IEICE TRANS. FUNDAMENTALS, VOL.E105–A, NO.11 NOVEMBER 2022

LETTER

Hardware Implementation of Euclidean Projection Module Based on Simplified LSA for ADMM Decoding

Yujin ZHENG[†], **Junwei ZHANG**[†], ***Nonmembers***, **Yan LIN**[††], ***Member***, **Qinglin ZHANG**[†], ***and*** **Qiaoqiao XIA**[†a)], ***Nonmembers***

SUMMARY The Euclidean projection operation is the most complex and time-consuming of the alternating direction method of multipliers (ADMM) decoding algorithms, resulting in a large number of resources when deployed on hardware platforms. We propose a simplified line segment projection algorithm (SLSA) and present the hardware design and the quantization scheme of the SLSA. In simulation results, the proposed SLSA module has a better performance than the original algorithm with the same fixed bitwidths due to the centrosymmetric structure of SLSA. Furthermore, the proposed SLSA module with a simpler structure without hypercube projection can reduce time consuming by up to 72.2% and reduce hardware resource usage by more than 87% compared to other Euclidean projection modules in the experiments.

key words: *simplified line segment projection algorithm (SLSA), alternating direction method of multipliers (ADMM), low-density parity-check (LDPC) codes*

1. Introduction

For low-density parity-check (LDPC) codes, Barman et al. proposed an linear programming (LP) decoding algorithm based on the alternating direction method of multipliers (ADMM) [1], which uses the idea of decomposition coordination to decompose a large problem into small local problems, thus improving the efficiency of the LP decoding algorithm.

The ADMM-LP decoding algorithm involves the Euclidean projection (EP) operation, which has a decisive impact on the hardware implementation of the ADMM-LP decoder in terms of throughput and hardware resource usage [2]. Therefore, researchers have proposed simplified projection methods to solve this problem. X. Zhang et al. proposed a cut search algorithm (CSA) to simplify the EP [3]. The CSA is less complex than the original projection algorithm described in [1], because the sort and desort operations are replaced by the cut search operation. However, the CSA still involves sort operation when solving the piecewise linear optimization problems, leads to a complexity of $O(d \log d)$, where d denotes the degree of check nodes. G. Zhang et al. proposed that the projection onto the parity polytope can be transformed to a projection onto a simplex [4]. The simplex projection method involves partial sort operations, and it has a linear-complexity of $O(d)$. In the worst case, the complexity of the simplex projection method reaches $O(d^2)$. Wasson et al. proposed the hardware implementations of projection onto the parity polytope and probability simplex [5]. Then, Wasson et al. implemented the ADMM decoder on a Field Programmable Logic Gate Array (FPGA) platform [6], [7]. However, the above projections are exact projections with high complexity, which reduces the ADMM-LP decoder throughput and increase the hardware resource consumption. In 2017, Jiao et al. proposed a lookup table (LUT) based projection method that further simplifies the projection algorithm [8]. Samhan developed a modified version of the Wasson's ADMM-LP decoder based on the LUT projection method, realizing a bitwidth reduction of about 50% on FPGA [9]. Thameur et al. presented the hardware implementations of three projection methods, namely exact Euclidean projection (EEP) [5], the iterative Euclidean projection (IEP) [10], and the LUT-based Euclidean projection (LEP) [11]. Recently, Xia et al. proposed a line segment projection algorithm (LSA) without sorting and iterative operations [12]. The LSA can save the average projection time by 43% compared with the CSA.

In this letter, we propose a simplified line segment algorithm (SLSA) with a simpler structure and fewer computational variables in Sect. 2. Then, we present the hardware design of the SLSA and quantization scheme in Sect. 3. Furthermore, we evaluate the impact of different quantization bitwidths on frame error rate (FER) performance, latency and hardware resource usage of the proposed SLSA module in Sect. 4. Finally, the proposed SLSA module is compared with other EP modules to further measure hardware performance.

2. Proposed Simplified Line Segment Algorithm

2.1 Line Segment Algorithm

In the EP, we use $\mathbb{PP}_d$ to represent the parity check polytope, as defined in Eq. (1). $\mathbb{PP}_d$ is the convex hull of all the length-d binary vectors with any even number of 1s, where d is the check node degree.

$$\mathbb{PP}_d = \mathrm{conv}\left(\left\{\boldsymbol{x} \in \{0,1\}^d \mid \|\boldsymbol{x}\|_1 \ \ is\ even\right\}\right) \tag{1}$$

Then, we use $\prod_{\mathbb{PP}_d}(\mathbf{v})$ to represent projecting a vector

Manuscript received December 28, 2021.
Manuscript revised April 8, 2022.
Manuscript publicized May 20, 2022.

[†]The authors are with College of Physical Science and Technology, Central China Normal University, Wuhan, 430079, China.
[††]The authors is with Wuhan Digital Engineering Institute, Wuhan, 430079, China.
a) E-mail: xiaqq@mail.ccnu.edu.cn (Corresponding author)

DOI: 10.1587/transfun.2021EAL2114

Algorithm 1 Line Segment Projection Algorithm

Input: Vector $\mathbf{v} \in \mathbb{R}^d$
Output: Projection Vector $\mathbf{z}$
1: Projection $\mathbf{v}$ onto unit hypercube: $\mathbf{u} = \prod_{[0,1]} \mathbf{v}$
2: Initialize $\boldsymbol{\theta}_\mathbf{v}$: $\theta_{v,i} = \begin{cases} 1, & v_i > 0.5 \\ -1, & \text{else} \end{cases}$
3: **if** set $\{i|\theta_{v,i} = 1\}$ has an even number of elements **then**
4: $\quad i^* = \underset{i}{arg\min}\,(|v_i - 0.5|)$
5: $\quad \theta_{v,i^*} = -\theta_{v,i^*}$
6: **end if**
7: $\boldsymbol{V} = \{i : \theta_{v,i} = 1\}$
8: **if** $\boldsymbol{\theta}_\mathbf{v}^T \mathbf{u} \leqslant |\boldsymbol{V}| - 1$ **then**
9: $\quad \mathbf{z} = \mathbf{u}$
10: **else**
11: $\quad$ Odd-vertex $\boldsymbol{O} : O_i = \begin{cases} 0, & \theta_{v,i} = -1 \\ 1, & \theta_{v,i} = 1 \end{cases}$
12: $\quad$ Index $a = \underset{i}{arg\min}\,(|v_i - 0.5|)$
13: $\quad$ Index $b = \underset{i \neq a}{arg\min}\,(|v_i - 0.5|)$
14: $\quad$ Even-vertex $\mathbf{A} = \begin{cases} O_i, & \text{if } i \neq a \\ 1 - O_i, & \text{if } i = a \end{cases}$
15: $\quad$ Even-vertex $\mathbf{B} = \begin{cases} O_i, & \text{if } i \neq b \\ 1 - O_i, & \text{if } i = b \end{cases}$
16: $\quad \mathbf{z} = \prod_{L_{\mathbf{AB}}} (\mathbf{v})$
17: **end if**
18: **return** $\mathbf{z}$

$\mathbf{v} \in \mathbb{R}^d$ onto $\mathbb{PP}_d$, $\mathbb{PP}_d$ can be described as a quadratic program [1]. To solved this problem, Xia et al. proposed a fast projection method called LSA [12]. The key of LSA is to use the projection onto the line segment to replace the projection onto the parity check polytope, which simplifies the projection operations.

Algorithm 1 describes the process of the LSA. Define $\mathbf{v} \in \mathbb{R}^d$ as the input vector to be projected, $\mathbf{z}$ as the output projection vector. We use $\mathbf{u}$ denotes the projection of $\mathbf{v}$ onto unit hypercube, $\boldsymbol{\theta}_\mathbf{v}$ denotes indicator vector. if $\mathbf{u}$ satisfied $\boldsymbol{\theta}_\mathbf{v}^T \mathbf{u} \leqslant |\boldsymbol{V}| - 1$, then we can ensure $\mathbf{u} \in \mathbb{PP}_d$ and $\mathbf{u}$ is projection result. Otherwise, we need to find the two nearest even vertexes $\mathbf{A}$ and $\mathbf{B}$ to the vector $\mathbf{v}$ as the two endpoints of the projected line segment $L_{\mathbf{AB}}$. Then, we can calculate the projection of $\mathbf{v}$ onto $L_{\mathbf{AB}}$ as described in [12].

2.2 Simplified Line Segment Algorithm

In the original LSA, we need to determine whether $\mathbf{u}$ satisfied $\boldsymbol{\theta}_\mathbf{v}^T \mathbf{u} \leqslant |\boldsymbol{V}| - 1$. Therefore, there are two branches of the algorithm so that the original LSA is not suitable for implementing pipelines in hardware.

Here, we improve the following points to simplify the LSA:

(1) We propose to remove the hypercube projection judgment (Line 8 − 9) in Algorithm 1. For all given vector $\mathbf{v}$, we use the projection of $\mathbf{v}$ onto $L_{\mathbf{AB}}$ as the final projection result.
(2) Both indicator vector $\boldsymbol{\theta}_\mathbf{v}$ and the Odd-vertex $\boldsymbol{O}$ have only two values, therefore, we can use a binarized vector $\boldsymbol{\beta}_\mathbf{v}$ to replace $\boldsymbol{\theta}_\mathbf{v}$ and $\boldsymbol{O}$.

Algorithm 2 SLSA

Input: Vector $\mathbf{v} \in \mathbb{R}^d$
Output: Projection Vector $\mathbf{z} \in \left[-\frac{1}{2}, \frac{1}{2}\right]^\mathbf{d}$
1: Initialize: $\boldsymbol{\beta}_\mathbf{v}$: $\beta_{v,i} = \begin{cases} 1, & v_i > 0 \\ 0, & else \end{cases}$
2: **if** set $\{i|\beta_{v,i} = 1\}$ has an even number of elements **then**
3: $\quad i^* = \underset{i}{arg\min}\,(|v_i - 0|)$
4: $\quad \beta_{v,i^*} = 1 - \beta_{v,i^*}$
5: **end if**
6: Index $a = \underset{i}{arg\min}\,(|v_i - 0|)$
7: Index $b = \underset{i \neq a}{arg\min}\,(|v_i - 0|)$
8: Even-vertex $\mathbf{A} = \begin{cases} \beta_{v,i}, & \text{if } i \neq a \\ 1 - \beta_{v,i} s, & \text{if } i = a \end{cases}$
9: Even-vertex $\mathbf{B} = \begin{cases} \beta_{v,i}, & \text{if } i \neq b \\ 1 - \beta_{v,i}, & \text{if } i = b \end{cases}$
10: $\mathbf{z} = \prod_{L_{\boldsymbol{AB}}} (\mathbf{v}) - 0.5$
11: **return** $\mathbf{z}$

(3) Considering the potential asymmetry of LSA with fixed-point in hardware implementation, we further improved the algorithm to a centrosymmetric one. We relocate the projected hypercube from $[0, 1]$ to $[-\frac{1}{2}, \frac{1}{2}]$.

Algorithm 2 describes the proposed SLSA. The input vector $\mathbf{v}$ can be adjusted to the projection hypercube $\left[-\frac{1}{2}, \frac{1}{2}\right]$ by $\mathbf{v} = \mathbf{v} - \frac{1}{2}$.

Comparing Algorithm 2 with Algorithm 1, the process is significantly simpler and has fewer variables, which benefits the hardware implementation.

3. Proposed Hardware Implementation of SLSA

3.1 Proposed SLSA Architecture

The proposed SLSA architecture is diagrammed in Fig. 1. The majority of steps in Algorithm 2 comprise straightforward operations that can be mapped directly to the FPGA platform with linear complexity and constant latency. It is worth noting that the hardware implementation of SLSA must involve *argmin*. *argmin* is the operation to find the minimum value, which we can achieve by recursively calling the two min-tree. This construction is efficient and has linear complexity.

In Fig. 1, $\mathbf{v}$ ($\mathbf{z}$) denotes the input (output) vector. The first step is to binarize the input vector $\mathbf{v}$ to obtain $\boldsymbol{\beta}_\mathbf{v}$. The next step is lines 2 − 5 of Algorithm 2, which determines whether $\boldsymbol{\beta}_\mathbf{v}$ is an odd vertex. If $\boldsymbol{\beta}_\mathbf{v}$ has no even number of elements, then $\boldsymbol{\beta}_\mathbf{v}$ is an odd vertex and we can find the two nearest even vertex $\mathbf{A}$ and $\mathbf{B}$ directly based on line 6 − 9 in Algorithm 2. Otherwise, we need to find the index i^* satisfying $\underset{i}{arg\min}\,(|v_i - 0|)$, followed by flipping the element as $1 - \boldsymbol{\beta_{v,i}}$, when $i = i^*$. Then, we can calculate the projection of a vector onto a line segment $L_{\boldsymbol{AB}}$.

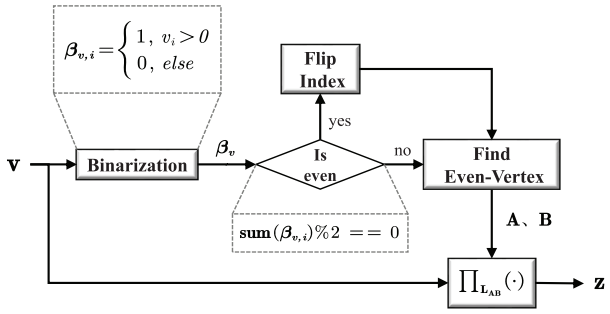

Fig. 1 Proposed SLSA hardware architecture.

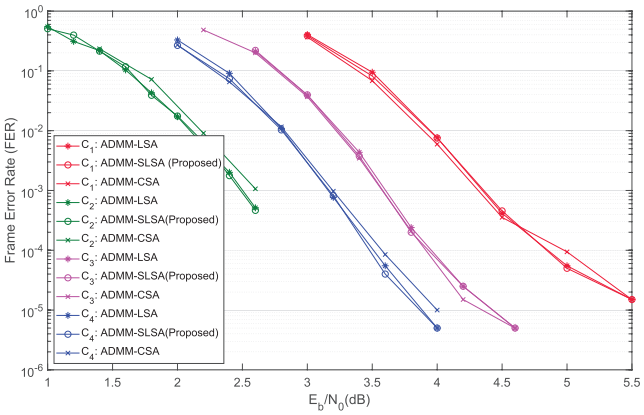

Fig. 2 FER performance of $C_1 - C_4$ with ADMM-SLSA decoder and other ADMM decoders.

3.2 Quantization Scheme

In the hardware implementation of SLSA, resource cost is essentially determined by the bitwidth of messages. In order to achieve a balance between resources and performance, a customised quantization scheme is used. For the binarised variables ($\boldsymbol{\beta}_\mathbf{v}$) in Algorithm 2, as well as the indicator variables (Index a, b), integer bits are used to represent them. When calculating the projections onto $L_{\mathbf{AB}}$, intermediate results are stored using fixed-points representation. Specifically, we can represent a fixed-point number as follow:

$$F_{\mathrm{p}}(S, I, P) \tag{2}$$

In (2), S indicates the sign bit, usually 1 bit. I and P indicate integer and fractional bitwidths respectively. In Sect. 4.2, we compare the quantization effects on FER performance for SLSA and LSA.

4. Simulation Results

In this section, we conduct experiments in terms of both FER performance and hardware implementation efficiency. Specifically, we compare the FER performance of CSA [4], LSA [12] and the proposed SLSA, quantization effects and resource scaling for the hardware design of SLSA.

4.1 FER Performance

In the simulation, the ADMM-LP decoder with penalty function l_2 [13] is employed and the additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation is assumed. In the experiments, we consider four codes conforming to IEEE802.16e standard: the $(576, 480)$ code C_1, the $(576, 288)$ code C_2, the $(576, 432)$ code C_3, and the $(576, 384)$ code C_4, The check node degrees of $C_1 - C_4$ are 20, $\{6, 7\}$, $\{14, 15\}$, and $\{10, 11\}$. The penalty parameter μ for $C_1 - C_4$ is 5.0, 3.0, 4.2, and 4.4, while the parameter α for $C_1 - C_4$ is 2.0, 1.4, 1.8, and 1.4, respectively. The maximum iteration number is set to be 300. The frame-error-rate curves are generated at a level of no less than 100 error frames for each point in the plots.

Figure 2 shows the FER performance of $C_1 - C_4$ for the ADMM-LP decoder with CSA, LSA and the proposed SLSA. The simulation results in Fig. 2 demonstrate that ADMM-SLSA decoder has a performance practically identical to decoder of ADMM-CSA and ADMM-LSA. Meanwhile, the SLSA is simpler in structure and can be mapped to a pipeline hardware architecture.

4.2 Quantization Effects

To illustrate the quantization effects of SLSA and LSA, we consider the code C_1 and the QC-LDPC code C_5 $(155, 64)$. For C_5, the check node degree is 5. The penalty parameter μ for C_5 is 3.0, while the parameter α is 1.4. The maximum iteration number is set to be 100. We then apply a uniform quantizer to projection vectors. Figure 3 shows the FER performance of C_1 and C_5 with $F_{\mathrm{p}}(1, 2, P)$ and $F_{\mathrm{p}}(1, 3, P)$ bitwidths. In Fig. 3 $(a), (b)$, the LSA has a poorer FER performance than SLSA with the same $F_{\mathrm{p}}(1, 2, P)$ bitwidths. The difference in fixed bitwidths between SLSA and LSA is due to the centrosymmetric structure of SLSA. We relocate the projected hypercube from $[0, 1]$ to $[-\frac{1}{2}, \frac{1}{2}]$ to eliminate the asymmetry of fixed-points representation. The SLSA with a minimum bitwidths of $F_{\mathrm{p}}(1, 2, 2)$ can achieve near floating performance while LSA need a minimum bitwidths of $F_{\mathrm{p}}(1, 3, 4)$.

4.3 Resource Scaling

To further evaluate the hardware efficiency, we also synthesis the hardware architecture proposed in Fig. 1 on the Xilinx ZCU102 FPGA platform which has 599550 LUTs, 548160 FFs, 2520 DSPs and 1824 BRAMs. For flexible development of validation, Vivado HLS tools was used for RTL architecture generation. In the simulations, we consider the degree of check nodes d of 5 for all Euclidean modules.

Table 1 shows the latency and hardware resource usage of SLSA module for different bitwidths including float precision. For float scheme, it yieds a latency of 70 clock cycles

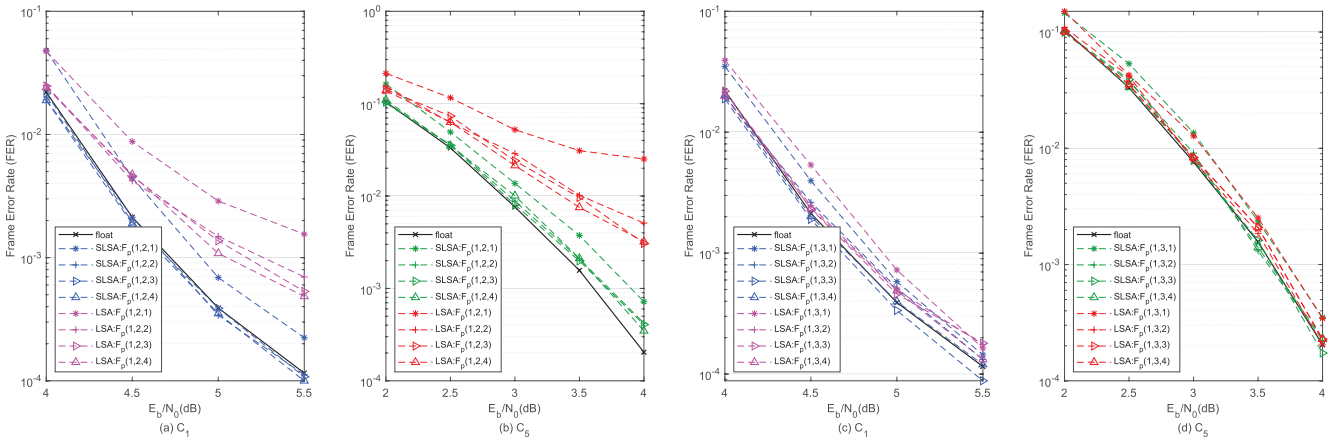

Fig. 3 FER performance of ADMM decoder with C_1 and C_5 at different $F_{\mathrm{p}}(S, I, P)$.

Table 1 Latency and resource scaling of SLSA modules.

Quantization Scheme	Float	$F_{\mathrm{p}}(1,5,P)$				$F_{\mathrm{p}}(1,4,P)$				$F_{\mathrm{p}}(1,3,P)$				$F_{\mathrm{p}}(1,2,P)$			
		1	2	3	4	1	2	3	4	1	2	3	4	1	**2**	3	4
Latency	70	15	14	14	14	14	14	14	14	15	14	14	15	14	**14**	14	14
Frequency(MHz)	183	291	288	287	291	289	291	288	287	288	289	288	288	287	**288**	289	288
Look-Up-Table (LUT)	10560	596	677	778	895	483	587	673	759	418	474	581	648	287	**416**	462	565
Flip Flop (FF)	8875	327	367	415	557	299	308	377	407	310	285	300	351	226	**308**	267	308
Digital Signal Processing (DSP)	15	0				0				0				0			

Table 2 Latency and resource scaling of SLSA, LSA, and CSA modules.

Algorithm	SLSA	LSA	CSA
Bitwidth	$F_{\mathrm{p}}(1,2,2)$	$F_{\mathrm{p}}(1,3,4)$	$F_{\mathrm{p}}(1,4,10)/F_{\mathrm{p}}(1,0,14)$
Latency	**14**	34	41
Frequency(MHz)	**288**	286	237
LUT	**416**	931	3415
FF	**308**	732	7927
DSP	**0**	0	2

at a maximum working frequency of 183 MHz. Due to the small degree of check nodes ($d = 5$) of SLSA module, logical resources can be used for storage instead of BRAMs resources. In terms of Table 1, the quantization scheme has around 80% less latency than the float scheme and the hardware resource cost is significantly reduced. Moreover, the quantified SLSA module does not require DSP resources. Combined with Sect. 4.2, Fig. 3 (*b*) demonstrate that $F_{\mathrm{p}}(1,2,2)$ representation is the better choice in terms of balancing hardware resources and FER performance.

Furthermore, we compare the latency and hardware resource usage of SLSA module with other EP modules: LSA and CSA [5] in Table 2. The CSA module can achieve near floating performance with a minimum input bitwidths of $F_{\mathrm{p}}(1,4,10)$ and minimum output bitwidths of $F_{\mathrm{p}}(1,0,14)$. The SLSA module yields the lowest latency of 14 at the highest working frequency of 288 MHz among three EP modules. All above EP modules can achieve near floating precision FER performance while SLSA has a simpler structure and a lower complexity compared with CSA. Moreover, the SLSA has a centrosymmetric structure without hypercube projection which is superior over LSA. The SLSA module has a 56.3% reduction in time consuming compared with LSA module and has a 72.7% reduction in time consuming compared with CSA module. Moreover, the SLSA module reduces the LUTs cost by 55.3% and FFs cost by 57.9% compared with LSA module and reduces the LUTs cost by 87.8% and FFs cost by 96.1% compared with CSA module. Thus, the proposed SLSA module is more efficient and has a lower hardware resource usage.

5. Conclusion

The Euclidean projection is a main challenge in ADMM hardware decoder. In this letter, we propose a hardware-friendly Euclidean projection algorithm called SLSA to solve this problem. It's proven by simulation results that the proposed SLSA module yields a lower latency and a lower hardware resource usage. For further research, we can apply the SLSA module in ADMM hardware decoder to improve throughput and reduce hardware resource usage.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62101204), by the Natural Science Foundation of Hubei Province under grant 2020CFB474, and by

the Fundamental Research Funds for the Central Universities under grant CCNU20ZT002.

References

[1] S. Barman, X. Liu, S.C. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," IEEE Trans. Inf. Theory, vol.59, no.12, pp.7870–7886, 2013.

[2] H.B. Thameur, B. Le Gal, N. Khouja, F. Tlili, and C. Jego, "Hardware design of euclidean projection modules for ADMM LDPC decoding," 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp.73–76, IEEE, 2018.

[3] X. Zhang and P.H. Siegel, "Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers," 2013 IEEE International Symposium on Information Theory, pp.1501–1505, IEEE, 2013.

[4] G. Zhang, R. Heusdens, and W.B. Kleijn, "Large scale LP decoding with low complexity," IEEE Commun. Lett., vol.17, no.11, pp.2152–2155, 2013.

[5] M. Wasson and S.C. Draper, "Hardware based projection onto the parity polytope and probability simplex," 2015 49th Asilomar Conference on Signals, Systems and Computers, pp.1015–1020, IEEE, 2015.

[6] M. Wasson, "Hardware-based linear program decoding with the alternating direction method of multipliers," Master's thesis, University of Toronto, 2016.

[7] M. Wasson, M. Milicevic, S.C. Draper, and G. Gulak, "Hardware-based linear program decoding with the alternating direction method of multipliers," IEEE Trans. Signal Process., vol.67, no.19, pp.4976–4991, 2019.

[8] X. Jiao, J. Mu, Y.C. He, and C. Chen, "Efficient ADMM decoding of LDPC codes using lookup tables," IEEE Trans. Commun., vol.65, no.4, pp.1425–1437, 2017.

[9] O. Samhan, "Application-specific fixed-point design of the alternating direction method of multipliers for linear program decoding," Master's thesis, University of Toronto, 2019.

[10] H. Wei and A.H. Banihashemi, "An iterative check polytope projection algorithm for ADMM-based LP decoding of LDPC codes," IEEE Commun. Lett., vol.22, no.1, pp.29–32, 2017.

[11] X. Jiao, Y.C. He, and J. Mu, "Memory-reduced look-up tables for efficient ADMM decoding of LDPC codes," IEEE Signal Process. Lett., vol.25, no.1, pp.110–114, 2017.

[12] Q. Xia, Y. Lin, S. Tang, and Q. Zhang, "A fast approximate check polytope projection algorithm for ADMM decoding of LDPC codes," IEEE Commun. Lett., vol.23, no.9, pp.1520–1523, 2019.

[13] X. Jiao, H. Wei, J. Mu, and C. Chao, "Improved ADMM penalized decoder for irregular low-density parity-check codes," IEEE Commun. Lett., vol.19, no.6, pp.913–916, June 2015.