

Design of a Practical Scheme for Ultra Wideband Communication

Yiyin Wang, Rene van Leuken, Alle-Jan van der Veen

Circuit & Systems, Department of Electrical Engineering
Delft University of Technology, the Netherlands

Abstract—In the design of a packet-oriented impulse-radio UWB communication system, the main challenge at the receiver is to have a fast synchronization to the coded pulses, along with a detection of the message. We consider schemes that are straightforward to implement in practical systems and propose two methods to realize the synchronization algorithms: a serial and a parallel method. The algorithms for synchronization and demodulation are implemented in a receiver prototype based on an FPGA.

I. INTRODUCTION

The overlay on existing frequency allocations, along with promises of high data rates, low cost and low complexity, makes ultra-wideband (UWB) an attractive technology for wireless communication. A practical UWB communication scheme is given by the delay-hopped transmitted-reference communication system (DHTR system) proposed by Hoctor and Tomlinson [1-3]. It is based on the transmission of pairs of pulses whose correlation carries the information: this is unchanged after convolution by the propagation channel since both pulses experience the same distortion. For synchronization and detection, the individual channel coefficients do not have to be estimated, which makes this a much more attractive scheme than some of the proposed rake transceivers.

Our aim in this paper is to consider a practical implementation of the digital parts of the DHTR system on an FPGA prototype board. We propose two hardware architectures for detection and synchronization: a serial and a parallel architecture. We will show the synthesis results for the serial architecture: this gives an indication on how fast the transceiver can be and how many resources it employs.

This research is part of the AIRLINK project [4] where other work packages consider the antenna design, analog electronics, and communication/networking layers. In the design, the relatively low clock speed of an FPGA is offset by its high degree of parallelism and I/O capabilities, so that nonetheless an acceptable data rate can be achieved.

The paper is organized as follows. In Section II, we introduce the system structure of the DHTR system, its working principles, and explain how we model a simplified version of the system. In Section III, algorithms for synchronization and demodulation are presented. We propose two architectures to implement the algorithms. Section IV shows the synthesis results for the serial architecture. Finally, conclusions are drawn in Section V.

II. DHTR SYSTEM

The delay-hopped transmitted-reference communication system transmits pulses in pairs (as a doublet). The first pulse is used as a reference and the second is used to carry the information. The pulses are separated by a short time interval, which is known by both the transmitter and the receiver in advance. This separation changes from doublet to doublet according to a user-specific “delay code”. The analog part of the receiver correlates the received signal with several time shifts using a bank of delay lines, integrates the results, and subsequently samples the outputs for digital synchronization and demodulation. Thus, the analog parts of the system do not contain any time-dependent or parametric parts, which is important since they run at maximum speed.

The transmitted messages are represented by symbols. A symbol is composed of several doublets, where, similar to CDMA systems, each doublet represents a chip. Fig. 1 shows an example of the signal pattern to transmit one symbol. In the design of our system, a symbol s consists of 8 chips $c[k]$, $k=1, \dots, 8$, where each chip is a frame with a duration of 20ns. In the frame, two narrow pulses $g(t)$ form a doublet $d(t)$. The first pulse is a fixed reference pulse, and the second has a polarity which depends on the symbol s and is modulated by $c[k]$. The doublet can be formulated as

$$d(t) = g(t) + c[k] * s * g(t - D[k]) \quad k=1, \dots, 8. \quad (1)$$

Here, s is the symbol value and can be -1 or 1; $c[k]$ is the chip value, which can also be -1 or 1, and constitutes a user-specific code. $D[k]$ is the user-specific delay time sequence, with values that are a multiple of 0.5ns. In our system, we use 5 possible values, $D[k] \in \{0.5ns, \dots, 2.5ns\}$.

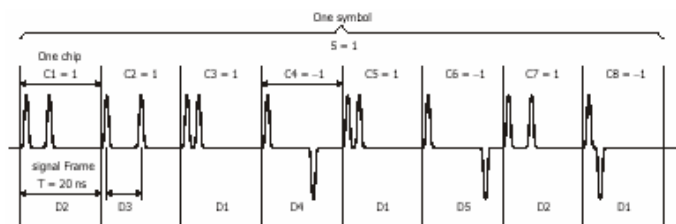


Figure 1. The structure of one transmitted symbol

A block diagram of the transceiver system is shown in Fig. 2. The upper part of Fig. 2 indicates the transmitter and the lower part depicts the receiver. The pulses transmitted by

the antenna in the transmitter go through the wireless channel and are received by the antenna in the receiver. The dashed line divides the digital part from the analog part of the transmitter and the receiver. All the digital parts are implemented on an FPGA.

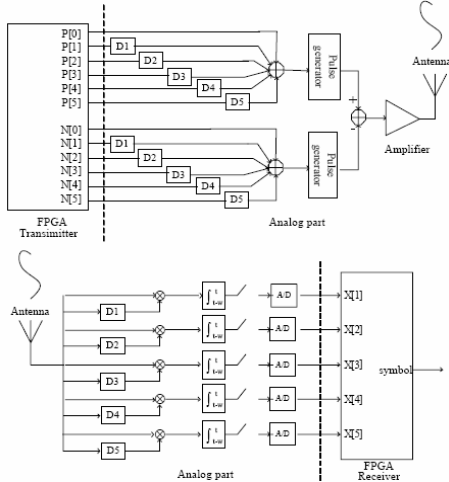


Figure 2. Schematic block diagram of the DHTR system

Twelve output pins of the FPGA are used to represent the two pulses in a frame and their displacements. In particular, binary pins P_0, \dots, P_5 represent the presence of a positive pulse at displacements of $0, \dots, 5$ times $0.5ns$, and similarly N_0, \dots, N_5 represent the presence of negative pulses. If there is a signal to transmit, then either P_0 or N_0 will be logically '1' to represent the reference pulse, and only one of $P_1, \dots, P_5, N_1, \dots, N_5$ will be '1' to represent the signal pulse and its time offset. The time shifts are implemented by analog delay lines ($D1, D2, \dots, D5$ in the figure; in practice, we use a slightly different scheme with only a single tapped delay line). The delayed signals are added together and sent into the pulse generators (one for a positive pulse and a separate one for a negative pulse), which generate narrow pulses in sequence. The analog pulses are added together, sent into the amplifier and transmitted by the antenna. The pulses are transmitted and then received by the antenna in the receiver. The received signals go through a bank of delay lines (the same delay periods as used in the transmitter), are correlated, integrated over a period of $20ns$, and sampled by an A/D converter. This procedure gives a strong positive (or negative) response in those delay branches that match the delay of the transmitted doublet, and approximately a zero response for other (non-matching) delays. The digital samples are sent to the digital part of the receiver implemented on an FPGA for synchronization and demodulation.

To be able to test the digital part of the receiver system, we have also implemented a simple channel emulator, running on an FPGA, as shown in Fig. 3. It has inputs $P_0, \dots, P_5, N_0, \dots, N_5$ as generated by the digital part of the transmitter, and generates the corresponding outputs X_1, \dots, X_5 suitable for the digital part of the receiver. The implemented emulator uses an ideal response:

$$X_j[k] = A_{ij} * c[k] * s \quad A_{ij} \geq 0; \\ k = 1, \dots, 8; \quad D[k] \in \{0.5ns, \dots, 2.5ns\}; \\ i = D[k]/0.5; \quad j \in \{1, \dots, 5\} \quad (2)$$

where A_{ij} is a gain parameter which depends on the transmitted delay index i and the received delay index j , and is related to a channel correlation coefficient. Ideally, if $i=j$, $A_{ij}=A$, else $A_{ij}=0$ [3].

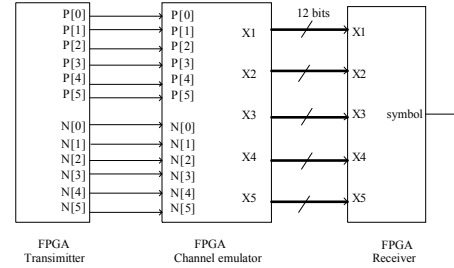


Figure 3. DHTR system emulator

III. RECEIVER SYNCHRONIZATION AND DEMODULATION

A. Detection and synchronization

At the receiver, the user code and delay time sequence are known information and they are repeatedly used for every symbol. The first thing to do at the receiver is to synchronize, i.e., to detect whether we have received a valid user signal and to find the position of the first chip of a symbol, taking into account an unknown integer delay. The detection is done by matching the desired user code (chip code and delay code) with a single symbol. Every symbol is composed of 8 chips, and since each chip corresponds to a sample, there are 8 possible positions for the first chip. We have to check them all to find the best match, which corresponds to a maximum correlation with the code. If a message was transmitted, then for each sample, one of X_1, \dots, X_5 will have a value of $+A$ or $-A$, namely the output corresponding to the transmitted delay between the pulses of that frame. If we are synchronized, then for the k -th chip, we choose $X_j[k]$ according to the user specified delay time sequence, $j=D[k]/0.5$, multiply with the corresponding chip value $c[k]$, and sum the results over 8 chips: this gives a matched output of $r = 8As$. From r , we can estimate A as $1/8 |r|$. If we are not synchronized, the samples will not add coherently, and r and the estimated A will be a small number. Thus, at the proper synchronization position, we will get the maximum estimated A . To synchronize, we check all 8 possible offsets, get 8 estimated values for A and find the maximum one— \max_A , and the corresponding position— \max_index .

To test whether there was a signal at all, we need to compare the maximum estimated A to a threshold value. This threshold value can be determined by analyzing the variance of the value that we will obtain in the case of noise-only. After choosing a desired false alarm level, it can be determined with the help of statistical signal processing theory. This would require knowledge of the noise power. In the absence of this information, we use the average value of

the estimated A over all possible positions as the threshold. If the maximum estimated A is α times larger than the average, we decide that we have received a desired user signal, otherwise it is just noise. Again, the correct α should be determined using statistical signal processing theory. In the text below, we use $\alpha=3$.

After synchronization, we can demodulate the symbol. In fact, the algorithm for demodulation is similar to the algorithm for estimating A , since it requires the correlation sum r . At the correct position, the receiver starts to demodulate. It chooses $X_j[k]$ according to the user specified delay time sequence, multiplies it with the user code and accumulates the products. After accumulation for 8 chips, the sign of the sum is the demodulated symbol. The demodulation is described by the following pseudo code:

```

r = 0;
for (k=0; k<8; k++)
    j = D[k]/0.5;
    r = r + Xj[k]*c[k];
symbol = sign(r);

```

B. Serial synchronization architecture

Following the algorithm described above, we propose two FPGA implementations structures for synchronization. The first one is a serial architecture. The 8 possible offset positions are checked in a serial way. For each offset, the correlation calculations are the same. Since they do not happen concurrently, the corresponding resources can be reused which saves a lot of area. The operations are assigned to every cycle according to the algorithm. Fig. 4 shows the arrangement for all the cycles. In the first cycle, the first sample read in is used as the starting point. The receiver reads in a new sample every clock tick of 20ns. As shown in Fig. 4, the numbers in the first row of every group is the cycle range. The numbers in *italic* represent the sequence of each batch of 8 samples. The numbers in **bold** represent the sequence of 8 chips we use to compute r and estimate A . Each batch of 8 chips starts at a different offset position, by skipping one sample in between. The numbers in the right column show the corresponding start position of the batch in black referred to the batch in *italic*. For example, in the row of the second batch of 8 samples in *italic*, the **bold 0** corresponds to the *italic 1*. At this time we check position 1 as the start point. The stars * are the skipped samples in order to have some offset to check different positions.

At the clock tick corresponding to a skipped sample, we have time to (i) calculate the absolute value of the estimated A , (ii) accumulate the value for the computation of the mean, and (iii) compare the absolute value with max_A , which is the record of the maximum A seen so far. If the new A is larger than this maximum, we update max_A and max_index . The comparison and the accumulation can be done concurrently because they use different function resources without any data dependency. For each batch of 8 samples, an accumulation is done to get the estimated A and every cycle there is an addition operation. To balance the operation of every cycle, we can schedule the calculation of $3*avg_A$,

which is used as a detection threshold, at cycle 73 as shown Fig. 4.

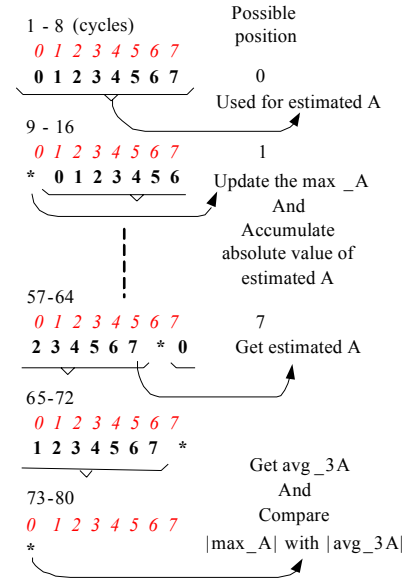


Figure 4. Serial synchronization: arrangement of the cycles

At cycle 73, we decide whether the signal was detected. If not, we continue another scanning round, otherwise we skip a specific number of samples to synchronize to the computed offset position (max_index) and start to demodulate. For example, if the proper position is 7, then we skip 7 samples and start demodulation at cycle 80. In the most ideal case, 74 cycles are needed for synchronization.

A disadvantage of this scheme is that synchronization requires 9 symbol periods, even if effectively only a single symbol period is used to detect the signal: this is not efficient from a signal processing point of view.

C. Parallel synchronization architecture

Since FPGAs have abundant resources, an alternative architecture is a parallel one. We can check all 8 possible offset positions concurrently and thus reuse the same samples for different positions. Fig. 5 shows the function blocks for this kind of receiver. For each possible offset position i , there is a function block P_i ($i=0, \dots, 7$) that operates on the samples starting with a corresponding offset, as triggered by a time-shifted 'start' signal. Every P_i block calculates the average estimated A (signal A_i in the figure), demodulates the sample (signal S_i), and updates the average value of the estimated A every 160ns (8 chips or 1 symbol period) at consecutive moments. These estimated A 's are compared to a threshold value to detect whether there was a signal and are also used to find the correct offset position. A selector function block selects the corresponding signal. This is the idea to do the synchronization using a parallel architecture. The serial method uses fewer resources, but takes a longer time to synchronize and is not efficient from a signal-processing viewpoint. The parallel method uses more resources, but is faster in synchronization: in fact it will detect the beginning of a packet as soon as it arrives.

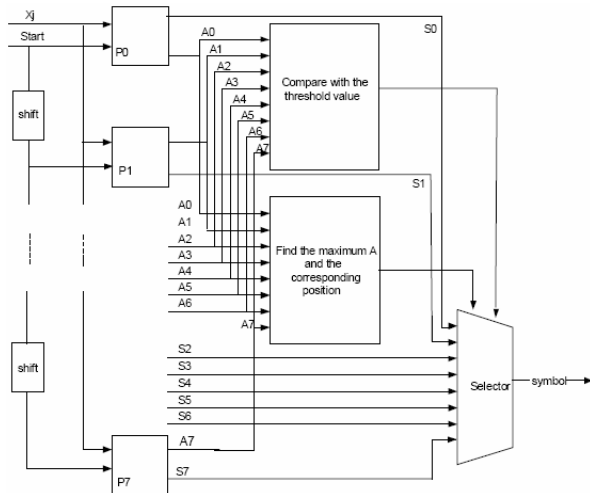


Figure 5. Receiver using parallel synchronization architecture

IV. RESULTS

In order to test the functionality of the DHTR system, we encapsulate it into a VHDL soft-core of an Atmel AVR micro-controller environment (shown in Fig. 6). The whole environment is implemented on a Xilinx Spartan3 (xc3s1500fg676-4) FPGA. A C-program, running on the AVR, enables communication between a user and the DHTR system. The user inputs a message using the keyboard. The message is transmitted to the AVR, translated into symbols by C program and sent to the transmitter hardware. The demodulated symbols from the receiver are collected by the AVR. They are translated into a received message and shown on the screen.

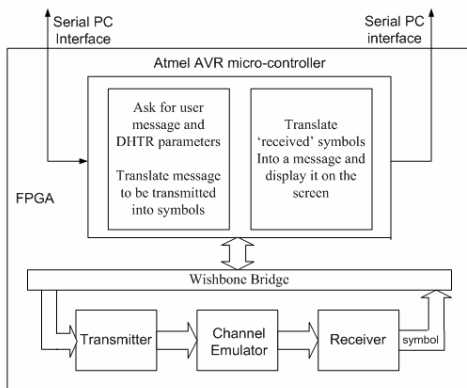


Figure 6. The architecture of the testing platform

The transmitter, the channel emulator and the receiver with the serial synchronization architecture are implemented on the FPGA as a Wishbone client. The total number of lines of the VHDL description for this DHTR system is less than 1500. The synthesis results of the system with the serial synchronization architecture are shown in Table I. As seen in the table, the system with the serial synchronization employs only a few resources, as indicated by the number of

TABLE I. THE SYNTHESIS RESULTS OF THE DHTR SYSTEM WITH THE SERIAL SYNCHRONIZATION ARCHITECTURE

	Achieved Frequency	Equivalent logic gates	Equivalent system gates
Transmitter and Channel emulator	171.1MHz	52	2080
Receiver	74.6MHz	233	9320

equivalent logic gates. As mentioned before, a disadvantage of this scheme is its slow acquisition time: it will take at least $(c+1)*c+1$ (where c is the number of chips per symbol) times the cycle time to complete one scanning round. As c increases, this time increases quadratically. This is not desirable from a signal-processing viewpoint. From these numbers in Table I, we can extrapolate that the number of resources used in the parallel receiver will be about $c*233$ gates. The time spent in one synchronization cycle is $c+(c-1)$. When c increases, the area and the acquisition time both increase linearly. In most cases, this is a better solution. According to the specific application, we can combine these two architectures to make a tradeoff between the acquisition speed and the chip area.

V. CONCLUSIONS

This paper presents the design of the digital part of a Delay Hopped Transmitted Reference UWB communication system. It is a mixed signal system. The transmitter modulates the symbols according to a user specified code: a polarity and a delay time sequence. The major challenge in this design is to obtain synchronization at the receiver. Two ways are proposed to do synchronization: the serial method and the parallel method. The serial method uses fewer resources, because it can share the calculation resources when checking all possible positions in serial, but it takes a relatively long time to synchronize. The parallel method uses more resources, because it checks all possible offset positions concurrently, but it is therefore able to achieve synchronization within a single symbol period. The design was targeted towards implementation on an FPGA development boards with a Xilinx Spartan3 device. We built a complete test environment to verify our design. The test results proved the functionality of the design was correct.

REFERENCES

- [1] R. Hoor and H. Tomlinson, "Delay-Hopped Transmitted-Reference RF communication", IEEE conference on Ultra Wideband Systems and Technologies, 21-23 May 2002, pp. 265-269
- [2] N. van Stralen, A. Dentinger, K. Welles II, R. Gaus Jr., et al., "Delay Hopped Transmitted Reference experimental results", IEEE conference on Ultra Wideband Systems and Technologies, 21-23 May 2002, pp. 93-98
- [3] A. Trindade, Q. H. Dang and A. J. van der Veen, "Signal processing model for a transmit-reference UWB wireless communication system", IEEE conference on Ultra Wideband Systems and Technologies, 16-19 Nov. 2003, pp. 270-274
- [4] The AIRLINK project, TU Delft, Netherlands: <http://www.airlink.tudelft.nl>