3GPP TSG-RAN WG1 Meeting #123 **R1-2508841**
Dallas, USA, November 17th – 21st, 2025

Agenda item:	11.4.1 Channel coding for 6GR
Source:	AccelerComm
Title:	LDPC decoder throughput and chip area
Document for:	Discussion/Decision

1 Introduction

In the RAN1#122bis meeting held October 13th – 17th 2025, the following working assumption was reached regarding the channel coding study item [1]:

Working Assumption

- Study 6G data channel coding for higher throughput than 5G with acceptable performance-complexity tradeoff for both NW side and UE side,
  - Target peak data rate is assumed to be 2 times of the target peak data rate defined in TR38.913

Note: The other target throughput is up to company to report.

Note: Applicability of the potential channel code will be further discussed.

In the feature lead's summary of the RAN1#122bis channel coding study item [2], the following proposals are included:

Proposal 3.3.1-2-v1: For the evaluation of throughput in 6G data channel coding scheme(s), the following formula can be considered

$$\text{Throughput} = \frac{K \times f}{T} \times C$$

Where K is the number of information bits in one coded block, f is the operating frequency, T is the number of decoding cycles per code block, C is the number of decoder cores.
For LDPC code, $T = I \times T_{\text{iter}}$, where I is the maximum number of decoding iterations, and T_{iter} is the decoding cycles per iteration.

Proposal 3.3.1-1-v2: For the evaluation of 6G data channel coding scheme(s), at least the following metrics are considered

- BLER performance
- Decoding throughput/latency
- Complexity
- Area efficiency

Proposal 3.1.2-1-v3: For the study of 6G data channel coding for higher throughput than 5G with acceptable performance-complexity tradeoff, the following options can be considered

- Option 1: Study LDPC code as data channel coding for higher throughput than 5G
  - Option 1-0: Implementation based solutions e.g. improving clock frequencies or higher decoding parallelism.
  - Option 1-1: Increase the lifting size
  - Option 1-2: Reduce the maximum number of iterations, e.g., fast convergence LDPC code
  - Option 1-3: Increase the number of systematic columns
  - Option 1-4: Reduce the number of edges in LDPC BG
  - Option 1-5: Optimize parallelism, e.g., improve orthogonality between rows of LDPC BG
  - Other options are not precluded.
  - The above options may be combined.
  - The LDPC code is quasi-cyclic (QC-LDPC)
  - FFS: whether to use 5G LDPC BG(s) or define new LDPC BG(s)
  - FFS: applicable condition(s)

This contribution describes AccelerComm's views on LDPC decoder throughput and chip area for 6GR based on the above.

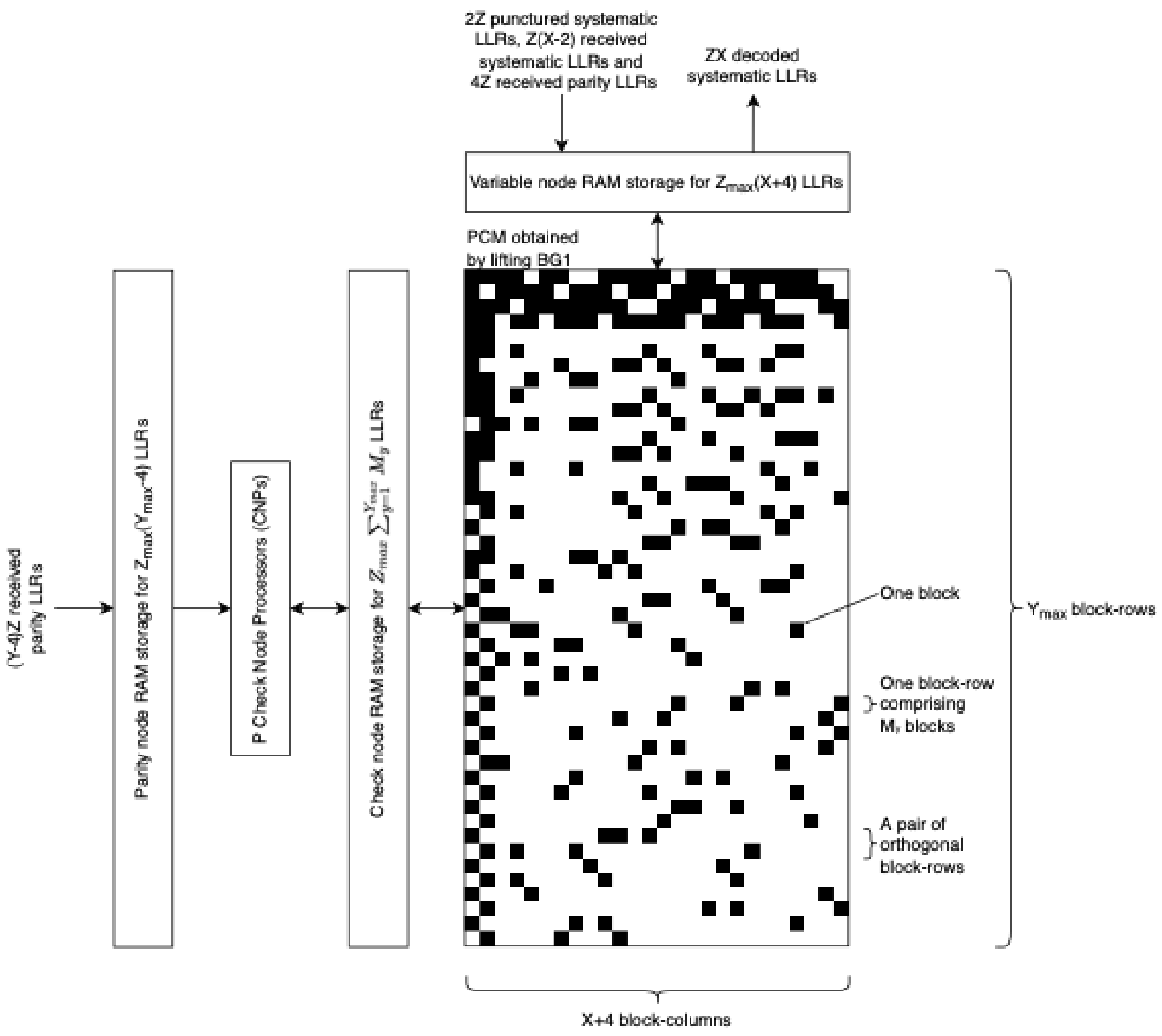
2 Layered Belief Propagation LDPC decoder algorithm

Figure 1 illustrates a block diagram for a layered belief propagation LDPC decoder in the case of BG1. The notation used throughout this Tdoc is listed in Table 1.

Notation/ Terminology	**Description**
base graph (BG)	A sparse matrix having Y_{max} rows and $X+Y_{max}$ columns, in which the non-null elements provide rotation values, where BG1 and BG2 are defined in [3]
parity check matrix (PCM)	A sparse matrix having ZY_{max} rows and $Z(X+Y_{max})$ columns, obtained by lifting the base graph
block	A Z by Z block of elements in the PCM obtained by lifting a corresponding non-null element of the base graph
block-column	A set of Z columns in the PCM obtained by lifting a corresponding column of the base graph
block-row	A set of Z rows in the PCM obtained by lifting a corresponding row of the base graph

sub-row	A set of (up to) P rows in a block-row, where there are ceil(Z/P) sub-rows in each block-row
Z_{max}	Maximum supported lifting size, where $Z_{max} = 384$ for 5G NR [3]
Z	LDPC lifting size used for a particular LDPC decoding operation, depending on information block length K
X	Number of systematic block-columns in the PCM, where X = 22 for BG1 and X = 10 for BG2
K	Information block length, where K = ZX when there are no filler bits
Y_{max}	Number of block-rows in the PCM, where $Y_{max} = 46$ for BG1 and $Y_{max} = 42$ for BG2
Y	Number of block-rows used in the PCM, depending on coding rate R
M_y	Number of blocks in the first X+4 block-columns of the y^{th} block-row of the PCM
R	LDPC coding rate, where R = X/(X+Y-2) when ZY parity bits are used
I	Maximum number of LDPC decoding iterations performed
P	LDPC decoder parallelism
D	Pipeline depth
T_{iter}	Number of clock cycles required to process each LDPC decoding iteration
f	FPGA or ASIC clock frequency
C	Number of parallel LDPC decoder cores
A_1, A_2, A_3	Baseline chip areas for RAM, row-parallel logic and block-parallel logic

Table 1. Notation and terminology used throughout this contribution.



2Z punctured systematic LLRs, Z(X-2) received systematic LLRs and 4Z received parity LLRs
ZX decoded systematic LLRs
Variable node RAM storage for $Z_{max}(X+4)$ LLRs
PCM obtained by lifting BG1
(Y-4)Z received parity LLRs
Parity node RAM storage for $Z_{max}(Y_{max}-4)$ LLRs
P Check Node Processors (CNPs)
Check node RAM storage for $Z_{max}\sum_{y=1}^{Y_{max}} M_y$ LLRs
One block
Y_{max} block-rows
One block-row comprising M_y blocks
A pair of orthogonal block-rows
X+4 block-columns

Figure 1. Block diagram of a layered belief propagation LDPC decoder for the case of BG1.

The layered belief propagation LDPC decoding algorithm proceeds as follows.

```
1. Load 2Z punctured systematic LLRs (having values of 0), Z(X-2) received systematic LLRs
   and ZY received parity LLRs into the variable and parity node RAMs
2. Set all check node RAM values equal to 0
3. For each of the I iterations
4.     For each of the Y block-rows in the PCM
5.         For each of the ceil(Z/P) sub-rows in the block-row
6.             Initialize a minimum and a second-minimum to infinity or to the value of a
               corresponding received parity LLR loaded from the parity node RAM for
               each of the (up to) P rows in the sub-row
7.             For each of the $M_y$ blocks in the first X+4 block-columns of the block-row
8.                 For each of the (up to) P rows in the sub-row
```

```
9.                      Load a corresponding LLR from the variable node RAM,
                        selected according to the rotation performed by the block
10.                     Subtract a corresponding LLR stored in the check node RAM
                        from the LLR loaded from the variable node RAM
11.                     Use the resulting difference to update the minimum and/or
                        second minimum if its value is lower
12.                 end
13.             end
14.             For each of the $M_Y$ blocks in the first X+4 block-columns of the block-row
15.                 For each of the (up to) P rows in the sub-row
16.                     Use the normalized min sum or offset min sum equation to adjust
                        the sign and value of the minimum or second minimum of this
                        parity check and to combine it with the corresponding LLR
                        loaded from the variable node RAM and the corresponding
                        LLR stored in the check node RAM, then use the resultant
                        value to overwrite the corresponding LLR in the variable node
                        RAM
17.                     Use the sign- and value-adjusted minimum or second minimum
                        of the parity check to overwrite the corresponding LLR in the
                        check node RAM
18.                 end
19.             end
20.         end
21.     end
22.     Determine whether early termination may be invoked
23. end
24. Output the ZX decoded systematic LLRs stored in the variable node RAM
```

3 Layered Belief Propagation LDPC decoder implementations

The implementation of a layered belief propagation LDPC decoder in an FPGA or ASIC may operate on the basis of a row-parallel architecture or a block-parallel architecture.

Row-parallel architecture

In this architecture, the Check Node Processor (CNP) of Figure 1 can perform all processing associated with all (up to) P rows of one sub-row of the PCM at the same time. More specifically, each iteration through steps 6 to 19 of the above-listed layered belief propagation algorithm is completed over a series of D clock cycles, where the pipeline depth D may have a value of 8, for example.

However, the processing of successive sub-rows in the same block-row may be pipelined, since there are no data dependencies between the processing of successive sub-rows in the same block-row. In this way, once the pipeline has been warmed up after D clock cycles, the processing of successive sub-rows completes in successive clock cycles, as shown in

```
function Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%CYCLES_PER_ITERATION_ROW_PARALLEL
%   This MATLAB function returns the number of clock cycles per iteration
%   for an LDPC decoder having a row-parallel architecture.
%
%   Syntax
%       Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%
%   Inputs Arguments
%       BG - 1 or 2
%       Y - Number of block-rows used in the PCM, depending on coding rate R
%       Z - LDPC lifting size
%       P - LDPC decoder parallelism
%       D - Pipeline depth
%
%   Output Arguments
%       Titer - Number of clock cycles per iteration

    % Beyond the first 20 block-rows in PCMs lifted from both BGs, each
    % pair of block-rows is orthogonal and can be processed at the same
    % time
    if Y <= 20
        Y2 = Y;
    else
        Y2 = ceil((Y-20)/2)+20;
    end

    % Assume D clock cycles to warm up the pipeline for each block-row and
    % then one sub-row processed per clock cycle thereafter.
    Titer = Y2*(ceil(Z/P)+D-1);
end
```

Figure 3.

By contrast, there are data dependencies between the processing of successive block-rows and so the processing of the last sub-row in one block-row cannot be pipelined with the processing of the first sub-row in the next block-row. This means that the pipeline must be warmed up again at the start of processing each block-row, as shown in

```
function Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%CYCLES_PER_ITERATION_ROW_PARALLEL
%     This MATLAB function returns the number of clock cycles per iteration
%     for an LDPC decoder having a row-parallel architecture.
%
%     Syntax
%             Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%
%     Inputs Arguments
%             BG - 1 or 2
%             Y - Number of block-rows used in the PCM, depending on coding rate R
%             Z - LDPC lifting size
%             P - LDPC decoder parallelism
%             D - Pipeline depth
%
%     Output Arguments
%             Titer - Number of clock cycles per iteration

    % Beyond the first 20 block-rows in PCMs lifted from both BGs, each
    % pair of block-rows is orthogonal and can be processed at the same
    % time
    if Y <= 20
        Y2 = Y;
    else
        Y2 = ceil((Y-20)/2)+20;
    end

    % Assume D clock cycles to warm up the pipeline for each block-row and
    % then one sub-row processed per clock cycle thereafter.
    Titer = Y2*(ceil(Z/P)+D-1);
end
```

Figure 3.

Note that after the first 20 block-rows in PCMs lifted from BG1 and BG2, the two block-rows in each successive pairing of block-rows are orthogonal to each other, as shown in Figure 1. More specifically, two block-rows are orthogonal to each other if there are no columns where they both have blocks. These pairings of orthogonal block-rows may be combined together and the row-parallel architecture may process them as if they were a single block-row. This reduces the number of loops around steps 4 to 21 of the above-listed algorithm required to complete each LDPC decoding iteration.

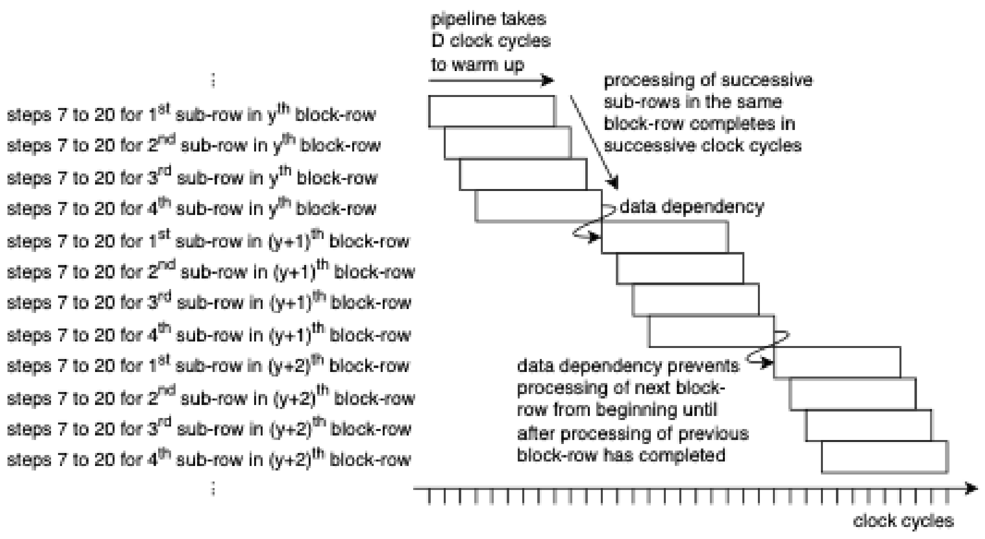

pipeline takes D clock cycles to warm up
processing of successive sub-rows in the same block-row completes in successive clock cycles
data dependency
data dependency prevents processing of next block-row from beginning until after processing of previous block-row has completed
steps 7 to 20 for 1st sub-row in yth block-row
steps 7 to 20 for 2nd sub-row in yth block-row
steps 7 to 20 for 3rd sub-row in yth block-row
steps 7 to 20 for 4th sub-row in yth block-row
steps 7 to 20 for 1st sub-row in (y+1)th block-row
steps 7 to 20 for 2nd sub-row in (y+1)th block-row
steps 7 to 20 for 3rd sub-row in (y+1)th block-row
steps 7 to 20 for 4th sub-row in (y+1)th block-row
steps 7 to 20 for 1st sub-row in (y+2)th block-row
steps 7 to 20 for 2nd sub-row in (y+2)th block-row
steps 7 to 20 for 3rd sub-row in (y+2)th block-row
steps 7 to 20 for 4th sub-row in (y+2)th block-row
clock cycles

Figure 2. Timing diagram for row-parallel architecture, for the case where there are ceil(Z/P)=4 sub-rows in each block-row.

Considering all of the above aspects, the number of clock cycles required to complete the processing of each LDPC decoding iteration using a row-parallel architecture may be calculated using the Matlab code of

```
function Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%CYCLES_PER_ITERATION_ROW_PARALLEL
%    This MATLAB function returns the number of clock cycles per iteration
%    for an LDPC decoder having a row-parallel architecture.
%
%    Syntax
%        Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%
%    Inputs Arguments
%        BG - 1 or 2
%        Y - Number of block-rows used in the PCM, depending on coding rate R
%        Z - LDPC lifting size
%        P - LDPC decoder parallelism
%        D - Pipeline depth
%
%    Output Arguments
%        Titer - Number of clock cycles per iteration

    % Beyond the first 20 block-rows in PCMs lifted from both BGs, each
    % pair of block-rows is orthogonal and can be processed at the same
    % time
    if Y <= 20
        Y2 = Y;
    else
        Y2 = ceil((Y-20)/2)+20;
    end

    % Assume D clock cycles to warm up the pipeline for each block-row and
    % then one sub-row processed per clock cycle thereafter.
    Titer = Y2*(ceil(Z/P)+D-1);
end
```

Figure 3.

```
function Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%CYCLES_PER_ITERATION_ROW_PARALLEL
%   This MATLAB function returns the number of clock cycles per iteration
%   for an LDPC decoder having a row-parallel architecture.
%
%   Syntax
%       Titer = cycles_per_iteration_row_parallel(BG, Y, Z, P, D)
%
%   Inputs Arguments
%       BG - 1 or 2
%       Y - Number of block-rows used in the PCM, depending on coding rate R
%       Z - LDPC lifting size
%       P - LDPC decoder parallelism
%       D - Pipeline depth
%
%   Output Arguments
%       Titer - Number of clock cycles per iteration

    % Beyond the first 20 block-rows in PCMs lifted from both BGs, each
    % pair of block-rows is orthogonal and can be processed at the same
    % time
    if Y <= 20
        Y2 = Y;
    else
        Y2 = ceil((Y-20)/2)+20;
    end

    % Assume D clock cycles to warm up the pipeline for each block-row and
    % then one sub-row processed per clock cycle thereafter.
    Titer = Y2*(ceil(Z/P)+D-1);
end
```

Figure 3. Matlab code for calculating T_{iter} for an LDPC decoder having a row-parallel architecture.

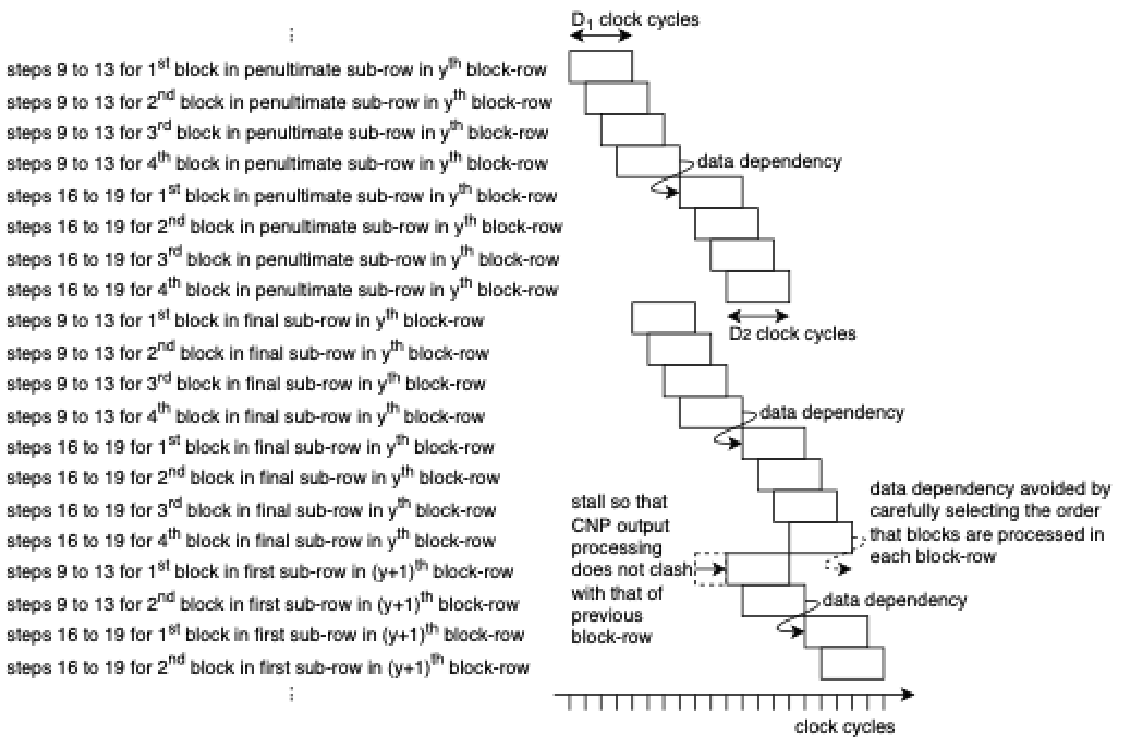
Block-parallel architecture

In this architecture, the CNP of Figure 1 can perform all processing associated with one block across all (up to) P rows of one sub-row of the PCM at the same time. More specifically, each iteration through the CNP input processing of steps 8 to 12 of the above-listed layered belief propagation algorithm is completed over a series of D_1 clock cycles, where the pipeline depth D_1 may have a value of 4, for example. Likewise, each iteration through the CNP output processing of steps 15 to 18 is completed over a series of D_2 clock cycles, where the pipeline depth D_2 may also have a value of 4, for example.

However, the CNP input processing of successive blocks in the same sub-row may be pipelined, since there are no data dependencies between this processing for successive blocks in the same sub-row. In this way, once the pipeline has been warmed up after D_1 clock cycles, the processing of successive sub-rows completes in successive clock cycles, as shown in Figure 5. Likewise, the CNP output processing of successive blocks in the same sub-row may be pipelined, with the processing of successive sub-rows completing in successive clock cycles after the pipeline has been warmed up after D_2 clock cycles. However, there are data dependencies between the CNP input and output processing and so all CNP input processing for all blocks in the sub-row must be completed, before the CNP output processing for any of the blocks in the sub-row can be started, as shown in Figure 5.

Furthermore, there are no data dependencies between the processing of different sub-rows in the same block-row. Owing to this, the CNP input processing of one sub-row can be tightly pipelined with the following CNP input processing of the next sub-row in the same block-row, as shown in Figure 5. Likewise, the CNP output processing of one sub-row can be tightly pipelined with the following CNP output processing of the next sub-row in the same block-row.

As described for the row-parallel architecture above, there are data dependencies between the processing of one block-row and the next. However, in the block-parallel architecture, we have freedom to choose to process the blocks in each block-row in any order. By carefully coordinating between the order that blocks are processed in one block-row and the next, we may ensure that all variable node RAMs are written to by the processing of one block-row before they are read by the processing of the next block-row, provided that the pipeline depths D_1 and D_2 are not excessively large. In this way, the CNP input processing of the last sub-row in one block-row may be tightly pipelined with the following CNP input processing of the first sub-row in the next block-row, as shown in Figure 5. Likewise, the CNP output processing of the last sub-row in one block-row may be tightly pipelined with the following CNP output processing of the first sub-row in the next block-row. For this reason, the throughput of the block-parallel decoder does not actually depend on the pipeline depths D_1 and D_2. However, in cases where a block-row contains more blocks than the next block-row, it is necessary to stall the processing of the first sub-row in the next block-row, in order to avoid a clash between its CNP output processing and that of the last sub-row in the block-row having more blocks, as shown in Figure 5.



steps 9 to 13 for 1st block in penultimate sub-row in yth block-row
steps 9 to 13 for 2nd block in penultimate sub-row in yth block-row
steps 9 to 13 for 3rd block in penultimate sub-row in yth block-row
steps 9 to 13 for 4th block in penultimate sub-row in yth block-row
steps 16 to 19 for 1st block in penultimate sub-row in yth block-row
steps 16 to 19 for 2nd block in penultimate sub-row in yth block-row
steps 16 to 19 for 3rd block in penultimate sub-row in yth block-row
steps 16 to 19 for 4th block in penultimate sub-row in yth block-row
steps 9 to 13 for 1st block in final sub-row in yth block-row
steps 9 to 13 for 2nd block in final sub-row in yth block-row
steps 9 to 13 for 3rd block in final sub-row in yth block-row
steps 9 to 13 for 4th block in final sub-row in yth block-row
steps 16 to 19 for 1st block in final sub-row in yth block-row
steps 16 to 19 for 2nd block in final sub-row in yth block-row
steps 16 to 19 for 3rd block in final sub-row in yth block-row
steps 16 to 19 for 4th block in final sub-row in yth block-row
steps 9 to 13 for 1st block in first sub-row in (y+1)th block-row
steps 9 to 13 for 2nd block in first sub-row in (y+1)th block-row
steps 16 to 19 for 1st block in first sub-row in (y+1)th block-row
steps 16 to 19 for 2nd block in first sub-row in (y+1)th block-row
D1 clock cycles
data dependency
D2 clock cycles
data dependency
stall so that CNP output processing does not clash with that of previous block-row
data dependency avoided by carefully selecting the order that blocks are processed in each block-row
data dependency
clock cycles

Figure 4. Timing diagram for block-parallel architecture, for the case where there are 4 blocks in the yth block-row and 2 blocks in the (y+1)th block-row.

Considering all of the above aspects, the number of clock cycles required to complete the processing of each LDPC decoding iteration using a block-parallel architecture may be calculated using the Matlab code of Figure 5.

```
function Titer = cycles_per_iteration_block_parallel(BG, Y, Z, P)
%CYCLES_PER_ITERATION_BLOCK_PARALLEL
%     This MATLAB function returns the number of clock cycles per iteration
%     for an LDPC decoder having a block-parallel architecture.
%
%     Syntax
%             Titer = cycles_per_iteration_block_parallel(BG, Y, Z, P)
%
%     Inputs Arguments
%             BG - 1 or 2
%             Y - Number of block-rows used in the PCM, depending on coding rate R
%             Z - LDPC lifting size
%             P - LDPC decoder parallelism
%
%     Output Arguments
%             Titer - Number of clock cycles per iteration

    % BG1 row weights
    My{1} = [19 19 19 19 2 7 8 6 9 8 6 7 6 5 6 6 5 5 5 5 5 4 4 5 4 4 ...
                3 4 4 4 4 4 4 4 4 4 3 4 4 3 4 3 4 4 3];

    % BG2 row weights
    My{2} = [8 10 8 10 3 5 5 5 3 4 4 4 3 4 4 3 4 4 3 3 3 3 2 3 3 2 4 2 ...
                3 2 4 2 3 3 3 3 3 2 3 3 3 3];

    % Extract the weights for the first Y block-rows of the BG and then
    % repeat each one for each of its ceil(Z/P) sub-rows
    row_weights = My{BG}(1:Y);
    subrow_weights = repelem(row_weights, ceil(Z/P));
    % Clock cycles are needed to read from VN RAM into the CNP and then

    % clock cycles are needed to write from CNP into VN RAM. Assume that

    % the writing of one sub-row can overlap with the reading of the next

    % sub-row. Assume that careful scheduling of which columns to write

    % first and which columns to read first can be achieved to satisfy all

    % data dependencies.

    Titer = sum(max(subrow_weights,[0,subrow_weights(1:end-1)]));

end
```

Figure 5. Matlab code for calculating T_{iter} for an LDPC decoder having a block-parallel architecture.

Chip area

The chip area of an ASIC implementation of an LDPC decoder (or equivalently the hardware resource utilisation of an FPGA implementation of an LDPC decoder) comprises two parts, RAM and logic, where the latter comprises flip-flops, gates, look-up tables, etc.

As shown in Figure 1, the RAM capacity of a layered belief propagation LDPC decoder is proportional to Z_{max}. Hence, doubling the Z_{max} of an LDPC decoder will double the chip area associated with RAM.

Observation 1: The chip area associated with the RAM of a layered belief LDPC decoder is proportional to its maximum supported lifting size Z_{max}.

By contrast, the chip area associated with the logic of a layered belief propagation LDPC decoder is proportional to P.

Observation 2: The chip area associated with the computational logic of a layered belief LDPC decoder is proportional to its parallelism P.

It is typical for the chip area associated with RAM to exceed that associated with logic in an ASIC implementation of a layered belief propagation LDPC decoder. Hence, ASIC implementations are particularly sensitive to the selection of Z_{max}.

By contrast, the fraction of an FPGA's logic resources that are occupied by the implementation of a layered belief propagation LDPC decoder typically exceeds the fraction of the FPGA's RAM resources that are occupied. Hence, FPGA implementations are particularly sensitive to the selection of P.

Architecture variations

The row-parallel and block-parallel architectures described above may be considered to be two ends of a spectrum of approaches to parallelism in the implementation of a layered belief propagation LDPC decoder. A hybrid of the two architectures in the middle of the spectrum may be implemented as a multi-block-parallel architecture, which may operate in a similar manner to the block-parallel architecture, but for two or more blocks at a time.

Furthermore, multi-code-block variants of both architectures may be implemented. In the case of the row-parallel architecture, a round-robin approach to processing more than one code block at a time may be adopted. More specifically, this allows the processing of one block-row of one code block to complete while the pipeline for a block-row of another code block is warming up, effectively removing the D-1 term from the calculation of T_{iter}. Here, it may be desirable to limit multi-code-block operation to cases where the sum of the lifting sizes of the different code block remains below Z_{max}, such that they can all be processed without a need for additional RAM. In the case of the block-parallel architecture, multiple code blocks may be processed at the same time in cases where the sum of their lifting sizes Z is does not exceed the parallelism P of the decoder.

While these architecture variations are not considered further in this contribution, it may be expected that the observations listed below remain valid for them.

4 Impact of increasing Z_{max}

In this section, we explore the impact of increasing Z_{max} with the aim of supporting a peak LDPC decoder throughput of 40 Gbps, which is double the 20 Gbps target of 5G NR.

Table 2 compares various different row-parallel (having a pipeline delay of D=8 clock cycles) and block-parallel layered belief propagation LDPC decoder implementations. The peak throughput for each of these implementations is obtained in the case of BG1, a code rate of R=0.92 and an information block length of K=8448. Table 2 quantifies the peak throughput for the various different implementations for the case of an ASIC clock frequency of f=1.5GHz and a maximum of I=8 iterations, according to

$$Throughput = \frac{K \times f}{I \times T_{iter}} \times C$$

where T_{iter} is calculated using the corresponding Matlab code listing provided above.

Architecture	**C**	**P**	**Z_{max}**	**RAM area**	**Logic area**	**Peak throughput (Gbps) for f=1.5GHz and I=8**
Row-parallel with D=8	1	32	384	A_1	A_2	20.8
Row-parallel with D=8	1	64	384	A_1	$2A_2$	30.5
Row-parallel with D=8	1	64	768	$2A_1$	$2A_2$	41.7
Row-parallel with D=8	2	32	384	$2A_1$	$2A_2$	41.7
Block-parallel	1	384	384	A_1	A_3	20.8
Block-parallel	1	768	384	A_1	$2A_3$	20.8
Block-parallel	1	768	768	$2A_1$	$2A_3$	41.7
Block-parallel	2	384	384	$2A_1$	$2A_3$	41.7

Table 2. Comparison of different layered belief propagation LDPC decoder implementations.

Table 2 shows that the 20 Gbps throughput requirement of 5G NR is met by C=1 instance of a row-parallel implementation having a parallelism of P=32 or by C=1 instance of a block-parallel implementation having a parallelism of P=384, when using the maximum lifting size from 5G NR of Z_{max}=384. These implementations use a baseline RAM area represented by A_1 and baseline logic areas represented by A_2 for the row-parallel implementation and A_3 for the block-parallel implementation. However, enhancement of these baseline implementations is required in order to meet the 40 Gbps throughput requirement of 6GR.

Simply doubling the parallelism P of the two baseline implementations is not sufficient for meeting the 40 Gbps throughput requirement of 6GR. More specifically, C=1 instance of a P=64 row-parallel implementation achieves a peak throughput of only 30.5 Gbps, with a RAM area maintained at A_1 and a doubled logic area of $2A_2$. Furthermore, C=1 instance of a P=768 block-parallel implementation achieves no increase in peak throughput at all, since its parallelism P exceeds the Z_{max}=384 of the 5G NR LDPC code. Again, a maintained RAM area of A_1 and a doubled logic area of $2A_3$ is obtained when doubling the parallelism P of the block-parallel implementation.

In order to meet the 40 Gbps throughput requirement of 6GR by doubling the parallelism P of the baseline implementations, it is necessary to also double the maximum lifting size to Z_{max}=768. However, this doubles the RAM area of both implementations to $2A_1$, alongside the doubling of logic areas to $2A_2$ and $2A_3$ associated with the doubling of the parallelism P.

Observation 3: Increasing the parallelism P of an LDPC decoder implementation only achieves a proportionally increased peak throughput if the maximum lifting size Z_{max} is also increased proportionately.

Observation 4: Increasing the parallelism P and maximum lifting size Z_{max} of an LDPC decoder by the same proportion will also increase its chip area by that proportion.

Alternatively, the same peak throughputs exceeding 40 Gbps can be obtained by simply using C=2 instances of the baseline implementations. This approach also uses the same doubled RAM area of $2A_1$, alongside the same doubled logic areas of $2A_2$ and $2A_3$.

Observation 5: The same increase in peak throughput and chip area can be achieved by simply using a proportionately increased number of instances C of the baseline LDPC decoder implementation.

Figure 6 plots the throughput of the LDPC decoder implementations from Table 2 as functions of coding rate R. These plots show that for both BG1 and BG2, as well as for both the row-parallel implementation and the block-parallel implementation, C=2 instances of a baseline implementation achieves the same throughput as C=1 instance having doubled parallelism P and doubled Z_{max}, across all coding rates R. However, it should be noted that this comparison is made at the longest supported information block lengths K, namely K=8448 for BG1 with Z_{max}=384, K=16896 for BG1 with Z_{max}=768, K=3840 for BG2 with Z_{max}=384 and K=7680 for BG2 with Z_{max}=768.

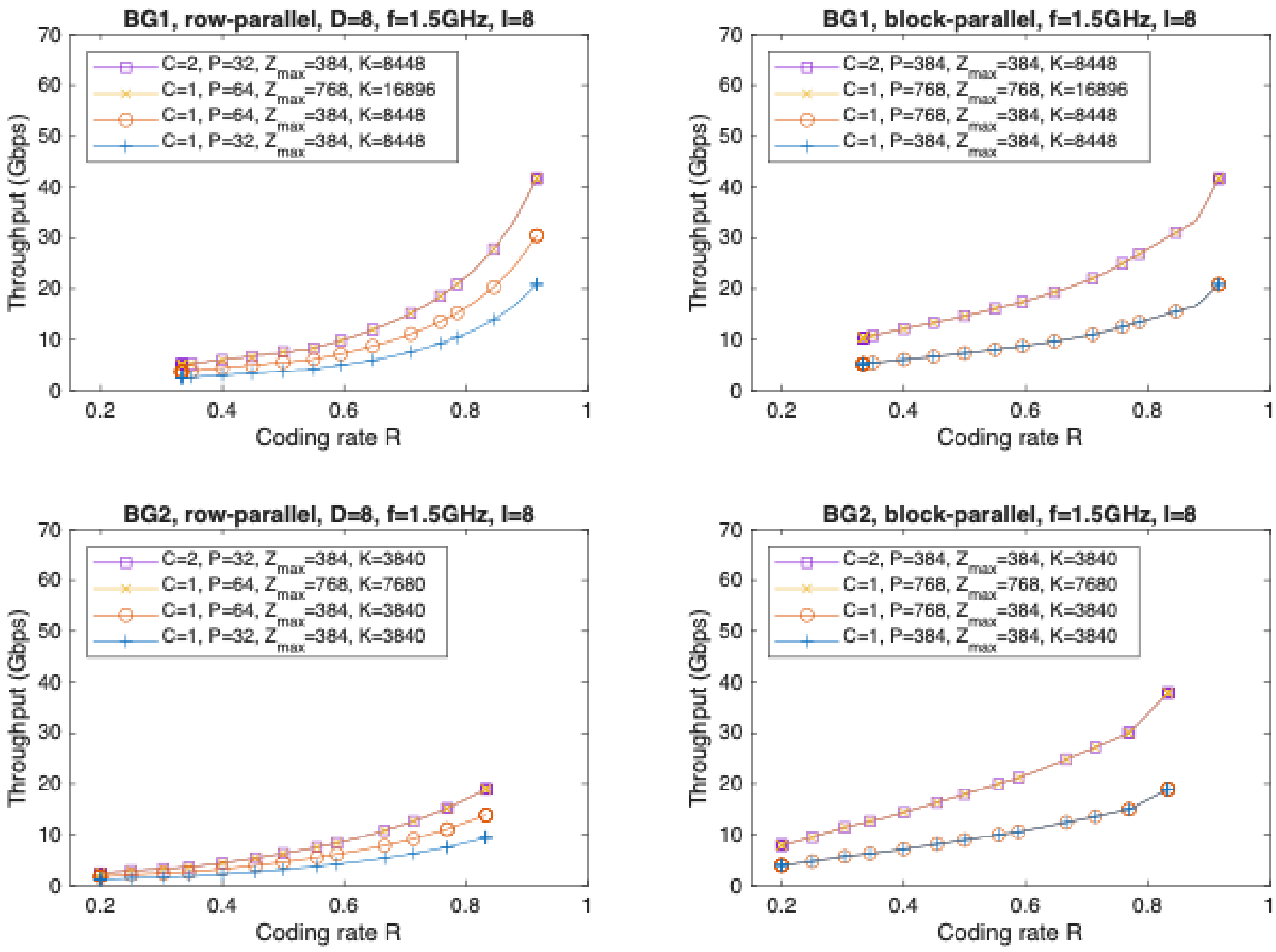


Figure 6. Throughput versus coding rate R for different layered belief propagation LDPC decoder implementations.

Figure 7 plots the throughput of the LDPC decoder implementations from Table 2 as functions of information block length K, at coding rates of 0.92 for BG1 and 0.83 for BG2. These plots illustrate how doubling the maximum lifting size to Z_{max}=768 enables support for longer information block lengths, which has the benefit of improving BLER performance. However, Figure 7 also shows that within the range of information block lengths supported at the baseline maximum lifting size of Z_{max}=384, C=2 instances of the baseline LDPC decoder implementations achieve higher throughputs than C=1 instance of the implementations having doubled Z_{max} and parallelism P. This may be attributed to the effect of pipeline warmup in the case of the row-parallel implementation and to the inability for a high parallelism P to be exploited when the lifting size Z is lower, in the case of the block-parallel implementation. The higher throughput of the C=2, Z_{max}=384 implementation is a particular advantage in the case of a gNodeB processing a high number of short code blocks in a direct-to-device NTN application, for example.

Observation 6: Increasing the maximum lifting size Z_{max} enables support for longer information block lengths K, where improved BLER performance is achieved.

Observation 7: At information block lengths within the range supported by the baseline maximum lifting size of Z_{max}=384, multiple instances of a baseline LDPC decoder implementation achieves a significantly higher throughput than a single instance of an LDPC decoder implementation having a proportionately increased Z_{max} and parallelism P.

Proposal 1: Do not increase the maximum lifting size of the 5G NR LDPC code above Z_{max}=384, unless this is deemed to be necessary for achieving BLER performance improvement.

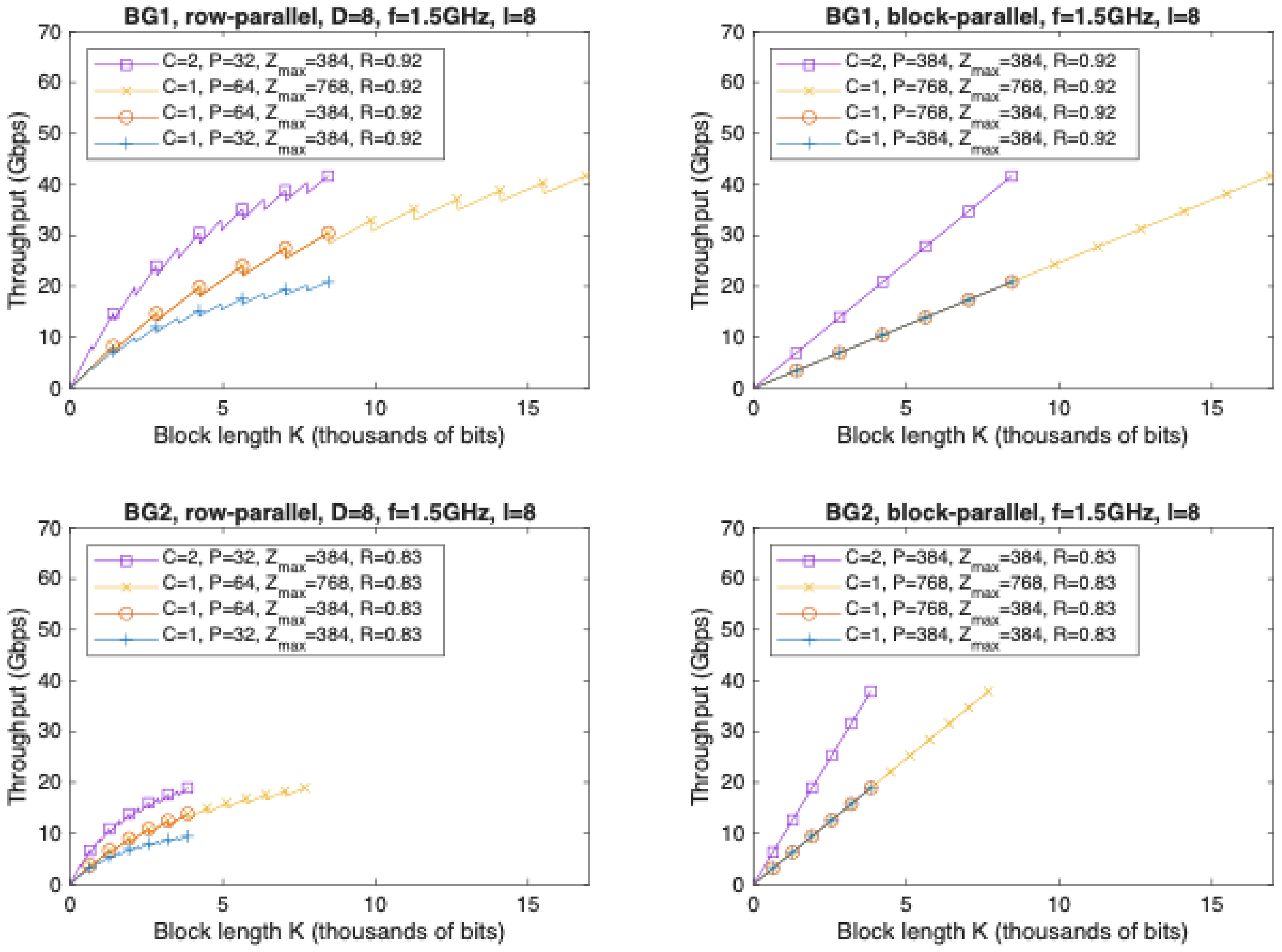


Figure 7. Throughput versus information block length K for different layered belief propagation LDPC decoder implementations.

5 Conclusion

In this contribution, AccelerComms's views on LDPC decoder throughput and chip area were shown, and the following observation and proposals were made:

Observation 1: The chip area associated with the RAM of a layered belief LDPC decoder is proportional to its maximum supported lifting size Z_{max}.

Observation 2: The chip area associated with the computational logic of a layered belief LDPC decoder is proportional to its parallelism P.

Observation 3: Increasing the parallelism P of an LDPC decoder implementation only achieves a proportionally increased peak throughput if the maximum lifting size Z_{max} is also increased proportionately.

Observation 4: Increasing the parallelism P and maximum lifting size Z_{max} of an LDPC decoder by the same proportion will also increase its chip area by that proportion.

Observation 5: The same increase in peak throughput and chip area can be achieved by simply using a proportionately increased number of instances C of the baseline LDPC decoder implementation.

Observation 6: Increasing the maximum lifting size Z_{max} enables support for longer information block lengths K, where improved BLER performance is achieved.

Observation 7: At information block lengths within the range supported by the baseline maximum lifting size of Z_{max}=384, multiple instances of a baseline LDPC decoder implementation achieves a significantly higher throughput than a single instance of an LDPC decoder implementation having a proportionately increased Z_{max} and parallelism P.

Proposal 1. Do not increase the maximum lifting size of the 5G NR LDPC code above Z_{max}=384, unless this is deemed to be necessary for achieving BLER performance improvement.

6 References

[1] Chair notes RAN1#122bis.

[2] R1-2508176, “Final FL summary for 6G channel coding,” ZTE, Apple (Moderator).

[3] TS 38.212