Synchronization algorithm and FPGA implementation for Transmit-Reference UWB receiver

Hai Viet Nguyen, Manh Hoang Tran
School of Electronics and Communications
Hanoi University of Science and Technology
EDA Group - C9 401
Email: vietnhk52@edabk.org, tmanhhoang@gmail.com

***Abstract*—This paper proposes a practical synchronization algorithm for Transmit-Reference UWB receiver which uses sliding window and supports flexible sampling rates. Additionally, a Simulink model and implementation of synchronization algorithm in receiver by hardware description language (HDL) is also developed using model-based design. Finally, we assess impact of signal-noise-rate (SNR), number of bits quantization and sampling rate on bit-error-rate (BER).**

***Index Terms*—Transmitted-Reference, Ultra-Wideband, model-based design, synchronization algorithm.**

I. Introduction

Since approved by FCC in 2002 [1], Ultra-Wideband has received much attention in both academia and industries. Transmit-Reference transceiver, first proposed in [2], [3], is one of the more practical schemes with reduced receiver's complexity. Data symbols can be detected without channel coefficients estimation at the expense of lower data rate and inferior noise performance (due to cross-correlation terms between signal and noise).

Several published papers develop and extend the concept of TR-UWB to support higher rates, multiple delays, multi-users [4]–[6]. However, not many TR-UWB deal with synchronization and pratical implementation issues. In [7], the synchronization algorithm is proposed for asynchronous multiuser TR-UWB but it already assumes perfect synchronization at frame level. In [8], the authors propose both sample-level and symbol-level timing methods for the modified TR-UWB system in which pulses are more closely spaced. However, the algorithms' complexities grow quickly because of the large matrix pseudo-inverse and Fourier transform.

This paper focuses on a more practical synchronization algorithm for the generic TR-UWB which has only one frame per symbol and assumes no interference between pulses or frames. Moreover, we implement the algorithm in FPGA and propose a method using model-based design to develop, simulate and test the whole TR-UWB system by combining Simulink and HDL together. Many practical issues including quantization bits, sub-Nyquist sampling rates are also considered.

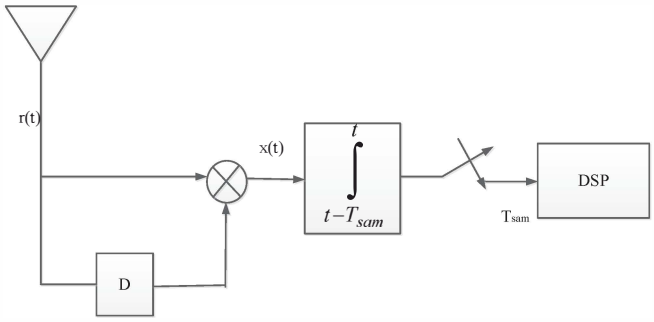

Fig. 1. TR-UWB receiver

II. TR-UWB transceiver

We consider TR-UWB transceiver in its most generic form where BPSK symbols s_i are mapped into frames, each of which consists only two pulses separated by D second. The first pulse is the reference $g(t)$ (Gaussian mono-cycle), while the second pulse, delayed by D, contains information $s \cdot g(t-D)$. The frame period T_f and the delay value D are chosen such that there is no interference between consecutive pulses or consecutive frames (i.e. no IPI and no IFI). For simplicity, we choose $T_f = 3D$ and $D > T_h$, where T_h is the maximum channel delay spread. The transmitted signal is:

$$y(t) = g(t) + s \cdot g(t-D) \tag{1}$$

The received signal (at antenna's output) is:

$$r(t) = h(t) + s \cdot h(t-D) + n(t) \tag{2}$$

where $h(t)$ is the convolution between UWB pulse $g(t)$ and the physical channel impulse response $h_p(t)$: $h(t) = g(t) * h_p(t)$, and $n(t)$ is the white Gaussian noise.

Since there is no IPI and no IFI, by cross-correlating the signal with the delay-by-D version of itself as in Fig. 1, we have

$$x(t) = r(t)r(t-D) \approx s \int h^2(t)dt + \text{noise} \tag{3}$$

978-1-4673-2493-9/12/$31.00 ©2012 IEEE

Therefore, data symbol can be simply detected by

$$\hat{s} = sign\left\{\int_{T_f} r(t)r(t-D)dt\right\} \tag{4}$$

III. Synchronization algorithm

TR-UWB receiver uses integrate and dump operation (followed by a comparator) to detect data symbols. While this is a very simple operation with low sampling rate (one sample per frame), it still needs to know where to start integration and where to dump properly. From Fig. 3 illustrating signal $x(t)$ after cross-correlating, we can see that in the whole frame, there is only one sub-frame from D to $2D$ which contains useful information $sh^2(t)$ while other sub-frames contain only noise and the cross-correlating terms between signal and noise. From now on, the sub-frames which contain information are also called as the data sub-frames.

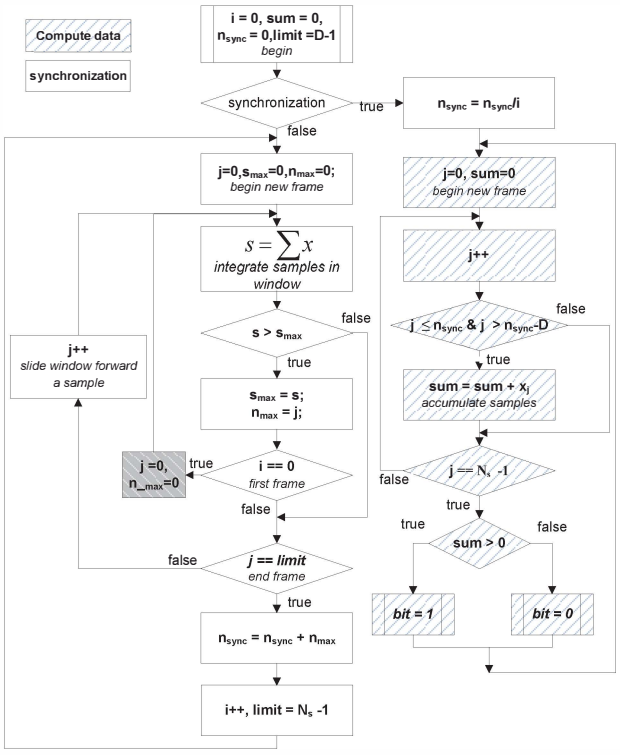

Fig. 2. Synchronization algorithm flow chart

The synchronization algorithm aims at: (i) determining the start of frames and (ii) detecting the data sub-frames, given an arbitrary received signal segment. While we can synchronize by detecting the peak of $x(t)$ in Fig. 3, the result will be poor when noise becomes comparable to signal level. By noticing that all the $sh^2(t)$ are with the same polarity (because $h^2(t)$ is always positive), we can collect all these channel coefficients' energy together by using sliding window method. The main idea of our proposed synchronization algorithm is to use sliding window one third of a frame size to accumulate received signal power and then detect the peak value to achieve both (i) and (ii) at the same time.

As illustrated in the algorithm's flow chart in Fig.2, when the first sample (can be anywhere in the frame) arrives at the receiver, synchronization mode starts and the sliding window integration is used with size $N_D = \frac{N_f}{3}$, where N_f is number of ADC generated samples per frame. Additionally, a variable counter is needed. In the first frame, this variable counter is set to 0 with its upper limit N_D, and the first sample is within the integration window. When another sample arrives, the variable counter increases by 1 and the window is slid forward one sample. All of samples in the window are integrated as in (5). Subsequently, the absolute value of the resulting integration s is compared with the present one s_{max}. If $|s| > s_{max}$, then $|s|$ is saved as s_{max} and the variable counter is reset to 0.

$$s = \sum_{j=0}^{D-1} x_j \tag{5}$$

When the variable counter reaches its limit N_D for the first time, it means the integration window is position exactly over the end of first frame. Therefore, the start of second frame is nearest to the start of frame as shown in Fig. 3. From second frame, limit of variable counter equals N_f. We continue to integrate all of samples in window and get from variable counter n_{max} value at which $|s|$ is maximum . At the end of each frame, counter equals limit, we add n_{max} to n_{sync} . Finally, when synchronization mode stops, n_sync is computed by (6).

Now that once synchronization is achieved, the receiver returns to its data mode by using integrate and dump, and detects data symbols by taking sign of sum in (7).

$$n\_sync = \frac{\sum_{j=0}^{\ell-1} n\_max_j}{\ell} \tag{6}$$

where ℓ is the number of synchronization bits.

$$sum = \sum_{j=n\_sync-N_D+1}^{n\_sync} x_j \tag{7}$$

IV. Implementation

Model-Based Design is transforming the way engineers and scientists work by moving design tasks from the lab and field to the desktop. It enables faster, more cost-effective development of dynamic system, including control systems, signal processing, and communications systems. In Model-Based Design, a system model is at the center of the development process, from requirements development, through design, implementation, and testing. The model is an executable specification that is continually refined throughout the development process. After model development, simulation shows whether the model works correctly. It is used for systems that include software (Simulink) and hardware (hardware description language-HDL).The Model-Based Design flow is shown in Fig. 4

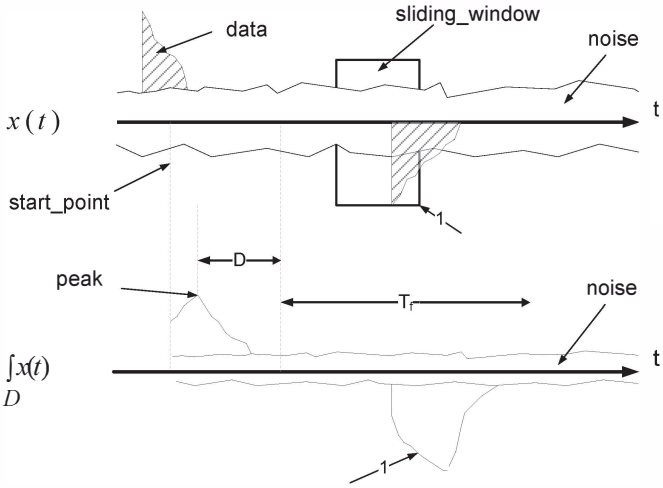

Fig. 3. Synchronization example

In our TR-UWB system, both software and hardware are included. Transmitter, channel and some parts in receiver are implemented in Simulink. Synchronization algorithm in receiver is implemented using Verilog HDL.

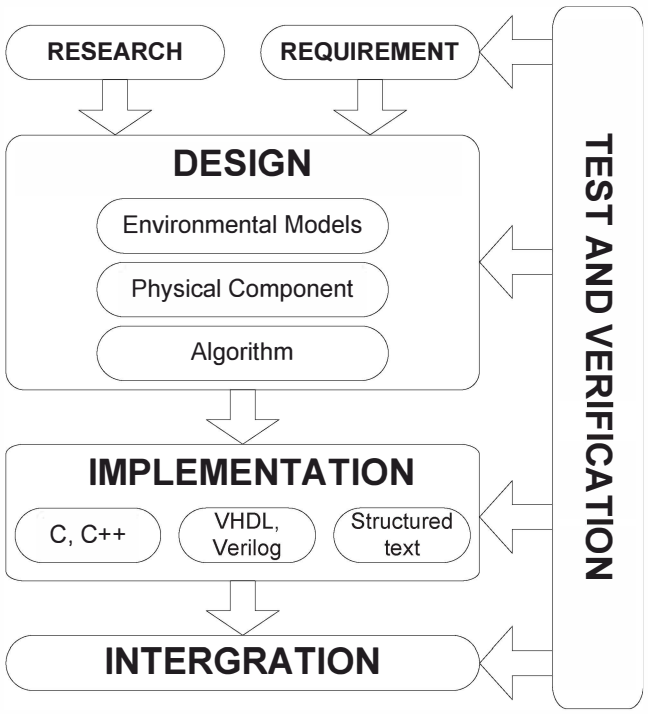

Fig. 4. Model-based design flow

A. Design

As analysed above,our TR-UWB system consists of transmitter, channel and receiver. At transmitter, randomly generated bit stream is created and used to control Gaussian pulse sequence. This signal is then sent through the channel block which includes: CM1 channel, white Gaussian noise. At receiver block, the received signal is multiplied by the delay-by-D version of itself .

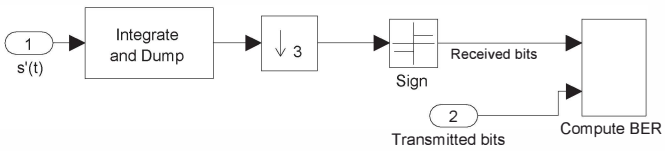

Fig. 5. Receiver model in case of perfect synchronization

In case of perfect synchronization assumption, signal go through "Result and Calculate Error" block, which execute tasks as follows: sampling, demodulating signal and computing BER. Fig.5 shows the structure of this block in Simulink.

In case of synchronization algorithm, signal arrives randomly in a frame. After passing through the multiplier, it goes to baseband processor shown in Fig. 6. In here, synchronization will be executed. The baseband processor consists of two parts: $datapath$ and $controller$. All of $datapath$ processing are controlled by $controller$ block (Fig. 10) which has Finite State Machine flow shown in Fig. 11. There are three state of receiver: (i)$Initial$ before starting operation,(ii) $Synchronization$ starting on processing signal and executing synchronization algorithm, (iii) $Caculation$ calculating data after synchronization.

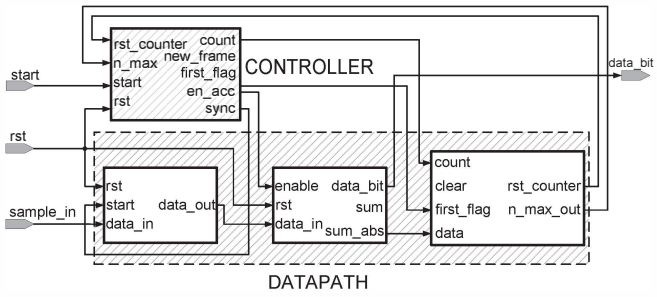

Fig. 6. Architecture of baseband processor

First, state of receiver is $Initial$, $datapath$ do not operate. Then, state of receiver changes to $Synchronization$, $datapath$ performs accumulation of samples in sliding window and peak determination. Accumulation processing is executed as follows:

$$s_{n-1} = x_{n-1} + x_{n-2} + ... + x_{n-N_D+1} + x_{n-N_D} \quad (8)$$

Next, all value in memory will be shifted right, and the next sum is :

$$s_n = x_n + x_{n-1} + ... + x_{n-N_D+1+2} + x_{n-N_D+1} \quad (9)$$

From $(8), (9)$, we have :

$$s_n = sum_{n-1} + (x_n - x_{N_D}) \quad (10)$$

After signal is sampled and quantized, it will be sent to $sliding\_window$ block (Fig. 7) which calculates $x_n - x_{N_D}$

is shown in (10). It has a memory and a subtracter. At time n, sample $(n - N_D)^{th}$ is out of the memory and sample n^{th} go to this block. We subtract sample n^{th} from sample $(n - N_D)^{th}$. Result will be sent to $accumulator$ block (Fig. 8) to implement accumulation as (10). Next, $|s_n|$ is computed in $accumulator$ and sent to $detect_max$ block (Fig. 9) which determines peak. When the next $data\_sum\_abs$ goes in, it is compared with the current maximum value. If it is greater than the current maximum value, the current maximum value will be change by $data\_sum\_abs$ at the next clock and the position of $data\_sum\_abs$ will be saved. After each frame, a value of the position n_max, where the sum is maximum, is outed and came to $controller$. In here, n_max value is added together.after that, result is shifted m bit, with m is number of synchronization bit as (6).

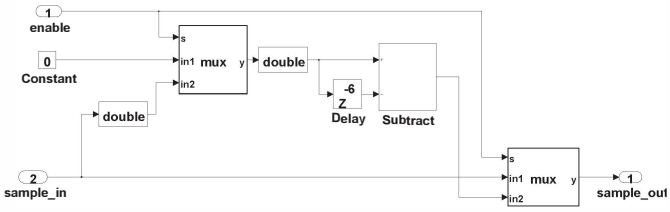

Fig. 7. $Sliding\_window$ model

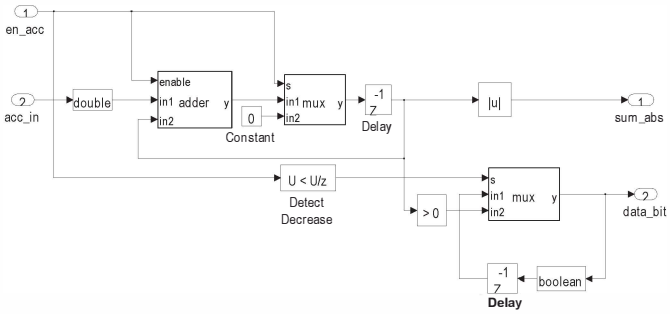

Fig. 8. $Accumulator$ model

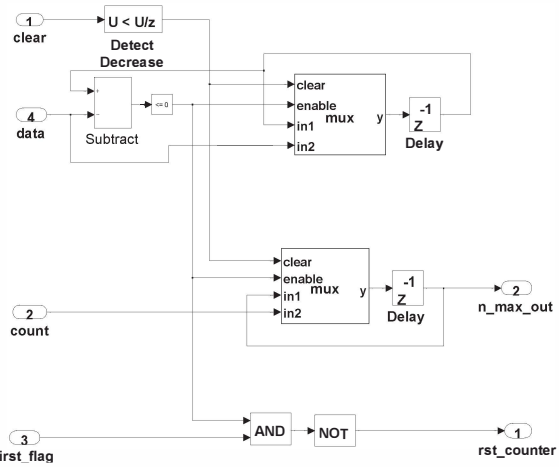

Fig. 9. $Detect_max$ model

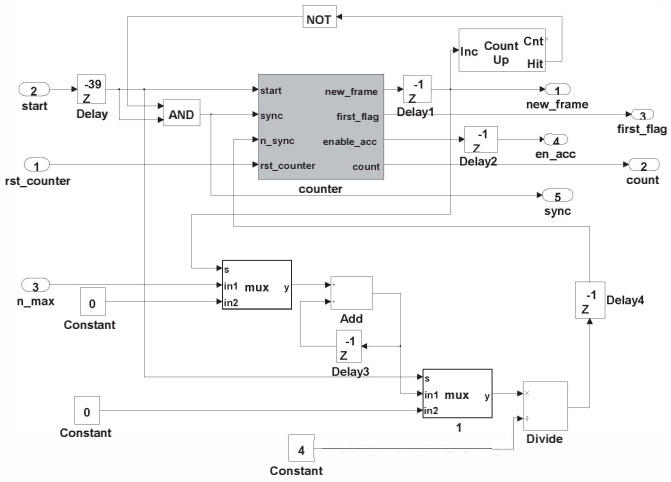

Fig. 10. $Controller$ model

Finally, $controller$ uses value n_sync to direct operation of $accumulator$ and $sliding_window$ block in $Caculation$ state. They only accumulate samples in one third of frame which carries data.

B. Implementation

In this section, the baseband processor above is implemented using Verilog HDL. Our top-module has four inputs and one output, shown in TABLE I. Input value of $sample\_in$ port is an m-bit integer array, which is the output of the quantizer with input $x(t)$.

In this paper, all blocks shown in Fig. 6 is described by Verilog HDL. The Verilog files will be add to $Black_Box$ blocks that can be used to incorporate either VHDL or Verilog HDL into the Simulink model. Fig. 12 shows the models of our design: $sliding\_window$, $accumulator$, $detect\_max$ and $controller$. Moreover, we add $Reset\&Start$ blocks which create reset and start signals, $Gateway\_in$ blocks sample and convert input of type Simulink to Xilinx fixed point type. Our design runs with input clk. But to synchronize with system, this input is hidden by adding a ce signal to the Verilog file. After that, we generate a $HDLhwcosim$ block and insert into our receiver.

TABLE I
IN/OUT SIGNAL OF OUR DESIGN

Port	In Out	Type	Number of bits
clk	input	Boolean	1
rst	input	Boolean	1
$start$	input	Boolean	1
$sample\_in$	input	Signed	m
$data\_bit$	output	Boolean	1

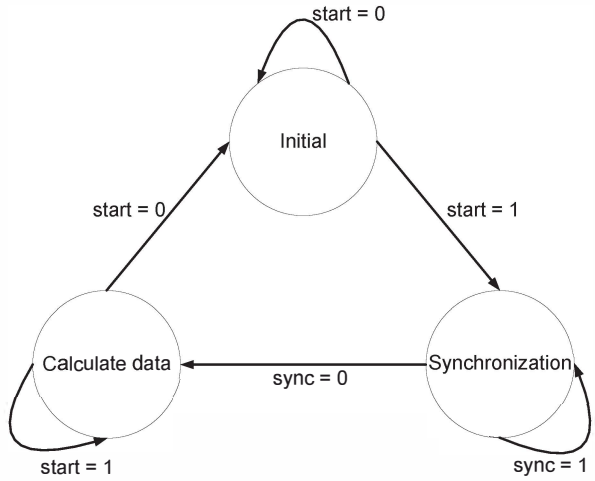

Fig. 11. FSM flow

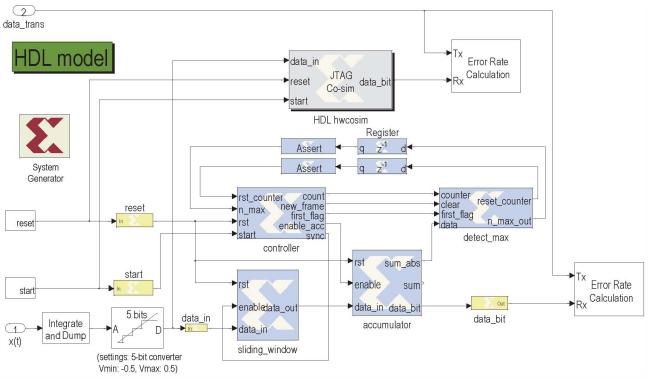

Fig. 12. HDL Design in Simulink model

(SNR). BER performance are compared in following cases: C1 - perfect synchronization assumption (in Simulink), C2 - synchronization algorithm (in Simulink), C3 - HDL implementation without Integrate and Dump, C4 - HDL implementation with Integrate and Dump. In C3 and C4, plots of BER vs. number of quantization bits and sampling rate are shown. Fig. 13 shows the BER performance gain of TR-UWB in C1, C2, C3 and C4.

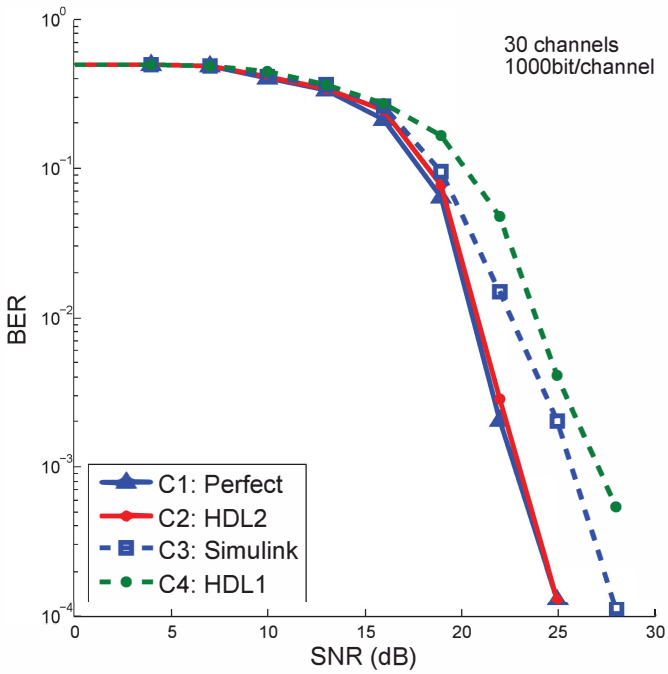

Fig. 13. BER vs. SNR

V. Synthesis and Simulation

A. Synthesis

We synthesis my design by Xilinx ISE. TABLE II shows synthesis report of our design with 8 quantization bits :

B. Simulation results

Our TR-UWB system is simulated with Gaussian mono-cycle pulses of width $2ns$.The spacing between two pulses in each frame is $60ns$, and frame period is $180ns$. Bit stream are generated randomly. IEEE channel model CM1 is chosen. Simulink and System Generator are used to simulate and compute bit-error-ratio (BER) versus signal-noise-ratio

TABLE II
Synthesis Report on Spartan 6 device XC6SLX45 package CSG324

Block	$Mainblock$	$Detect\_max$
Number of Slice Registers	52	15
Number of slide LUTs	72	27
Number of fully used LUT-FF pairs	49	14
Number of bonded IOBs	33	33
Max frequency	400 MHz	300 MHz

In Fig. 13, we run C3 with : sample rate is 400 MHz, number of bits quantization is 8. And we run C4 with : sample rate is 100 MHz, number of bits quantization is 8.

From Fig. 13, it can be seen that BER performance of the cases C2, C3, C4 are quite close to the ideal case C1 (assuming perfect synchronization). The gap is only 3 dB. Moreover, Fig. 14 shows that in case C4, when we reduce sampling rate and the number of quantization bits, the BER does not change significantly. Actually, 50 MHz sampling rate and 4 bit quantization are good enough.

VI. Conclusions

In this paper, we have developed and implemented a simple synchronization for TR-UWB receiver. This algorithm is flexible enough so that its parameters can be easily tweaked to support different hardware platforms. It can be shown, by simulation, that we can achieve sufficiently good performance with a low sampling rate and fewer quantization bits. Moreover, by using model-based design, we can readily extend and implement the algorithm to more complicated cases with multiple frames per symbols as specified in IEEE 802.15.4a standard.

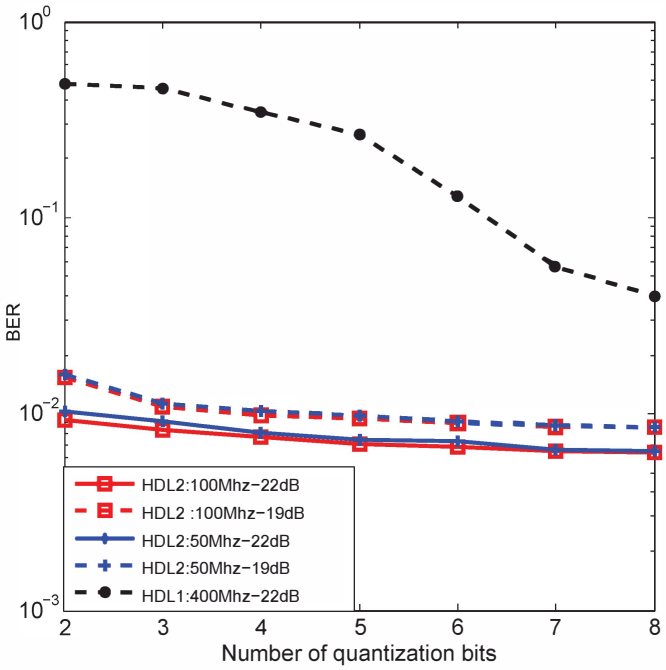

Fig. 14. BER vs. number of quantization bits and sampling rate

REFERENCES

[1] "FCC notice of proposed rule making, revision of part 15 of the commission's rules regarding ultra-wideband transmission systems," Tech. Rep. ET-Docket 98-153, Federal Communications Commission, Washington, D.C., Apr. 2002.

[2] N. van Stralen, A. Dentinger, K. Welles, R. Gauss, R. Hoctor, and H. Tomlinson, "Delay hopped transmitted reference experimental results," in *IEEE Conference on Ultra Wideband Systems and Technologies*, pp. 93–98, 2002.

[3] R. Hoctor and H. Tomlinson, "Delay-hopped transmitted-reference RF communications," in *IEEE Conference on Ultra Wideband Systems and Technologies*, pp. 265–270, 2002.

[4] K. Witrisal, M. Pausini, and A. Trindade, "Multiuser interference and inter-frame interference in UWB transmitted reference systems," in *IEEE Conference on Ultra Wideband Systems and Technologies*, (Kyoto, Japan), May 2004.

[5] G. Leus and A. J. van der Veen, "A weighted autocorrelation receiver for transmitted reference ultra wideband communications," in *IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, (NY), June 2005.

[6] Q. Dang, A. van der Veen, A. Trindade, and G. Leus, "Signal model and receiver algorithms for a transmit-reference ultra-wideband communication system," *IEEE Journal on Selected Areas in Communications*, Apr. 2006.

[7] R. Djapic, G. Leus, A. Trindade, and A. van der Veen, "Blind synchronization in mulituser transmit-reference UWB systems," in *Proc. of the European Signal Processing Conference (EUSIPCO 2005)*, (Antalya (TR)), Sept. 2005.

[8] G. L. Yiyin Wang and A. van der Veen, "Digital receiver design for transmitted reference ultra-wideband systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, 2009.