FPGA Implementation of Prefix-Free Code Distribution Matching for Probabilistic Constellation Shaping

Qinyang Yu[1,2], Steve Corteselli[2], and Junho Cho[2*]
1. Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, 200444 Shanghai, China
2. Nokia Bell Labs, 791 Holmdel Road, Holmdel, NJ 07733, USA
**email: junho.cho@nokia-bell-labs.com*

Abstract: We implement rate-adaptable prefix-free code distribution matching in an FPGA, demonstrating its real-time feasibility with substantially less hardware resources than low-density parity-check coding. © 2020 The Authors

1. Introduction

Distribution matching (DM) and forward error correction (FEC) are two essential components of a probabilistically shaped (PS) coded modulation system [1] with the probabilistic amplitude shaping (PAS) architecture [2]. There exist DM and FEC implementations that jointly approach the capacity of the additive white Gaussian noise channel to within a practically inconsiderable gap, such as the constant composition DM (CCDM) [3] and the irregular low-density parity check (LDPC) code [4]. However, the feasibility of real-time hardware for DM and FEC should be shown under stringent real-world constraints such as the latency, throughput, and the resource area, in order to declare a *pragmatically* achieved information rate (IR) in high-speed optical fiber communications. As for the FEC, this is widely agreed among transmission experiments, hence only pragmatic codes are used to claim an *achieved* IR, that are proven through extensive hardware implementations to be real-time feasible (see. e.g., [5]). As for the DM, on the other hand, virtually *no* existing experimental results justify the real-time feasibility of the used DM; indeed, to the best of our knowledge, there is only one very recent paper on hardware-based DM implementation [6] published to date.

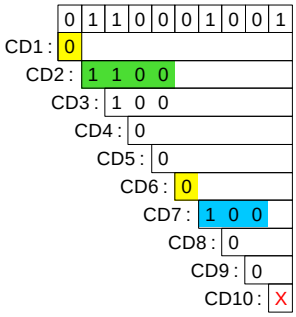
In this paper, we show that the rate-adaptable (RA) prefix-free code DM (PCDM) [7, 8] is a real-time feasible DM for high-speed optical transmission, by implementing it in a field-programmable gate array (FPGA). We demonstrate that PCDM can be processed in a *massively parallel* manner in each block, thereby minimizing the latency and maximizing the throughput, whereas most other known DMs such as CCDM [3], shell mapping [9], and enumerative sphere shaping [10] have limited parallelism [11, Table 3]. We also show that PCDM consumes substantially smaller area than LDPC to achieve a target throughput on a selected FPGA platform.

2. Rate-Adaptable PCDM

PCDM is a look-up table (LUT)-based DM that uses variable-length prefix-free codes with a framing method, enabling data transmission in accordance with the optical transport network's frame structure. With a reference to the example LUT shown in Fig. 1(a), a PCDM encoder observes an incoming bit stream on the fly until it finds the first matching left entry from the LUT, and produces the corresponding symbols in the right entry; e.g., a bit stream "01100…" is mapped by the codebook of Fig. 1(a) to the symbol stream "111111, 1113, …" The encoder repeats this process until all the input bits in a length-B block are encoded. The problem of variable-length output is solved by a *framing* method that switches the codebook to that of a uniform quadrature amplitude modulation (QAM) at a proper time during encoding, see [7, 8] for details. The encoder then always produces a length-S output *symbol* block from the length-B input *bit* block, accomplishing a fixed DM rate (also called the *shaping factor*) of $\beta = B/S$ bit/symbol in each block. We implement 15 PCDM codebooks for 15 different shaping factors, as shown in Table 1. Here, we fix the output length S and change the input length B for an optical transponder to flexibly change the IR at

Input Bits	Output Symbols
0	111111
100	113
1010	111113
1011	11113
1100	1113
1101	1311
1110	3111
111100	133
111101	3113
1111100	1313
1111101	3131
1111110	3311
11111110	3133
111111110	3313
111111111	3331

(a)



(b)

Fig. 1. (a) A PCDM codebook for $\beta = 0.5$, and (b) its parallel codeword detector.

Table 1. Fifteen PCDM codebooks implemented in our FPGA

QAM template	16-QAM							64-QAM							
Input block length (B)	90	120	150	180	210	240	270	360	390	420	450	480	510	540	570
Output block length (S)	300														
Shaping factor ($\beta = B/S$)	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
IR	1.8	2	2.2	2.4	2.6	2.8	3	3.2	3.4	3.6	3.8	4	4.2	4.4	4.6

a fixed symbol rate. When used with fixed rate-0.8 FEC, and with 16- or 64-ary QAM templates in the PAS architecture, the codebooks realize PS QAM with the IRs ranging from 1.6 to 4.8 bit/symbol in 0.2-bit/symbol increments (cf. Table 1), where 1.6 and 4.8 bit/symbol are realized using uniform QAMs. At 125 GBaud, for example, these IRs span net data rates from 400 Gb/s to 1.2 Tb/s in 50-Gb/s increments in two polarizations. Each of the 15 LUTs has ≤ 16 rows that have ≤ 9 bits and ≤ 12 symbols in the left and right entries, respectively. Figure 2 shows the *shaping gap*, i.e., the difference in average symbol energy that the 15 PCDM encoders use compared to an ideal DM for the same target shaping factor, for $S = 300$, 600, and without framing. Note that for the same number of *symbols* S in each output block, the DM in binary domain produces S and $2S$ *bits*, respectively, for 16- and 64-QAMs in the PAS architecture.

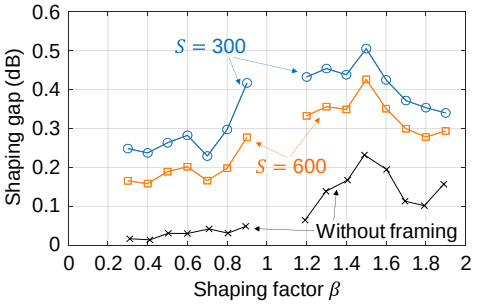


Fig. 2. Shaping gap of the PCDM codebooks.

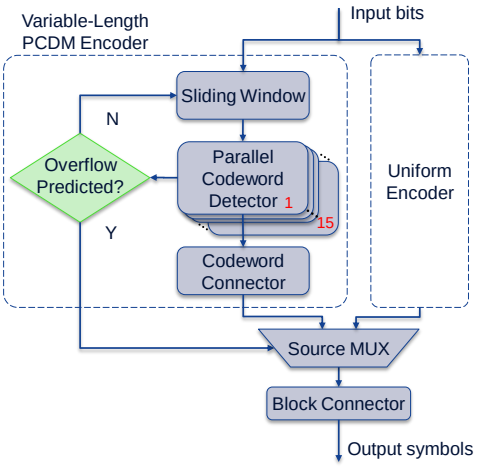


Fig. 3. Universal architecture for RA-PCDM.

3. Universal Hardware Architecture for PCDM

We design a *universal* hardware architecture for fine-grained RA-PCDM, as shown in Fig. 3, which consists of a *Variable-Length PCDM Encoder*, a *Uniform Encoder*, a *Source Multiplexer (SMUX)*, and a *Block Connector (BC)*. In the variable-length PCDM encoder, a length-W *Sliding Window (SW)* fetches W bits from an incoming bit stream. In each window, following the method presented in [12], a set of W parallel *Codeword Detectors (CDs)* simultaneously searches for the potential codewords; e.g., Fig. 1(b) illustrates how the bit stream "0110001001…" is encoded in a length-10 window in parallel, where only the matching codewords from the 1st, 2nd, 6th, and 7th CDs (highlighted in colors) are produced while all the other candidate codewords are discarded. As seen from the example, the last few bits in the window may be left un-encoded due to the non-existence of a matching codeword (this induces increased latency, as discussed below), hence the window begins from the last un-encoded bit in the next step. There are 15 sets of parallel CDs (cf. red numbers in Fig. 3), each set of which is dedicated to one codebook, collectively realizing the 15 shaping factors presented in Table 1. The subsequent *Codeword Connector (CC)* aggregates the valid codewords into a seamless symbol stream. Based on the cumulative input and output lengths of the CDs, an *Overflow Predictor (OP)* detects the position from which the uniform encoder should be used to accomplish fixed-length framing (as per the method described in [7, 8]). The SMUX chooses the output symbols either from the variable-length PCDM encoder or from the uniform encoder, depending on the switching signal from the OP. Finally, the BC produces the probabilistically-shaped symbols, framed in a fixed-length block. Operation of a PCDM *decoder*, and hence its architecture, is almost identical to the encoder.

Importantly, this universal architecture enables massively parallel processing. Specifically, a *parallel factor* of $P = W - U + 1$ is realized, where U is the length of the longest word in the left LUT entry, which accounts for the residual un-encoded bits in each window. In our case, the codebook with the largest shaping factor of $\beta = 1.9$ represents the critical case that determines the latency and throughput of a whole RA-PCDM encoder, which has the overhead $U = 6$. Therefore, when implemented in a fully pipelined manner, our PCDM encoder finishes encoding a block every $T \coloneqq \lceil B/P \rceil$ clock cycles, where $\lceil x \rceil$ denotes the least integer $\geq x$. The latency for processing a block is then $L \coloneqq T/F$ µs with F being the clock frequency in MHz. The net data rate is FB/T Mb/s, and the throughput of the encoder, which is defined by convention as the number of *output bits* per unit time, amounts to $F(2S)/T$ Mb/s.

4. FPGA Implementation Results

We use the Kintex Ultrascale XCKU040 FPGA [13] as a target platform, which has 242,400 configurable logic block (CLB) LUTs, 484,800 CLB registers, and 600 block random access memories (BRAMs) of 36 Kb each. When a PCDM encoder is synthesized for various window sizes, ranging from 16 to 64, a great portion of the utilized resources is the CLB LUTs and the rest is only the CLB registers (which are more abundant in the FPGA), as shown for an example window size of 24 in Table 2; we therefore use the throughput per CLB LUT to evaluate the *area efficiency* of a PCDM encoder. Figure 4 summarizes the synthesis results for various window sizes, in terms of the area, clock frequency, latency, throughput, and the area efficiency. We use 12 and 13 pipeline stages for $W \leq 32$ and $W = 64$, respectively. It can be observed that the latency and throughput can be continuously enhanced by increasing the

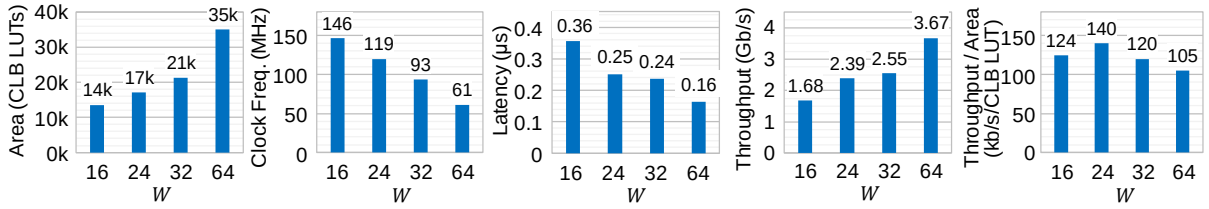


Fig. 4. RA-PCDM synthesized in FPGA with various window sizes.

window size, at the expense of a decreasing area efficiency. This allows the PCDM encoder (and the decoder in the same way) to achieve stringent real-time processing requirements for high-speed optical transmission. For the case of $W = 24$, seven instances of a pair of PCDM encoder and decoder can be synthesized on a target FPGA platform, achieving a throughput of 16.7 Gb/s.

Table 2. Resource utilization for $W = 24$

Resource	SW	CD+CC +OP	SMUX	BC	Total
CLB LUT	4376	5708	5732	1272	17090
CLB Register	66	372	2408	1200	5246
BRAM	0	0	0	0	0

Unfortunately, it is not possible to fairly compare our PCDM implementation with other DM implementations, since it is very difficult from the only existing hardware-based DM implementation [6] to infer the implemented rate adaptability and shaping gap. Therefore, we compare our results with a recent RA-LDPC implementation on the same FPGA platform [14], in which FEC alone performs rate adaptation to realize 8 different IRs with 16- and 64-QAMs. Although the RA-LDPC encoder and decoder use a significant amount of CLB registers and BRAMs, we give the RA-LDPC the advantage of evaluating the area efficiency based solely on the CLB LUT. Depicted in Fig. 5 are the throughputs per CLB LUT for the fixed rate-0.8 LDPC [14], RA-LDPC [14], RA-PCDM (W=24), and RA-PCDM-LDPC (W=24), the last implementation of which combines the fixed rate-0.8 LDPC and RA-PCDM to realize a PS coded modulation system as a whole. Here, the throughput is calculated with respect to the *entire* PS QAM encoder output bits since, in the PAS architecture, FEC needs to yield 1.5× more bits than DM in the critical case of $\beta = 1.9$; also, the numbers of CLB LUTs for the encoder and decoder are scaled such that the encoding and decoding throughputs are the same (these lead to the difference in the area efficiencies of RA-PCDM between Figs. 4 and 5; namely, 140 kb/s/CLB LUT $\div 2$ (en-, decoders) $\times 1.5 \approx 104.9$ kb/s/CLB LUT). It can be seen from Fig. 5 that, compared to the fixed-rate LDPC (1st column), only marginal area should be invested to realize the fine-grained rate adaptability using the RA-PCDM-LDPC (4th column), which comes with the additional benefit of shaping gains. Compared to the RA-LDPC (2nd column), the RA-PCDM-LDPC realizes ~2× finer granularity of the IR using less hardware resources, again with the additional benefit of shaping gains.

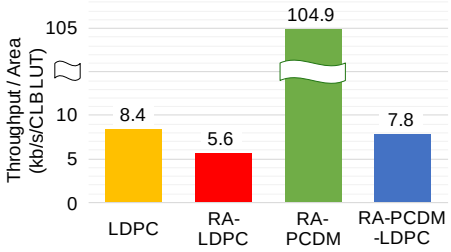


Fig. 5. PS-QAM Throughput/Area of FPGA implementations.

5. Conclusion

We implemented parallel RA-PCDM in FPGA, demonstrating that RA-PCDM can be implemented in real time with substantially less hardware resources than RA-LDPC. The parallel factor can be flexibly increased to fulfill the latency and throughput requirements of high-speed optical transmission, and the block length can also be increased to reduce the shaping gap (cf. Fig. 2) with a slightly increased hardware cost.

References

[1] J. Cho and P. J. Winzer, "Probabilistic constellation shaping for optical fiber communications," *J. Lightw. Technol*. **37**(6), 1590–1607 (2019).
[2] G. Böcherer et al., "Bandwidth efficient and rate-matched low-density parity-check coded modulation," IEEE Trans. Commun. **63**(12), 4651–4665 (2015).
[3] P. Schulte and G. Böcherer, "Constant composition distribution matching," *IEEE Trans. Inf. Theory* **62**(1), pp. 430–434 (2016).
[4] T. J. Richardson et al., "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory* **47**(2), pp. 619–637 (2001).
[5] P. Hailes et al., "A survey of FPGA-based LDPC decoders," *IEEE Commun. Surveys Tutorials* **18**(2), 1098–1122 (2015).
[6] T. Yoshida et al., "FPGA implementation of distribution matching and dematching," in *Proc. ECOC*, Dublin, Ireland, Sep., 2019, Paper M.2.D.2.
[7] J. Cho, "Prefix-free code distribution matching for probabilistic constellation shaping," *IEEE Trans. Commun.*, to be published.
[8] J. Cho and P. J. Winzer, "Multi-rate prefix-free code distribution matching," in *Proc. OFC*, San Diego, CA, USA, Mar. 2019, Paper M4B.7.
[9] P. Schulte and F. Steiner, "Divergence-optimal fixed-to-fixed length distribution matching with shell mapping," *IEEE Wireless Commun. Lett.* **8**(2), pp. 620–623 (2019).
[10] Y. C. Gültekin et al., "Approximate enumerative sphere shaping," in *Proc. ISIT*, Vail, CO, USA, Jun. 2018, pp. 676–680.
[11] Y. C. Gültekin et al., "Probabilistic Shaping for Finite Blocklengths: Distribution Matching and Sphere Shaping," 2019, arXiv:1909.08886[eess.SP].
[12] J. Nikara et al., "Multiple-symbol parallel decoding for variable length codes," *IEEE Trans. VLSI Syst.* **12**(7), 676–685 (2004).
[13] Xilinx, "UltraScale architecture and product data sheet: overview," DS890 (v3.10), Aug. 2019.
[14] X. Sun et al., "Run-time reconfigurable adaptive LDPC coding for optical channels," *Optics express* **26**(22), pp.29319–29329 (2018).