Optimization Strategies for Low-Latency 5G NR LDPC Decoding on General Purpose Processor

Mody Sy

Optimization Strategies for Low-Latency 5G NR LDPC Decoding on General Purpose Processor

Mody Sy
EURECOM, 06410 BIOT, France
Email: mody.sy@eurecom.fr

***Abstract*—In recent years, with the progression of the computational abilities of General-Purpose Processors (GPPs), there has been a heightened interest in the implementation of software Low-Density Parity-Check (LDPC) decoders. This investigation provides a comprehensive analysis of the most effective strategies for optimizing the decoding latency of 5G LDPC on GPPs. Our proposed optimization mechanisms consist of the implementation of Advanced Vector Extensions 512 (AVX-512) instructions in computationally intensive routines and the application of code transformation techniques, specifically optimization through unrolling to tackle the primary computational challenges and minimize the overall latency of the decoder. To assess the efficiency of our proposed techniques, thorough simulations were carried out to determine the decoding time. Our findings indicate that the implementation of the aforementioned optimization techniques on computational routines causing time bottlenecks can lead to a significant reduction of at least 30% in computational delay, even under unfavorable conditions. This discovery demonstrates the feasibility of developing a low-latency software 5G NR LDPC decoder on an x86 architecture.**

***Index Terms*—5G NR, LDPC Decoder, x86 target, AVX-512, OpenAirInterface**

I. Introduction

The Low Density Parity Check (LDPC) codes, first proposed by Robert Gallager in 1962 [1], have garnered significant interest among the scientific community in developing efficient coding solutions for various communication standards. Due to their performance being near the capacity of Gaussian channels and the feasibility of parallelizing their decoding algorithms in software or hardware, they have been adopted by recent standards such as DVBS2 [4], [8], 5G NR, and others. However, the current LDPC decoders still face several limitations in terms of latency, throughput, and power consumption, creating a gap in performance that needs to be bridged. The channel decoding component remains the most computationally intensive aspect of the receiver, and thus, the decoder delay has a direct impact on the overall decoding throughput. With the advent of 5G, the requirement for low-latency decoding has become increasingly crucial. While ASIC [7] and FPGA technologies offer solutions to this trade-off, they also entail significant design expenses. Thus, developing low-latency and high-throughput LDPC decoders remains a challenging task. The advent of software targets, such as graphics processing units (GPUs) and general purpose processors (GPPs) [11] [4] [8], is allowing the development of programmable, reconfigurable and scalable software-based low-density parity-check (LDPC) decoders. However, the computation complexity associated with LDPC decoding techniques cannot be ignored. In this regard, the capabilities of modern x86 general-purpose processors can be leveraged by utilizing their resources and features. Moreover, the integration of vector units or extensions in modern processors can significantly reduce the execution time of a program and enhance the efficiency of parallelism by utilizing the AVX-512 (Intel Advanced Vector Extended) instruction sets. The optimization of the decoding algorithm is essential for real-time decoding and is based on OpenAirInterface [12] particularly an x86 target. The performance of the decoder is restricted by a few steps of the algorithm, which serve as computational bottlenecks. Profiling, which is the step prior to optimization, is crucial in this regard as optimization cannot be achieved without measurement. The paper is structured as follows: Section II presents the foundation of 5G NR LDPC, Section III describes the optimization procedures, the results and performance analysis are presented in Section IV and finally, Section V concludes the paper.

II. 5G NR LDPC Foundation

The Low-density Parity-check (LDPC) coding technique has been widely adopted for the uplink and downlink shared transport channels in 5G New Radio (NR) communication systems. This coding method is well-suited for 5G NR shared channels due to its high throughput, low latency, efficient decoding complexity and rate compatibility. The performance of LDPC codes in 5G NR demonstrates an error floor at or below the 10^5 block error rate (BLER), which is a significant advantage over traditional coding techniques. The QC-LDPC family serves as the foundation for 5G NR LDPC codes. NR LDPC code is constructed from a Base Graph Matrix (BG) of dimension $M \times N$, designated as $\boldsymbol{H}_{BG}$. The selection of the $\boldsymbol{H}_{BG}$ matrices in the 5G NR coding process is based on the coding rate and the length of the transport block or code block. There are two base graphs, $BG1$ with dimensions of $N = 68$ and $M = 46$ optimized for large information block sizes of $K \leq 8448$ and high coding rates between $1/3 \leq R \leq 8/9$, and $BG2$ with dimensions of $N = 52$ and $M = 42$ optimized for small information block sizes of $K \leq 3840$ and lower coding rates between $1/5 \leq R \leq 2/3$. These codes are suitable for high reliability scenarios due to their ability to achieve additional coding gain at low-code rates. The maximum number of information bits for $BG1$ is $K = 22Z_c$ and for $BG2$ is $K = 20Z_c$, where Z_c is the lifting size. There are 51 lifting sizes ranging from 2 to 384 for each base graph. The parity check matrix, denoted as $\mathbf{H}$, is obtained by replacing each element of H_{BG} with a cyclic permutation identity matrix, $I(P_{ij})$. In other words, each element of H_{BG} is replaced by the corresponding Cyclic Permutation Matrix (CPM). The size of the matrix $\mathbf{H}$ is $m \times n$, with $m = M \times Z_c$,

979-8-3503-3412-8/23/$31.00 ©2023 IEEE

$n = N \times Z_c$, and $k = n - m = (N - M) \times Z_c$. Both $BG1$ and $BG2$ have similar structures. There are various effective LDPC encoding techniques because of the structure and features of base graphs. Due to the structure and features of the base graphs, there are various effective Low-Density Parity-Check (LDPC) encoding techniques. A new, high throughput and low complexity encoding technique has been proposed in [13] The schematic representation of the PUSCH processing from the MAC layer to the physical layer is illustrated in Figure 1. This figure depicts the transmit-end for the uplink transmission and encompasses various processing stages, including transport block CRC attachment, code block segmentation with accompanying CRC attachment, LDPC encoding, rate matching, code block concatenation, and modulation. The transmission procedure involves the addition of a Cyclic Redundancy Check (CRC) to the transport block. CRC serves as an error detection code, utilized to calculate the Block Error Rate (BLER) after the decoding process. Consequently, the entire transport block is employed to calculate the parity bits of the CRC. Let us assume that the transport message, prior to the attachment of the CRC, is represented as $a(0), a(1), \ldots, a(A-1)$, where A denotes the size of the transport block message. The parity bits, $p(0), p(1), \ldots, p(L-1)$, are then added to the message, with L being the number of parity bits employed. If $A > 3824$, the number of parity bits, L, is 24. In the case where $A \leq 3824$, L is equal to 16. The resulting message, after the attachment of the CRC, is represented as $b(1), b(2), \ldots, b(B)$, where B denotes the size of the transport block information, including the CRC bits, and $B = A + L$. In 3GPP NR, the selection of the Low-Density Parity-Check (LDPC) base graph is dependent on the size of the transport block message (A) and the transport block coding rate (R). According to the 3GPP 38212 specification, if $A \leq 292$, or if $A \leq 3824$ and $R \leq 0.67$, or if $R \leq 0.25$, LDPC BG2 is utilized. Otherwise, LDPC BG1 is employed. The result of the code block segmentation and Cyclic Redundancy Check (CRC) attachment process is a sequence of code blocks, represented as $c_r(1), c_r(2), \ldots, c_r(K_r)$, where $K_r = K'_r + L$. Here, K' is the number of bits in the r^{th} code block and L is the attached CRC for that code block. Each code block message is then encoded independently. The input bit sequence in 3GPP NR is represented as $\boldsymbol{c} = [c(0), c(1), \ldots, c(K_r - 1)]^T$, where K_r is the number of information bits in the code block. The redundant bits, also known as parity bits, are denoted as $\boldsymbol{w} = [w(0), w(1), \ldots, w(N + 2Z_c - K_r + 1)]^T$. The LDPC-coded bits are represented as $d(0), d(1), \ldots, d(N_r - 1)$. Each code block is fed to the LDPC encodeder using the following procedure [10]:

1) Find the set with index iLS which contains Z_c in [10].
2) Set $d_{k-2Z_c} = c_k, \forall k = 2Z_c, \ldots, K_r - 1$
3) Generate $N_r + 2Z_c - K_r$ parity bits $\boldsymbol{w} = [w(0), w(1), \ldots, w(N + 2Z_c - K_r + 1)]^T$ such that $\boldsymbol{H} \times [\boldsymbol{c}\ \boldsymbol{w}]^T = \boldsymbol{0}$
4) The encoding is performed in $GF(2)$.
5) Set $d_{k-2Z_c} = w_{k-K_r}, \forall k = K_r, \ldots, N_r + 2Z_c - 1$

In the rate matching process, each code block is subjected to independent rate matching. The resulting output sequences are denoted as f_{rk}, where r ranges from 0 to $C - 1$ and k ranges from 0 to $E_r - 1$. The value of E_r represents the number of rate matched bits for the r^{th} code block. Subsequently, code block concatenation is performed, which involves transforming all code block messages into a sequence of transport block messages. The output from this block is a sequence of bits represented by $g(0), g(1), \ldots, g(E-1)$. The receiving chain serves as the inverse counterpart of the transmitting chain. The decoding of Low-Density Parity-Check (LDPC) codes is carried out on each code block individually, and a range of decoding techniques can be implemented. Among these, belief propagation (BP) methods, which rely on iterative exchange of messages between bit nodes and check nodes, are the most commonly utilized for LDPC decoding. Although the BP method presents a considerable computational complexity, it offers near-optimal decoding performance [16]. To achieve a better trade-off between performance and complexity, various efficient decoding algorithms based on Min-Sum (MS) approximation have been proposed in the literature [14], [15], [17], [19]–[21]. However, layered message passing decoding has been shown to improve convergence time, making it a suitable option for Ultra-Reliable Low-Latency Communications (URLLC).

In order to approach the next section with a clear insight, let us start by examining the fundamental principle of the minimum sum decoding algorithm, as described in the Algorithm 1. Assume that $q_{ij}(b)$ is the probability that $c_i = b$, $b \in \{0, 1\}$, given extrinsic information from all check nodes, excluding check node j and channel sample y_i, and $r_{ji}(b)$ is the probability of the $j - th$ check equation being satisfied given $c^i = b$ and the other bits have separable distribution represented by $\{q_{ij'}\}_{j' \neq j}$.

The notation is as follows:

- $\mathcal{B}_j$ = Bit nodes connected to check node j,
- $\mathcal{B}_j/i$ =bit nodes connected to check node j,excluding Bit node i
- $\mathcal{C}_i$ = check nodes connected to variable node i
- $\mathcal{C}_i/j$ = check nodes connected to variable node i, excluding check node j

The subsequent principle may be used to illustrate the layered decoding principle as described in the work of WANG [22] :

- Each layer independently processes variable node operations and checks node operations.
- The input Log-Likelihood Ratio (LLR) of the present layer is derived from the output LLR of the previous layer.
- The ultimate output LLR produced by the decoding algorithm will serve as the basis for the final decision-making process.
- The input Log-Likelihood Ratio (LLR) of the current layer can be updated using the following equation: $\mathcal{L}_{k+1,i} = \mathcal{L}_{k,i} - \mathcal{L}_{k+1,i'}$. Here, $\mathcal{L}_{k+1,i}$ represents the updated input LLR of layer $k+1$, while $\mathcal{L}_{k,i}$ is the output LLR of the previous layer and $\mathcal{L}_{k+1,i'}$ represents the old input LLR of layer $k+1$.

III. Optimisation Procedures

A. Profiling step

The experiments is carried out on a high-performance server equipped with an Intel® Xeon® Gold 6154 processor. This

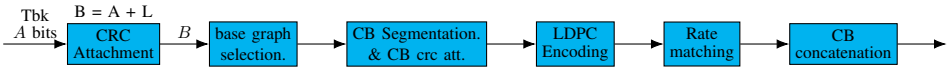

Fig. 1. LDPC encoding chain

Algorithm 1: Log-Likelihood Min Sum Algorithm

Input:
The channel log likelihoods : $\mathcal{L}_i \in \mathbb{R}^n$:
Maximum # of iterations : MAX-ITER
Description of the parity check matrix using $\mathcal{B}(j)$ and $\mathcal{C}(i)$.
Output: Estimated code word: $\hat{\boldsymbol{c}} \in \{0,1\}^n$:

Initialization :
for *each i, and for each* $j \in \mathcal{C}(i)$ **do**
  $\mathcal{L}(q_{ij}) = \mathcal{L}_i$
Check Node to Variable Node Step (horizontal step):
for *each check node* j **do**
  for *each variable node* $i \in \mathcal{B}(j)$ **do**
    $$\mathcal{L}(r_{ji}) = \left(\prod_{i' \in \mathcal{B}_{j/i}} \operatorname{sign}\left(L\left(q_{i'j}\right)\right)\right) \cdot \min_{i' \in \mathcal{B}_{j/i}} \left(\left|\mathcal{L}\left(q_{i'j}\right)\right|\right)$$
Variable Node to Check Node Step (vertical step)
for *each variable node* i **do**
  for *each check node* $j \in \mathcal{C}(i)$ **do**
    $$\mathcal{L}(q_{ij}) = \mathcal{L}_i + \sum_{j' \in \mathcal{C}_i/j} \mathcal{L}(r_{j'i})$$
    Also compute the output likelihoods
    $$\mathcal{L}(Q_i) = \mathcal{L}_i + \sum_{j' \in \mathcal{C}_i} \mathcal{L}(r_{j'i})$$
Hard decision:
for *each* i **do**
  $\hat{c}_i = 1$ if $\mathcal{L}(Q_i) < 0$ else $\hat{c}_i = 0$
Parity Check:
if $\boldsymbol{H}\hat{\boldsymbol{c}}^T = \mathbf{0}$ **then**
  return $\hat{\boldsymbol{c}}$
otherwise, $\boldsymbol{if}$ # ITER $<$ MAX-ITER
goto Check Node to Variable Node Step
else
  return $\hat{\boldsymbol{c}}$ and indication of coding failure.

TABLE I
PROFILING CASE BG1, B=8448, 5 ITERATIONS

Routines	**Time [μ s]**		
	R=1/3	***R=2/3***	***R=8/9***
llr2CnProcBuf	15.931	7.196	2.507
cnProc (per iteration)	19.847	12.604	9.546
cnProcPc (per iteration)	0.218	0.378	0.341
bnProc (per iteration)	2.817	0.436	0.767
bnProcPc(per iteration)	4.149	2.043	1.006
cn2bnProcBuf (per iteration)	12.437	4.171	1.732
bn2cnProcBuf (per iteration)	11.407	3.984	1.648
llrRes2llrOut	0.865	0.516	0.411
llr2bit	0.382	0.196	0.167
ldpc_decoder [total]	273.502	131.489	78.316

TABLE II
PROFILING CASE BG2, B =3840, 5 ITERATIONS

Routines	**Time [μ s]**		
	R=1/5	***R=1/3***	***R=2/3***
llr2CnProcBuf	0.790	0.473	0.258
cnProc (per iteration)	9.061	5.530	1.681
cnProcPc (per iteration)	7.341	3.804	2.154
bnProc (per iteration)	0.265	0.387	0.175
bnProcPc(per iteration)	1.472	0.925	0.456
cn2bnProcBuf (per iteration)	2.807	1.632	0.701
bn2cnProcBuf (per iteration)	7.833	4.362	1.381
llrRes2llrOut	7.175	4.162	1.386
llr2bit	0.665	0.470	0.247
ldpc_decoder [total]	145.015	82.63	33.389

processor boasts 18 cores, a clock speed of 3.00 GHz, AVX-512 instruction set support, 2 FMA units, and a L3 cache of 24.75 MB. To assess the execution time of the decoder, we utilize the OAI time measurement tool, specifically the $time\_meas.h$ library. The utilization of this tool facilitated the execution of the profiling phase, thereby enabling the identification of crucial limitations and bottlenecks. In order to mitigate any potential impacts from the operating system, the decoder is executed on a singular core, and the test computer is operated in a console mode. We evaluate the decoding time for both short (3840) and long (8448) block lengths. The number of decoder iterations as well as the coding rate are considered. Tables I and II present the profiling results as a function of coding rate, with the number of iterations fixed at 5 for both short (BG_2) and long (BG_1) block lengths;

The results of our experiments demonstrate that as the number of iterations increases, the latency of the system grows correspondingly, resulting in a near doubling of the delay. Furthermore, we have noted the impact of the coding rate on the latency, where although lower coding rates lead to improved decoding performance, the decoding delay still plays a critical role in determining system performance. We can observe from the profile findings that some algorithmic processes are computational time bottlenecks. The decoder's total effectiveness is constrained by these bottlenecks or hot spots. The aforementioned findings demonstrate that processing of CNs, BNs, and messages passing between them constitute up a significant part of the decoder. In order to minimize decoder latency, the primary focus of optimization efforts should be directed towards the routines $cnProc$, $bnProc$, $bnProcPc$, $cn2bnProcBuf$, $bn2cnProcBuf$, and $llr2cnProcBuf$. The procedure of $cnProc$ in particular is computationally intensive, and as a result, constitutes the primary bottleneck in the decoding process.

B. Optimization Step

In the pursuit of optimizing decoder workload and reducing time complexity, it is imperative to conduct a thorough examination of the available alternatives and possibilities. The choice of decoding algorithm, the design of the parity matrix, the optimization of the algorithm's code, and the consideration of the target architecture all play a crucial role in ensuring a real-time decoding process. Utilizing cutting-edge features of contemporary processors, such as Single Instruction Multiple Data (SIMD) intrinsics, can result in a significant enhancement of the decoding algorithm's performance. The utilization of the SSE, $AVX2$, and $AVX-512$ instruction sets has been shown to enable various forms of parallelism specifically for 5G NR LDPC decoding. To achieve optimal throughput and minimize latency in the decoding process, the novel features of these conventional instruction sets have been exploited in the loading, storing, and preprocessing operations, as proven in the literature [2], [4], [8]. Furthermore, the implementation of an LDPC decoder utilizing the $AVX-512$ instruction set has demonstrated a higher degree of parallelism, resulting in a reduction in latency and an increase in throughput.

1) **AVX-512 instruction sets in CNs/BNs processing:** The optimization process will prioritize the acceleration of the CNs and BNs routines, which have been identified as the performance bottlenecks of the algorithm. To fully utilize the capabilities of the vector processing units, we will revise the decoding algorithm to incorporate the latest Intel AVX-512 instructions. These instructions, featuring larger vector units and registers, will significantly enhance the speed of the check node (CN) and variable node (BN) processing operations. Theoretically, the performance speed of the decoder can be estimated to be two times greater by utilizing AVX-512 instructions instead of AVX2. The implementation of the decoder can operate using either AVX2 or AVX-512 versions, dependent upon the microarchitecture of the target processor or the instruction sets supported by the execution environment.

The optimized processing flow of CNs is depicted in Listing 1 .

```
1
2
3
4   // CN processing
5
6   if (BG==1)
7   {
8       switch (R)
9       {
10          case 13:
11          {
12              #ifdef __AVX512BW__
13              nrLDPC_CNP_BG1_R13_AVX512(...);
14              #else
15              nrLDPC_CNP_BG1_R13_AVX2(...);
16              #endif
17              break;
18          }
19          case 23:
20          {
21              #ifdef __AVX512BW__
22              nrLDPC_CNP_BG1_R23_AVX512(...);
23              #else
24              nrLDPC_CNP_BG1_R23_AVX2(...
25              #endif
26              break;
27          }
28          case 89:
29          {
30              #ifdef __AVX512BW__
31              nrLDPC_CNP_BG1_R89_AVX512(...);
32              #else
33              nrLDPC_CNP_BG1_R89_AVX2(...);
34              #endif
35              break;
36          }
37      }
38  }
39  else
40  {
41      switch (R)
42      {
43          case 15:
44          {
45              #ifdef __AVX512BW__
46              nrLDPC_CNP_BG2_R15_AVX512(...);
47              #else
48              nrLDPC_CNP_BG2_R15_AVX2(...);
49              #endif
50              break;
51          }
52          case 13:
53          {
54              #ifdef __AVX512BW__
55              nrLDPC_CNP_BG2_R13_AVX512(...);
56              #else
57              nrLDPC_CNP_BG2_R13_AVX2(...);
58              #endif
59              break;
60          }
61          case 23:
62          {
63              #ifdef __AVX512BW__
64              nrLDPC_CNP_BG2_R23_AVX512(...);
65              #else
66              nrLDPC_CNP_BG2_R23_AVX2(...);
67              #endif
68              break;
69          }
70      }
71  }
```

Listing 1. AVX-512 instruction sets in CNs processing

2) **Enhancement of Processing Buffers:** In the decoding process, buffers have been introduced as a means of temporarily storing the information that is transmitted between the CNs and BNs nodes. This enables the implementation of vectorization in a more efficient manner. However, it has been observed that the transfer of messages into buffers is a time-consuming process. To resolve this issue, a highly optimized memcpy function has been developed. The optimized memcpy function employs the use of the REP MOVSB (Repeat While Equal Move String Instructions -Byte) instruction, which is well-suited for modern x86 processors. This implementation significantly improves the efficiency of message transfer into buffers and overall performance of the decoding process.

3) **Unrolling process-based optimization of routines:** Additionnaly, we introduce the loop unwinding technique as a means of efficiently allocating processor registers and mitigating the delay caused by processing at the control node (CN) and bit node (BN). This approach is motivated by the recognition that many programs are characterized by prolonged execution times within loops. To reduce this processing cost, loops can be transformed into repeating sequences of equivalent, independent operations. While loop unwinding can significantly enhance program execution speed, it also increases the space complexity of the program, resulting in a trade-off between space and time that may not be desirable, particularly for embedded systems.

IV. Numerical Results and Discussion

The optimization results on a selected instance are depicted in Figure 2 and Figure 3. The resulting decoding times after the optimizations are compared to the measurements obtained during the profiling phase and are presented in Tables I and

II to highlight the benefits of the optimization efforts. The optimization outcomes were analyzed based on the employed coding rate and the maximized block lengths for the BG1 and BG2 configurations, with a fixed iteration count of 5. The studied decoders are respectively named reference NRLDPC decoder and optimized NRLDPC decoder. The correlation

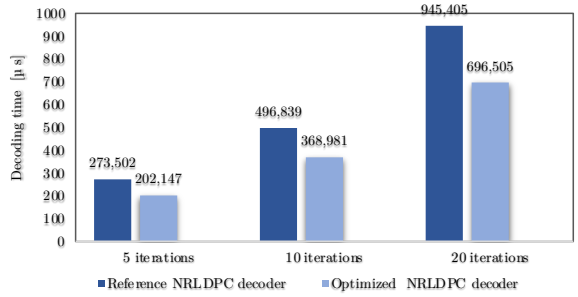

Fig. 2. Optimization balance - BG1, B=8448, R=1/3 i= 5, 10, 20 iterations.

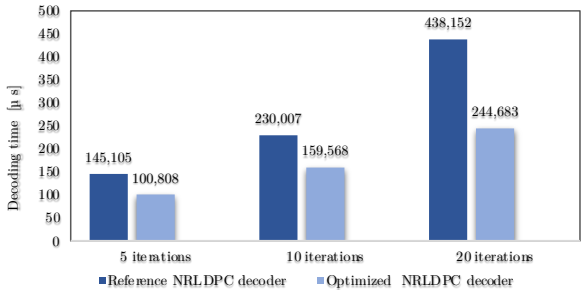

Fig. 3. Optimization balance - BG2, B=3840, R=1/5, i= 5, 10, 20 iterations.

between iterations and decoding time is evident. An increase in the number of iteration results in a substantial increase in the delay. Additionally, the decoding complexity of LDPC codes increases exponentially with the size of the code, which limits their practical application.

Through the implementation of various optimization strategies, our proposed $LDPC$ decoder exhibits a decreased latency compared to the established baseline or reference implementation. Consequently, the implementation of the suggested optimization techniques across all procedures is anticipated to result in a decoder with a substantial reduction in computational time, potentially by a minimum of 50%. It is important to emphasise that the optimization efforts in this investigation were exclusively directed towards the primary hotspots that acted as critical bottlenecks. The results of the optimization efforts are presented in Figures 4 and 5.

V. Conclusion

This paper presented strategies for optimizing the decoding latency of 5G Low-Density Parity-Check (LDPC) on General Purpose Processors (GPPs). Our proposed optimization mechanisms consist of the implementation of Advanced Vector Extensions 512 (AVX-512) instructions in computationally intensive routines and the application of code transformation techniques, specifically optimization through unrolling. The aim of these strategies is to address the main computational bottlenecks and enhance the overall performance of the LDPC

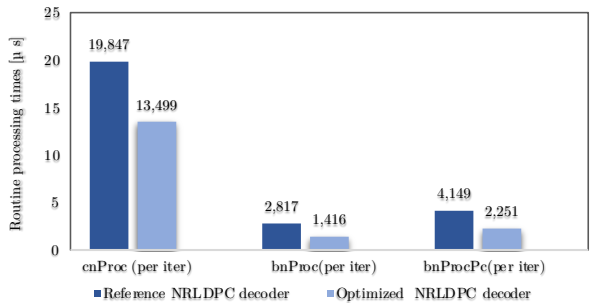

Fig. 4. CNs and BNs Processing times - BG1, B=8448, i=5, R=1/3

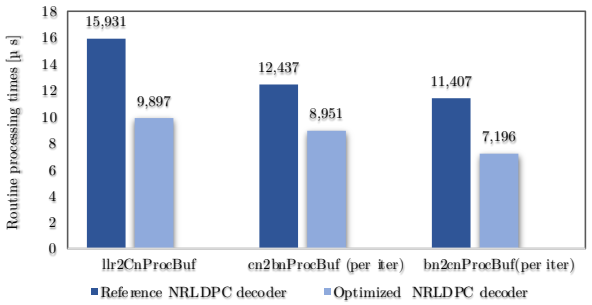

Fig. 5. CNs and BNs buffer computation times - BG1, B=8448, i=5, R=1/3

decoder. To validate the effectiveness of our optimization strategies, we conducted several simulations to evaluate the latency of the LDPC decoder on OpenAirInterface(OAI). Our findings indicated that the implementation of the aforementioned optimization techniques on computational routines causing time bottlenecks can lead to a significant reduction of at least 30% in computational delay, even under unfavorable conditions. This discovery demonstrated the feasibility of developing a low-latency software 5G NR LDPC decoder on an x86 architecture.

References

[1] R. GALLAGER, "Low-density parity-check codes, IRE Transactions on information theory," vol. 8, pp. 21 - 28, 1962.

[2] B. LeGAL and C. Jego," High-throughput multi-core LDPC decoders based on x86 processor," IEEE Trans. Parallel Distrib. Syst, vol. 27, pp. 1373-1386, May 2016.

[3] G. Eugene, "Scaling the Fast x86 DVB-S2 Decoder to 1 Gbps," IEEE Aerospace Conference, 02 March 2019

[4] Y. Xu, W. Wang, Z. Xu and X. Gao, "AVX-512 Based Software Decoding for 5G LDPC Codes," Signal Processing Systems IEEE International Workshop on, pp. 54-59, 2019 .

[5] J. ANDRADE, G. FALCAO and V. SILVA,A Survey on Programmable LDPC Decoders,IEEE Communications Surveys & Tutorials July 2019.

[6] G. Falcão, L. Sousa and V. Silva, "Massively LDPC decoding on multicore architectures,v IEEE Trans. Parallel Distrib. Syst., vol. 22, p. 309 – 322, 2011.

[7] G. Falcão, V. Silva and L. Sousa, "How GPUs can outperform ASICs for fast LDPC decoding," in Proc. ACM Int. conf. Supercomputing, p. 390–399, 2009.

[8] W. Ji, Z. Wu, K. Zheng, L. Zhao and Y. Liu, "Design and Implementation of a 5G NR System Based on LDPC in Open Source SDR," IEEE Globecom Workshops (GC Wkshps), pp. 9-13, December 2018.

[9] H. Wu et H. Wang, "A High Throughput Implementation of QC-LDPC Codes for 5G NR," IEEE Journal Article Publisher, vol. 77, 19 December 2019.

[10] 3GPP TS 38.212 V16.2.0, " Technical Specification Group Radio Access Network, Multiplexing and channel coding", July 2020.

[11] A. Alex , . K. Leonid, M. Jane , S. Fred and N. Scott , “5G NR LDPC Decoding Performance Comparison between GPU & FPGA Platforms,” 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 3 May 2019.

[12] F. Kaltenberger, A. Silva, A. Gosain, L. Wang and T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” Computer Networks., vol. 176, pp. 107-284, 2010.

[13] T. Nguyen, T. N. Tan, and H. Lee, “Efficient QC-LDPC Encoder for 5G New Radio”, Electronics, vol. 8, p. 668, June 2019

[14] Ming Jiang, Chunming Zhao, Li Zhang and Enyang Xu, ”Adaptive offset min-sum algorithm for low-density parity check codes,” in IEEE Communications Letters, vol. 10, no. 6, pp. 483-485, June 2006, doi: 10.1109/LCOMM.2006.1638623.

[15] M. Xu, J.Wu and M. Zhang, ”A modified Offset Min-Sum decoding algorithm for LDPC codes,” 2010 3rd International Conference on Computer Science and Information Technology, 2010, pp. 19-22, doi: 10.1109/ICCSIT.2010.5564884.

[16] K. Sun and M. Jiang, ”A Hybrid Decoding Algorithm for Low-Rate LDPC codes in 5G,” 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), 2018, pp. 1-5, doi: 10.1109/WCSP.2018.8555597.

[17] L. Jun, S. Jin, Z. Wang and L. Li, ”An improved min-sum based column-layered decoding algorithm for LDPC codes,” 2009 IEEE Workshop on Signal Processing Systems, 2009, pp. 238-242, doi: 10.1109/SIPS.2009.5336258.

[18] V. Savin, ”Self-corrected Min-Sum decoding of LDPC codes,” 2008 IEEE International Symposium on Information Theory, 2008, pp. 146-150, doi: 10.1109/ISIT.2008.4594965.

[19] C. Jones, E. Valles, M. Smith and J. Villasenor, ”Approximate-MIN constraint node updating for LDPC code decoding,” IEEE Military Communications Conference, 2003. MILCOM 2003., 2003, pp. 157-162 Vol.1, doi: 10.1109/MILCOM.2003.1290095.

[20] S. Myung, S. I. Park, K. J. Kim, J. Y. Lee et S. Kwon, ‘’Offset and normalized min-sum algorithms for ATSC 3.0 LDPC decoder,” EEE Trans. Broadcast., vol. 63, p. 734–739, Dec 2017.

[21] X. Meng, w. Jianhui et M. ZHANG, ‘’A modified offset min-sum decoding algorithm for LDPC codes,” Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. IEEE, pp. 19-22, 2010.

[22] LIFANG WANG “Implementation of Low-Density Parity-Check codes for 5G NR shared channels”,master Thesis, August 30, 2021