High-Speed Post-Processing in Continuous-Variable Quantum Key Distribution Based on FPGA Implementation

Shen-Shen Yang, Zhen-Guo Lu, and Yong-Min Li

***Abstract*—In a continuous-variable quantum key distribution (CV-QKD) system, the computation speed of the post-processing procedure, including information reconciliation (IR) and privacy amplification (PA), inevitably affects the practical secret key rate. IR and PA can be implemented in parallel using low-density parity-check (LDPC) codes and hash functions, respectively. We achieve high-speed hardware-accelerated post-processing procedure for Gaussian symbols on a field-programmable gate array (FPGA) by taking advantage of its superior parallel processing ability. To this end, the sum-product algorithm decoders and a modified LDPC codes construction algorithm adapted to FPGA's characteristics are developed and employed. Two different structures including multiplexing and non-multiplexing are designed to achieve the trade-off between the speed and area of FPGAs, so that an optimal scheme can be adopted according to the requirement of a practical system. Simulation results show that the maximum throughput can reach 100 M symbols/s. We verified the correctness of the post-processing procedures when implemented on the Xilinx VC709 evaluation board, which is populated with the Virtex-7 XC7VX690T FPGA and provided some possible solutions to obtain better performance when more advanced FPGAs are available. The scheme can be applied readily for real-time key extraction and effectively reduce power consumption of the CV-QKD system.**

***Index Terms*—Continuous-variable quantum key distribution (CV-QKD), FPGA, post-processing, real-time.**

I. Introduction

In QUANTUM key distribution (QKD) [1]–[6], Alice sends quantum states in which keys are encoded to Bob through a quantum channel, and Bob measures the received states with random bases. Then, a secret key secure in the information-theoretic sense can be extracted and shared between Alice and Bob. A QKD system typically includes three steps: (1) quantum state preparation, distribution, and measurement; (2) data sifting and parameter estimation; and (3) post-processing procedure. The last can be divided into two main parts, information reconciliation (IR), in which Alice and Bob use error correction codes (ECCs) to correct errors in sifted keys to obtain identical corrected keys, and privacy amplification (PA), which eliminates the leaked information by using universal classes of hash functions to map long-corrected keys to shorter secure keys.

There are three types of QKD protocols [2]: discrete-variable QKD (DV-QKD), continuous-variable QKD (CV-QKD) [3]–[5], and distributed phase reference QKD (DPR-QKD) [7]. CV-QKD has attracted much attention in recent years, mainly because its information is encoded in the amplitude and phase quadratures of quantum states so that high-sensitive low-noise heterodyne or homodyne detection techniques can be employed. It also has potential higher secret key rates over metropolitan areas. However, the IR procedure in CV-QKD systems is relatively sophisticated since it works in low signal-to-noise ratio (SNR) conditions. With the progress of experimental research [8]–[11], the system's repetition rate is becoming increasingly higher, and the data throughput of the post-processing procedure must be fast enough; otherwise, it will reduce the actual key rate.

Several schemes have been proposed for IR of Gaussian symbols, such as slice reconciliation [12]–[15], multidimensional reconciliation [16]–[18], and other schemes [19], [20], which are applicable to a certain range of SNRs. As shown in Fig. 1, the slice reconciliation can obtain a higher secret key rate in the case of shorter distance and larger SNR (from 1 to 15), while multidimensional reconciliation, usually combined with multi-edge type low-density parity-check (LDPC) codes [17], [21], [22] or Raptor codes [23], [24], is optimal for a smaller SNR from 0.01 to 1 and is limited to 1 bit per pulse. For IR in CV-QKD, very large block ECCs are usually required to achieve high reconciliation efficiency on the additive white Gaussian noise channel (AWGNC). Because of the large block codes and many iterations required in IR, the throughputs of the error correction algorithms is very limited on a central processing unit (CPU). One feasible way to improve the throughput is to exploit hardware-based acceleration. Based on graphics processing unit (GPU), a throughput of 9.17 Mb/s is obtained for multidimensional reconciliation [22]. This record is further improved to 30.39 Mb/s [25] (also based on a GPU). Currently, high-speed slice reconciliation based on hardware acceleration has not been achieved.

Manuscript received January 16, 2020; revised February 28, 2020 and March 29, 2020; accepted March 30, 2020. Date of publication April 6, 2020; date of current version July 23, 2020. This work was supported in part by the National Key R&D Program of China under Grant 2016YFA0301403, in part by the National Natural Science Foundation of China (NSFC) under Grant 11774209 and Grant 61378010, in part by the Key R&D Project of Shanxi Province under Grant 201803D121065, and in part by the Shanxi under Grant 1331KSC. *(Corresponding author: Yong-Min Li.)*

The authors are with the State Key Laboratory of Quantum Optics and Quantum Optics Devices, Institute of Opto-Electronics, Collaborative Innovation Center of Extreme Optics, Shanxi University, Taiyuan 030006, China (e-mail: 201612607013@email.sxu.edu.cn; 201812607010@email.sxu.edu.cn; yongmin@sxu.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2020.2985408

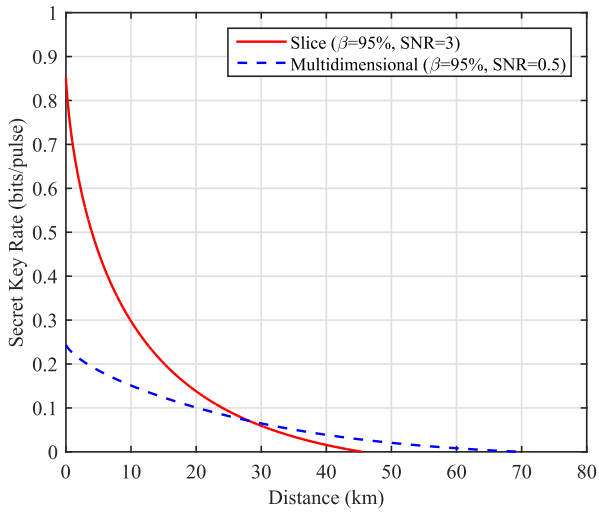

Fig. 1. Performance of slice reconciliation vs multidimensional reconciliation with the same reconciliation efficiency $\beta = 95\%$. Other parameters are excess noise $\xi = 0.01$, efficiency of receiver's detector $\eta = 0.64$, electronic noise at Bob's side $V_{el} = 0.1$.

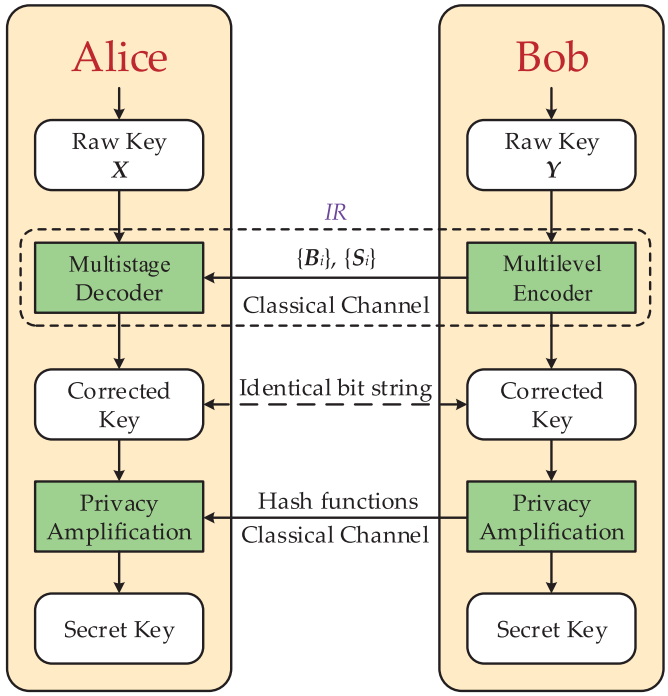

Fig. 2. Schematic of the post-processing with slice reconciliation and PA in the CV-QKD system. $\{\boldsymbol{B}_i\}$ are the levels corresponding to the less significant bits, and $\{\boldsymbol{S}_i\}$ are the syndromes of the significant bits calculated according to the parity-check matrix of LDPC codes.

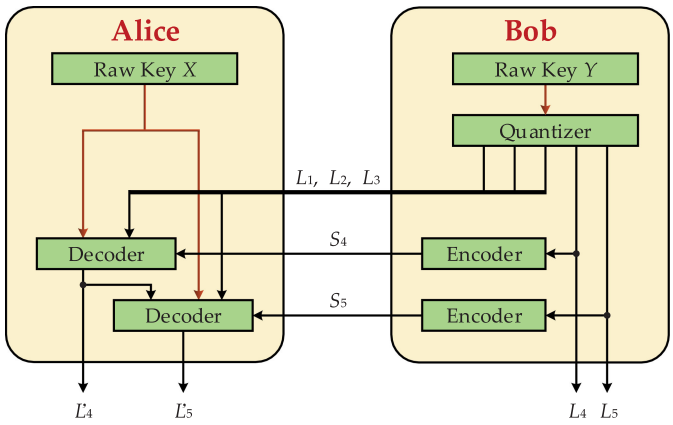

Fig. 3. Slice reconciliation based on five levels of encoding and decoding. $RawKeyX$ and Y are correlated Gaussian symbols obtained after base sifting in a CV-QKD system. $Quantizer$ is used to quantize the Gaussian symbols Y and obtain five bit strings L_1, L_2, L_3, L_4, and L_5; S_4 and S_5 are syndromes generated by LDPC encoding modules $Encoder$ of levels 4 and 5, respectively; L'_4 and L'_5 are corrected secret keys output from the LDPC decoding modules $Decoder$ of levels 4 and 5.

Field-programmable gate arrays (FPGAs) are very attractive when designing prototypes [26]–[29] and manufacturing small-production-run devices, and their in-system programmability makes them substantially cost-effective. In addition, FPGAs are good at binary operations and can configure the degree of parallelism more flexibly, so that it facilitates control over the trade-off between data throughput and hardware resource requirements. Furthermore, the power consumption of an FPGA is much lower than that of a GPU and other hardware.

In this work, we achieve a high-speed post-processing procedure based on FPGA hardware acceleration. To this end, the sum-product algorithm decoders and a modified LDPC codes construction algorithm adapted to FPGA characteristics are developed and employed. Two different structures, including multiplexing and non-multiplexing, are designed to achieve the trade-off between the area and speed of FPGAs, so that one can adopt optimal schemes according to the requirements of a practical system. By implementing multiple code words decoding simultaneously, the throughput of slice reconciliation can reach 100 M symbols/s on a Xilinx Virtex-7 FPGA. Finally, we integrate IR and PA [30] to realize a complete real-time post-processing procedure.

The rest of this paper is organized as follows. In Section II, we briefly review the post-processing procedure in a CV-QKD system. Then, we present the two different structures of the post-processing procedure and the design of LDPC decoders in Section III. In Section IV, we present detailed implementation results for the post-processing procedure on FPGA. In Section V, we give a brief summary and discussion.

II. Post-Processing in CV-QKD

Fig. 2 shows a typical structure of the post-processing procedure with reverse slice reconciliation and PA in CV-QKD systems. In the IR part, LDPC codes are employed to achieve high-speed and high-efficiency IR. The PA is implemented by the Toeplitz matrix, which is a special universal hash function.

A. Slice Reconciliation Protocol

Slice reconciliation [12]–[15] is an error correction scheme for Gaussian symbols using binary error correcting codes. It works in different ways on both parties; see Fig. 3 for a schematic

description of the slice reconciliation. For reverse slice reconciliation, Bob uses the quantizing function $Q:\mathbb{R}\to\{0,1\}^m$ to transform each continuous Gaussian variable Y_i into an m-bit label $\{B_j(Y_i)\}, j=1,...,m$. Next, Bob uses multilevel coding (MLC) by encoding each individual level j of label bits independently as the syndrome of an error correcting code with rate $R_j(1\le j\le m)$. To recover Bob's m-bit label $\{B_j\}$, the sender Alice employs multistage decoding (MSD) using its own source X as side information. Finally, the two parties share identical corrected keys.

In our scheme, raw key Y are quantified to five levels. The information included in the first three levels is very small and disclosed directly, and the latter two levels are encoded and successively decoded. More precisely, Bob must send the quantized bit strings $\{B_1(Y_i)\}$, $\{B_2(Y_i)\}$, and $\{B_3(Y_i)\}$ of the first three levels and the syndromes S_4 and S_5 of the last two levels to Alice. For simplicity, hereafter we use L_1, L_2, and L_3 to represent $\{B_1(Y_i)\}$, $\{B_2(Y_i)\}$, and $\{B_3(Y_i)\}$.

The efficiency β of the slice reconciliation is given by

$$\beta=\frac{H\left[Q(Y)\right]-m+\sum_{i=1}^{m}R_i}{I\left(X;Y\right)}, \tag{1}$$

where $Q(Y)$ is the discrete symbol by quantizing Y, $H[Q(Y)]$ is the information entropy of $Q(Y)$, and $I(X;Y)$ is the mutual information between Alice and Bob.

B. Standard LDPC Codes

LDPC codes [31] is a good performance error correction code and can be used to achieve high-efficiency IR very close to the Shannon limit. The performance of LDPC decoders greatly affects the reconciliation efficiency in CV-QKD. To implement LDPC decoders based on hardware, the approaches of each step should be carefully designed to achieve the best performance, including decoding algorithms, message propagation schedule, construction of check matrix, and so on.

1) Decoding Algorithms: LDPC codes are usually decoded using a belief propagation (BP) algorithm, in which messages typically in the form of a logarithmic-likelihood ratio (LLR) are iteratively passed in both directions along the edges between connected nodes.

The two most dominating LDPC decoding algorithms are the sum-product algorithm (SPA) and the min-sum algorithm (MSA). Note that two modifications of the MSA, called the normalized MSA and offset MSA [32], have been proposed to improve performance. Although MSAs are easy to implement on FPGA, the highest reconciliation efficiency in our test that can be achieved is less than 90%. Therefore, the SPA is chosen in our design.

2) Message Propagation Schedule [33]: The schedule of the LDPC decoding process determines the order in which variable nodes and check nodes are processed, as well as whether multiple nodes are processed in parallel. Flooding, layered belief propagation (LBP) [34], and informed dynamic scheduling are three widely used schedules. LBP tends to converge to the correct code word with fewer iterations and therefore has lower computational complexity, and the memory size required for a LBP decoder is half of that required for a flooding decoder. In consideration of these features, the LBP schedule is adopted in our scheme.

3) Construction of Parity Check Matrix (PCM): In addition to the size of matrix block and the degree distributions of its nodes, the position of the edges also has a significant impact on error correction performance. A number of techniques have been proposed, such as random construction, the progressive edge growth (PEG) algorithm, and quasi-cyclic (QC) codes. Previous work showed that the PEG algorithm has better performance at SNR ~ 3 [14], while random construction exhibits better performance at SNR ~ 1 [15]. QC codes are defined by a parity-check matrix constructed from an array of $q\times q$ cyclically shifted identity matrices or $q\times q$ zero matrices. QC codes impose a highly regular parity-check matrix structure with a sufficient degree of randomness offset to achieve near-Shannon-limit error correction performance while reducing the implementation complexity of the decoder. QC codes also reduce data permutation and memory access complexity by eliminating random, unordered memory access patterns.

In the proposed scheme, a small base matrix is first constructed using the PEG algorithm or random construction, depending on the specific SNRs, and a large block matrix is successively obtained by quasi-cyclic expansion. When implementing the LDPC decoders based on FPGA, there are inevitably some clock delays between the writing and reading of LLR data during the calculating procedure. To use the pipeline structure to improve the throughput, the structure of the matrix must have enough interval between the two nodes in the same column to avoid read-write conflicts.

C. Practical Secret Key Rate

Accounting for the finite-size effects [35], the secret key rate of a reverse reconciliation CV-QKD system is given by $K_{finite}=(n/N\ )(\beta I_{AB}-\chi_{BE}-\Delta(n))$, where n is the number of data used to distill the secret key, N is the number of sifted data after quantum transmission and measurement, $N-n$ is the number of data used for parameter estimation, I_{AB} is the mutual information between Alice and Bob, χ_{BE} is the Holevo bound, and $\Delta(n)$ is the finite-size offset factor.

In the above, the secret key rate is derived without considering the actual throughput of the post-processing procedure; in other words, we assume that the post-processing procedure is real-time. In a practical QKD system, the computation time of the post-processing procedure is limited. More precisely, the practical key rate is affected by a factor $\alpha=PP_{out}/PP_{in}$, where PP_{in} and PP_{out} represent the post-processing input and output rates, respectively [36]. In general, we have $0<\alpha\le 1$. Furthermore, raw key blocks incorrectly decoded in a QKD system are usually simply discarded by the sender and receiver. As a result, the final key rate are also affected by the frame error rate (FER) of the LDPC decoder.

Taking into account all the previous factors in the CV-QKD system, the practical secret key rate is given by

$$K_{prac}=\alpha\left(1-FER\right)\left(\frac{n}{N}\right)\left(\beta I_{AB}-\chi_{BE}-\Delta(n)\right). \tag{2}$$

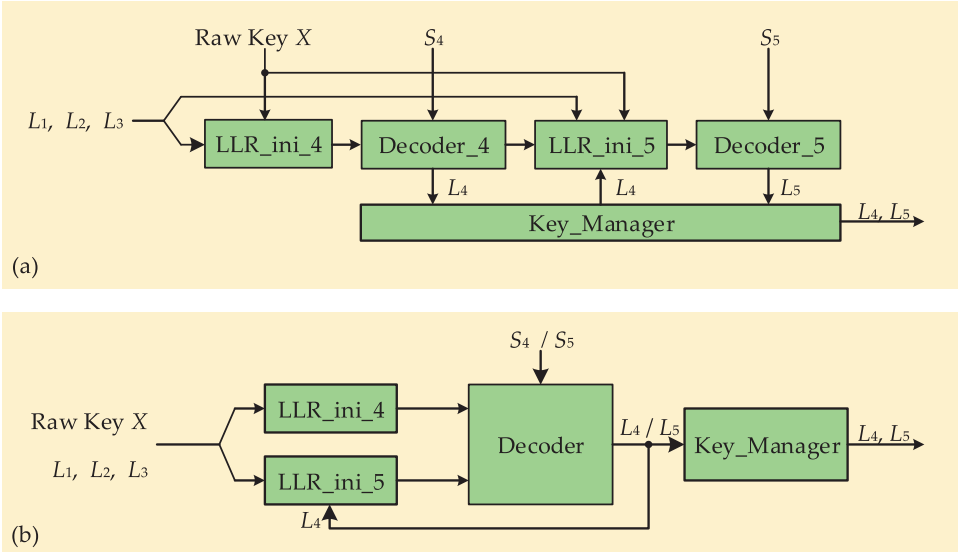

Fig. 4. (a) Fully pipelined non-multiplexed structure and (b) two-level multiplexing structures in IR. LLR_ini_4 and LLR_ini_5 modules are used to generate the initial LLR data of levels 4 and 5 before iterative decoding, respectively. $Decoder\_4$ and $Decoder\_5$ modules in (a) are FPGA-based LDPC decoders in levels 4 and 5, respectively. $Decoder$ module in (b) is multiplexed in two levels. $Key\_Manager$ module is used to store the corrected secret keys and manage their inputs and outputs.

III. Design and Implementation

In this section, we will describe the details of how to achieve the post-processing procedure on FPGA. FPGAs have the advantage of parallel computing and low power consumption, but its hardware resources are limited, particularly the block memory and logic gates (LUTs). Therefore, a crucial challenge is to achieve the post-processing procedure using only the limited hardware resources. Although the storage medium can employ an external DDR, the access rate of a DDR usually cannot meet the high-speed requirements, which is necessary for high-speed LDPC codes decoding.

The schematic of the logic structure of the post-processing procedure is shown in Fig. 2. It consists of several main modules, i.e., a multilevel encoder, multistage decoder, privacy amplification, and classic communication through classical channels. The implementation of multilevel encoders on an FPGA is relatively simple. First, the Gaussian symbols are quantized, and then matrix multiplication is performed to calculate the syndromes. The multilevel encoders consume very little hardware resources and can provide high throughput.

A. Implementation of the Multistage Decoder

The implementation of a multistage decoder on an FPGA is complex and consumes more hardware resources. To decrease the consumption of hardware resources, two approaches are adopted: 1) using short-code-length LDPC codes, and 2) using as few bits as possible to represent the natural number during the calculation. When the SNR is relatively high, for example, SNR $\sim$ 3, the code length is short. In this case, to improve the throughput, we create two or more decoders on an FPGA, and each decoder performs the fourth- and fifth-level decoding in parallel. Fig. 4(a) shows the fully pipelined non-multiplexed structure in levels 4 and 5 using two independent decoders, $Decoder\_4$ and $Decoder\_5$, which can have different parallel parameters.

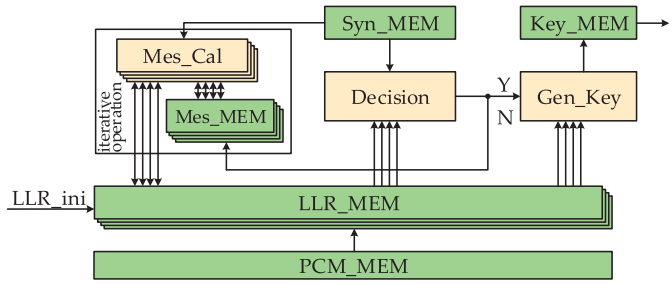

Fig. 5. Logic structure of LDPC decoders. Green blocks are memory modules, and pink blocks are calculation modules. The five memories store the parity check matrix, LLRs, syndromes, node messages, and secret key, respectively.

When the SNR is low, for example, SNR $\sim$ 1, the consumption of the hardware resources is high due to long code length. To address this issue, we propose a multiplexing method. Owing to the high architectural consistency, there are minor differences between the two LLR initialization modules of levels 4 and 5. Considering that the required hardware resources are small, we create two LLR modules and operate them in parallel. Note that in this structure, the quasi-cyclic extension parameters of the two-level check matrix must be the same. Fig. 4(b) shows the structure of two-level multiplexing decoding in the IR part. In the multiplexing structure, not only are the LUTs resources reused, more importantly, the very limited storage resources can be reused as well.

Fig. 5 shows the logic structure of the LDPC decoder. After the LLR message is initialized, it is written into LLR_MEM,

which is true dual port RAM. The Mes_Cal module calculates the messages of each node and updates LLR data. After completing the decoding process, the $Decision$ module determines whether the decoding is successful. If the decoding is successful, the corrected key is generated by the Gen_Key module; otherwise, the iterative operation is performed again. In the LDPC decoder, all data are calculated in parallel using fixed-point numbers to save memory resources.

B. Integration of IR and PA

For PA, we designed the Rhomboid-Block Operation algorithm, which consists of parallel multiply-accumulator units, in which each unit performs multiple single-bit multiplication. We can exert control over the trade-off between throughput and hardware resource consumption by choosing a reasonable parallel parameter. For a more detailed description of the FPGA-based PA algorithm, please refer to [30].

When integrating IR and PA, one of the key points is that the processing speeds of these two parts must match each other to avoid wasting limited hardware resources. Owing to the simplicity of the PA algorithm, its throughput can be much higher than that of the IR algorithm. Therefore, the throughput of the PA algorithm can be adjusted according to the throughput of IR. In addition, the corrected key is stored in the $Key\_Manager$ module in Fig. 4. We can easily read the keys from this module if the parallel parameter of PA are consistent with IR; otherwise, the storage format of the key must be transformed.

C. Throughput

In a continuously running CV-QKD system, the throughput of the entire post-processing procedure depends on the slowest sub-procedure when a non-multiplexed structure is used. In our case, it is the LDPC decoder. Its throughput can be estimated by

$$T_{LDPC} \approx \frac{f \cdot q}{(1-R) \cdot N_{node} \cdot N_{iter}}, \tag{3}$$

where f is the clock frequency of FPGAs, q is the quasi-cyclic parameter, R is the code rate of the LDPC codes, N_{node} is the average number of nodes in each row of a basic matrix, and N_{iter} is the average number of iterations to execute a decoding algorithm. As shown in (3), the throughput increases linearly with increasing FPGA's clock frequency and quasi-cyclic parameter q. Therefore, these two parameters should be as large as possible.

To derive (3), we assume that all the clocks are occupied by the algorithm. Note that when running the post-processing procedure on an FPGA, some additional clocks are needed for the purposes of control, delay, etc. However, the portion of these additional clocks is negligible, and the actual throughput is very close to the estimated result obtained by (3), which is useful in estimating the throughput in advance when designing a scheme.

The throughput of the multiplexed structure is equal to half the average throughput of the two decoders.

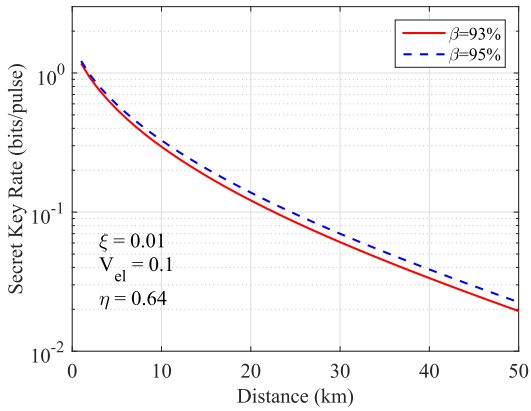

Fig. 6. Secret key rate with reconciliation efficiencies of 93% and 95%. Other parameters are excess noise $\xi = 0.01$, electronic noise $V_{el} = 0.1$, and efficiency of receiver's detector $\eta = 0.64$.

IV. Results

In this section, we summarize the implementation results of the proposed scheme. Our main goal is to achieve high throughput post-processing procedure using the FPGA's limited hardware resources. According to the above theoretical analysis, the larger the parallel parameter, the greater the throughput, but it also requires more hardware resources. To determine the parallel parameter P_{LDPC} of LDPC decoders and P_{PA} of PA, the effects of the speed and area on the design must be investigated. In Table I, we present the design space exploration of LDPC decoders and PA under different parameters. Note that the simulation results are consistent with the theoretical analysis. The throughputs of LDPC decoders and PA increase linearly with the increase of their parallel parameters. The larger the throughput, the more LUTs must be consumed, but the storage resources will not increase. From [30], we know that the throughput of PA decreases linearly with the increasing block size when the other parameters remains constant. Based on these results, we can choose the optimal parameters to avoid wasting hardware resources in a practical CV-QKD system.

Table II presents the detailed parameters of the performance of IR and post-processing based on a CPU or FPGA. We present the performance for two typical SNR values, 1 and 3. The correlated raw keys X and Y are generated using C language and downloaded to FPGA. It is known that LDPC codes will achieve better performance if a longer block length is used. In [15], we achieved information reconciliation with 95% efficiency by using LDPC codes with a block length of 10^6. However, it is a challenge to implement the LDPC decoder with a very large code length based on FPGA because the storage resources of an FPGA are limited. After careful evaluation, we chose two relatively short block lengths. For SNR $\sim$ 1 and 3, the block lengths L of LDPC codes for each level are 349,952 and 262,144, respectively. For the calculation of the optimal code rates, please refer to [15]. The practical rates presented in Table II are determined after actual decoding tests. The length of syndromes S can be calculated using the formula $S = (1-R)L$. In this case, the reconciliation efficiency can reach 93% and the algorithm can be implemented in state-of-the-art FPGAs. In Fig. 6, we compare

TABLE I
DESIGN SPACE EXPLORATION

Scheme	LDPC decoders [a]				PA [b]			
Block Length	262,144				349,952			
Parallel Parameter	1	32	64	128	32	64	128	256
LUTs	1,313	22,870	57,894	113,880	802	2,032	7,031	26,571
Flip-flops	1255	17594	47227	68500	343	489	797	489
Block RAM (bit)	35.41 M	35.41 M	35.41 M	35.41 M	3,520	3,520	3,584	3,584
Throughput (Mb/s)	1.70	49.53	100.90	198.13	2.92	11.69	46.76	186.98

[a] Code rate is 0.98.
[b] Compression ratio (equal to the secret key rate of CV-QKD) is 0.1.

TABLE II
PERFORMANCE OF IR AND POST-PROCESSING BASED ON CPU AND FPGA

	Information Reconciliation					Post-Processing
Hardware	CPU		FPGA			FPGA
SNR	1.0	3.0	1.0	3.0	3.0	3.0
Block Length	349,952	262,144	349,952	262,144	262,144	262,144
Code Rates	0.115/0.836 [a]	0.43/0.98	0.115/0.836	0.43/0.98	0.43/0.98	0.43/0.98
Length of Syndromes	309,707/57,392	149,422/5,242	309,707/57,392	149,422/5,242	149,422/5,242	149,422/5,242
Initial Bit Error Ratios	0.3108/0.0412	0.1534/0.0041	0.3108/0.0412	0.1534/0.0041	0.1534/0.0041	0.1534/0.0041
Reconciliation Efficiency	93.02%	93.06%	93.02%	93.06%	93.06%	93.06%
Scheme	—	—	Multiplexing	Multiplexing	Non-multiplexing	Multiplexing
Parallel Parameters	64/64	64/64	64/64	64/64	128/64	64/64/128 [b]
Average Iterations	82/48	29/10	93/65	38/11	30/10	38/11
LUTs	—	—	83058	42273	146542	42360
Flip-flops	—	—	47226	34883	95066	34896
Block RAM (Mb)	—	—	51.7	35.41	62.82	62.86
Implementation Status	Measured	Measured	Simulated	Measured	Simulated	Measured
Total Latency	36.9 s	10.5 s	39.1 ms	14.1 ms	11.8 ms	21.5 ms
FER	5%	3%	14%	9%	11%	9%
Throughput (symbols/s)	26.9 K	32.7 K	14.83 M	38.18 M	100.90 M	38.18 M

[a] Values on left-hand (right-hand) side of the symbol "/" belong to levels 4 and 5 in slice reconciliation, respectively.
[b] First two values belong to levels 4 and 5 in slice reconciliation, and the third value belongs to PA.

the impact of reduced reconciliation efficiency on key rate and transmission distance. It can be found that the influence of the reconciliation efficiency on the key rate is trivial when the excess noise $\xi = 0.01$, electronic noise $V_{el} = 0.01$, and efficiency of receiver's detector $\eta = 0.64$.

From Table II, we note that more average iterations are required on FPGA, and the FER is also increased to some extent. This is because FPGAs use fixed-point arithmetic to suppress the hardware resource requirements, with the cost being that it also sacrifices the accuracy of relevant calculations. If an FPGA with larger storage resources is employed, a longer block length LDPC decoder and more accurate fixed-point numbers can be used, and the IR performance can reach that of a CPU in principle. Although the FER and reconciliation efficiency degrades to some extent, the throughput is improved by three orders of magnitude. According to (2), the practical secret key rate will be greatly boosted.

The latency of IR, including the latency of the two LLR initializations and two LDPC decoders, is independent of whether multilevel multiplexing is used. The total latency per post-processing procedure is the sum of the latencies of IR and PA. The FER is affected by several factors, including the code lengths, the code rates, and the accuracy of fixed-point numbers. It is crucial to optimize all the relevant parameters to achieve the best overall performance.

As mentioned earlier, the throughput of the entire post-processing procedure or IR depends on the LDPC decoders, so the value of non-multiplexing structure shown in Table II is equal to the throughput of the LDPC decoders, while the value of multiplexing structure is equal to half the average throughput of the two decoders. We use the Xilinx VC709 evaluation board, which is populated with the Virtex-7 XC7VX690T FPGA with 433,200 LUTs and 866,400 flip-flops, as well as 52,920 kb BRAMs, to implement the multiplexing structure and entire post-processing

procedure at SNR = 3, and the throughput reaches 38.18 M symbols/s. In this work, we choose the PA block size to be consistent with that of the IR to facilitate testing. In order to mitigate finite-size effects in an actual CV-QKD system, more FPGA resources can be employed to implement the PA with very large block size. We also give the simulation results of the multiplexing structure with SNR = 1 and the non-multiplexing structure with SNR = 3, which can be implemented on other FPGAs, such as the Virtex-7 XC7VX1140T with a total of 712,000 LUTs, 1,424,000 flip-flops, and 67,680 kb BRAMs [37].

V. Discussion

In this work, we designed and demonstrated high-efficiency post-processing procedure in CV-QKD based on a Xilinx Virtex-7 FPGA. The hardware-accelerated post-processing algorithm reaches throughputs of 100.90 M symbols/s and 14.83 M symbols/s, which are the highest speeds at SNR = 3 and 1, respectively, to the best of our knowledge. If better FPGAs are utilized, such as the Xilinx Virtex UltraScale+ FPGA, the performance of the current post-processing procedure can be improved further. With the rapid advance of FPGA technology, available block memory in chips will become larger, and, in this case, better performance of the post-processing procedure can be expected.

At present, chip-based CV-QKD techniques are undergoing rapid development [11], placing high demands on the integration and power consumption of the post-processing module. Since FPGAs are easy to integrate, and its power consumption is low, it is a very promising secret key distillation engine for chip-based CV-QKD systems.

References

[1] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 145–195, Jan. 2002.

[2] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, no. 3, pp. 1301–1350, Sep. 2009.

[3] C. Weedbrook *et al.*, "Gaussian quantum information," *Rev. Mod. Phys.*, vol. 84, no. 2, pp. 621–669, May 2012.

[4] E. Diamanti and A. Leverrier, "Distributing secret keys with quantum continuous variables: Principle, security and implementations," *Entropy*, vol. 17, no. 9, pp. 6072–6092, Aug. 2015.

[5] Y.-M. Li, X.-Y. Wang, Z.-L. Bai, W.-Y. Liu, S.-S. Yang, and K.-C. Peng, "Continuous variable quantum key distribution," *Chin. Phys. B*, vol. 26, no. 4, Apr. 2017, Art. no. 040303.

[6] S. Pirandola *et al.*, "Advances in quantum cryptography,", 2019, *arXiv:1906.01645*.

[7] D. Bacco *et al.*, "Two-dimensional distributed-phase-reference protocol for quantum key distribution," *Sci. Rep.*, vol. 6, no. 1, Dec. 2016, Art. no. 36756.

[8] X. Wang, W. Liu, P. Wang, and Y. Li, "Experimental study on all-fiber-based unidimensional continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 95, no. 6, Jun. 2017, Art. no. 062330.

[9] W. Liu, X. Wang, N. Wang, S. Du, and Y. Li, "Imperfect state preparation in continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 96, no. 4, Oct. 2017, Art. no. 042312.

[10] N. Wang, S. Du, W. Liu, X. Wang, Y. Li, and K. Peng, "Long-distance continuous-variable quantum key distribution with entangled states," *Phys. Rev. Appl.*, vol. 10, no. 6, Dec. 2018, Art. no. 064028.

[11] G. Zhang *et al.*, "An integrated silicon photonic chip platform for continuous-variable quantum key distribution," *Nature Photon.*, vol. 13, no. 12, pp. 839–842, Dec. 2019.

[12] G. Van Assche, J. Cardinal, and N. J. Cerf, "Reconciliation of a quantum-distributed gaussian key," *IEEE Trans. Inf. Theory*, vol. 50, no. 2, pp. 394–400, Feb. 2004.

[13] P. Jouguet, D. Elkouss, and S. Kunz-Jacques, "High-bit-rate continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 90, no. 4, Oct. 2014, Art. no. 042329.

[14] Z. Bai, X. Wang, S. Yang, and Y. Li, "High-efficiency gaussian key reconciliation in continuous variable quantum key distribution," *Sci. China Phys. Mech. Astron.*, vol. 59, no. 1, Jan. 2016, Art. no. 614201.

[15] Z. Bai, S. Yang, and Y. Li, "High-efficiency reconciliation for continuous variable quantum key distribution," *Jpn. J. Appl. Phys.*, vol. 56, no. 4, Apr. 2017, Art. no. 044401.

[16] A. Leverrier, R. Alleaume, J. Boutros, G. Zemor, and P. Grangier, "Multidimensional reconciliation for a continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 77, no. 4, Apr. 2008, Art. no. 042325.

[17] P. Jouguet, S. Kunz-Jacques, and A. Leverrier, "Long-distance continuous-variable quantum key distribution with a Gaussian modulation," *Phys. Rev. A*, vol. 84, no. 6, Dec. 2011, Art. no. 062317.

[18] Q. Li, X. Wen, H. Mao, and X. Wen, "An improved multidimensional reconciliation algorithm for continuous-variable quantum key distribution," *Quantum Inf. Process.*, vol. 18, no. 1, Jan. 2019, Art. no. 25.

[19] X.-Q. Jiang, P. Huang, D. Huang, D.-K. Lin, and G.-H. Zeng, "Secret information reconciliation based on punctured low-density parity-check codes for continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 95, no. 2, Feb. 2017, Art. no. 022318.

[20] H. Mani, T. Gehring, C. Pacher, and U. L. Andersen, "Reconciliation of weakly correlated information sources utilizing generalized exit chart,", 2020, *arXiv:1812.05867*

[21] X. Wang, Y. Zhang, Z. Li, B. Xu, S. Yu, and H. Guo, "Efficient rate-adaptive reconciliation for continuous-variable quantum key distribution," *Quantum Inf. Comput.*, vol. 17, pp. 1123–1134, Nov. 2017.

[22] M. Milicevic, C. Feng, L. M. Zhang, and P. G. Gulak, "Quasi-cyclic multi-edge LDPC codes for long-distance quantum cryptography," *NPJ Quantum Inf.*, vol. 4, Apr. 2018, Art. no. 21.

[23] S. J. Johnson, A. M. Lance, L. Ong, M. Shirvanimoghaddam, T. C. Ralph, and T. Symul, "On the problem of non-zero word error rates for fixed-rate error correction codes in continuous variable quantum key distribution," *New J. Phys.*, vol. 19, no. 2, Feb. 2017, Art. no. 023003.

[24] C. Zhou, X. Wang, Y. Zhang, Z. Zhang, S. Yu, and H. Guo, "Continuous-variable quantum key distribution with rateless reconciliation protocol," *Phys. Rev. Appl.*, vol. 12, no. 5, Nov. 2019, Art. no. 054013.

[25] X. Wang, Y. Zhang, S. Yu, and H. Guo, "High speed error correction for continuous-variable quantum key distribution with multi-edge type LDPC code," *Sci. Rep.*, vol. 8, no. 1, Jul. 2018, Art. no. 10543.

[26] H.-F. Zhang *et al.*, "A real-time QKD system based on FPGA," *J. Lightw. Technol.*, vol. 30, no. 20, pp. 3226–3234, Oct. 2012.

[27] A. Tanaka *et al.*, "High-speed quantum key distribution system for 1-Mbps real-time key generation," *IEEE J. Quantum Electron.*, vol. 48, no. 4, pp. 542–550, Apr. 2012.

[28] N. Walenta *et al.*, "A fast and versatile quantum key distribution system with hardware key distillation and wavelength multiplexing," *New J. Phys.*, vol. 16, no. 1, Jan. 2014, Art. no. 013047.

[29] Z. Yuan *et al.*, "10-Mb/s quantum key distribution," *J. Lightw. Technol.*, vol. 36, no. 16, pp. 3427–3433, Aug. 2018.

[30] S.-S. Yang, Z.-L. Bai, X.-Y. Wang, and Y.-M. Li, "FPGA-based implementation of size-adaptive privacy amplification in quantum key distribution," *IEEE Photon. J.*, vol. 9, no. 6, Dec. 2017, Art. no. 7600308.

[31] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge Univ. Press, 2008.

[32] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[33] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A survey of FPGA-based LDPC decoders," *IEEE Commun. Surveys Tut.*, vol. 18, no. 2, pp. 1098–1122, May 2016.

[34] P. Radosavljevic, A. de Baynast, and J. R. Cavallaro, "Optimized message passing schedules for LDPC decoding," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Beijing, China, 2005, pp. 591–595.

[35] A. Leverrier, F. Grosshans, and P. Grangier, "Finite-size analysis of a continuous-variable quantum key distribution," *Phys. Rev. A*, vol. 81, no. 6, Jun. 2010, Art. no. 062343.

[36] P. Jouguet and S. Kunz-Jacques, "High performance error correction for quantum key distribution using polar codes," *Quantum Inf. Comput.*, vol. 14, pp. 329–338, Mar. 2014.

[37] "7 series FPGAs overview," San Jose, CA, USA. 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf