Implementation of an All-Digital DRC for 100BASE-FX

Gang Luo[1], Qiangang Wang[1], Yujia Liu[2], Yuping Zhang[3(✉)], Hanyue Sun[1], and Qicheng Zhou[4]

[1] CYG SUNRI CO., LTD., Shenzhen, China
wangqg@cyg.com
[2] University of Electronic Science and Technology of China, Chendu, China
[3] Chengdu Technological University, Chendu, China
zhangyuping@cdtu.edu.cn
[4] Chengdu University of Information Technology, Chendu, China

Abstract. This paper proposes a blind oversampling data recovery algorithm LUT-DRC (Data Recovery Algorithm Based on Look-Up Table) for 100Base-FX. The LUT-DRC can recover all data in an Ethernet packet of any length, even a clock jitter of $8\,ns \pm 0.03125\,ns$ at the transmitter. The LUT-DRC core consists of only 470 LUT6, 1 block RAM, and 1 PLL (Phase-Locked Loop, PLL) and has an estimated power consumption of 10 mW at 125 Mbps. LUT-DRC was implemented on a PANGO PGL25 FPGA device and tested using a NuStreams-700 network tester. No CRC (Cyclic Redundancy Check, CRC) errors were found during data transfer testing using $2.5 * 10^8$ Ethernet packets. The characteristics of LUT-DRC and its performance make it suitable for any FPGA to implement 100BASE-FX communication without a 100BASE-FX PHY (Physical Layer Transceiver, PHY) chip.

Keywords: DRC · Oversampling · Jitter tolerance · FPGA

1 Introduction

The 100BASE-FX communication systems are generally implemented based on PHY chips [15]. However, in a communication system with an FPGA device, it is an effective method to reduce system power consumption and improve system stability by integrating the PHY into the FPGA. The 100Base-FX Ethernet adopts FDDI (Fiber Distributed Data Interface, FDDI) physical layer standard. The signal transmitted on the physical layer is the 4B5B asynchronous serial signal encoded by NRZI (Non return to zero, inverted, NRZI) [1]. Therefore, the key point to implementing a 100BASE-FX PHY transceiver inside an FPGA is the all-digital CDR (Clock and Data Recovery, CDR) circuit. FPGA device vendors only deliver the hardware IP on high-end products for the 100BASE-FX communication solutions based on SerDes (Serializer/Deserializer, SerDes). For

S. Yu et al. (Eds.): TridentCom 2022, LNICST 489, pp. 59–69, 2023.
https://doi.org/10.1007/978-3-031-33458-0_5

example, Xilinx Artix7, Spartan series and Intel Arria10, Stratix10 series [5,7,9]. However, It is challenging to implement the all-digital CDR circuit on FPGA without SerDes.

In recent years, there has been an increasing amount of literature on BO-CDR (Blind Oversampling Clock and Data Recovery, BO-CDR). For example, Seoul National University [10], The University of Tokyo [4] and University of Minnesota [8]. Previous research has established that BO-CDR is the main methods of implementing digital CDR [12]. It is based on the direct sampling method or phase discriminative coding method [10]. For example, The all-digital BO-CDR designed by The University of Tokyo in 2009 uses phase discrimination coding logic, the designed circuit is placed on the Altera Stratix GX FPGA evaluation board for verification, and its jitter tolerance is 0.9 unit interval (Unit Interval, UI).

The discrimination coding method uses a clock with a frequency M times the data rate to sample the data. Then select the sample value with the encoding value equal to $1/2M$ as the correct sample data [6,17]. This method has several advantages. Firstly, oversampling and data recovery can be implemented in a digital circuit, which makes it easy to implement in different digital systems. Secondly, BO-CDR can be locked to serial data immediately during an oversampling cycle, which makes instantaneous phase acquisition possible [8]. Thirdly, BO-CDR is realized in digital circuit, which eliminates the influence of noise signal. But BO-CDR has some disadvantages. BO-CDR needs a high-frequency encoding clock. When the data rate is n Mbps, the encoding clock is required to be $m*n$ MHz (m is the oversampling rate), which is a great challenge for FPGAs [13]. Because the timing closure is too difficult with the frequency increase in FPGA [2]. Secondly, since BO-CDR only recovers $1bit$ of sample value per cycle, this algorithm can perform well when the reference clock and data source frequencies are the same. But its resistance to clock jitter is very weak, there is a high BER (Bit Error Ratio, BER) when clocks or data drift away from each other. Third, it is complicated to implement the direct sampling method using digital circuits.

Defosez, Marc et al. Used PLL to generate the same frequency multiphase clock, then use both rising and falling edges to sample data [6]. This method reduces the clock frequency from $m*n$ MHz to n MHz or $1/2*n$ MHz. Gao Ning et al. Added a filter structure between the oversampling and the data recovery unit, to reduce the BER [13]. However, there is no good method to improve the clock jitter resistance while reducing the logic complexity.

In this paper, a LUT-DRC algorithm based on look-up table is proposed to solve above problems, which makes it possible to implement a 100Base-FX in FPGAs. And the DRC-Table obeys the basic principles and data laws of BO-CDR. Extensive test experiments have shown that the data recovery unit implemented by this method meets the performance requirements of 100BASE-FX.

2 LUT-DRC

The primary approach of the LUT-DRC is to analyze all possible data oversampling values and edge cases based on the data characteristics during transmission. These characteristics include features of FDDI in 100BASE-FX, oversampling factors, and NRZI code features. We specify the location of the data to be recovered in each case to form a DRC Table. The recovered data will be stored in a FIFO (First In, First Out, FIFO) to achieve continuous asynchronous serial data recovery. After being processed by the rate balancing module, the data will output smoothly at the original transmission rate.

This method has a simple logical structure and implementation. The rate balancing unit after the LUT-DRC turns the data stream from non-uniform into uniform, solving the problem of data bit lossing due to the clock and data rate imbalance. Our algorithm provides a new idea to implement 125 Mbps asynchronous serial data recovery in FPGAs, thus making it possible to implement 100BASE-FX PHY transceivers inside FPGAs.

2.1 Overall Structure

The overall structure of LUT-DRC is shown in Fig. 1. The input of the LUT-DRC is 125 Mbps differential serial data, and the output is a set of source synchronization signals containing clock and data. The LUT-DRC consists of three parts, signal pre-processing unit, data recovery unit, and rate balancing unit.

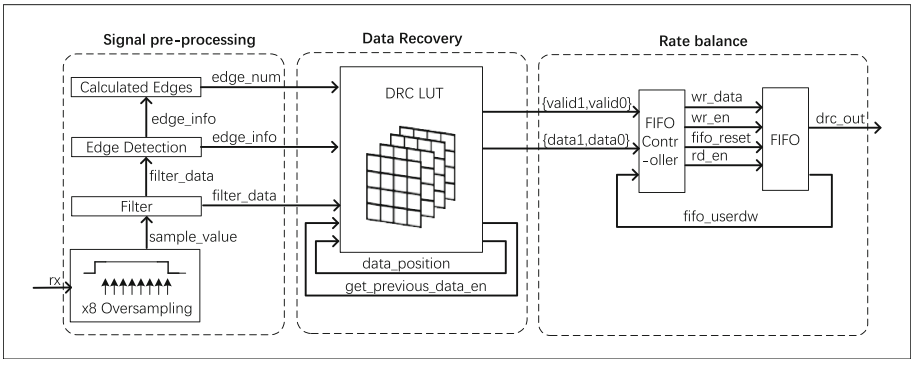

Fig. 1. LUT-DRC system schematic

Signal Pre-processing Unit. It processes the differential input serial data of 125 Mbps in four steps. (1) Oversample the signal by eight times. (2) Filter the oversampled signal to suppress signal interference caused by non-monotonic or overshoot [13]. (3) Calculate the edge position of the current cycle. (4) Calculate

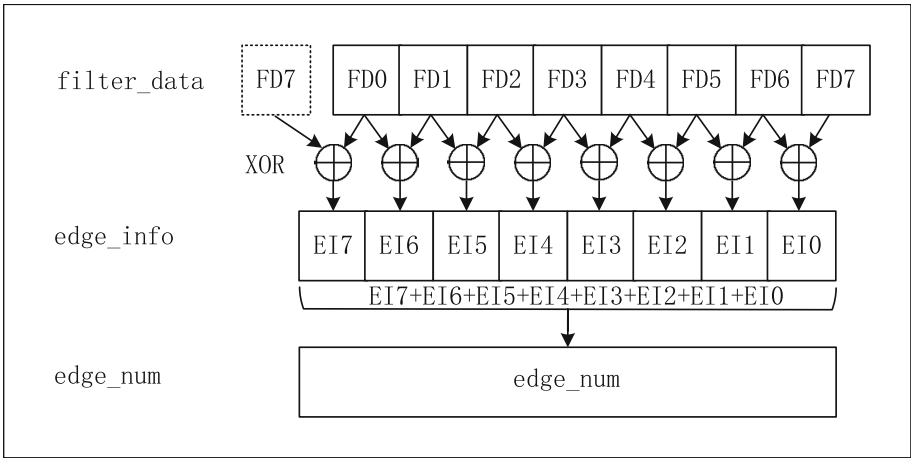

Fig. 2. Schematic diagram of pre-processing mechanism

the number of edges in the current cycle. The schematic of data pre-processing is shown in Fig. 2.

Data Recovery Unit: It recovers data from sampled values based on the output of the signal pre-processing unit and feedback signals from DRC-Table.

Rate Balance Unit: It turns non-homogeneous data streams caused by the clock or data jitter into a homogeneous, uninterrupted data stream.

2.2 Data Recovery Unit

The core of our data recovery algorithm is the DRC Table, implemented by LUTs. We set the following rules based on the LUT-DRC system's characteristics to infer the DRC Table.

1. *Specifying the optimal sample output position: According to BO-CDR theory, the middlemost sample after the data edge is considered optimal recovery data. Therefore, for the x8 oversampling system, we specify the 3rd bit after the data edge as the optimal sample position, as shown in Fig.* 3.
2. *Specifying the setup time: According to the description of the optimal sample output position, a 3-bit wide setup time, Tsu, is required, as shown in Fig.* 3.
3. *Specifying the hold time: For signals, the hold time is generally required to be shorter than the setup time* [3,11]*, so an additional sample interval (1 bit) is required behind the optimal sample output position. Hold time is shown as Th in Fig.* 3.
4. *Specifying the minimum symbol width: Each symbol must be sampled at least five times steadily, according to the setup time and hold time requirements, as shown in Fig.* 3.

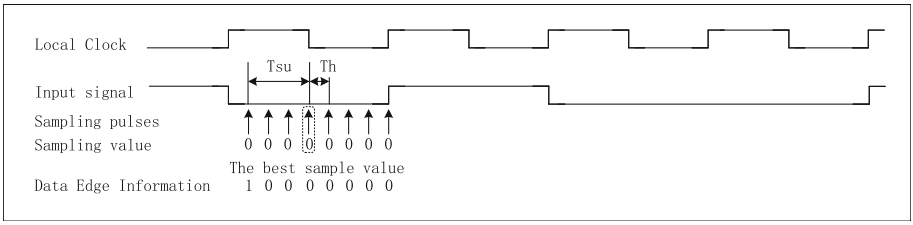

Fig. 3. LUT-DRC Timing Diagram

Infer the possible sampling signal and edge information according to the four rules above.

1. *Two edges in the sampled data: According to rule No.4, each symbol is sampled at least five times. Thus, there must be greater or equal to 4 zeros after each transition edge. Since LUT-DRC uses x8 oversampling, there will be at most two edges and at least one edge per cycle, and there must be at least four consecutive zeros between two edges. Therefore, all six edge-info permutations of the sampled signal with two edges are listed in the edge-num = 2 column, as shown in Fig. 4.*
2. *One edge in the sampled data: Typically, when a clock matches the data rate, a symbol will be sampled eight times in a cycle, and only one edge signal will be generated. The edge position is related to the initial state and may occur at any bit of the sampled value. All eight edge-info permutations of the sampled signal with a single edge are listed in the edge-num = 1 column, as shown in Fig. 4.*

Infer the Residual Flag and Locations of Recovered Data

When the edge is at the lower 0 to 2 bits of the sampled data, the optimal sample position specified by rules No. 2 and 3 is not in the current cycle and will only be output in the next cycle. In such a case, the data corresponding to one edge is left to recover to the next cycle. The formulae for calculating the valid signal of the residual edge and position information are defined by Eq. 2 and Eq. 3.

$$Output\ position\ of\ data\ =\ edge\ position - \ 3 \tag{1}$$

$$Residual\ edge\ signal\ =\ (rightmost\ edge\ position\ <\ 3)\ ?\ 1 : 0 \tag{2}$$

$$Residual\ data\ position\ =\ rightmost\ edge + 7 - \ 2 \tag{3}$$

Based on the inferred edge information and Eq. 1 to 3, the DRC Table is designed as shown in Fig. 4. The DRC Table has four inputs: the edge-num and edge-info in the current sampling cycle, the valid signal get-previous-data-en of the residual edge calculated in the previous cycle, and the data-position of the previous edge. The outputs are data1, data0, and their valid bit valid1, valid0 of the current cycle, and the edge information get-previous-data-en and data-position for the next cycle. The data bits and their valid bits are output to the rate balancing unit as recovered data for the current cycle, and the edge residual is fed to the DRC Table as two inputs for the next cycle.

Input		Output		
edge_num	edge_info	recovered data	get_previous_data_en	data_position
0	xxxxxxxx	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[data_position],1'b0};	0	/
	xxxxxxxx	{valid1,valid0} <= {1'b0,1'b0}; {data1,data0} <= {1'b0,1'b0};	0	/
1	xxxxxxxx	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[data_position]};	0	/
	10000000	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[4],1'b0};	0	/
	01000000	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[3],1'b0};	0	/
	00100000	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[2],1'b0};	0	/
	00010000	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[1],1'b0};	0	/
	00001000	{valid1,valid0} <= {1'b1,1'b0}; {data1,data0} <= {filter_data[0],1'b0};	0	/
	00000100	{valid1,valid0} <= {1'b0,1'b0}; {data1,data0} <= {1'b0,1'b0};	1	7
	00000010	{valid1,valid0} <= {1'b0,1'b0}; {data1,data0} <= {1'b0,1'b0};	1	6
	00000001	{valid1,valid0} <= {1'b0,1'b0}; {data1,data0} <= {1'b0,1'b0};	1	5
2	10000001	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[4]};	1	5
	10000001	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[4]};		
	10000010	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[4]};	1	6
	10000010	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[4]};		
	10000100	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[4]};	1	7
	10000100	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[4]};		
	01000001	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[3]};	1	5
	01000001	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[3]};		
	01000010	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[3]};	1	6
	01000010	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[3]};		
	00100001	{valid1,valid0} <= {1'b1,1'b1}; {data1,data0} <= {filter_data[data_position],filter_data[2]};	1	5
	00100001	{valid1,valid0} <= {1'b0,1'b1}; {data1,data0} <= {1'b0,filter_data[2]};		

Fig. 4. LUT-DRC Table

2.3 Rate Balance Unit

FIFO is often used for CDC (Clock Domain Crossing, CDC), data bit width conversion, and data buffering in high-speed serial communications [16]. For example, in the JESD204B protocol, FIFO is used to data align and CDC [14]. In LUT-DRC, the function of FIFO is CDC and data bit width conversion.

FIFO Control Circuit - Write Control Principle. The FIFO write control circuit uses a 9-bit leftward shift register. Valid0 and valid1 signals determine whether we shift data0 or data1, or both into the shift register. For example, suppose valid0 is asserted, then data0 will be shifted into the shift register. Due to the mismatch between the clock and data rate, 2-bit or 1-bit may shift into the shift register per cycle, so we need a 9-bit shift register. When the shift register contains only 8-bit data, we write the lower 8-bit data to the FIFO. Otherwise, we write the upper 8-bit data to the FIFO instead. All cases are shown in Fig. 5.

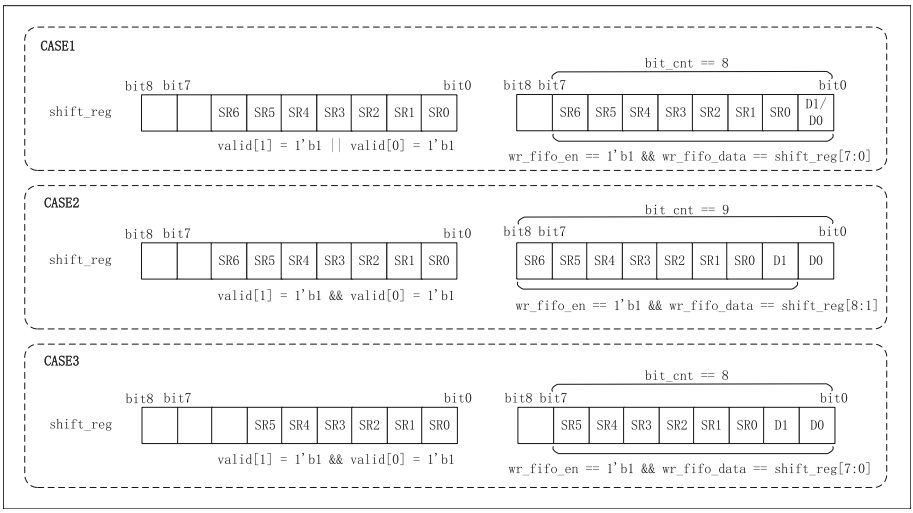

Fig. 5. Rate blance Structure

FIFO Control Circuit - Read Control Principle. The circuitry after DRC requires only 1-bit per cycle and does not allow any interruption in an Ethernet message frame. Otherwise, it will cause RMII (Reduced Media-Independent Interface, RMII) timing recovery errors. Therefore, the FIFO read control module is designed as follows: (1) Reset the FIFO read/write pointer after detecting the JK code in the transmission message. (2) Read FIFO when the userdw signal is equal to the threshold value till the FIFO is empty.

2.4 Anti-jitter Performance of the LUT-DRC

The ability of the proposed LUT-DRC to tolerate persistent slow and fast clock jitter depends on the userdw value in the rate balancing module. For the userdw, having a larger or smaller value is not better. If the value is too small, there is a risk that the FIFO will be read empty, and the Ethernet packet data will be interrupted when the data rate at the sending side is slower than the local clock rate. Suppose the value is too large and the data rate at the sending side is faster

than the local clock rate. In that case, there is a risk that the FIFO will be reset by the arrival of a new frame of Ethernet messages before the previous messages read out from the FIFO. Thus the previous frame of Ethernet messages may not be fully recovered. The maximum length of an Ethernet frame is 1.5 kByte, which is 12288 symbols. According to Eq. 4 and 5, when userdw is 48 bits, LUT-DRC has the same resistance to fast and slow clock jitter, allowing the symbol width to be lengthened by 0.0312 ns or shortened by 0.03125 ns, respectively. Such a jitter resistance is multiple times higher than the ±0.00040 ns specified by IEEE 802.3.

$$\textit{Slow clock tolerance} = \textit{userdw} * \textit{8 ns / 12288} \tag{4}$$

$$\textit{Fast clock tolerance} = \textit{(96 - userdw)} * \textit{8 ns / 12288} \tag{5}$$

For the single mutation tolerance of a single cycle, the allowable signal mutation length is ±3 ns, as specified in Sect. 2.2. The mutation length of a symbol is ±3 ns, thus allowing a symbol width to mutate from 8 ns to 5 ns or 11 ns.

3 Evaluation

In the experiment, the LUT-DRC algorithm is implemented based on the PGL25-6MBG324 device from PANGO. The development environment is PDS 2020.3.SP2 and ADS by SHENZHEN PANGO MICROSYSTEMS CO., LTD.

3.1 RTL Design Results

The resource utilization of the proposed LUT-DRC algorithm based on FPGA implementation is shown in Table 1. From the table, LUT-DRC utilized only 470 LUT6 and has a low power consumption of 10 mW. The PLL can be shared when multiple PHYs are implemented in the same FPGA, while other resources are related to the number of PHYs instantiated.

Table 1. Resource Utilization and Power

Resource Utilization		
Resource	Utilization	Used for
LUT6	470	RTL Logic
DRAM	1	Rate balance FIFO
APMs	0	–
GPLL	1	x8 oversample clock
IOL	1	x8 oversample
Power		
Total Power		10 mW

3.2 Simulation Results

The simulation results with −0.04 ns and +0.032 ns jitter in the transmit clock are shown in Fig. 6 and Fig. 7. When the symbol period is shortened to 7.96 ns or extended to 8.032 ns due to the clock jitter, the DRC at the receiver side can still recover the original data and restore the symbol period to 8 ns. It shows that the LUT-DRC has a higher clock jitter resistance than IEEE 802.3 standard. When the transmitter clock cycle is less or greater than 8 ns, 2 bits or 0 bit data is recovered during this cycle to balance the clock jitter. The data valid signal dout-valid in such case is shown as ② in Fig. 6 and Fig. 7, respectively.

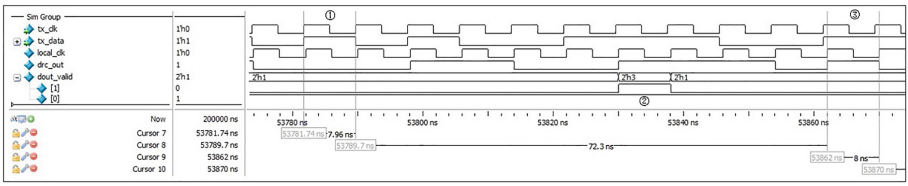

Fig. 6. Fast clock jitter simulation of LUT-DRC

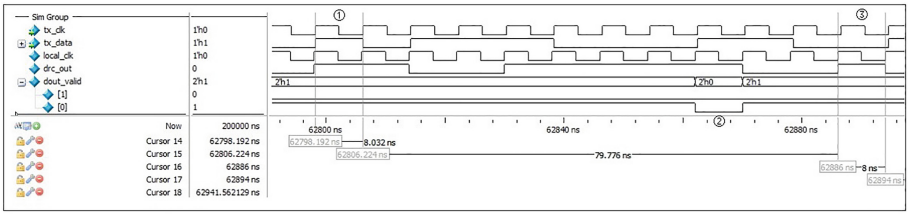

Fig. 7. Slow clock jitter simulation of LUT-DRC

3.3 Test Results

The 100Base-PHY transceiver is implemented based on LUT-DRC, then connected to the network tester and the internal MAC of the FPGA. The test structure is shown in Fig. 8. During the test, the network tester sends 64 Byte, 448 Byte, 832 Byte, 1216 Byte, and 1518 Byte length Ethernet messages to the test board with a minimum frame interval of 96 bits. The data received by the test board will go through LUT-DRC and MAC (Media Access Control, MAC), then loop back to the tester. Results are shown in Table 2.

As shown in Table 2, both the bit error rate and the CRC error rate are zero after transferring $2.5 * 10^8$ packets to the LUT-DRC under the extreme condition specified by the Ethernet data messaging protocol (minimum message spacing of 96 bits, minimum message frame length of 64 bytes, and maximum message frame length of 1518 bytes).

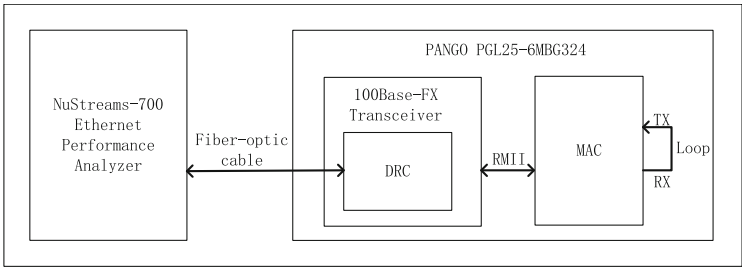

Fig. 8. Test structure

Table 2. Test results statistics

Trial	Frame	Rate	Max	Min	Avg	Tx Packets	Rx packets
1	64	100.00	0.00	0.00	0.00	178570800	178570800
1	448	100.00	0.00	0.00	0.00	32050800	32050800
1	832	100.00	0.00	0.00	0.00	17605200	17605200
1	1216	100.00	0.00	0.00	0.00	12135600	12135600
1	1518	100.00	0.00	0.00	0.00	9752400	9752400

4 Conclusion

This paper proposes and evaluates LUT-DRC, a blind oversampling data recovery algorithm based on the look-up table approach. It has several advantages over the existing BO-CDR algorithm. Firstly, it has a more straightforward implementation than the existing methods. Secondly, it has a high clock jitter resistance, enabling it to recover data with ±0.0312ns of transmitter clock jitter. This performance is higher than the requirements of IEEE802.3 standard. The LUT-DRC was validated on FPGA devices and achieved ideal results with no errors in over 200 million packets of Ethernet messages. Such test results demonstrate that LUT-DRC can be used to implement clock data recovery in FPGAs. We also gives a direct relationship between the clock jitter resistance and the FIFO userdw signal in the LUT-DRC design process. The relationship can provide a certain theoretical calculation basis for the design of DRC in other application scenarios.

References

1. IEEE Standard for Ethernet. IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015), pp. 1–5600 (2018)
2. Aggarwal, R.: FPGA place & route challenges. In: Proceedings of the 2014 on International Symposium on Physical Design, pp. 45–46 (2014)
3. Balef, H.A., Jiao, H., de Gyvez, J.P., Goossens, K.: An analytical model for interdependent setup/hold-time characterization of flip-flops. In: 2017 18th International Symposium on Quality Electronic Design (ISQED), pp. 209–214. IEEE (2017)

4. Bushnaq, S., Nakura, T., Ikeda, M., Asada, K.: All digital baseband 50 Mbps data recovery using 5× oversampling with 0.9 data unit interval clock jitter tolerance. In: 2009 12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp. 206–209. IEEE (2009)
5. Xilinx Datasheet. Artix-7 FPGAS data sheet: DC and AC switching characteristics v1. 18 (2015)
6. Defossez, M.: LVDS 4x asynchronous oversampling using 7 series FPGAS and ZYNQ-7000 ap socs. Xilinx Inc. (2017)
7. Cyclone V Device Handbook, vol. 1, Device Overview and Datasheet (2012)
8. Hsieh, M., Sobelman, G.E.: Architectures for multi-gigabit wire-linked clock and data recovery. IEEE Circ. Syst. Mag. **8**(4), 45–57 (2008)
9. Ismail, K., Ismail, T., Mostafa, H.: Design and implementation of CDR and SerDes for high speed optical communication networks using FPGA. In: 2016 18th International Conference on Transparent Optical Networks (ICTON), pp. 1–3. IEEE (2016)
10. Kim, J., Jeong, D.-K.: Multi-gigabit-rate clock and data recovery based on blind oversampling. IEEE Communications Magazine **41**(12), 68–74 (2003)
11. Kim, K.-C.: Measurement of setup and hold time in a CMOS DFF for a synchronizer. J. Korea Inst. Electron. Commun. Sci. **10**(8), 883–890 (2015)
12. Lin, Y.H., Tu, S.H.L.: Implementation of an oversampling data recovery receiver for serial link communications. In: Seventh International Symposium on Signal Processing and Its Applications, vol. 1, Proceedings, pp. 613–616. EURASIP; IEEE; IEEE French Chapter, Paris, France, 01–04 July 2003
13. Gao, N., Zhang, Z., Fang, Y., Guo, Y., Liu, L.-L.: The implementation of a high-performance blind oversampling clock data recovery circuit. Microelectron. Comput. **31**(6), 137–140 (2014)
14. Saheb, H., Haider, S.: Scalable high speed serial interface for data converters: using the JESD204B industry standard. In: 2014 9th International Design and Test Symposium (IDT), pp. 6–11. IEEE (2014)
15. Sugimoto, N., Fukushima, S., Sakai, Y., Oguchi, K., Akatsu, Y.: A small optical Ethernet PC card for fiber-to-the-notebook pcs and its applications. In: Optical Transmission, Switching, and Subsystems II, vol. 5625, pp. 491–497. SPIE (2005)
16. Sung, G.-M., Tung, L.-F., Wang, H.-K., Lin, J.-H.: USB transceiver with a serial interface engine and FIFO queue for efficient FPGA-to-FPGA communication. IEEE Access **8**, 69788–69799 (2020)
17. Yang, C.-K.K., Farjad-Rad, R., Horowitz, M.A.: A 0.5-m CMOS 4.0-Gbit/s serial link transceiver with data recovery using oversampling. IEEE J. Solid-State Circ. **33**(5), 713–722 (1998)