A Resource- and Power-Efficient Implementation of PSS Synchronization on FPGA in Vehicular Networks

Chengxiang Ge[1], Fei Peng[1], Shan Cao[1], Zhiyuan Jiang[1(✉)], and Terng-Yin Hsu[2]

[1] School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China
jiangzhiyuan@shu.edu.cn

[2] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Abstract. This paper proposes a resource- and power-efficient implementation of primary synchronization signal (PSS) synchronization on FPGA in vehicular networks. Firstly, the received signal is preprocessed. Secondly, the synchronization signal is detected and its timing position is calculated. Thirdly, a sequence is extracted for frequency offset estimation. Finally, the received original signal is marked with the frame header and compensated with proper frequency offset. The hardware implementation scheme is based on FPGA and uses the advantages of FPGA high-speed pipeline computing to realize real-time sampling and synchronization of transmitter and receiver, the consumption of Slice Look-Up-Table (LUT) is 9545 and the total on-chip power is 1.213 w.

Keywords: Synchronization · FPGA · Primary Synchronization Signal(PSS)

1 Introduction

Long term evolution vehicle (LTE-V) is an emerging vehicular communication technology that is widely-adopted by many countries. LTE-V (Release 14) [1] which often referred to Cellular V2X (C-V2X) technology is defined in the 3rd Generation Partnership Project (3GPP) community. V2X means that it can realize vehicle to everything communication including vehicle to vehicle, vehicle to pedestrian, vehicle to network, vehicle to cloud, and vehicle to infrastructure. LTE based C-V2X allows vehicles to communicate with each other directly without the need for base station infrastructures and is expected to be a critical enabler for connected and autonomous vehicles [5]. Through the wireless network, vehicle shares basic information such as position and vehicle speed with the surrounding vehicles, so that driver can know the vehicle's position and status around [3]. LTE-V technology is based on OFDM technology, which requires the use of synchronization technology in the design of the receiver to ensure the strict orthogonality of the transmitted signal and the received signal.

K. Rabie et al. (Eds.): IoTaaS 2022, LNICST 506, pp. 135–147, 2023.
https://doi.org/10.1007/978-3-031-37139-4_13

This paper proposes a resource- and power-efficient implementation of Primary Synchronization Signal (PSS) synchronization on FPGA in vehicular networks. Firstly, the received signal is preprocessed. Second, the synchronization signal is detected and its position in the time domain is calculated. Thirdly, a sequence is extracted for frequency offset estimation. Finally, the received original signal is marked with frame header and proper frequency offset compensation is carried out to prepare for subsequent decoding operations. The hardware implementation scheme is based on FPGA, which uses the advantages of FPGA high-speed pipeline computing to achieve real-time sampling and synchronization of transmitter and receiver. The consumption of Slice Look-Up-Table (LUT) is 9545 and the total on-chip power is 1.213 w.

2 Physical Layer of LTE-V

In this section, a brief introduction of LTE-V frame structure and Synchronous signal structure are provided.

2.1 LTE-V Frame Structure

The frame structure of LTE-V is shown in Fig. 1. T_f is the period of a wireless frame, T_s is the period of each signal sampling point. A wireless frame consists of 20 slots with a total length of 10 ms. Every two slots form a subframe with a length of $30720T_s$, which is 1 ms. Each subframe consists of 14 Single-carrier Frequency-Division Multiple Access (SC-FDMA) symbols where 9 symbols are available for data transmission [2]. The synchronization technique is used to mark the frame header of the frame to ensure that the receiver can determine the starting point of each received signal subframe.

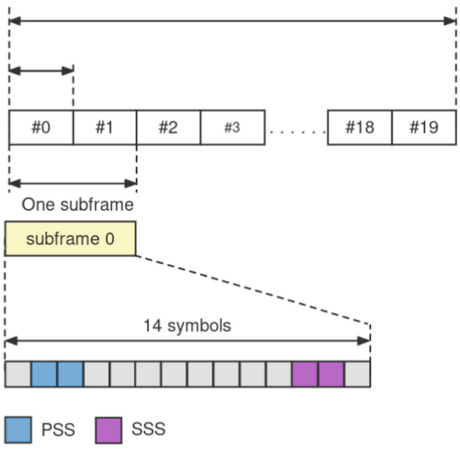

Fig. 1. LTE-V frame structure and position of synchronization signal.

2.2 Synchronization Sequence

This paper studies the detection of PSS. As shown in Fig. 1, the signal is transmitted on the second and third SC-FDMA symbols of subframe 0, which are adjacent to each other. PSS sequence is generated in frequency domain by using Zadoff Chu sequence [6], which has good mutual property and can resist certain noise and frequency offset. The expression generated by PSS sequence is defined as

$$d_u(n) = \begin{cases} e^{-j\frac{\pi un(n+1)}{63}} & n = 0, 1, 2, \ldots, 30 \\ e^{-j\frac{\pi un(n+1)(n+1)}{63}} & n = 31, 32, 33, \ldots, 61 \end{cases}, \tag{1}$$

where u is the root index of different Zadoff Chu sequence, and n is the index of each point of the sequence, this paper choose two sequences, which respectively corresponding to the sequence generated by $u = 26$ and $u = 37$.

2.3 Frequency Offset

In the process of signal transmission, due to Doppler effect and clock source error between transmitter and receiver, the received signal will have frequency offset relative to the signal sent by transmitter. It can be judged intuitively by analyzing the spectrum diagram. Frequency offset will cause Cyclic Redundancy Check (CRC) failure at the receiver, and the packet cannot be solved correctly. Therefore, in synchronization technology, frequency offset estimation is necessary for the received signal at the receiver, and original signal need to be compensated with proper frequency offset.

3 Proposed Synchronization Method

3.1 Traditional Synchronization Methods

The traditional synchronization method mainly includes correlation calculation and traversing the calculation results to find the maximum value, in which correlation calculation includes cross-correlation, auto-correlation and partial cross-correlation [4], such as (2), (3), (4).

$$R = \max_n \left[\left| \sum_{i=0}^{N_{FFT}-1} Y(i) X^*(i+n) \right|^2 \right], \tag{2}$$

$$R = \max_n \left[\left| \sum_{i=0}^{N_{FFT}/2-1} Y(i+n) Y^* \left[(N_{FFT} - i) + n \right] \right|^2 \right], \tag{3}$$

$$R = \max_{n} \left[\sum_{m=0}^{N_{PB}-1} \left| \sum_{i=m(N_{FFT}/N_{PB})}^{\frac{(m+1)N_{FFT}}{N_{PB}}-1} Y(i)X^*(i+n) \right|^2 \right], \tag{4}$$

where R is the calculated correlation peak value result, $X(i)$ is the local reference PSS signal, and $Y(i)$ is the received signal containing PSS information. The first cross-correlation algorithm cross correlates the received signal containing PSS information with the local PSS sequence, and then traverses all the results to get the maximum value. The second auto-correlation method is to auto correlate the received signal containing PSS information, and also to traverse all results to find the maximum value. The third kind of partial correlation splits the first kind of cross-correlation algorithm into multiple segments for cross correlation, then sums each result up, and finally traverses to get the maximum value.
The above three methods can calculate the location of the PSS, however, if use FPGA to achieve these methods directly, the resource consumption is large. Therefore, a new synchronization method is proposed in this paper, which uses cross-correlation to resist the interference of frequency offset and noise, and uses the characteristics of PSS cross-correlation peak to determine the peak only by local traversal, and it is convenient for hardware implementation.

3.2 Proposed Correlation Method

The cross correlation algorithm selects two segments of signals in time domain for cross correlation and then sum the results up, which can resist the influence of frequency offset. The receiver performs real-time cross-correlation between the received signal and the locally stored PSS sequence, and calculates the energy of the signal. By comparing the peak value of the cross-correlation result and the result of energy calculation, the location of the PSS is realized. The calculation formula is as follows

$$R_{pss128}(n) = \left| \sum_{i=0}^{127} d_{pre}(n+i) \cdot S_{pss128^*}(i) \right|^2 + \left| \sum_{i=0}^{127} d_{pre}\left(n + \frac{2048+144}{16} + i\right) \cdot S_{pss128}{}^*(i) \right|^2, \tag{5}$$

where n is the index of each cross correlation calculation result, i is the serial number of 128 sampling points selected for calculation, $d_{pre}(i)$ is the signal generated by low-pass filtering and down sampling of the original signal received by the receiver, $S_{pss128}(i)^*$ is the reference primary synchronous sequence generated according to the protocol and stored locally.

The pre-processed received signal is cross-correlated with the local reference PSS sequence to determine whether there is PSS in the corresponding original signal. Since two PSS reference sequences with different root indexes are selected when designing the physical layer protocol, two synchronization sequences are

used for parallel calculation. The expression of the energy calculation of the received signal is as follows as

$$E(n) = \sum_{i=0}^{127} |d_{pre}(n+i)|^2 + \sum_{i=0}^{127} |d_{pre}(n+(2048+144)/16+i)|^2 . \quad (6)$$

where n is the index of the sample points. In order to compare with the cross-correlation results, 128 sampling points are selected for calculation. The energy result is obtained after the auto-correlation calculation of the received signal d_{pre}. Two sequences in the time domain is taken here. If the real-time cross-correlation result and the energy calculation result meet (7) and (8), it is considered that the peak value of the synchronization signal has been detected.

$$R_{pss128}(r) > \varepsilon \cdot E(r). \quad (7)$$

$$R_{result}(r) = \max_{n} \left[(R_{pss128}(r) \right]. \quad (8)$$

Where r is the index of corresponding position, ε is the energy threshold coefficient, which will be fine-tuned according to the specific hardware implementation results, n is the number of sampling points. The starting position of the synchronization signal can be obtained by selecting the maximum peak value within the length of half subframe $15360T_s$ in the time domain, as shown in (8).

3.3 Frequency Offset Estimation and Compensation Algorithm

The frequency offset compensation of the signal needs to be implemented to avoid the subsequent decoding affected by the frequency offset of the received signal in the actual transmission process. Therefore, after the synchronization correlation is completed, it is necessary to estimate the frequency offset of a sequence corresponding to the peak's position in the original input signal, and the calculation formula is shown as

$$C_{pss} = \sum_{i=0}^{63} \left(d_{new}(i) \cdot {S_{pss128}}^*(i)\right)^* \cdot \left(d_{new}(i+64) \cdot {S_{pss128}}^*(i+64)\right). \quad (9)$$

$$FFO_1 = \frac{1}{\pi} \operatorname{angle}\left(C_{pss}\right). \quad (10)$$

where $d_{new}(i)$ is the signal after the input signal is filtered and 16 times down sampled. The starting position is the position of the PSS. ${S_{pss128}}^*(i)$ is the conjugate of the primary synchronization sequence stored locally in Read-Only

Memory (ROM). C_{pss} is the cumulative value of frequency offset calculation. The frequency offset value FFO_1 of decimal times can be calculated by calculating the arc-tangent of C_{pss}. After the frequency offset estimation is completed, the original signal is compensated for the frequency offset according to the value of the frequency offset estimation. The compensation formula is as follows

$$d_{out}(n) = d_{in}(n)e^{-j2\pi\varepsilon\frac{n}{N}}, \tag{11}$$

where $d_{in}(n)$ is the input of frequency offset compensation and $d_{out}(n)$ is the output, ε is the normalized frequency offset. N is the number of Fast Fourier Transform (FFT) points, and n is the index.

3.4 Simulation Result

The implementation of the proposed algorithm is on MATLAB R2019a. In this paper, two PSS sequences generated by the different root index are adopted, and one of them is detected in Fig. 2. It can be seen that a main peak value and two secondary peaks are detected. If no synchronization signal is detected, the cross-correlation result is shown in Fig. 3.

Compared with the three methods mentioned above, the algorithm used in this paper only uses 128 points, taking into account the frequency offset resistance and the complexity of hardware implementation. Figure 4 shows the cross-correlation results of 128 points without segmentation (M = 1), with two segments (M = 2) and with four segments (M = 4), respectively. Where M is the number of segments. When M = 2, the signal with length of 128 points is divided into two data with length of 64 points for cross-correlation calculation. Theoretically, the more the number of segments, the stronger the anti-noise ability of cross-correlation. However, given the complexity of the hardware implementation, the non-segmented (M = 1) approach is chosen here.

4 Hardware Implementation Structure

4.1 Proposed Hardware Implementation Method

Hardware implementation is based on the proposed synchronization algorithm, and the implementation block diagram is shown in Fig. 5. Divided by function, there are three main modules: synchronization module, frequency offset estimation and compensation module, and timing mark module. In the synchronization module, signal preprocessing, cross-correlation calculation, energy calculation and peak value comparison functions are completed. In the frequency offset estimation and compensation module, a segment of signal is extracted according to the position of the PSS to estimate the frequency offset and compensate the frequency offset of the original signal. Finally, the frame header is marked in the timing marking module to facilitate subsequent processing.

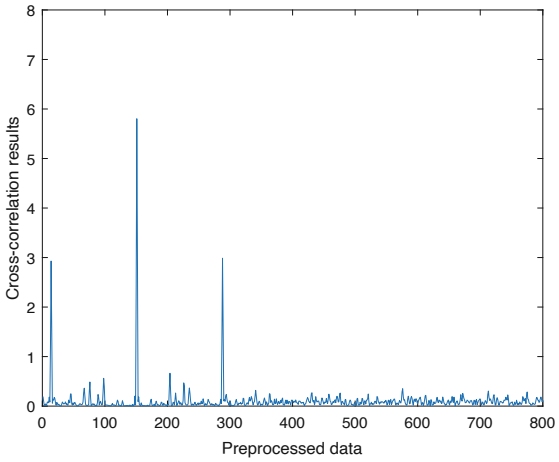

Fig. 2. Cross correlation results when PSS is detected.

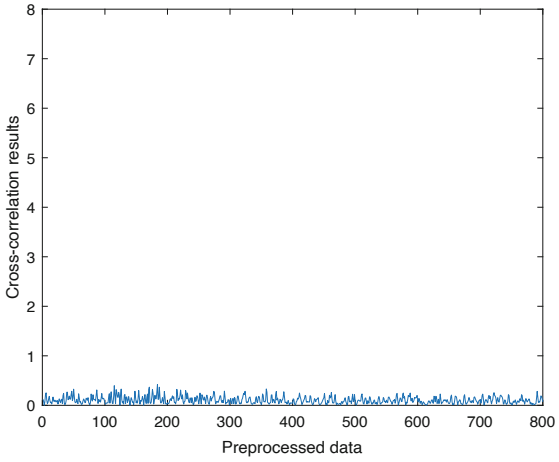

Fig. 3. Cross correlation results when no PSS is detected.

4.2 Implementation of Cross-Correlation

The calculation of Finite Impulse Response (FIR) is shown in (12) as

$$y(k) = \sum_{n=0}^{N-1} a(n)x(k-n) \quad k = 0, 1, \ldots. \tag{12}$$

where $x(k-n)$ is the input data, $y(k)$ is the output of FIR IP, $a(n)$ is the tap coefficients, and N is the filter order.

Since the calculation method of cross correlation is very similar to that of FIR, this paper uses FIR IP core to complete the calculation of cross correlation.

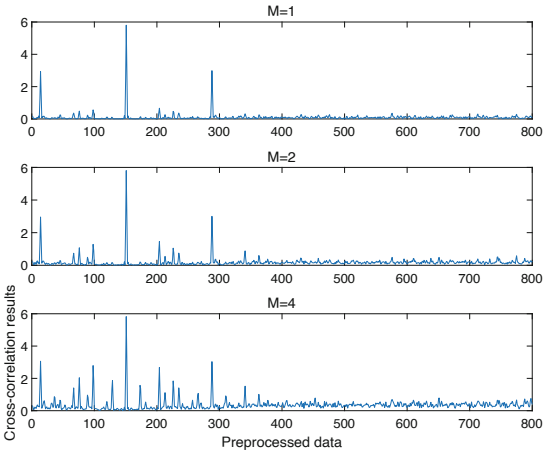

Fig. 4. Cross correlation results with different number of segments.

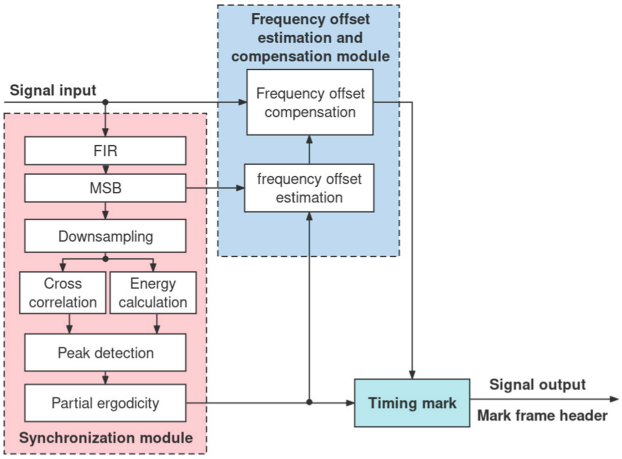

Fig. 5. Hardware implementation structure.

Figure 6 is the implementation structure of FIR. In the actual FPGA design, the FIR IP core will adopt a more optimized structure to realize the automatic selection of the smallest time-sharing multiplication and accumulation unit according to the user-specified throughput, and then optimize the use of the final hardware resources. Since the form of the input data is a complex number, the original algorithm needs to be processed accordingly to facilitate the implementation of FPGA. Assume that $X(k)$ and $Y(k)$ are two complex numbers, according to the multiplication association law, the complex multiplication of each point can be split to calculate the real part and imaginary part of the complex multiplication result, which can be expressed as

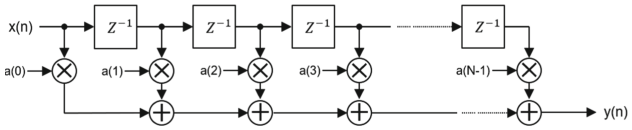

Fig. 6. Implementation of FIR.

$$\begin{aligned} Y(k) \cdot X^*(k) &= [Y_r(k) + Y_i(k) \cdot j] \cdot [X_r(k) - X_i(k) \cdot j] \\ &= [Y_r(k) \cdot X_i(k) + Y_i(k) \cdot X_i(k)] \\ &+ [Y_i(k) \cdot X_r(k) - Y_r(k) \cdot X_i(k)] \cdot j. \end{aligned} \tag{13}$$

The real and imaginary parts of the local reference PSS sequence are respectively imported into two FIR IP cores as the tap coefficients of FIR. Each FIR IP core is configured as two channels with the same tap coefficients, so that the real and imaginary parts of the final cross-correlation results can be calculated, as shown in Fig. 7.

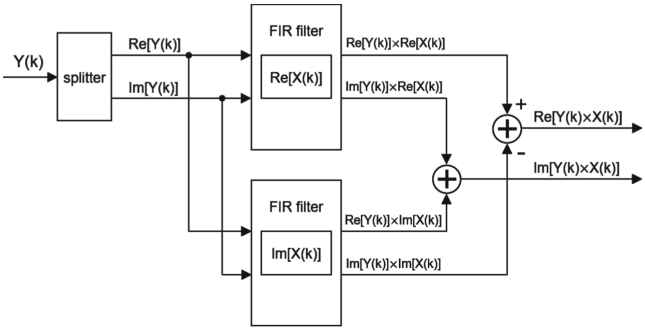

Fig. 7. Cross correlation implementation architecture using FIR IP core.

4.3 Energy Calculation

According to (6), the implementation structure of energy calculation is shown in Fig. 8. Two segments of signals are selected for calculation in the time domain. The final energy calculation result is obtained by adding the calculation results of the first segment of signals and the second segment of signals in the Random Access Memory (RAM) buffer.

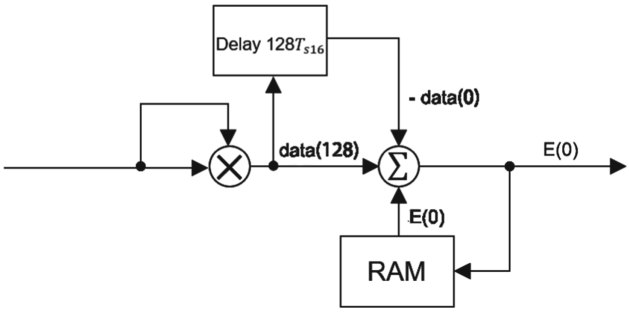

Fig. 8. Energy calculation architecture.

4.4 Frequency Offset Estimation and Compensation

Refer to (9) (10), the architecture of frequency offset estimation is shown in Fig. 9. Use RAM to buffer the preprocessed signal. When the location of the synchronization signal is detected, the 128 point data at the corresponding location are extracted from RAM and dot product with the local reference PSS in ROM. The result of frequency offset estimation can be calculated by ARCTAN module of CORDIC IP core. Figure 10 shows the implementation structure of frequency offset compensation. The calculation result of frequency offset estimation is used to compensate the original input signal. Here, the rotation mode of CORDIC IP core is used. The input phase needs to be preprocessed to $(-\pi, \pi)$ to meet the configured CORDIC input range.

4.5 Timing Mark

After completing the detection of the PSS, the frame header position of the next subframe can be deduced according to the LTE-V frame structure and the position of the synchronization signal. As shown in Fig. 11, point A is the position of the PSS, and point B is the moment when the synchronization signal is detected. The frame header of the next subframe is after the delay t, which is the position of point C. The relative positions of A, B and C are fixed, and the delay parameter t is a fixed number that can be calculated. As shown in Fig. 3 above, cross correlation may calculate multiple peaks, including the main peak and the secondary peak, and the correct location is the location of the main peak. Traditional methods determine the maximum peak value by traversing all the results, but consume too much hardware resources. The method proposed in this paper makes use of the characteristics of synchronous signals. The distance between the first sub peak and the last peak is within the length of half sub frame, therefore, only half the length of the sub frame after the first sub peak need to be selected for traversal.

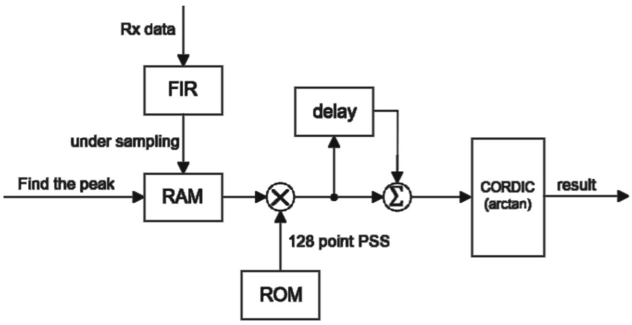

Fig. 9. Architecture of frequency offset estimation.

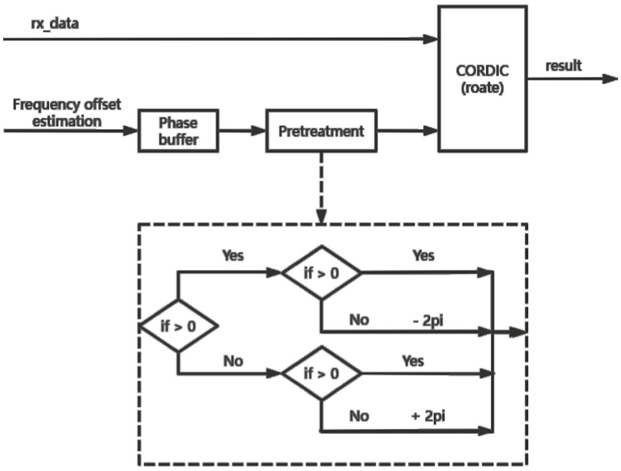

Fig. 10. Architecture of frequency offset compensation.

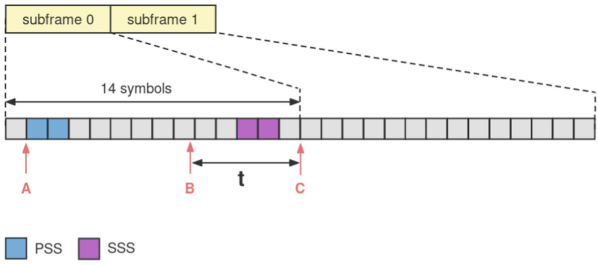

Fig. 11. Timing mark.

5 FPGA Implementation Results

FPGA implementation includes Register Transfer Level (RTL) coding, the simulation and comprehensive implementation on VIVADO 2019.1, and the test on

a Xilinx Virtex-7 XC7Z035FFG676-2 FPGA. The simulation results are shown in Fig. 12, where the output is the cross-correlation result, which is consistent with the simulation results of MATLAB R2019a. The final test result on FPGA is shown in Fig. 13, where the frame header can be marked correctly. The overall test in the subsequent project shows that the two boards is correctly synchronized and the video can be transmitted from transmitter to receiver. Resource consumption is shown in Table 1, and the energy consumption analysis is shown in Fig. 14.

Fig. 12. Simulation result on VIVADO 2019.1.

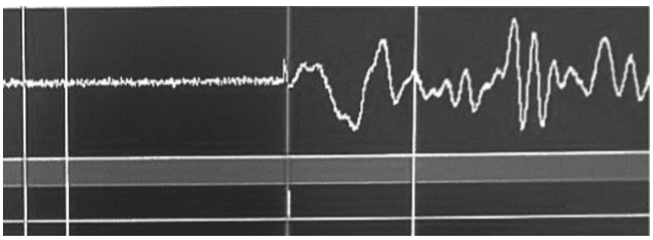

Fig. 13. Test result on a Xilinx Virtex-7 XC7Z035FFG676-2 FPGA.

Table 1. The resource consumption

TYPE	Slice LUTs	Slice Registers	Block RAM Tile
number	9545	11549	14

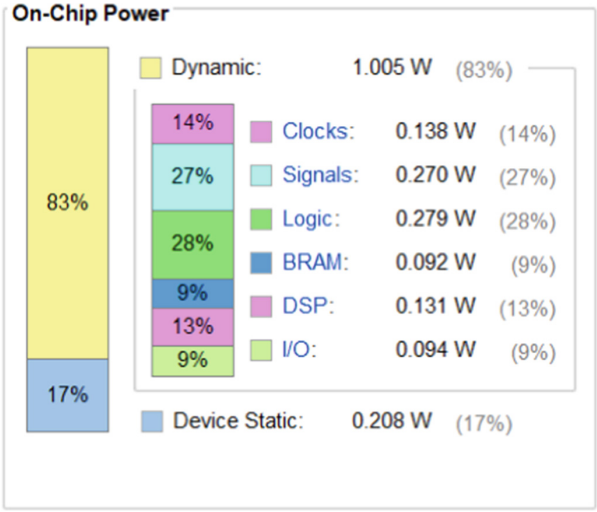

Fig. 14. Energy consumption analysis.

6 Conclusion

Based on the structure of LTE-V physical layer and the characteristics of the synchronization signal, a resource- and power-efficient implementation of PSS synchronization on FPGA in vehicular networks is proposed. Referring to traditional algorithm, an optimized method has been developed, which is suitable for implementation for FPGA. After completing comprehensive simulation analysis on MATLAB R2019a and digital Design on VIVADO 2019.1, this method is finally tested on a Xilinx Virtex-7 XC7Z035FFG676-2 FPGA. Evaluations based on test results show that the proposed design work well in LTE-V system.

References

1. Study on lte-based v2x services (v14.0.0 release 14) 3gpp (2016)
2. Physical channels and modulation (release 14) (2018)
3. et al, Z.Y.: Implementing its applications by lte-v2x equipment-challenges and opportunities, pp. 120–124 (2018)
4. M. J. Shim, J. S. Han, H.J.R., Choi, H.J.: A frequency synchronization method for 3gpp lte ofdma system in tdd mode, pp. 864–868 (2009)
5. Saifuddin, M., Zaman, M., Toghi, B., Fallah, Y.P., Rao, J.: Performance analysis of cellular-v2x with adaptive & selective power control, pp. 1–7 (2020)
6. Popovic, B.M., Berggren, F.: Primary synchronization signal in e-utra. In: 2008 IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, pp. 426–430 (2008)