumulated and compared to a certain threshold. These two steps (i.e., XNORing and accumulation) represent a simple correlation process. If the correlation result exceeds a positive threshold value then a full agreement (between the generated PN sequence in the receiver and the received coded data) has occurred. Therefore, a logic one is received. On the other hand if the correlation result exceeds a negative threshold value then a full disagreement has occurred. Therefore, a logic zero is received.

It’s obvious from the above discussion that the PN sequence generated in the receiver must be in phase with the received PN sequence, also the position of the received pulse must be exactly determined. Hence, the problem is to achieve two-level synchronization. Level one is determining the position of the desired user’s Gaussian pulse in order to activate the LNA and ADC only in these periods (to save power). The second is to align the phase of the generated PN sequence in the receiver with the received PN sequence.

2. SYSTEM SPECIFICATION
1. The system uses normal Gaussian pulse for UWB transmission
2. The system uses OOK as a pulse modulation scheme
3. The system uses a PN sequence of length 31 chips
4. The system uses a TDM/CDMA technique scheme for multi user access

3. TOTAL SYSTEM OVERVIEW
The system consists of two main blocks, namely, WindowSync and PN_Correlator blocks(Fig.1). The former is responsible of performing the first level of synchronization, while the latter manages the second level. The interaction between the two blocks takes place, as will be described in the following section, in order to make synchronization smart.

3.1 Main Algorithm Overview
The system takes the serial output of the ADC as an input, and generates a clock called PhiChip which has a frequency that is the same as the chip rate preset by the transmitter and the receiver. Furthermore, the system divides each PN chip, through a clock called PhiSeg, into 4 logical segments, which are then processed individually according to the mode of operation (acquisition or tracking, to be shown later).
pulse location belongs to the desired user or not (i.e., whether this pulse location contains the user PN sequence or not). The window sync system knows the PN correlation result through a feedback signal called X.

After determining the segment containing the desired pulse sequence, the system enters the tracking mode where the desired pulse must be centered within the segment to track any drift of it. Some power issues are also introduced to be discussed in the implementation.

4 IMPLEMENTATION OF WINDOW SYNC

The system takes the serial output of the ADC as an input, and generates a clock called Phi Chip which has a frequency that is the same as the chip rate preset by the transmitter and the receiver. The serial binary input data (coming from the one bit resolution ultra high speed ADC) is first converted, through the serial to parallel block, into a parallel bus of 32 bits carrying the data contents of the 32 successive samples of the current segment. The main objective of this block is to reduce the processing speed (i.e., the output parallel samples have a rate of change that’s as lower as 32 times than the input serial data), and to divide each PN chip into 4 logical segments, each segment contains 32 samples.

The parallel bus is, then, introduced to four matched filter. Each filter of them accomplishes the task of parallel correlating its 16 input samples with predetermined Gaussian pulse template coefficients. The Gaussian pulse is assumed to be16-sample width. Thus, the system makes use of four matched filters each of them is 16-sample width and they are placed 8 samples apart from each other. The first matched filter (i.e., M0) is responsible for processing the inter-segment samples (i.e., the last 8 samples of the previous segment and the first 8 samples of the current segment). On the other hand the remaining matched filters process the intra-segment samples i.e., with M2 in the segment center and considered as the target position.

Each matched filter output is, then, passed through a threshold detector to compare its value with a threshold in order to determine whether a pulse is present in this region or not. Mn (0 TO 3) is called the equivalent output of the matched filters since they represent the data (‘1’ if there is a pulse, ‘0’ if not) that resides in this region of the segment. The following block in the signal path is the matrix block. WindowMotherControl uses the Matrix to: save the data contents of the different positions within the chip in a matrix format which has a row index indicated by the segment number through the input named Seg. Also it is used to determine the segment (or the window of observation) from which the data pulses is read in order to be sent to the PN correlator. This is done by the SegIndex input port which determines the row of the matrix from which the output data is obtained Note the difference between the Seg and SegIndex input ports, the former is a simple counter that is updated each segment by the ClkGenerator in order to indicate the current segment (i.e., 0, 1, 2, 3) and thus Seg is used for writing in the Matrix. On the contrary, the SegIndex is used for reading from the Matrix and it has a constant value that can be changed only by the WindowMotherControl to shift the window of observation; finally the WindowMotherControl uses the matrix to shift the window by simply incrementing or decrementing SegIndex. Actually, there are two matrices: matrix (0): used in the tracking mode, and matrix(1): used in the acquisition mode. Multiplexing between the two matrices is made by MatrixSelect input port which is controlled by WindowMotherControl. The matrix takes as an input a bus of 4 bits carrying the equivalent matched filter outputs (i.e., Mn (0 TO 3)) and the Seg signal which indicates the current processed segment. Writing to the matrix is done at the negative edge of PhiSeg to avoid the inputs glitching, as follows:

If MatrixSelect is ‘1’ (i.e., the acquisition matrix is selected) then:
Mn (i, 1) = Mn (0); Mn (i, 2) = Mn (2); Mn (i, 3) = Mn (1) OR Mn (3); where i = seg.

While if MatrixSelect is ‘0’ (i.e., the tracking matrix is selected) then:
Mn (i, 1) = Mn (1); Mn (i, 2) = Mn (1) OR Mn (2) OR Mn (3); Mn (i, 3) = Mn (3); where i = seg.

Finally, the data of the selected matrix row is latched by 3 Dff’s (one for each bit) which are triggered by PhiChip to obtain Data (0 To 2). This is done to ensure that the produced chip data are synchronized with the produced chip clock. Note, also, that in acquisition we send, to the PN correlator, the inter segment matched filter output as well as the intra segment ones, while in the tracking mode we are concerned only with the intra segment contents. This is because in the acquisition the pulse may reside in between two successive segments while in the tracking we already have a well centered pulse in a well known segment. A final observation about this block is the ORing action made in evaluating Mn(i,2) during tracking. This is smartly done to enable a continuous stream of data path (i.e., Data(2)) during the pulse drift (e.g., if the pulse is drifted from the position of Mn(2) to Mn(3) or to Mn(1) we will have Data(2) unchanged since it contains the ORing of the three positions). The continuous path of data is a requirement of the PN correlator in order not to lose any bit of information during tracking (as will be seen in the next section). The above argument is done assuming one user per segment, but it is still valid if we have multi user per segment, as for long PN sequences the right PN sequence can be distinguished even if it’s ORed with other PN sequences.

The next block in the signal path is the ZeroDetector. One of the main characteristics of any PN sequence that there can’t be any successive chips of zero value for more than (N-1) where N is the number of DFFs used for generating the sequence. Thus for our 31 chip PN sequence, it’s obvious that if window (or...
segment) contains no pulses for more than 4 successive chips then we can conclude that this segment contains no data (from our user or others) and the window must be shifted. The ZeroDetector starts counting the successive zeros of its input Data (0:2) just after the Zenable input signal goes logic HIGH and resumes counting until Zenable goes LOW. Some noise margin is taken for this process, as this block needs that Zenable remains HIGH for 15 successive PN chips to perform the zero detection tests. After the test, Zero is ‘1’ if the zero detection occurs and vice versa. Zenable and Zero signals contribute handshaking between the WindowMotherControl and the ZeroDetector.

Finally, we have to describe the main unit of the window sync system which is the WindowMotherControl block. This unit is responsible for controlling and monitoring the system mode of operation, positioning the window, supervising the zero detectors and the power control unit and handling the handshaking with the PN correlator. It’s implemented by a logical finite state machine (FSM) which is clocked by PhiChip. The main algorithm that this unit executes is:

1. Initiate the system
2. Enter the acquisition mode of operation:
3. Wait for 15 PN Chips to perform the zero detection, then check the Zero signal.

If Zero = ‘1’ then there is no pulse in this segment and we have to look at the next segment by incrementing SegIndex and loop to step2. While If Zero = ‘0’ then there is a pulse but we have to wait for the decision of the PN correlator whether this pulse belongs to our user or not, thus, go to step4.

4. The WindowMotherControl is now waiting for an estimated period of 6 data bits (i.e., 6 * 31 chips). After this period it checks the feedback signal (X) as follows (check the acquisition matrix section):

   If X= ‘000’ then there is no correct pulse in this segment, thus increment PhiSeg and loop to step2.

   If X = “100” then its M0 (i.e., the inter segment one) which gives the right pulse, and thus we have to shift PhiSeg in order to center this pulse in the segment. This is simply done by updating the value of PhaseSelect. Then go to step5 (i.e., tracking).

   If X = “010” then its M2 (i.e., the middle one) which gives the right pulse, and we do nothing with PhiSeg. Thus, go directly to tracking.

   If X = “001” then its M1 or M3 which gives the right pulse, and we have to do nothing (for the time being) with PhiSeg until we know which one of them is the right one. But, at least, we know that the pulse is now intra segment. Thus, go directly to tracking.

5. This is the tracking step, in which:
   Wait for other 2 data bits to investigate the effect of the changes made to PhiSeg.

6. Check the feedback signal X each PN chip. If X = “110” or “011” then the pulse has drifted, thus, PhiSeg must be updated corresponding to the value of X and loop to step6 after waiting for 2 data bits (long enough for the PN correlator to update itself without losing data).

   If X = “010” then M2 contains the right pulse, and we enter the deep tracking mode of operation. Thus, loop to step6.

   If X = “011” then M3 contains the right pulse, and the pulse has drifted to the right, thus, PhiSeg must be updated correspondingly and loop to step6.

   If X = “000” then the pulse has drifted outside the whole segment or a drift has occurred during one data bit, thus go to step7.

7. We have to wait one data bit to investigate the reason behind having X equal to “000”. The first half of this data bit is dedicated for the zero detector and the whole data bit is left for the PN correlator to take a decision. In both cases, if the zero test shows a zero detection or the PN correlator decides that the current pulse is no more belongs for our user, the system return back to acquisition (i.e., step2). If not, the system is back to tracking (step 6)

Since the UWB receiver must keep a low power usage, a power control unit is added to the system. The main criterion of this unit is as follows: in the tracking mode and as the right segment (i.e., in which the desired pulse resides) has been correctly obtained, there’s no need to read any other contents in the chip. Thus, the output of this block is a signal that activates the LNA and ADC only in the periods of the coming pulse in order to save power.

5 IMPLEMENTION OF PN CORRELATOR
The implemented PN correlator can monitor the 3 positions sent by the window sync at the same time. But first we will discuss the PN correlator receiving single position.

5.1 Used Algorithm
There are two modes of operation, which are the acquisition and tracking. In the acquisition mode, the system has 16 running correlators that correlate the received coded data with 16 different PN phases generated in the receiver tending to find the PN phase which is in phase with the received PN code.
The correlation time is 31 chip long (time for a PN code to be repeated), if no phase of the 16 generated phases proved to be the right one the generated phases are stopped for 16 chip (in order to be sure that the coming phase is one of the phases generated inside the receiver) and we begin the correlation again.

When we find the right phase the PN-correlator begins to operate in the tracking mode with only 3 running correlators to save power. These correlators correlate the received PN code with the right, preceding and trailing phases. This achieves continuous monitoring for the position of the right PN phase.

Since in the tracking mode we just have 3 running correlators, the system is only capable of tracking the code for a sudden shift of only one PN chip so that one of the running correlators can lock. If a larger step in the phase took place and the right phase was completely lost the system must return back to acquisition.

5.2 The Main Building Blocks
5.2.1 Mother Control
It is the block that controls all the other blocks and determines the mode of operation (acquisition or tracking)

5.2.2 PN Generator:
It is the block that is capable of generating 6 different phases of a given PN-code and so we must have 3 generators. These 6 phases and that certain code is determined according to the initial conditions sent by the mother control. Note that we can enable and disable the generator by ANDing the clock to a signal called enable generators.

5.2.3 PN Phase Logic:
It is responsible of correlating the received PN-code with one of the generated phases. If the correlation result is bigger than a certain +ve value or smaller than a certain –ve value, the block sends to the mother control a logic ‘1’ value in the signal called it_is_me indicating that the phase which have been correlated by the block is the right one. And it also determines what is the received data bit (‘1’ or ‘0’) according to the polarity of the correlation results. This block can be enabled or disabled by the Mother control.

5.2.4 Tracker:
This block works only in the tracking mode and is disabled in the acquisition mode by the signal Begin tracking which is sent by the mother control. It monitors the data and the it_is_me signals of the three phases used in the tracking mode and sends feedback signals to the mother control about the place of the right phase through the signal K plus it passes the data of the right phase to the output.

5.2.5 MUX:
According to the control signal coming from the mother control this MUX selects three outputs out of 16 inputs. And there are two MUXs to pass the three data and the three it_is_me signals of the phases monitored in the tracking mode to the tracker.

5.3 Blocks Interaction
5.3.1 Acquisition Mode
1st of all the mother control initializes the system as follows:
1. Disables the tracker by setting the Begin tracking signal to ‘0’.
2. Enable generators by setting enable generators signal to ‘1’.
3. Load the generators with 16 successive phases of the PN code
4. Enable all the phase control logic blocks
5. Setting the values of the internal variables and signals

Then the mother control waits for 31 phi chips which is the time needed for the phase control logic to complete its task (the correlation process). After this it reads all the it_is_me signals from all the phase control logic blocks. If no one of the 16 phase proved to be the right one this means that we must load the generators with the other 16 phases. This is done by disabling the generators for 15 chips(Fig.3).

5.3.2 Tracking Mode
To enter the tracking mode the mother control sets some signals as follows:
1. Enable the tracker by using the signal begin tracking

On the other hand if the right phases were found to be the n^{th} phase, the generators are disabled for 31-n chips, then the received PN code will be in phase with the phase ‘0’(Fig.4). At this time the generators are loaded with the phases (26, 25, 24 ...0 ...6, 7, 8) and so the received coded data will lock on the phase generated from the middle of the generators and this gives a bigger tracking range.

5.3.3 Blocks Interaction
5.3.4 Acquisition Mode
5.3.5 Tracking Mode
2. Disable all the phase control logic except the three monitoring the right, preceding and trailing phases.
3. Set the signal named Select MUX in order to pass the data and the it_is_me signals of the three phases.

While working in the tracking mode the tracker continuously monitors the status of the three phases (is one of them the right one? and who is this one?) and sends the feedback signal K to the mother control indicating the place of the right phases. When K=3 this means that no one of these three phases is the right one.

5.4 How To Monitor Three Pulses At The Same Time
In the previous case, it is described how to lock on a single input, in order to decrease the acquisition time of the overall system it would be better to monitor three inputs from three different time slots and sends a feedback signal to the window sync. Telling it which one of the three is the right user.

This can be easily achieved by not only using 16 Phase control logic but [3 * 16] Phase control logic. Each 16 Phase control logic monitors a single pulse in a certain time slot. The it_is_me signals of each 16 Phase control logic of every 16 Phase control logic are ORed and sent back to the window sync (the output of each OR gate resembles a bit in a bus named “X” [0 to 2]) to tell the window sync where the right user PN phase is. Then the right pulse is passed to the rest of the system by using a 3*1 MUX that is controlled by the signal “X”.

6. EXPERIMENTAL RESULTS:
6.1 Simulation Results Of Window Sync Sub-System
As illustrated in Fig.5 and Fig.6, our user (DATA6) is periodically transmitting its pulses in segment number 2 (i.e., the third segment of the chip) for a certain period then he ceases to transmit. A note must be declared here, that the user transmits its pulse in segment 2, but the system senses, and thus considers, it, logically, in segment 3, this is due to the one segment latency countered in the Serial To Parallel block. There are 3 other users that send their pulses in the other three segments. RXDATA is the received data which is, essentially, the ORing of the the four users pulses. Fig.35 gives the total view while, Fig.37 focuses only the last section of the scenario, i.e., the tracking mode of operation.

6.1.1 Examining simulation results :
First of all the system is in the acquisition mode of operation (i.e., at the leftmost of Fig.5), where:

Acq_TrcBar and MatrixSelect signals are HIGH, indicating the acquisition mode of operation. LNAenable signal during acquisition is HIGH all the time because the position of user’s pulse isn’t known. X signal is “000” indicating that the user pulse is not found yet.

The system is now reading segment number 1 as indicated by the value of SegIndex which is “01”. As the pulse which exists in segment 1 belongs to another user then the feedback signal X is “000”. The window sync system recognizes the value of X and then decides to inspect the next segment which is segment 2. The zero detection test is being made first. Zenable goes high for a period of 15 chips, then, the Window Mother Control checks the Zero signal value which is ‘0’ at this time, i.e., there is a pulse in this segment.

As a result, the control will wait for another 6 data bits before it checks the value of X which is, also, “000”. The system now examines segment 3 (the user segment), but our user is not transmitting at this instance. Thus the zero detection test will show that there is no pulse exists in this segment, and, the control will decide to read from the next segment which is segment 0. The same procedures are done sequentially until, finally, the SegIndex (or SEG# is 3 -or ‘11’) while our user is transmitting. In this case the system will enter the tracking mode of operation in which:

Acq_TrcBar and MatrixSelect signals go LOW, indicating the tracking mode of operation. LNAenable signal is now being selectively HIGH only for periodic times while DATA6 is being transmitted. X signal is “010” indicating that the user pulse is now centered in the segment. Here the user pulse is already centered without any modifications of our own.
6.2 Simulation Results Of PN Correlator Sub-System:

6.2.1 Transition from acquisition to tracking:
Initially the begin tracking signal was set to ‘0’ declaring the acquisition mode till a certain phase (n) locks to the received phase (the_it_is_me of this phase was set to ‘1’). So the generators are disabled for a certain while (31-n chips) and it can be seen that the generators phases are kept constant for this period. After this period the generators are loaded by the signal load (raised to ‘1’) putting the phase ‘0’ in the middle. At the same time the begin tracking is raised to ‘1’ to enable the tracker. After 31 chip the_it_is_me signal of the middle phase is raised to ‘1’ indicating that it is the right phase.

6.2.3 Normal operation:
It can be easily seen that the output data is equal to zero at the end of the correlation time when the generated code is inverted with respect to the incoming one, And vice versa.

6.2.4 Transition from tracking mode back to acquisition mode:
At the beginning, the receiver was in locking state with the user code. This can be seen from observing the status of the signals Begin_tracking = ‘1’, K = “01”, It_is_me of the middle phase = ‘1’. Due to sudden ceases of the user to send its data, i.e., no right coded data to lock on, the phase control logic send zero (it_is_me of the middle phase ) signals residing with sending of the tracker to the K signal with status “11” indicating no locking. Even though, the system does not decide to return back to acquisition due to the suspected noise effect and wait for 3 data bits’ time instead. If the “K” signal remains the same, the system decide to return back to acquisition mode setting the “Begin_tracking” signal to ‘0’.

7. Conclusion

It is clear that this design can fulfill the following features:

1. Minimized acquisition time, typically 24 data-bits (6 data-bits * 4 segments) for preamble in the worst case.
2. Enhancing acquisition time applying zero detection method.
3. No data lose during tracking mode.
4. The system applies many power saving techniques. It controls its own power as well as the ADC and LNA power.
5. Area efficient by applying the time interleaving concept all over the system.
6. Dividing the system architecture into two major blocks makes the system highly flexible in the sense of:

- The system can work on different modulation schemes (e.g., OOK, BPSK or PPM) and different pulse shapes (e.g., normal Gaussian or doublet) with only little modifications in the window sync system without touching the PN correlator.
- The PN sequence length can be scaled up by scaling up only the PN correlator while the window sync system is almost not affected.