FPGA Prototyping of CCDM with On-line Configurable Probabilistic Distribution Based on Parallel Arithmetic Coding

Jingwei Song[1], Yan Li[1*], Xiaoshuo Jia[1], Zulin Liu[1], Kejia Xu[1], Jifang Qiu[1], Hongxiang Guo[1], Xiaobin Hong[1], Zhisheng Yang[1], and Jian Wu[1]

[1] *State Key Laboratory of Information Photonics and Optical Communications, Beijing Univ. of Posts & Telecom, Beijing, China*

[*] *liyan1980@bupt.edu.cn*

Abstract: The real-time CCDM and inverse CCDM were realized in an FPGA. The DM and inverse DM achieved a throughput of 16.8GBaud and supported on-line reconfiguration to realize different entropies with fine granularity. © 2024 The Author(s)

1. Introduction

In high capacity WDM systems, high order modulation formats have been widely used to approach the channel capacity. In the presence of fiber transmission impairments, the signal-to-noise ratio (SNR) is limited. In the additive white Gaussian noise channel, with a given SNR and the order of quadrature amplitude modulation (QAM), the maximum mutual information can be achieved if the QAM symbols are subject to the Maxwell-Boltzmann (MB) distribution [1]. To generate non-uniformly distributed symbols, PS techniques are necessary. Some representative PS schemes are, for example, constant composition distribution matcher (CCDM), many-to-one (MTO), hierarchical distribution matcher (HiDM) and enumerative sphere shaping (ESS) [2].

To deploy the PS techniques in practical systems, the FPGA implementation is of great importance as the practical digital communication systems are based on real-time DSP chips like ASICs or FPGAs. To be implemented in the FPGA, the PS scheme needs to have low complexity and can be re-organized in a parallel structure. Up to now, many works focusing on real-time implementation of different PS techniques have been reported as is shown in Table 1. Compared to other intensively investigated PS schemes listed in Table 1, the CCDM is the only one that can provide fine granularity of the output entropy, making CCDM adaptive to different scenarios, e.g., links with different distances, number of WDM channels or amplifiers with different noise figures, etc. However, in the mainstream view, the real-time CCDM is hard to be realized with the arithmetic coding (AC) [3], which is inherently a sequential procedure and is usually regarded incompatible with the parallelization structure of real DSP chips [4]. As far as we know, although CCDM has been widely used in high-capacity offline transmission experiments since it was proposed, there has not been any report on approaches to realize a real-time parallel CCDM.

In this paper, an effective method to realize the real-time CCDM and inverse CCDM in an FPGA chip is proposed. Based on the modified AC scheme, the CCDM can be implemented in a parallel structure with a high symbol rate. The FPGA verification proved that, with a codeword length of 48, the real-time CCDM and inverse CCDM for PS-16QAM can achieve a symbol rate of 16.8GBd. Besides, it can support on-line configuration of the probabilistic distribution and can thus realize an entropy from 2.5bit/symbol to 4.0bit/symbol with an information divergence (compared to the ideal MB distribution) lower than 0.04.

2. Principle of the parallel CCDM scheme

2.1. The Parallel Arithmetic Coding

The parallel arithmetic coding we used to make the CCDM suitable for FPGA implementation is a modification of the traditional AC introduced in [3]. In the AC-based CCDM, a bit sequence is mapped to a symbol sequence that is a permutation of a collection of symbols C, whose size equals to the codeword length. The symbols occurring in C belongs to an alphabet α, whose size equals to the number of the types of symbols [5]. The occurrence number of each type of symbols in C is used to achieve the desired distribution and can be on-line configured in the real-time CCDM. In the following example, the codeword length is 8, the alphabet α is {“A”, “B”}, the occurrence numbers of “A” and “B” are set to 6 and 2, respectively, and C is therefore {“A”, “A”, “A”, “A”, “A”, “A”, “B”, “B”}, leading to a distribution of the output sequence of {3/4, 1/4}.

Fig. 1 shows the detailed process of mapping 1011 to a sequence belonging to one of the permutations of C described above. The CCDM is consisted of 8 what we call “*Division and Selection*” modules (D&S). Each D&S module divides the input intervals into several parts and select one of them to send to the next layer. The module imports three signals from the previous. The first signal is the size of the current range, i.e., the number of remaining

intervals $N_{interval}$. The second one is N_{rem}, a vector that records the number of remaining symbols, whose size equals to the size of the alphabet α. The third one is I_{target}, which indicates the index of the target unit interval in the remaining intervals. Each module also outputs a symbol called S_{out} based on the result of selection. At last, the desired sequence is composed of the outputs of 8 D&S modules. In this example, the $N_{interval}$, N_{rem} and I_{target} inputted into the first layer is 28, {6,2} and 11 respectively. The value of $N_{interval}$ equals to the number of available permutations, which in this example is $C_8^6 = C_8^2 = 28$ [5]. The ratio of the sizes of the parts equals to the ratio of the occurrence numbers of remaining symbols, which is 6:2 in the first module since there are 6 unused "A"s and 2 unused "B"s. Therefore, interval 0 to 20 belong to the "part A" and the rest intervals belong to the "part B". Besides, the input bit sequence 1011, if treated as a binary number, can be translated to the decimal $(11)_{10}$, indicating that the target interval is the 12th of the 28 intervals. As is shown in Fig.1, it's clear that the target interval is inside the "part A". Thus, the first output symbol is "A" and the intervals belonging to "part A" are sent to the next D&S module. The N_{rem} is also updated to {5,2} as one "A" has been used. When delivered from one layer to another, e.g., from the 3rd D&S to the 4th D&S in Fig. 1, the remaining intervals need to be re-numbered from 0 to $N_{interval}$-1, and the I_{target} is also updated to indicate the target number in the new intervals. After the 8th D&S finishes its work, there will be 8 output symbols consisting the output sequence of the CCDM. The inverse CCDM follows the similar principle and is also split to 8 layers of D&S modules. The only difference is that the "selection" operation of inverse CCDM is determined by the received symbol.

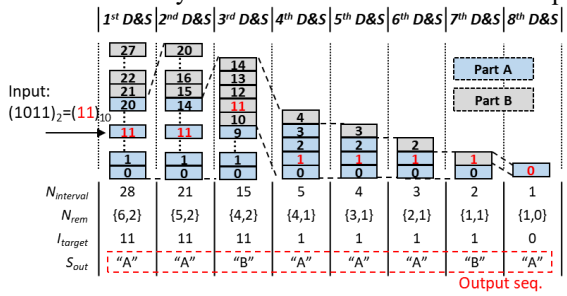

Fig. 1. The arithmetic coding scheme of (a) CCDM and (b) inverse CCDM

Table 1. Reported PS schemes

Scheme	Granularity of entropy	Latency	Rate loss	Real-time impl.
HiDM	coarse	low	low	[6,7]
MTO	coarse	low	high	[8]
ESS	coarse	medium	low	[9]
BWDM	coarse	low	not mentioned	[4]
CCDM (short)	fine	medium	medium	**Not reported**
CCDM (long)	fine	high	low	

2.2. The FPGA Implementation of CCDM

Based on the AC scheme, as Fig. 2(a) shows, the real-time CCDM and inverse CCDM with a codeword length of 48 are realized in a fully pipelined and parallel structure in a 16-nm FPGA (Xilinx XCVU13P). We assumed that the probabilistic amplitude shaping (PAS) scheme was used and the modulation format is 16QAM [5]. Therefore, we only illustrate the modules related to the amplitude bits in Fig. 2(a). In the left part of Fig. 2(a), the real-time CCDM is consisted of 48 D&S modules. The D&S modules execute the operations described in Fig. 1 and are fully-pipelined so that in every clock cycle, each D&S module can output one symbol. However, the symbols generated by these D&S modules in the same clock cycle don't belong to the same codeword. Therefore, extra delay modules at the output ports are needed to compensate the latency. The real-time inverse CCDM is also realized in the same way except the delay modules are moved to the input side. The Virtual IO allows us to set the values of some signals inside the FPGA on line. By controlling them, we can change the occurrence number of symbols. For example, if we want to generate a sequence consisted of 20 "A"s, 12 "B"s, 12 "C"s and 4 "D"s, we only need to set the $N_{interval}$ to $C_{48}^{20} \times C_{28}^{12} \times C_{16}^{12} = 92661430676953015213600$ and set the N_{rem} to {20,12,12,4} and the probabilistic distribution of the outputted symbols will be {0.4167,0.25,0.25,0.0833}.

Table. 2 Details of the implementations

Codeword Length	L=24 @ 350MHz			L=40 @ 350MHz			L=48 @ 350MHz		
Resources Utilization	CCDM	iCCDM	Total	CCDM	iCCDM	Total	CCDM	iCCDM	Total
LUT	44660	24087	70992	131781	83162	217121	198197	129937	330310
FF	45804	21658	71679	138735	66917	209869	212467	100729	317413
LUT per 1GBd	5316.667	2867.5	8451.429	9412.929	5940.143	15508.64	11797.4	7734.34	19661.3
FF per 1GBd	5452.857	2578.333	8533.214	9909.643	4779.786	14990.64	12646.8	5995.77	18893.6
Power (W)	0.704	0.368	1.166	2.192	1.4	3.708	3.636	2.727	6.48
Power per 1GBd (W)	0.08381	0.04381	0.13881	0.15657	0.1	0.26486	0.21643	0.16232	0.38571
Symbol rate (GBd)	8.4			14			16.8		

We also evaluated the complexity and power consumption of three CCDMs with different codeword lengths. The details of these implementations are listed in Table. 2. The complexity can be represented by the number of look-up-

tables (LUT) and flip-flops (FF). In our design, to make the comparison fair, the block random access memory (BRAM) is not used. These numbers are accurately calculated by the Vivado software after the implementation process. The highest real-time symbol rate achieved in the FPGA is 16.8GBaud. According to the results of FPGA verification, any distribution with a codeword length of 48 can be realized by on-line configuration. The placement of FPGA is given in Fig. 2(b). The picture of the evaluation board is showed in Fig. 2 (c).

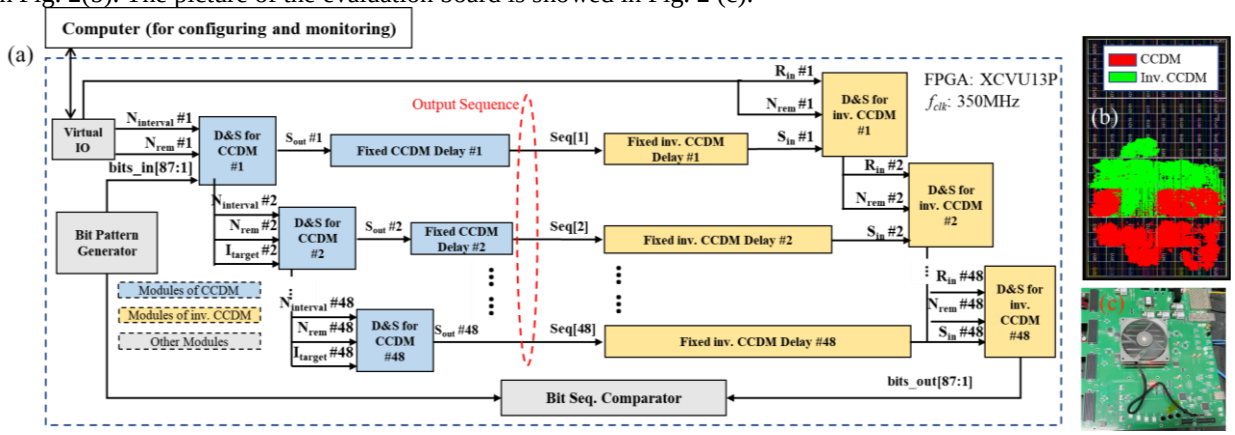

Fig. 2. (a) The structure of real-time parallel CCDM. (b) The FPGA placement (L=48). (c) The picture of the evaluation board.

3. Performance analysis

We evaluated some indexes that are important for the hardware implementation or PS and showed the results in Fig. 3. The distribution of CCDM's output is optimized to minimize the information divergence between the realized distribution with the MB distribution under different entropies. Fig. 3(a) shows the latency of the DM and inverse DM with a clock frequency of 350MHz, where a linearity relationship between the total latency and the square of the codeword length can be observed. Fig. 3(b) shows the complexity and symbol rate under different codeword length with a clock frequency of 250MHz. Fig. 3(c) demonstrate the rate loss and the information divergence with different target entropies and codeword lengths. The insets of Fig. 3(c) exhibit the distribution of the symbols outputted by the CCDM with a codeword length of 48. According to the results, the entropy is adjustable with fine granularity.

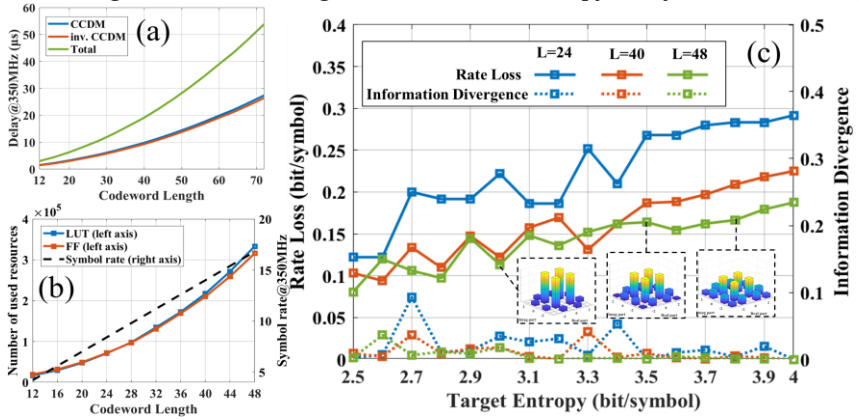

Fig. 3. Some important indexes: (a) The latency; (b) The complexity and symbol rate; (c) The rate loss and information divergence.

4. Conclusion

We have proposed a real-time CCDM and inverse CCDM scheme and realized it on an FPGA chip. The scheme is fully parallel and pipelined and allows on-line reconfiguration of the distribution. In the FPGA verification, the real-time symbol rate achieved 16.8GBaud.

Acknowledgment

This work is supported by National Key Research and Development Program of China (2022YFB2903201).

Reference

[1] J. Cho et al., J. Light. Technol, **37**(6), 1590-1607 (2019).
[2] Gültekin Y.C et al., Entropy, **22**(5),581 (2020).
[3] P. Schulte et al., IEEE Trans. Inf. Theory, **62**, 230-434 (2016).
[4] L. Zhang et al., J. Light. Technol, early access (2023).
[5] G. Böcherer, et al., J. Light. Technol, **37**(2), 230-244 (2019).
[6] L. Zhang et al., J. Light. Technol **40(**5), 1339-1345 (2022).
[7] T. Yoshida et al., OFC 2021, Tu6D.4.
[8] J. Fang et al., ICAIT 2023.
[9] L. Zhang et al., CLEO 2022, STh1I.2.