Robust and Low-Complexity Timing Synchronization Algorithm and its Architecture for ADSRC Applications

Jinsang KIM, Huynh TRONG ANH
Kyung Hee University, Republic of Korea
jskim27@khu.ac.kr

***Abstract*—5.9 GHz advanced dedicated short-range communications (ADSRC) is a short-to-medium range communication standard that supports both public safety and private operations in roadside-to-vehicle and vehicle-to-vehicle communication environments. The core technology of physical layer in ADSRC is orthogonal frequency division multiplexing (OFDM), which is sensitive to timing synchronization error. In this paper, a robust and low-complexity timing synchronization algorithm suitable for ADSRC system and its efficient hardware architecture are proposed. The implementation of the proposed architecture is performed with Xilinx Vertex-II XC2V1000 Field Programmable Gate Array (FPGA). The proposed algorithm is based on cross-correlation technique, which is employed to detect the starting point of short training symbol and the guard interval of the long training symbol. Synchronization error rate (SER) evaluation results and post-layout simulation results show that the proposed algorithm is efficient in high-mobility environments. The post-layout results of implementation demonstrate the robustness and low-complexity of the proposed architecture.**

Index Terms—advanced dedicated short range communication, cross-correlation method, field programmable gate array, orthogonal frequency division multiplexing, timing synchronization

I. INTRODUCTION

The ADSRC standard, which is used for intelligent transportation system (ITS) applications, is a short-to-medium range communication service. ADSRC standard provides very high data rates in circumstances where minimizing latency in the communication link and isolating relatively small communication zones are important [1].

The ADSRC standard is the extension of IEEE 802.11. Therefore, the physical layer (PHY) of ADSRC is based on OFDM technology. Key parameters of ADSRC's physical layer are summarized in Table I.

TABLE I. KEY PARAMETERS OF ADSRC PHY

Signal Bandwidth B (MHz)	10
Data Rate (Mbps)	3, 4.5, 6, 9, 12, 18, 24, 27
Modulation	BPSK, QPSK,16-QAM, 64-QAM
Code Rate	1/2, 2/3, 3/4
OFDM Symbol Duration (μs)	8
Guard Interval (μs)	1.6
Subcarrier Spacing (KHz)	156.25
Number of Pilot Tones	4
Number of Subcarriers	52

ADSRC can operate with data payload capabilities of 3, 4.5, 6, 9, 12, 18, 24, and 27 Mbps. In addition, ADSRC can support a very high data rate, which can be up to 54 Mbps, when 20 MHz bandwidth is used. The system uses 52 subcarriers which are modulated by using binary or quadrature phase shift keying (BPSK/QPSK), 16-quadrature amplitude modulation (16-QAM) or 64-quadrature amplitude modulation (64-QAM). Forward error correction coding is used with coding rate of 1/2, 2/3, or 3/4. An OFDM symbol has the total duration of 8 μs, which includes 1.6-μs guard interval.

OFDM has many advantages due to its capability to combat multi-path fading and high spectral efficiency [2]. OFDM also has been applied to many communication standards such as ADSL, IEEE 802.11a, and IEEE 802.16. However, the severe disadvantage of OFDM technique is that it is very sensitive to synchronization errors [3]. Moreover, ADSRC applications work in high-mobility environment where the speed can reach 120 mph. As a result, Doppler spread caused by high-mobility environment and multipath fading phenomenon cause performance degradation of ADSRC system.

In this paper, a robust and low-complexity timing synchronization scheme for ADSRC system is proposed. In the proposed algorithm, the cross-correlation-based technique is used to detect the starting point of a short training symbol. The guard interval of the first long training symbol is found by applying the same cross-correlation-based method. Therefore, fast timing synchronization can be achieved. Also, efficient and low-complexity hardware architecture of the proposed algorithm is developed. Then, the design is implemented in Xilinx Vertex II XC2V1000 FPGA. SER is utilized to evaluate the performance of the proposed algorithm. SER simulation in both Matlab and post-layout simulation using FPGA shows that the proposed design exhibits good performance in high-mobility environments.

The rest of this paper is organized as follows. In section II, the overview ADSRC system model is presented. The proposed ADSRC timing synchronization algorithm is given in section III. Section IV presents the proposed architecture of ADSRC timing synchronization algorithm. FPGA implementation of the proposed architecture is followed in section V. Simulation results and analyses of the design are given in Section VI. Section VII ends up with the conclusions.

This work was supported by Korea Research Foundation under Grant No. KRF-2006-D00337 and IDEC (CAD tool).

Digital Object Identifier 10.4316/AECE.2009.03008

II. Overview of ADSRC System Model

According to ADSRC standard, each ADRSC frame consists of three fields: preamble, signal and data field as shown in Figure 1. Preamble field is mainly used for timing synchronization, carrier frequency offset estimation and channel estimation. It comprises 10 identical short training symbols and 2 identical long training symbols. Each short training symbol has 16 samples. The number of samples in one long training symbol is 64. The interval between successive samples is 0.1μs. Signal field contains RATE and LENGTH information of ADSRC frame. Note that the signal field's bits are BPSK-modulated. Data field conveys the information data.

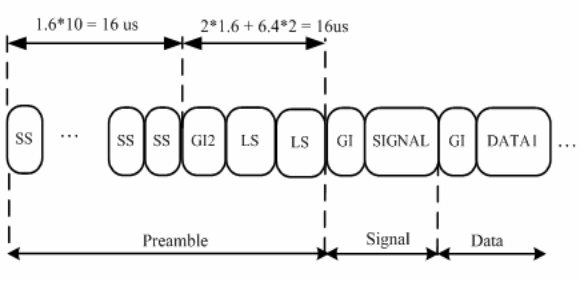

Figure 1. Frame structure of ADSRC system.

As shown in Figure 2, the generation of signal, data and preamble fields is performed at the transmitter. Then, the boundaries among these generated fields are shaped by a time-windowing function in order to smooth transitions. After that, the ADRC transmitted signal travels through the high-mobility channel. The effect of carrier frequency offset (CFO) is also modeled in this system. In addition, the channel parameters of outdoor urban/suburban environment provided by Joint Technical Committee (JTC) are employed for the ADSRC system [4]. Table II shows these parameters.

TABLE II. CHANNEL PARAMETERS

Tap	Channel A (rms Delay Spread = 0.4μs)		Channel B (rms Delay Spread = 12 μs)	
	Relative Delay (ns)	Average Power (db)	Relative Delay (ns)	Average Power (db)
1	0	-1.6	0	-2.5
2	100	-5.1	300	0
3	200	0	8900	-12.8
4	500	-7.6	12900	-10
5	1200	-6.9	17100	-25.2
6	1600	-27.6	20000	-16

At the receiver, the timing synchronization block finds the exact timing point of the received data. Then, the carrier frequency mismatch is corrected in frequency synchronization block. Next, guard removal (GR) block removes guard interval data and the OFDM symbols are converted to parallel data by FFT block. Based on training symbols and pilot tones, the channel estimation block estimates the channel coefficients. Then, equalizer block equalizes the received data with these estimated channel coefficients to improve ADSRC's system performance. The equalized data continued to be demodulated, depunctured, deinterleavered and decoded. Finally, output data are obtained after FEC decoding and descrambling.

III. Proposed ADSRC Timing Synchronization Algorithm

Various timing synchronization algorithms for OFDM-based systems have been published [5]-[13]. In [5], a timing metric, which is the ratio between the correlation function to the received signal energy, is proposed to find the symbol

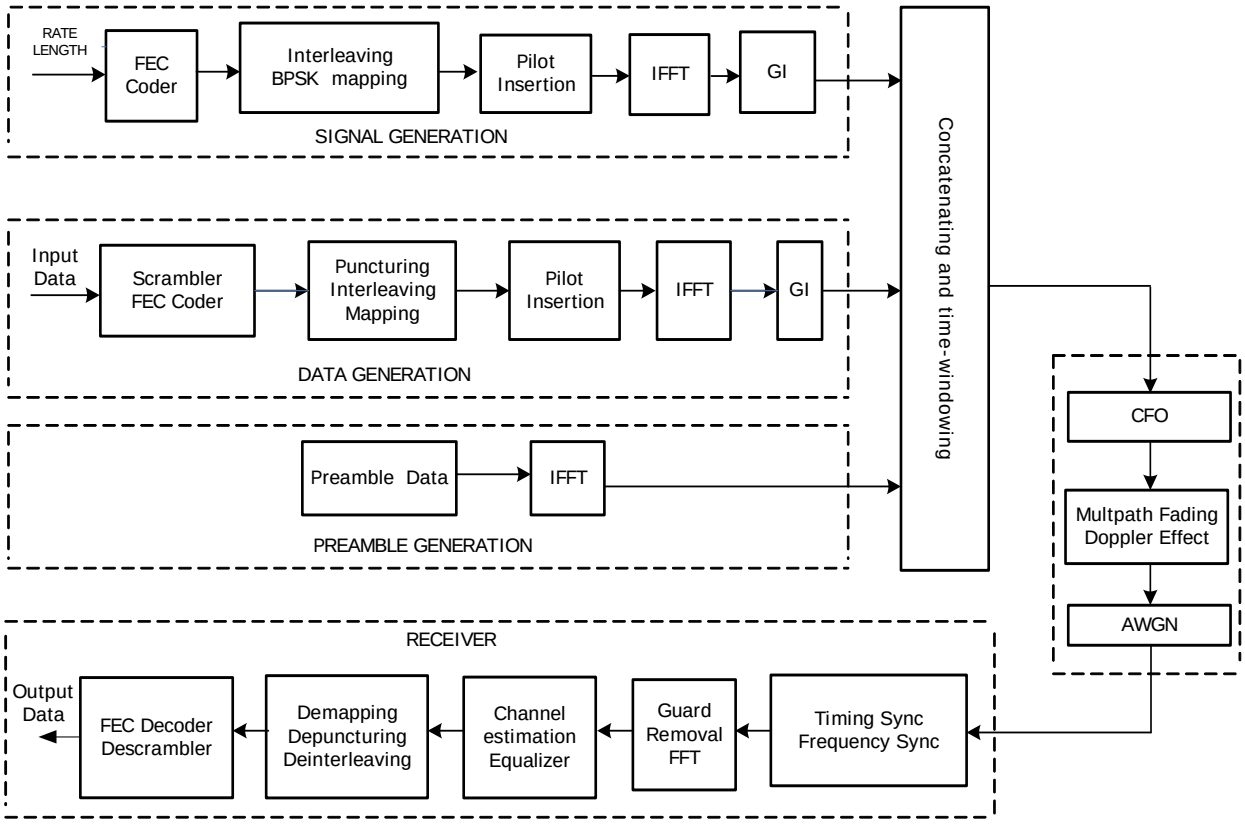

Figure 2. ADSRC baseband simulation model.

timing. A maximum likelihood (ML) timing synchronization method is proposed in [6]. The disadvantage of these algorithms is that they require high-complexity architecture, large hardware utilization and high power consumption. Moreover, the accuracy of these algorithms relies on the SNR estimation, which is often not accurate in high-mobility and multipath fading environments. In [7], a new timing synchronization which includes fine and coarse estimation steps is proposed. However, it requires high-complexity architecture, particularly in coarse estimation block. Moreover, the robustness of the algorithm in [7] also depends on the SNR estimation.

In this paper, a robust and low-complexity scheme for timing synchronization suitable for ADSRC systems is proposed. The proposed timing synchronization algorithm begins at an unknown position of the preamble. For example, the scheme is initialized at the 7th sample of the 7th symbol as shown in Figure 3.

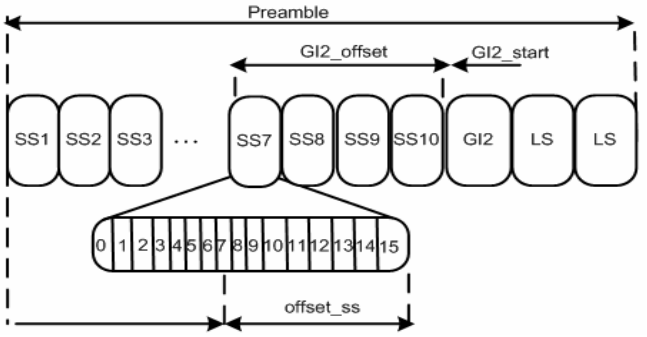

Figure 3. Proposed algorithm begins at the arbitrary point in preamble field of an ADSRC frame.

The proposed algorithm consists of two steps as follows.

Step 1: Detect the starting point of the short training symbol

Cross correlation function between r(n) and rs(k) can be written as follow:

$$C(n)=\left|\sum_{k=0}^{15} r(n+k)\,r_s^{*}(k)\right|,\quad 0\le n\le 15, \tag{1}$$

where r(n) is the received signal and $r_s(k)$ is the short training symbol. Note that $r_s(k)$ is known at both receiver and transmitter. The timing metric function $R_1(n)$, which is the square of cross-correlation function C(n), is used to find the starting point of a short training symbol:

$$\begin{aligned} R_1(n) &= \{C(n)\}^2 \\ &= \left|\sum_{k=0}^{15} r(n+k)\,r_s^{*}(k)\right|^2,\quad 0\le n\le 15 \end{aligned} \tag{2}$$

$$\text{offset-ss} = \arg\max_{0\le n\le 15}\{R_1(n)\} \tag{3}$$

where offset-ss is the timing offset of a short training symbol. Because one short training symbol has the period of 16 samples, the offset of a short training symbol is detected when $R_1(n)$ is evaluated in 16 consecutive samples. C(n) is utilized to find the starting position of a short training symbol to avoid a root square operation which consumes a large hardware utilization of ADRC timing synchronization design.

Step 2: Detect the starting point of GI2

$$R_2(m)=\left|\sum_{k=0}^{15} r(k+m)\,r_s^{*}(k)\right|^2 \tag{4}$$

$$m = 0,16,32,48,\ldots$$

$$GI2-\text{offset}=\arg\max_{m}\{R_2(m)<\theta\}. \tag{5}$$

After completing step 1, the exact timing position of a short training symbol is found. However, how many short training symbols left in the given ADSRC frame are still unknown. Therefore, the purpose of step 2 is to find the starting point of the guard interval (GI2) of the first long training symbol. The $R_2(m)$ calculates the square of cross-correlation value between the r(n) and $r_s(n)$. However, the argument m of $R_2(m)$ does not increase consecutively.

Instead, m increases with step size of 16. Therefore, $R_2(m)$ has the plateau form in the short symbol region. When received signal is within the guard interval region of long symbol, $R_2(m)$ decreases sharply. A threshold θ is used to detect the starting point of guard interval in the received signal.

IV. Efficient and Low-Complexity Hardware Architecture

The hardware architecture of the proposed algorithm is shown in Figure 4. The real and imaginary parts of the received signal (data-re [15:0] and data-im [15:0]) have the bit width of 16, which contains 4 integer bits and 12 fractional bits. According to the ADSRC standard, the frequency of system clock signal clk is 10 MHz. The detect-signal is used to declare that a new ADSRC frame has been detected. The timing metric block calculates the timing metric values $R_1(n)$ and $R_2(m)$ in both step 1 and step 2. For efficient resource utilization, data-re[15:0] and data-im[15:0] are truncated before they are used as inputs of the timing metric block. Therefore, the input data (cross-re and cross-im) of timing metric block has the bit width of 9, in which there are 4 integer bits and 5 fractional bits.

In step 1, the fine comparator block accepts the timing metric values from the cross-corr block. It finds the maximum peak among 16 consecutive timing metric values. When the maximum timing metric value is detected, the fine comparator block asserts the ss-detect and pointer-ss [4:0] signal to the control block. The pointer-ss [4:0] signal indicates the exact position of short symbol. Based on pointer-ss [4:0] information, the control block generates the addr-buff [4:0], which is used in the ADSRC buffer to supply the exact data for the timing metric block in step 2.

After step 1 is done completely, the control block sends the coarse-start signal to initialize the operation of the coarse comparator. The purpose of coarse comparator block is to find the offset of guard interval 2 in step 2. Note that the fine comparator block is idle in step 2 and the coarse comparator block is idle in step 1 by using clock gating technique. Therefore, power consumption of the design can be reduced significantly.

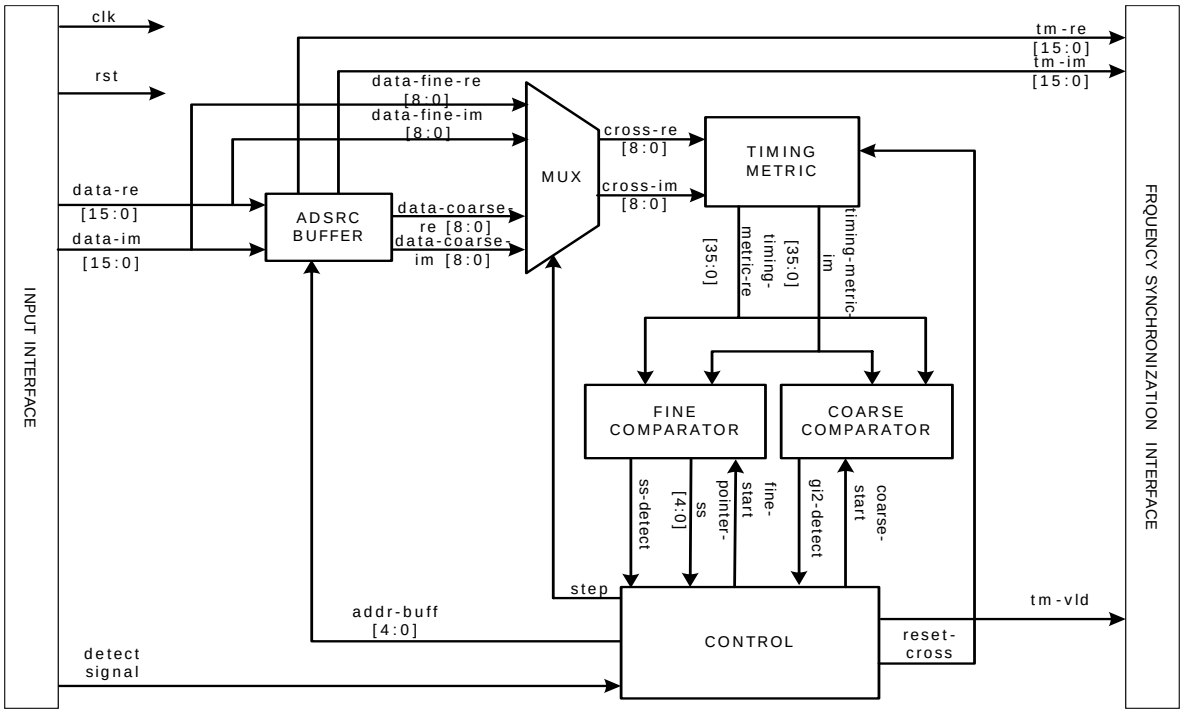

Figure 4. The Proposed Hardware Architecture of ADSRC Timming Syncroniyation.

V. FPGA Implementation of the Proposed Architecture

The ADRC timing synchronization design was coded in VHDL, synthesized using SynplifyPro 8.0, placed and routed in Xilinx ISE 6.0 and implemented on a Virtex-II XC2V1000 FPGA. Active-HDL 6.2 tool is used for HDL simulation. Post-Layout simulation is performed in ModelSim SE 6.0a. Table III shows the hardware utilization result by SymplifyPro 8.0. Majority of hardware resource is occupied by the timing metric and the ADSRC buffer blocks.

TABLE III. SYNTHESIS RESULT

Block	Register	LUTs	MULCY	XORCY	Multiplier (MULT 18x18)
Fine comparator	58	90	30	5	0
Coarse comparator	1	10	8	0	0
MUX	18	18	0	0	0
Control	46	26	8	10	0
Timing metric	1964	1675	1385	1421	8
ADSRC buffer	1164	933	0	0	0

TABLE IV. POST LAYOUT RESULT

Item	Amount	Percentage
Number of SLICEs	2463	48%
Number of Extenal IOBs	78	23%
Number of Multiplier (MULT18X18s)	8	25%
Number of BUFGMUXs	1	6%
Average Power	426 (mW)	
Maximum Frequency	84 (MHz)	

Using pinout and area constraint editor (PACE) tool of Xilinx, the floor plan of the design is created. The floor planed area of the timing metric and the ADSRC buffer is much larger than the remaining blocks of the design. In addition, pin assignment is also performed using PACE tool. After that, placed and routed process is performed.

Table IV shows final post-layout results of the design. The ADSRC timing synchronization design occupies 2463 slices of the target device, which is equivalent to 14 % of hardware utilization. The number of 18x18 multipliers is 8, which corresponds to 25% multiplier resource of FPGA chip. Maximum clock frequency of the design is 84 MHz. Power consumption of the design is evaluated by using Xilinx XPower Tool. XPower tool accepts the value change dump (VCD) file which is generated from ModelSim. The total estimated power consumption of the design is 426 mW.

VI. Simulation Result and Analyses

The simulation methodology of the proposed architecture is given in Figure 5. The transmitter, CFO and channel environment, which includes AWGN, multipath fading, and Doppler effects, are modeled in Matlab. The ADRC timing synchronization is coded by both Matlab and VHDL.

Figures 6 and 7 show the curves $R_1(n)$ and $R_2(m)$ in the channel environment which has EbNo of 10 dB, 120 mph velocity, and normalized CFO of 0.2. $R_1(n)$ curve reaches its maximum value when n=8, which indicates that the exact timing point of the short symbol is located at the 8th sample of the received signal. $R_2(m)$ has a quite plateau shape when m is within from 1 to 8. However, $R_2(m)$ begins decreasing sharply when m = 9, which indicates the starting point of the GI2. Based on these results, it can be concluded that the post layout simulation results of $R_2(m)$ and $R_1(n)$ are similar to those of Matlab simulation.

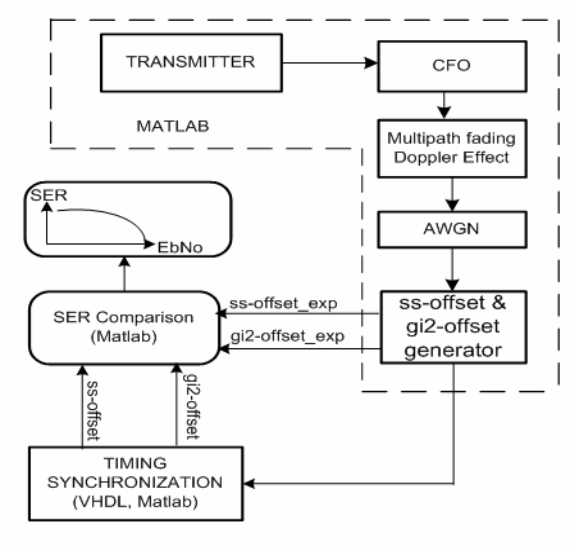

Figure 5. Simulation Environment.

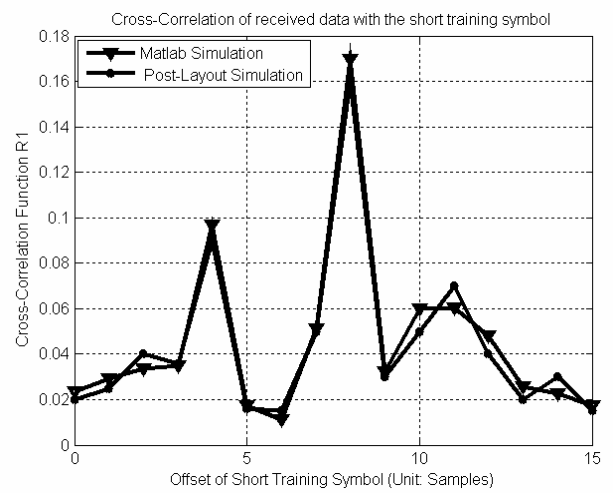

Figure 6. $R_1(n)$ reaches the maximum peak when n = 8.

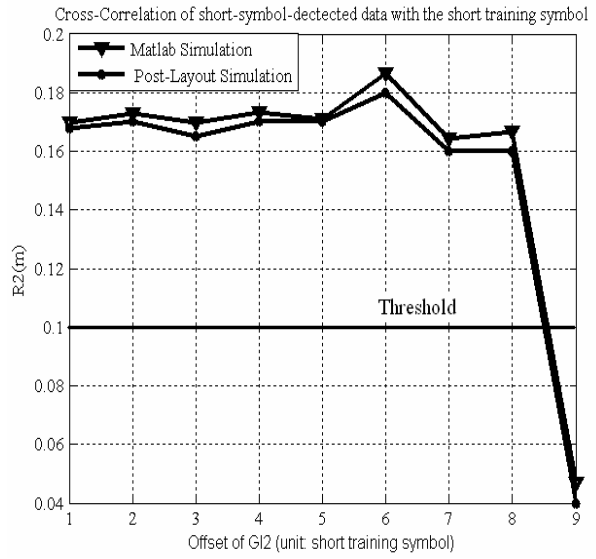

Figure 7. $R_2(m)$ is lower than the threshold value when m = 9.

SER parameter is used to evaluate the performance of the proposed architecture. Figure 8 shows SER curves of the proposed scheme when the vehicle moves at the velocity of 100 mph in channel A. Normalized CFO value is 0.25. From Figure 8, it can be concluded that SER in Matlab simulation is close to SER from post layout simulation. In channel B which has the delay spread (4μs) larger than the guard interval (1.6μs) of ADSRC system, the proposed architecture also demonstrates its robustness as shown in Figure 9. Figure 10 shows the comparison of SER post layout simulation between channel A and channel B at the speed of 120 mph.

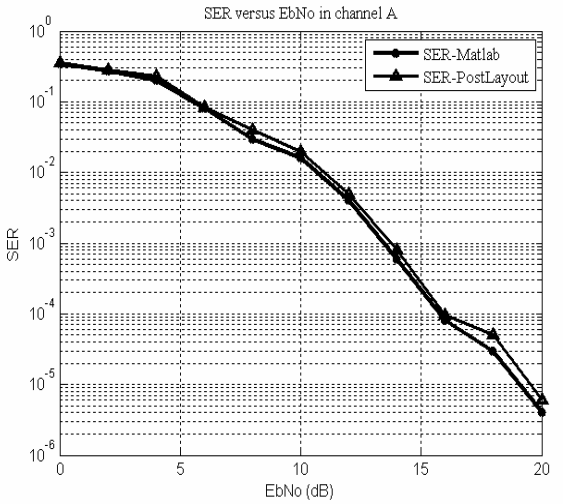

Figure 8. SER comparison between Matlab and Post-layout Simulation in Channel A.

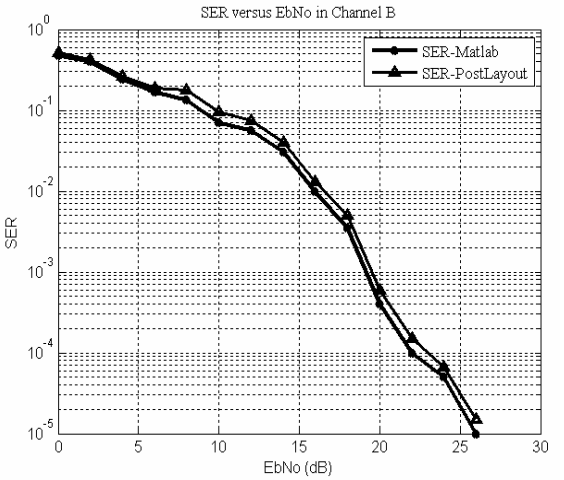

Figure 9. SER comparison between Matlab and Post-layout Simulation in Channel B.

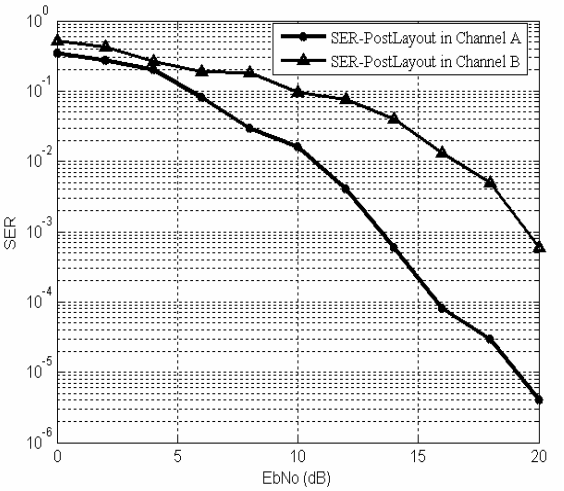

Figure 10. SER comparison between Channel A and Channel B environments.

VII. CONCLUSION

In this paper, a robust and low-complexity ADSRC timing synchronization algorithm and its efficient hardware architecture are proposed. Cross-correlation technique is used to detect the starting point of both a short training symbol and guard interval of long training symbol. The design was implemented with target device of Xilinx Vertex-II XC2V1000 FPGA. Post-layout results show that the proposed architecture is efficient in terms of speed, area, and hardware utilization. The performance of the proposed algorithm is evaluated using SER parameter. SER simulation results from both Matlab and post layout demonstrate that the proposed algorithm exhibits robust performance in high-mobility environment.

REFERENCES

[1] http://www.leearmstrong.com/DSRC%20Home/Standards%20Programs/North%20American/DSRC_Tutorial_06-10-021/sld091.htm

[2] L. Hanzo, M. Munster, B.J Choi and T. Keller, "OFDM and MC-CDMA for Broadband Multi-User Communications", WLANs and Broadcasting, Wiley, 2003.

[3] Shinsuke Hara, Ramjee Prasad, "Muticarrier Technologies for 4G Mobile Communications", Artech, 2003.

[4] Joint Technical Committee on Wireless Access, "Technical Report on RF channel characterization and System Deployment Modeling", Paper No. JTC(AIR) 1994.09.23-065R6, Sep. 23,1994.

[5] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM", IEEE Transaction on Communications, vol. 45, pp.1613-1621, Dec. 1997.

[6] J.J. van de Beek, M. Sandell, "ML Estimation of Time and Frequency Offset in OFDM Systems", IEEE transactions on signal processing, vol.45, pp.1800-1805, July 1997.

[7] S. Chang and B. Kelley, "Time synchronization for OFDM-based WLAN systems", IEEE Electronics Letters, vol. 39, pp. 1024-1026, June 2003.

[8] N. Chan, M Tanaka and R.Heaton, "OFDM Timing Synchronization under Multi-path Channels", VTC-2003 Spring, vol. 1, pp. 378-382, May 2003.

[9] Feng Lu, Takm Ohseki, Hiroyasu Ishikawa and Hideyuki Shinonaga, "On Symbol Timing for OFDM based Mobile Communications Systems", Global Telecommunications Conference, vol. 1, pp. 273-277, Nov. 2002.

[10] Kun-Wah Yip, Yik-Chung Wu and Tung-Sang Ng, "Timing-synchronization analysis for IEEE 802.11a wireless LANs in frequency-nonselective Rician fading environments", IEEE transactions on Wirelss Communications, vol. 3, pp. 387-394, March 2004.

[11] C. Williams, M.A. Beach and S. McLaughlin, Robust "OFDM timing synchronization", IEEE Electronics Letters, vol. 41, pp. 751 -752, June 2005.

[12] K.Wang, J. Singh and M. Faulkner, "FPGA implementation of OFDM-WLAN synchronizer", Electronics Design, Test and Applications, 2004 EEE Electronics Letters, pp. 89 -94, Jan 2004.

[13] Yin-Tsung Hwang, Kuo-Wei Liao and Chien-Hsin Wu, "FPGA realization of an OFDM frame synchronization design for dispersive channels", ISCAS'03, vol. 2, pp. 256-259, May 2003.