LDPC Decoding: VLSI Architectures and Implementations

Module 2: VLSI Architectures and Implementations

Kiran Gunnam

NVIDIA

kgunnam@ieee.org

Outline

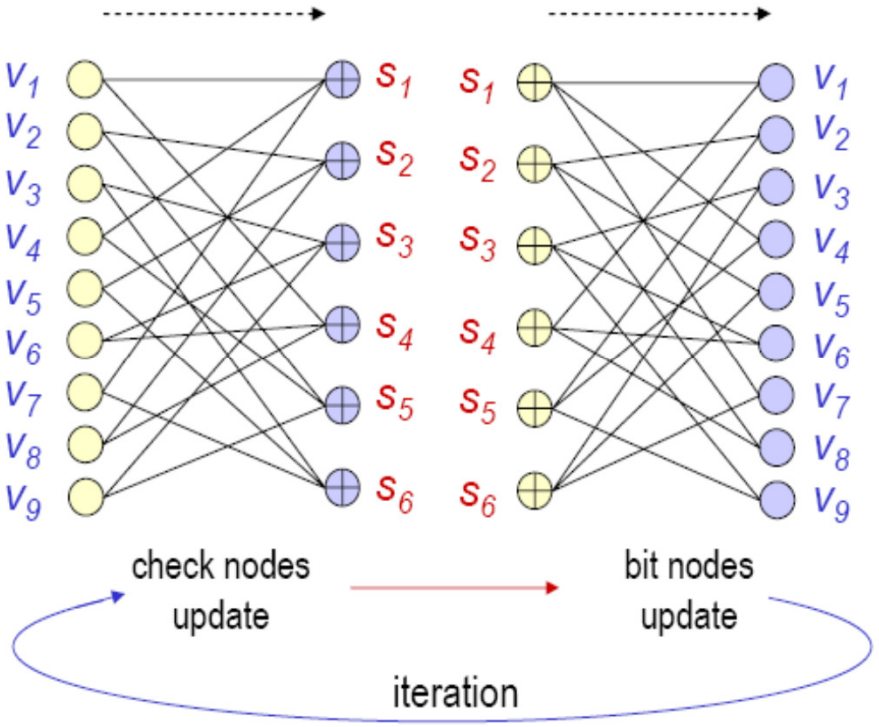
- Check Node Unit Design
- Non-layered Decoder Architecture
  - Block Serial Processing
  - From Throughput requirements to design specifications
- Layered Decoder Architecture
  - Block Serial Processing
  - Block Serial Processing and scheduling for Irregular H matrices
  - Block Parallel Processing
  - From Throughput requirements to design specifications
- Case Study of Decoders for 802.11n and Flash Channel

LDPC Decoding, Quick Recap 1/5

$$H = \begin{bmatrix} 1\,0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,1\,0\,1\,0\,0\,0\,0\,1 \\ 0\,0\,1\,0\,1\,0\,1\,0\,0 \end{bmatrix} \begin{matrix} \text{check} \\ \text{nodes} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{matrix}$$

$v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9$

variable nodes



Bit nodes (also called variable nodes) correspond to received bits.

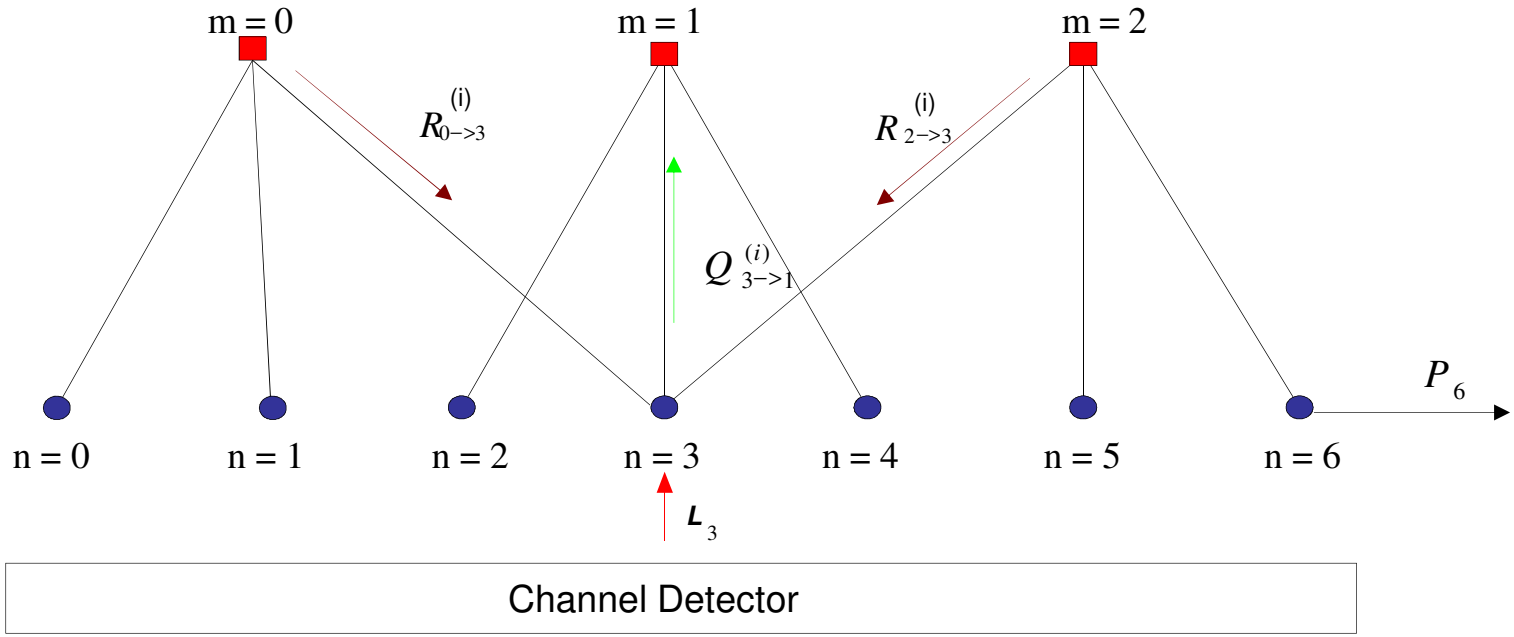
Check nodes describe the parity equations of the transmitted bits.

eg. v1+v4+v7= 0; v2+v5+v8 =0 and so on.

The decoding is successful when all the parity checks are satisfied (i.e. zero).

LDPC Decoding, Quick Recap 2/5

- There are four types of LLR messages
  - Message from the channel to the n-th bit node, L_n
  - Message from n-th bit node to the m-th check node $Q^{(i)}_{n->m}$ or simply $Q^{(i)}_{nm}$
  - Message from the m-th check node to the n-th bit node $R^{(i)}_{m->n}$ or simply $R^{(i)}_{mn}$
  - Overall reliability information for n-th bit-node P_n



LDPC Decoding, Quick Recap 3/5

Notation used in the equations

x_n is the transmitted bit n ,

L_n is the initial LLR message for a bit node (also called as variable node) n , received from channel/detector

P_n is the overall LLR message for a bit node n ,

$\hat{x}_n$ is the decoded bit n (hard decision based on P_n) ,

[Frequency of P and hard decision update depends on decoding schedule]

$\mathrm{M}(n)$ is the set of the neighboring check nodes for variable node n ,

$\mathrm{N}(m)$ is the set of the neighboring bit nodes for check node m .

For the i^{th} iteration,

$Q_{nm}^{(i)}$ is the LLR message from bit node n to check node m ,

$R_{mn}^{(i)}$ is the LLR message from check node m to bit node n .

LDPC Decoding, Quick Recap 4/5

(A) check node processing: for each m and $n \in \mathrm{N}(m)$,

$$R_{mn}^{(i)} = \delta_{mn}^{(i)} \kappa_{mn}^{(i)} \qquad (1)$$

$$\kappa_{mn}^{(i)} = \left| R_{mn}^{(i)} \right| = \min_{n' \in \mathrm{N}(m) \setminus n} \left| Q_{n'm}^{(i-1)} \right| \qquad (2)$$

The sign of check node message $R_{mn}^{(i)}$ is defined as

$$\delta_{mn}^{(i)} = \left(\prod_{n' \in \mathrm{N}(m) \setminus n} \mathrm{sgn}\left(Q_{n'm}^{(i-1)} \right) \right) \qquad (3)$$

where $\delta_{mn}^{(i)}$ takes value of $+1$ or -1

LDPC Decoding, Quick Recap 5/5

(B) *Variable-node processing*: for each n and $m \in M(n)$:

$$Q_{nm}^{(i)} = L_n + \sum_{m' \in \mathrm{M}(n) \backslash m} R_{m'n}^{(i)} \qquad (4)$$

(C) P Update and Hard Decision

$$P_n = L_n + \sum_{m \in M(n)} R_{mn}^{(i)} \qquad (5)$$

A hard decision is taken where $\hat{x}_n = 0$ if $P_n \geq 0$, and $\hat{x}_n = 1$ if $P_n < 0$.

If $\hat{x}_n H^T = 0$, the decoding process is finished with $\hat{x}_n$ as the decoder output; otherwise, repeat steps (A) to (C).

If the decoding process doesn't end within some maximum iteration, stop and output error message.

Scaling or offset can be applied on R messages and/or Q messages for better performance.

On-the-fly Computation

Our previous research introduced the following concepts to LDPC decoder implementation

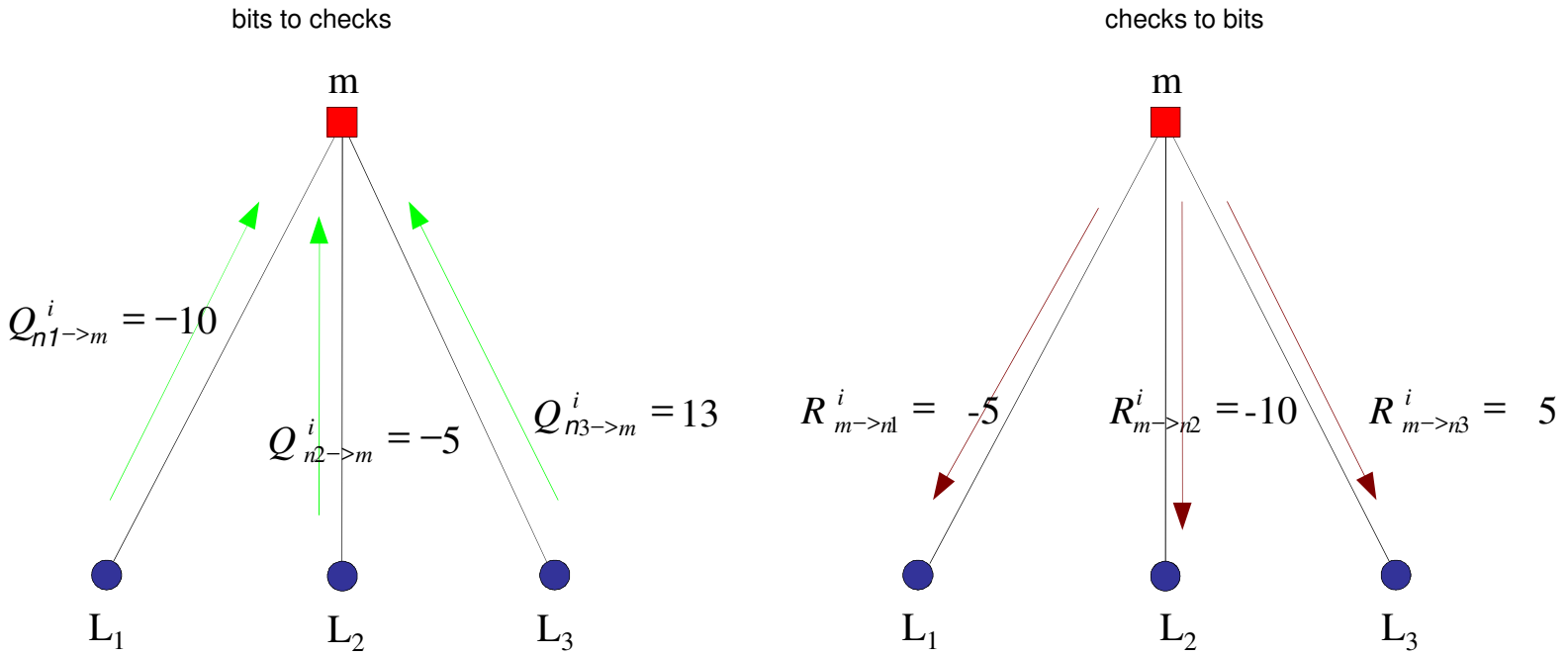
[1-10] presented at various IEEE conferences. ICASSP'04,Asilomar'06,VLSI'07,ISWPC'07, ISCAS'07, ICC'07, Asilomar'08. References P1 and P2 are more comprehensive and are the basis for this presentation.

1. Block serial scheduling
2. Value-reuse,
3. Scheduling of layered processing,
4. Out-of-order block processing,
5. Master-slave router,
6. Dynamic state,
7. Speculative Computation
8. Run-time Application Compiler [support for different LDPC codes with in a class of codes. Class:802.11n,802.16e,Array, etc. Off-line re-configurable for several regular and irregular LDPC codes]

All these concepts are termed as On-the-fly computation as the core of these concepts are based on minimizing memory and re-computations by employing just in-time scheduling. For this presentation, we will focus on concepts 1-4.

Check Node Unit (CNU) Design

$$\kappa_{m\to l}^{(i)} = \left|R_{m\to l}^{(i)}\right| = \min_{l' \in \mathrm{N}(m)\setminus l} \left|Q_{l'\to m}^{(i-1)}\right|$$



Check Node Unit (CNU) Design

$$\kappa_{mn}^{(i)} = \left|R_{mn}^{(i)}\right| = \min_{n' \in \mathrm{N}(m) \setminus n} \left|Q_{n'm}^{(i-1)}\right| \qquad (2)$$

The above equation (2) can be reformulated as the following set of equations.

$$M1_m^{(i)} = \min_{n' \in \mathrm{N}(m)} \left|Q_{mn'}^{(i-1)}\right| \qquad (6)$$

$$M2_m^{(i)} = 2nd\min_{n' \in \mathrm{N}(m)} \left|Q_{mn'}^{(i-1)}\right| \qquad (7)$$

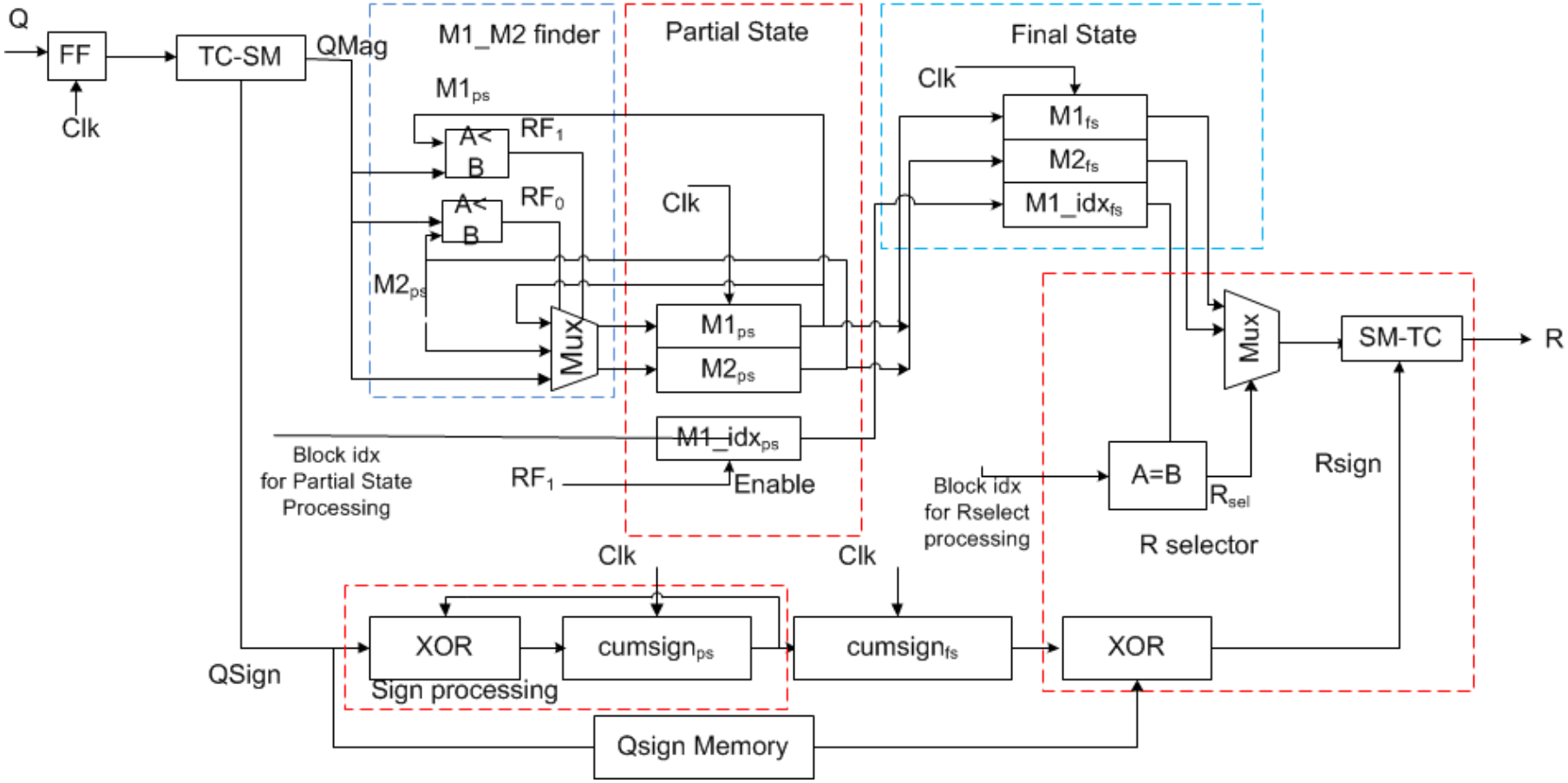
$$k = Min1Index \qquad (8)$$

$$\kappa_{mn}^{(i)} = M1_m^{(i)}, \ \forall n \in \mathrm{N}(m) \setminus k$$

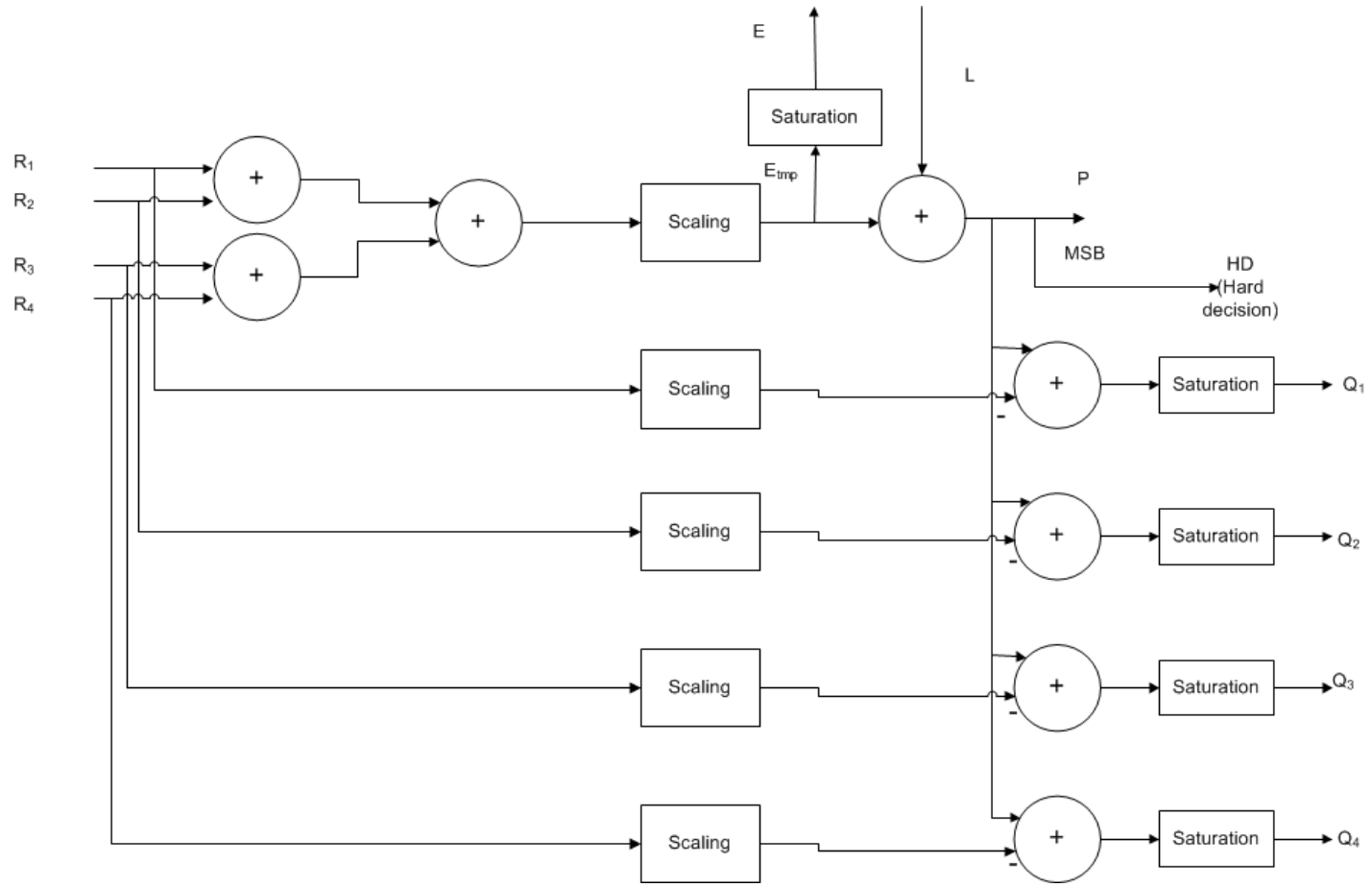
$$= M2_m^{(i)}, n = k \qquad (9)$$

- Simplifies the number of comparisons required as well as the memory needed to store CNU outputs.
- Additional possible improvements: The correction has to be applied to only two values instead of distinct values. Need to apply 2's complement to only 1 or 2 values instead of values at the output of CNU.

CNU Micro Architecture for min-sum

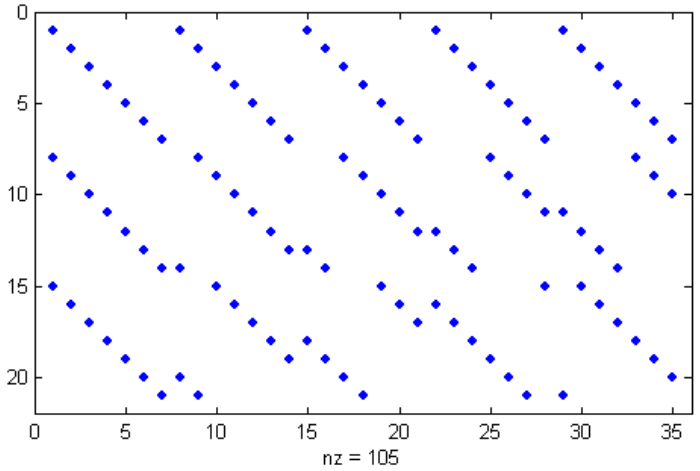


VNU Micro Architecture



Example QC-LDPC Matrix

$$H = \begin{bmatrix} I & I & I & \dots & I \\ I & \sigma & \sigma^2 & \dots & \sigma^{r-1} \\ I & \sigma^2 & \sigma^4 & \dots & \sigma^{2(r-1)} \\ : & & & & \\ I & \sigma^{c-1} & \sigma^{(c-1)2} & \dots & \sigma^{(c-1)(r-1)} \end{bmatrix}$$



$$\sigma = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & .0 & 0 \\ : & & & & \\ 0 & 0 & \dots & .1 & 0 \end{bmatrix}$$

Example H Matrix, Array LDPC

r row/ check node degree=5

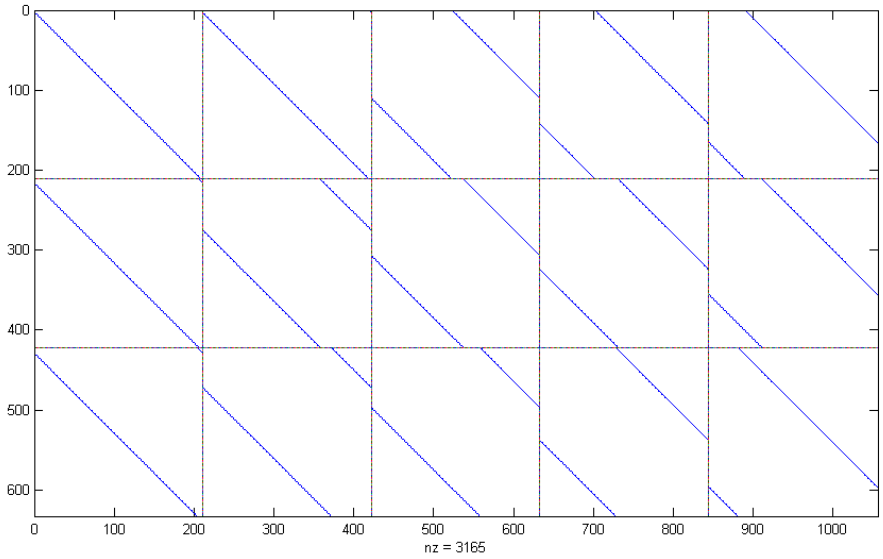
c columns/variable node degree=3

Sc circulant size=7

*N=Sc*r=35*

Example QC-LDPC Matrix

$$S_{3\times5} = \begin{bmatrix} 2 & 3 & 110 & 142 & 165 \\ 5 & 64 & 96 & 113 & 144 \\ 7 & 50 & 75 & 116 & 174 \end{bmatrix}$$



Example H Matrix

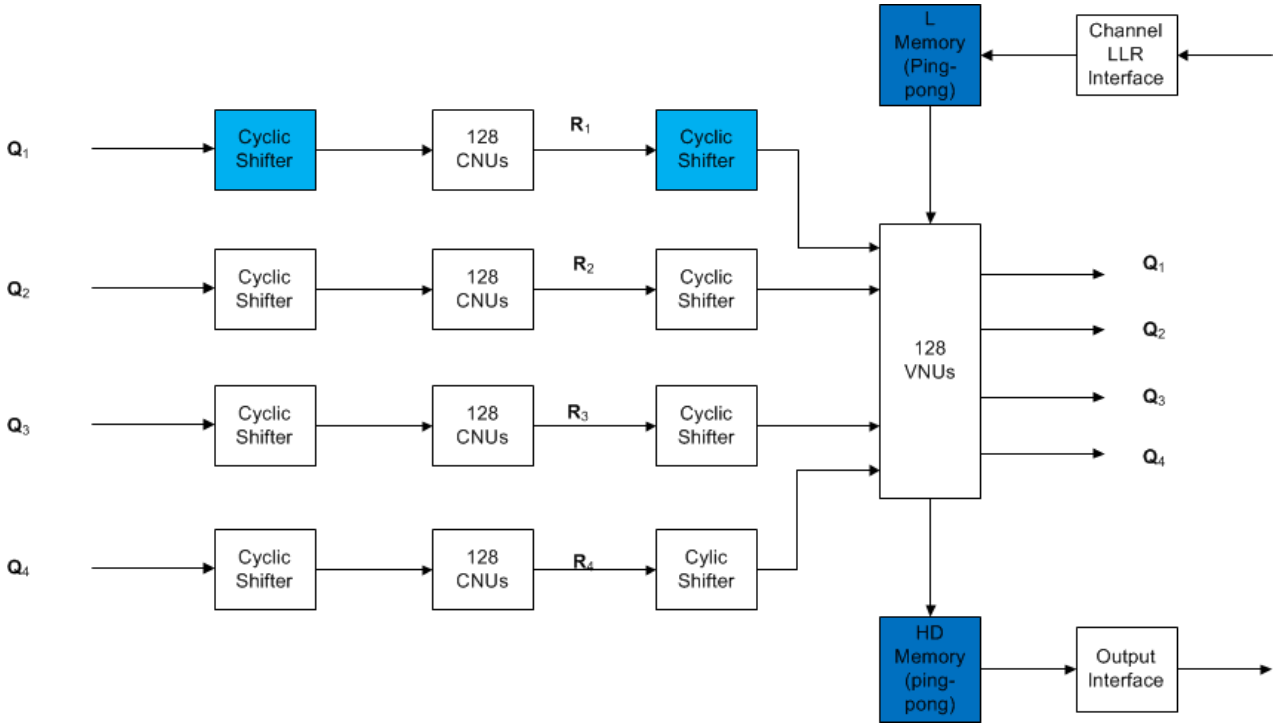
r row degree=5

c column degree =3

Sc circulant size=211

*N=Sc*r=1055*

Non-Layered Decoder Architecture



Supported H matrix parameters

r row degree=36

c column degree =4

Sc ciruclant size=128

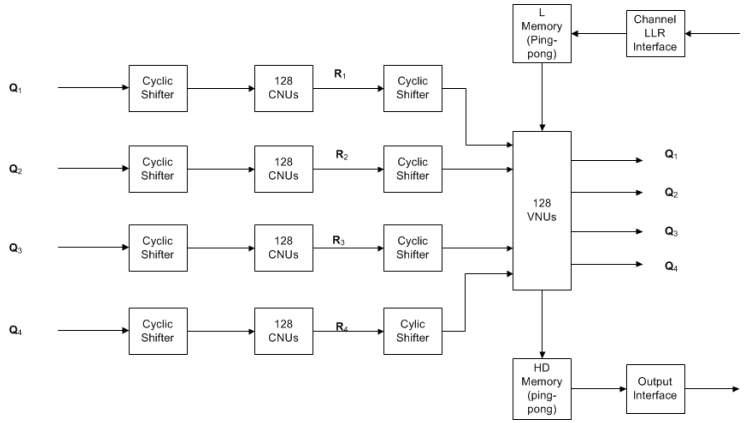
*N=Sc*r=4608*

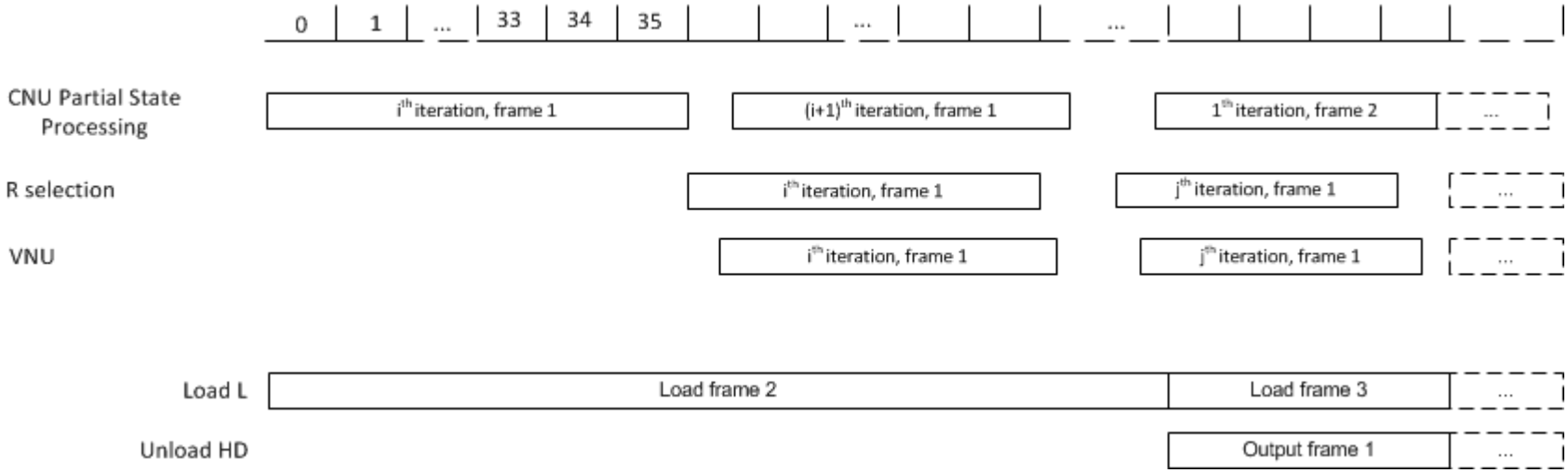
*L Memory=> Depth 36, Width=128*5*

HD Memory=> Depth 36, Width=128

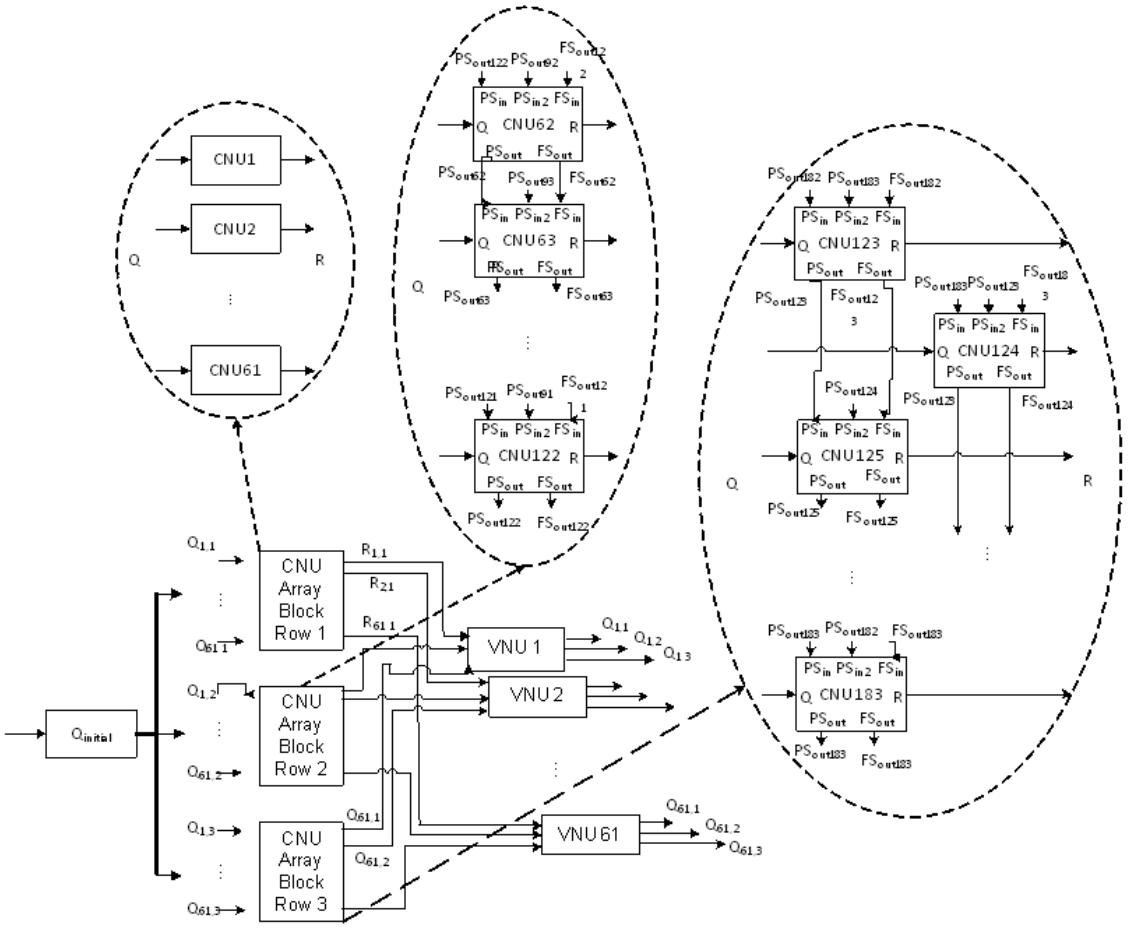
Possible to remove the shifters (light-blue) by re-arranging H matrix's first layer to have zero shift coefficients.

Pipeline for NLD





Non-Layered Decoder Architecture for Array LDPC codes



Supported H matrix parameters

r row degree=32

c column degree =3

Sc ciruclant size=61

*N=Sc*r=1952*

From Throughput Requirements to Design Specification

- Requirements
  - Throughput in bits per sec.
  - BER
  - Latency
- BER would dictiate Number of Iterations and degree profile(check node degrees and variable node degrees).
- Circulant Size (Sc)
- Number of Columns processed in one clock (Nc)
- Number of bits processed per clock=Throughput/clock frequency
- Sc*Nc=Nb*Iterations
- Sc is usually set to less than 128 for smaller router.

Layered Decoder Architecture

Optimized Layered Decoding with algorithm transformations for reduced memory and computations

$\vec{R}_{l,n}^{(0)} = 0, \vec{P}_n = \vec{L}_n^{(0)}$ [Initialization for each new received data frame], (9)

$\forall i = 1,2,\cdots,it_{\max}$ [Iteration loop],

$\forall l = 1,2,\cdots,j$ [Sub-iteration loop],

$\forall n = 1,2,\cdots,k$ [Block column loop],

$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)}, \quad (10)$$

$$\vec{R}_{l,n}^{(i)} = f\left(\left[\vec{Q}_{l,n'}^{(i)}\right]^{S(l,n')}, \forall n' = 1,2,\cdots,k\right), \quad (11)$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(i)}, \quad (12)$$

where the vectors $\vec{R}_{l,n}^{(i)}$ and $\vec{Q}_{l,n}^{(i)}$ represent all the R and Q messages in each $p \times p$ block of the H matrix, $s(l,n)$ denotes the shift coefficient for the block in l^{th} block row and n^{th} block column of the H matrix. $\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)}$ denotes that the vector $\vec{Q}_{l,n}^{(i)}$ is cyclically shifted up by the amount $s(l,n)$

k is the check-node degree of the block row.

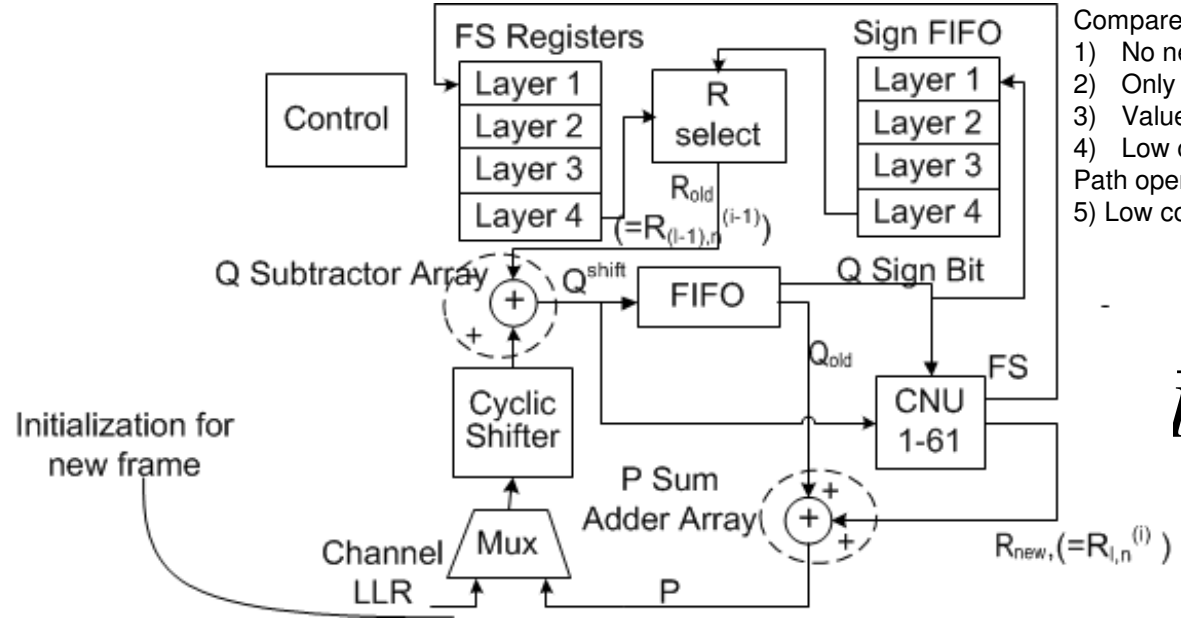
A negative sign on $s(l,n)$ indicates that it is a cyclic down shift (equivalent cyclic left shift).

$f(\cdot)$ denotes the check-node processing, which embodiments implement using, for example, a Bahl-Cocke-Jelinek-Raviv algorithm ("BCJR") or sum-of-products ("SP") or Min-Sum with scaling/offset.

Layered Decoder Architecture

Our work proposed this for H matrices with regular mother matrices.

Compared to other work, this work has several advantages
1) No need of separate memory for P.
2) Only one shifter instead of 2 shifters
3) Value-reuse is effectively used for both Rnew and Rold
4) Low complexity data path design-with no redundant data Path operations.
5) Low complexity CNU design.



$$\vec{R}_{l,n}^{(i)} = f\left(\begin{bmatrix}\vec{Q}_{l,n'}^{(i)}\end{bmatrix}^{S(l,n')}, \quad \forall n' = 1,2,\cdots,k\right)$$

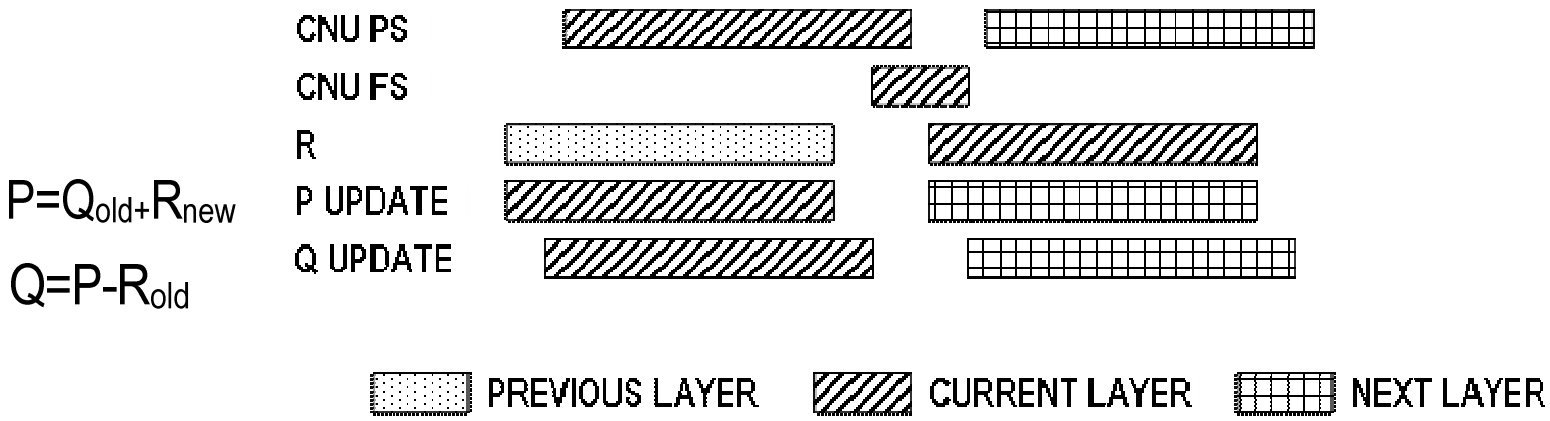
$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)}$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(i)}$$

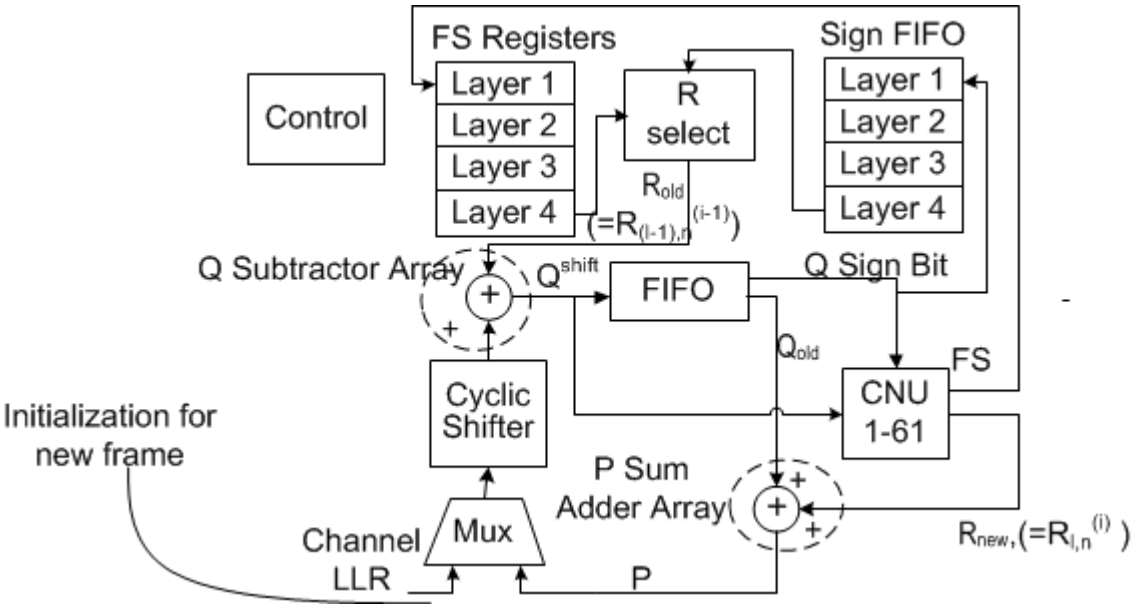
$Q=P-R_{old}$

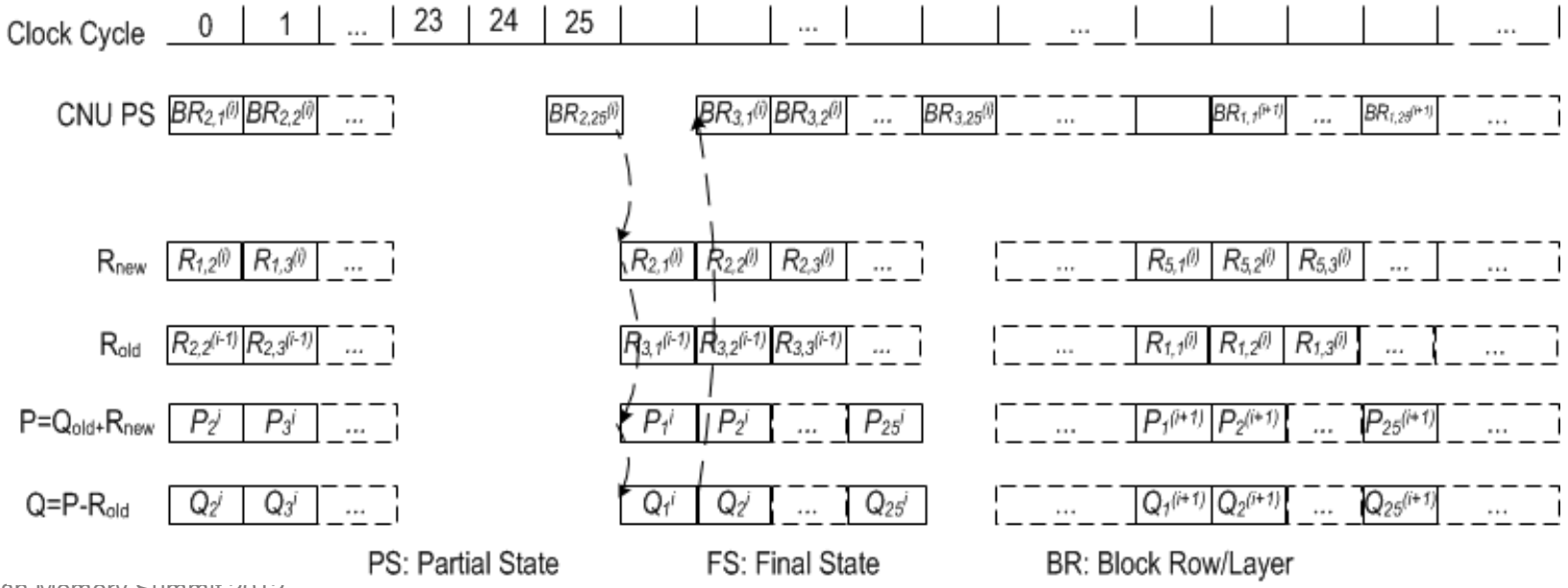
$P=Q_{old}+R_{new}$

Data Flow Diagram



Data Flow





Parameters used here: *r row degree=25, c column degree =3*

Irregular QC-LDPC H Matrices

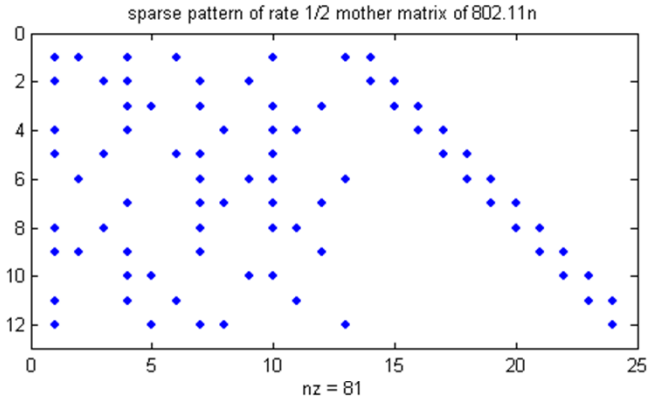
$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,n_b-2} & \mathbf{P}_{1,n_b-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{P}_{m_b-1,0} & \mathbf{P}_{m_b-1,1} & \mathbf{P}_{m_b-1,2} & \cdots & \mathbf{P}_{m_b-1,n_b-2} & \mathbf{P}_{m_b-1,n_b-1} \end{bmatrix} = \mathbf{P}^{H_b}$$

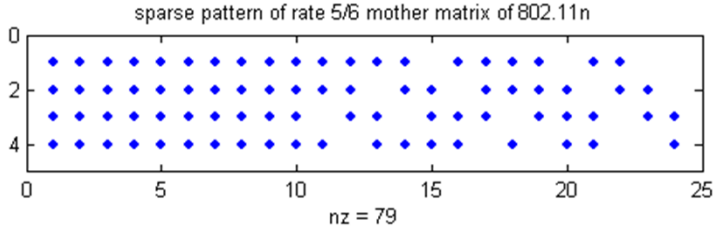
Different base matrices to support different rates.

Different expansion factors (z) to support multiple lengths.

All the shift coefficients for different codes for a given rate are obtained from the same base matrix using modulo arithmetic

Irregular QC-LDPC H Matrices



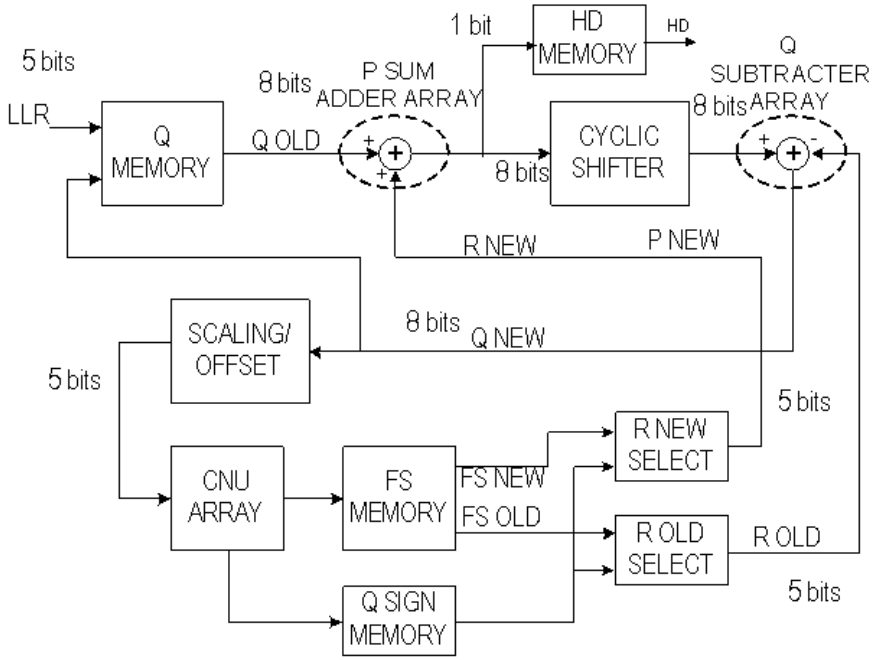


Irregular QC-LDPC H Matrices

- Existing implementations show that these are more complex to implement.
- These codes have the better BER performance and selected for IEEE 802.16e and IEEE 802.11n.
- It is anticipated that these type of codes will be the default choice for most of the standards.
- We show that with out-of-order processing and scheduling of layered processing, it is possible to design very efficient architectures.
- The same type of codes can be used in storage applications (holographic, flash and magnetic recording) if variable node degrees of 2 and 3 are avoided in the code construction for low error floor

Hocevar, D.E., "A reduced complexity decoder architecture via layered decoding of LDPC codes," IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004. .pp. 107- 112, 13-15 Oct. 2004

Layered Decoder Architecture



Our work proposed this for H matrices with irregular mother matrices.

Compared to other work, this work has several advantages

1) Q memory (some times we call this as LPQ memory) can be used to store L/Q/P instead of 3 separate memories- memory is managed at circulant level as at any time for a given circulant we need only L or Q or P.
2) Only one shifter instead of 2 shifters
3) Value-reuse is effectively used for both Rnew and Rold
4) Low complexity data path design-with no redundant data

Path operations.

5) Low complexity CNU design.
6) Out-of-order processing at both layer and circulant level for all the processing steps such as Rnew and PS processing to eliminate the pipeline and memory access stall cycles.

Data Flow Diagram

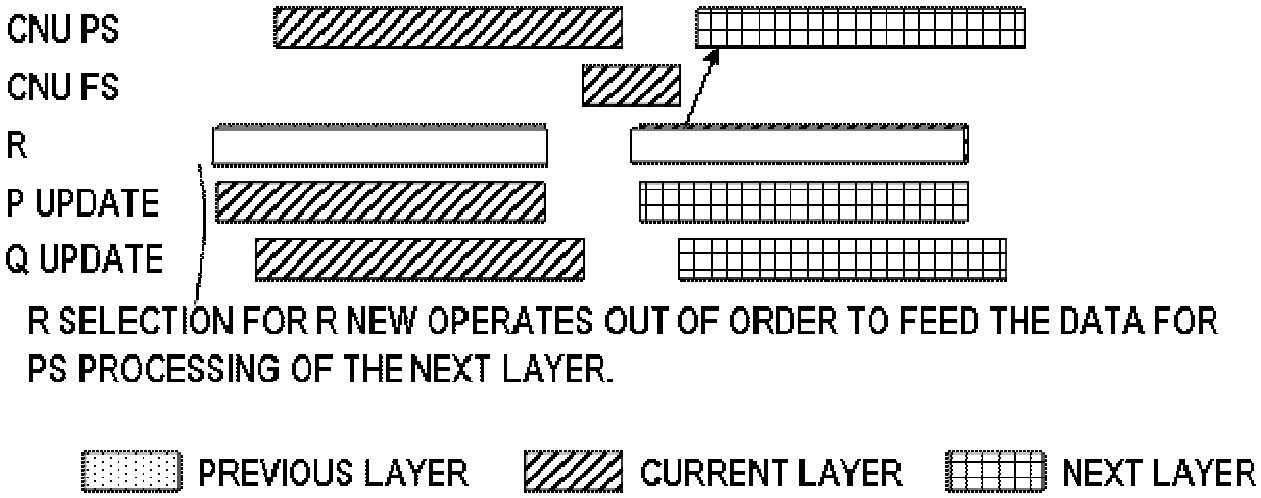
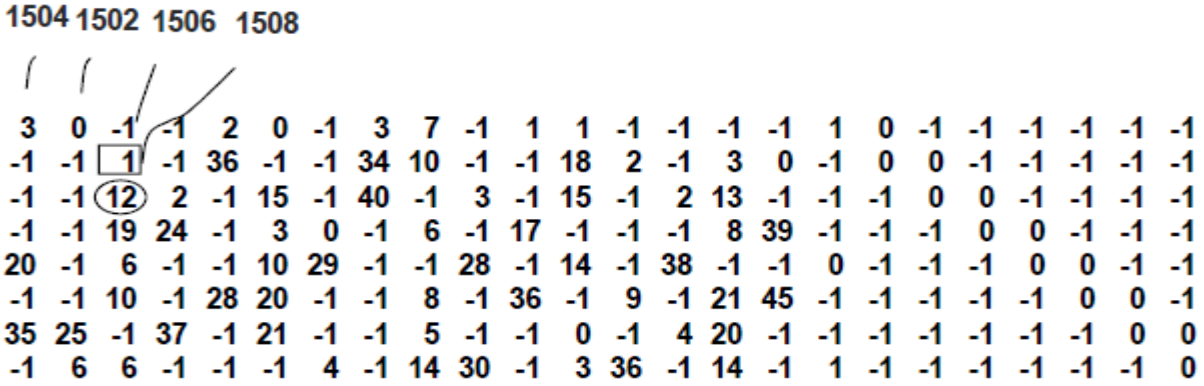


Illustration for out-of-order processing

Rate 2/3 code. 8 Layers, 24 block columns. dv, column weight varies from 2 to 6. dc, row weight is 10 for all the layers.



The following are the parameters of the circulant 1508 marked with the circle (denote this as the specified circulant):
The specified circulant 1508 belongs to 3rd layer.
This is the first non-zero circulant in this layer. So, the block number *bn* for the specified circulant 1508 is 1.
The circulant index *ci* for this specified circulant 1508 is 21.
The block column *bc* for this specified circulant 1508 is 3.
This specified circulant 1508 takes the updated P message from the circulant 1506 marked with the rectangle. So, circulant 1506 is the dependent circulant of the circulant 1508. The dependent circulant 1506 has a circulant index *ci* of 11. So, the dependent circulant index *dci* of the circulant 1508 is 11.
The layer of the dependent circulant 1506 is 2. So the dependent layer *dl* of the circulant 1508 marked with the circle is 2.
The block number of the dependent circulant 1506 is 1. So, the dependent block number *db* of the specified circulant 1508 is 1
The shift coefficient of the specified circulant 1508 is 12. Thus, the shift matrix coefficient *sm* of the specified circulant 1508 is 12. The H matrix has a circulant (i.e. identity matrix of size 96 x 96 that is cyclically shifted right by the amount 12) corresponding to 12 entry 1508 in the S matrix. Note that a non-zero circulant in the H matrix corresponds to 1 entry in the H_b matrix.
The shift coefficient of the dependent circulant 1506 is 1. So, the delta shift matrix coefficient *dsm* of the specified circulant 1508 is 12-1=11.
The specified circulant 1508 is the second non-zero circulant in the 3rd block column. Since the specified circulant 1508 is NOT the first non-zero circulant in its block column, the specified circulant takes the updated P message from the dependent circulant 1506 in all the iterations. Therefore, the use channel value flag *ucvf* of the specified circulant 1508 is 0.

Illustration for out-of-order processing

Rate 2/3 code. 8 Layers, 24 block columns. dv, column weight varies from 2 to 6. dc, row weight is 10 for all the layers.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	3	0	-1	-1	2	0	-1	3	7	-1	1	1	-1	-1	-1	-1	1	0	-1	-1	-1	-1	-1	-1
1	-1	-1	1	-1	36	-1	-1	34	10	-1	-1	18	2	-1	3	0	-1	0	0	-1	-1	-1	-1	-1
2	-1	-1	12	2	-1	15	-1	40	-1	3	-1	15	-1	2	13	-1	-1	-1	0	0	-1	-1	-1	-1
3	-1	-1	19	24	-1	3	0	-1	6	-1	17	-1	-1	-1	8	39	-1	-1	-1	0	0	-1	-1	-1
4	20	-1	6	-1	-1	10	29	-1	-1	28	-1	14	-1	38	-1	-1	0	-1	-1	-1	0	0	-1	-1
5	-1	-1	10	-1	28	20	-1	-1	8	-1	36	-1	9	-1	21	45	-1	-1	-1	-1	-1	0	0	-1
6	35	25	-1	37	-1	21	-1	-1	5	-1	-1	0	-1	4	20	-1	-1	-1	-1	-1	-1	-1	0	0
7	-1	6	6	-1	-1	-1	4	-1	14	30	-1	3	36	-1	14	-1	1	-1	-1	-1	-1	-1	-1	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

Non-zero circulants are numbered from 1 to 80. No layer re-ordering in processing. Out-of-order processing for Rnew. Out-of-order processing for Partial state processing.

Illustration for 2nd iteration with focus on PS processing of 2nd layer.

Rold processing is based on the circulant order 11 16 17 18 20 12 13 14 15 19 and is indicated in green.

Rnew is based on the circulant order 72 77 78 58 29 3 5 6 8 10 and is indicated in blue.

Q memory, HD memory access addresses are based on the block column index to which the green circulants are connected to.

Q sign memory access address is based on green circulant number.

Superscript indicates the clock cycle number counted from 1 at the beginning of layer 2 processing.

Out-of-order layer processing for R Selection

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

Normal practice is to compute R new messages for each layer after CNU PS processing.

However, here we decoupled the execution of R new messages of each layer with the execution of corresponding layer's CNU PS processing. Rather than simply generating Rnew messages per layer, we compute them on basis of circulant dependencies.

R selection is out-of-order so that it can feed the data required for the PS processing of the second layer. For instance Rnew messages for circulant 29 which belong to layer 3 are not generated immediately after layer 3 CNU PS processing .

Rather, Rnew for circulant 29 is computed when PS processing of circulant 20 is done as circulant 29 is a dependent circulant of circulant of 20.

Similarly, Rnew for circulant 72 is computed when PS processing of circulant 11 is done as circulant 72 is a dependent circulant of circulant of 11.

Here we execute the instruction/computation at precise moment when the result is needed!!!

Out-of-order block processing for Partial State

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

Re-ordering of block processing . While processing the layer 2,

the blocks which depend on layer 1 will be processed last to allow for the pipeline latency.

In the above example, the pipeline latency can be 5.

The vector pipeline depth is 5.so no stall cycles are needed while processing the layer 2 due to the pipelining. [In other implementations, the stall cycles are introduced – which will effectively reduce the throughput by a huge margin.]

Also we will sequence the operations in layer such that we process the block first that has dependent data available for the longest time.

This naturally leads us to true out-of-order processing across several layers. In practice we wont do out-of-order partial state processing involving more than 2 layers.

Overview of Schedule Optimization

- The decoder hardware architecture is proposed to support out-of-order processing to remove pipeline and memory accesses or to satisfy any other performance or hardware constraint. Remaining hardware architectures won't support out-of-order processing without further involving more logic and memory.
- For the above hardware decoder architecture, the optimization of decoder schedule belongs to the class of NP-complete problems. So there are several classic optimization algorithms such as dynamic programming that can be applied. We apply the following classic approach of optimal substructure.
- Step 1: We will try different layer schedules(j! i.e j factorial of j if there are j layers).
- Step 2:Given a layer schedule or a re-ordered H matrix, we will optimize the processing schedule of each layer. For this, we use the classic approach of optimal substructure i.e. the solution to a given optimization problem can be obtained by the combination of optimal solutions to its sub problems. So first we optimize the processing order to minimize the pipeline conflicts. Then we optimize the resulting processing order to minimize the memory conflicts. So for each layer schedule, we are measuring the number of stall cycles (our cost function).
- Step 3: We choose a layer schedule which minimizes the cost function i.e. meets the requirements with less stall cycles due to pipeline conflicts and memory conflicts and also minimizes the memory accesses (such as FS memory accesses to minimize the number of ports needed and to save the access power and to minimize the more muxing requirement and any interface memory access requirements.

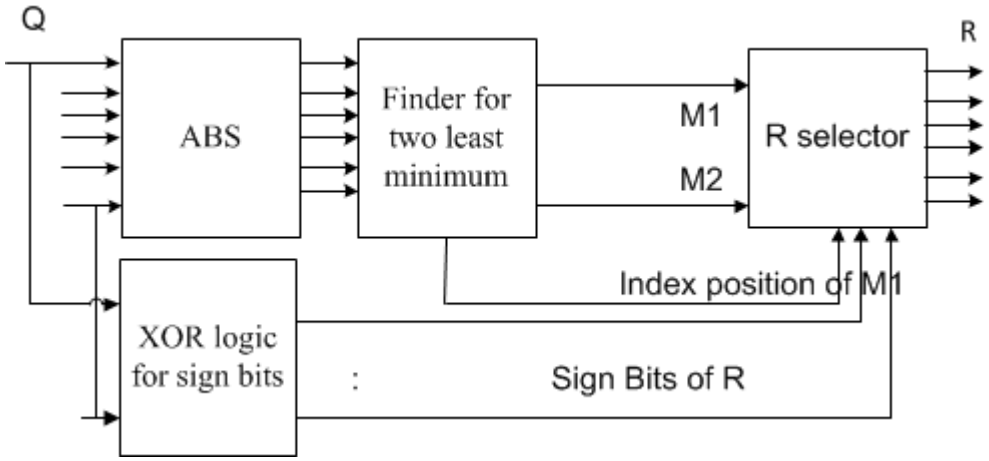
Memory organization

- Q memory width is equal to circulant size *8 bits and depth is number of block columns for 1-circulant processing.
- HD memory width is equal to circulant size *1 bits and depth is number of block columns for 1-circulant processing.
- Qsign memory width is equal to circulant size *1 bits and depth is number of non-zero circulants in H-matrix for 1-circulant processing.
- FS memory width is equal to circulant size*(15 bits(=4 bits for Min1+4 bits for Min2 +1 bit for cumulative sign+6 bits for Min1 index).
- FS memory access is expensive and number of accesses can be reduced with scheduling.
- For the case of decoder for regular mother matrices: FS access is needed one time for Rold for each layer; is needed one time for R new for each layer.
- For the case of decoder for irregular mother matrices: FS access is needed one time for Rold for each layer; is needed one time for R new for each non-zero circulant in each layer.

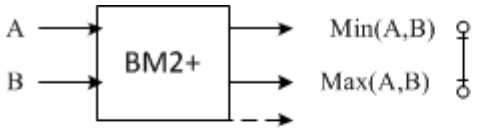
From Throughput Requirements to Design Specification

- Requirements
  - Throughput in bits per sec.
  - BER
  - Latency
- BER would dictiate Number of Iterations and degree profile(check node degrees and variable node degrees).
- Circulant Size (Sc)
- Number of Circulants processed in one clock (NSc)
- Number of bits processed per clock=Throughput/clock frequency
- **Sc*NSc=Nb*Iterations*Average Variable Node degree**
- Sc is usually set to less than 128 for smaller router.

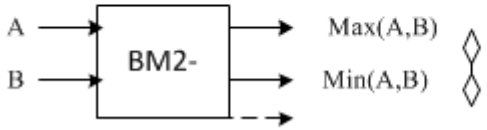
Parallel CNU



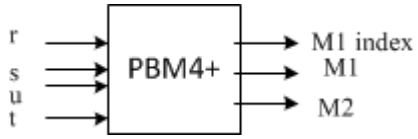
Parallel Min1-Min2 finder



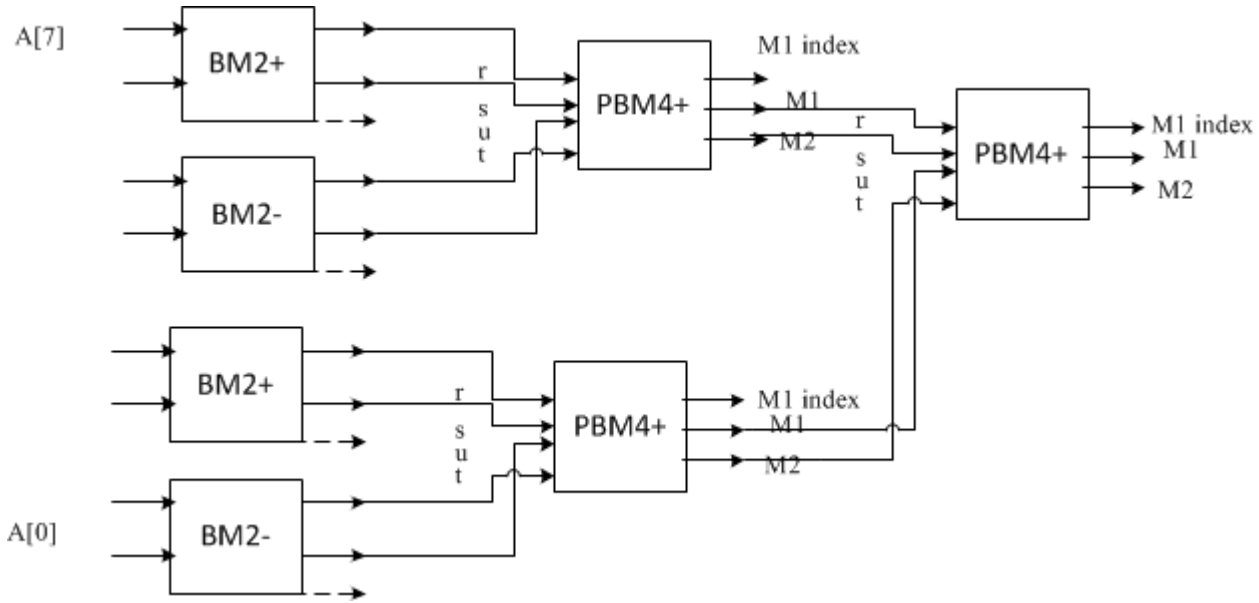
Symbol to indicate the BM2+ network



Symbol to indicate the BM2- network

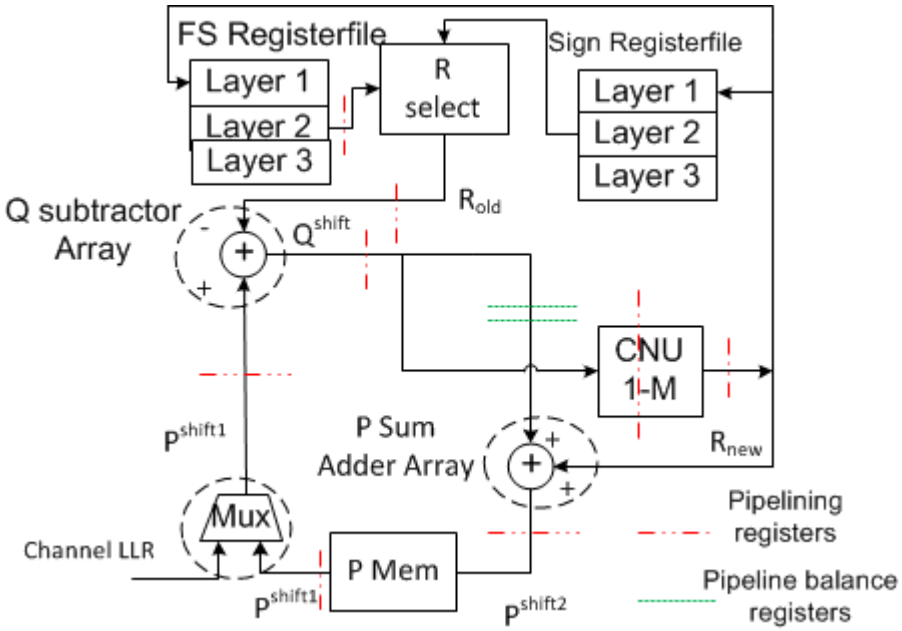


The inputs r,s,t,u form two bitonic sequences. r and s form a bitonic sequence of increasing order(i.e r<s). t and u form a bitonic sequence of decreasing order(i.e t>u).



Min1-Min2 finder using hierarchical approach of using PBM4+ to build PBM8+

Block Parallel Layered Decoder



Compared to other work, this work has several advantages
1) Only one memory for holding the P values.
2) Shifting is achieved through memory reads. Only one memory multiplexer network is needed instead of 2 to achieve delta shifts
1) Value-reuse is effectively used for both Rnew and Rold
2) Low complexity data path design-with no redundant data Path operations.
5) Low complexity CNU design with high parallelism.
6) Smaller pipeline depth

Here M is the row parallelization (i.e. number of rows in H matrix Processed per clock).

From Throughput Requirements to Design Specification

- Requirements
  - Throughput in bits per sec.
  - BER
  - Latency
- BER would dictate Number of Iterations and degree profile(check node degrees and variable node degrees).
- Regular code is assumed(i.e. uniform check node and variable node degrees)
- Circulant Size (Sc)=Code Length/Check Node Degree
- Number of rows processed in one clock (Nr)
- Number of bits processed per clock=Throughput/clock frequency
- **Nr=Nb*Iterations*Variable Node degree/Check Node degree**

Layered Decoder Throughput Results-FPGA, 802.11n

FPGA IMPLEMENTATION RESULTS THE MULTI-RATE DECODER. FULLY COMPLIANT TO IEEE 802.11N(SUPPORTS $z_0 = 27, 54, 81$ AND ALL THE CODE RATES) (DEVICE, XILINX 2V8000FF152-5, FREQUENCY 110MHZ)

	$M = 27$	$M = 54$	$M = 81$	Available
Slices	1836	3647	5514	46592
LUT	3317	6335	9352	93184
SFF	1780	3560	5341	93184
BRAM	33	65	97	168
Memory(bits)	23376	39360	55344	
Throughput(Mbps)				
$z_0 = 81$	119	238	356	
$z_0 = 54$	119	238	178	
$z_0 = 27$	119	119	119	

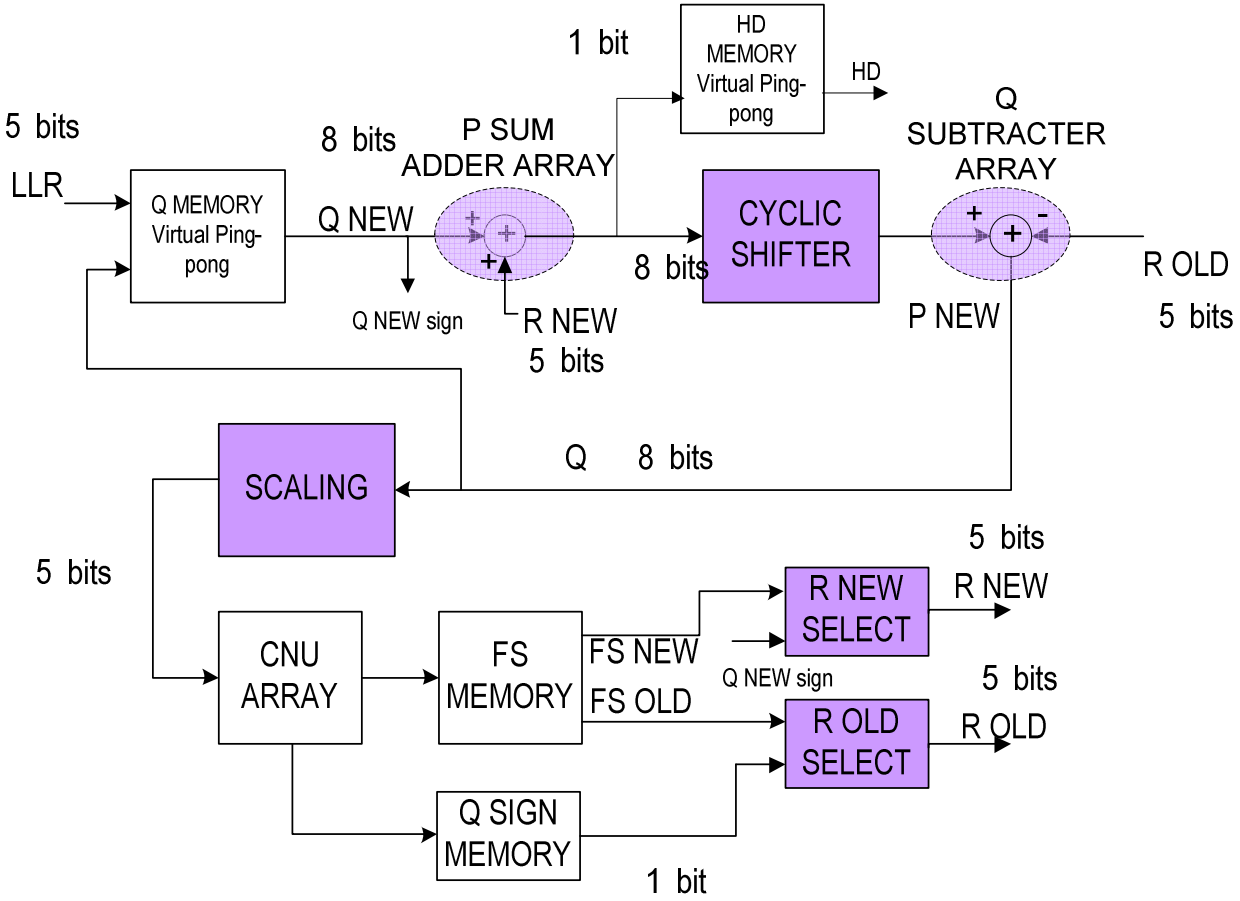
Layered Decoder Throughput Results-ASIC, 802.11n

ASIC IMPLEMENTATION RESULTS THE MULTI-RATE DECODER FOR $M = 81$ (0.13 μ M TECHNOLOGY [15], FREQUENCY 500MHZ)

Resource	Area (mm^2)	Component	Power (mW)
CNU	0.444	Memory	62.3
VNU	0.067	leakage	0.07
Storage	1.039	Clock	27.1
Flip-flop	0.022	wiring	13.5
Shifter	0.124	active power	135.35
wiring	0.085	total power	238.4
total	1.7816		
Throughput(Mbps)	541, 1082 and 1618 for $z_0 = 27, 54$ and 81		

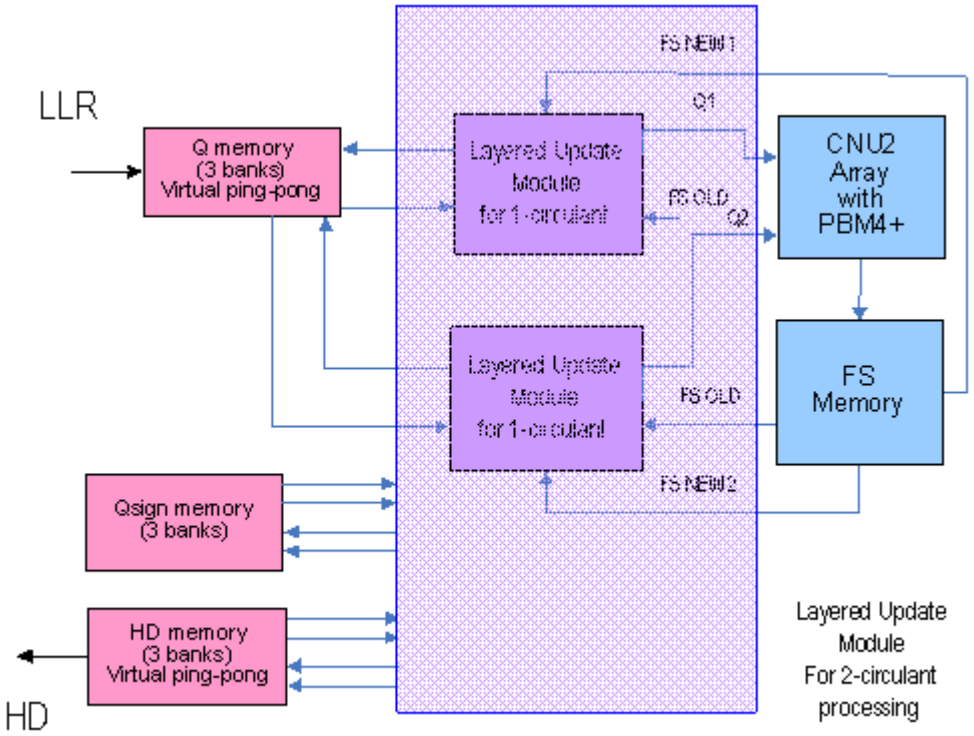
Proposed decoder takes around 100K logic gates and 55344 memory bits.

Layered Decoder for Flash Channel



Decoder similar to that of Slide 26. One-circulant processing. Serves medium throughput applications. Modules highlighted in Lavender comprise layered update module (LUM).

Layered Decoder for Flash Channel



Two-circulant processing. Serves high-throughput applications while retaining the flexibility to support multiple codes.

Note: In some of our designs which needed limited flexibility, we used the block parallel architecture on Slide 37 as a better candidate for area and power.

Design considerations

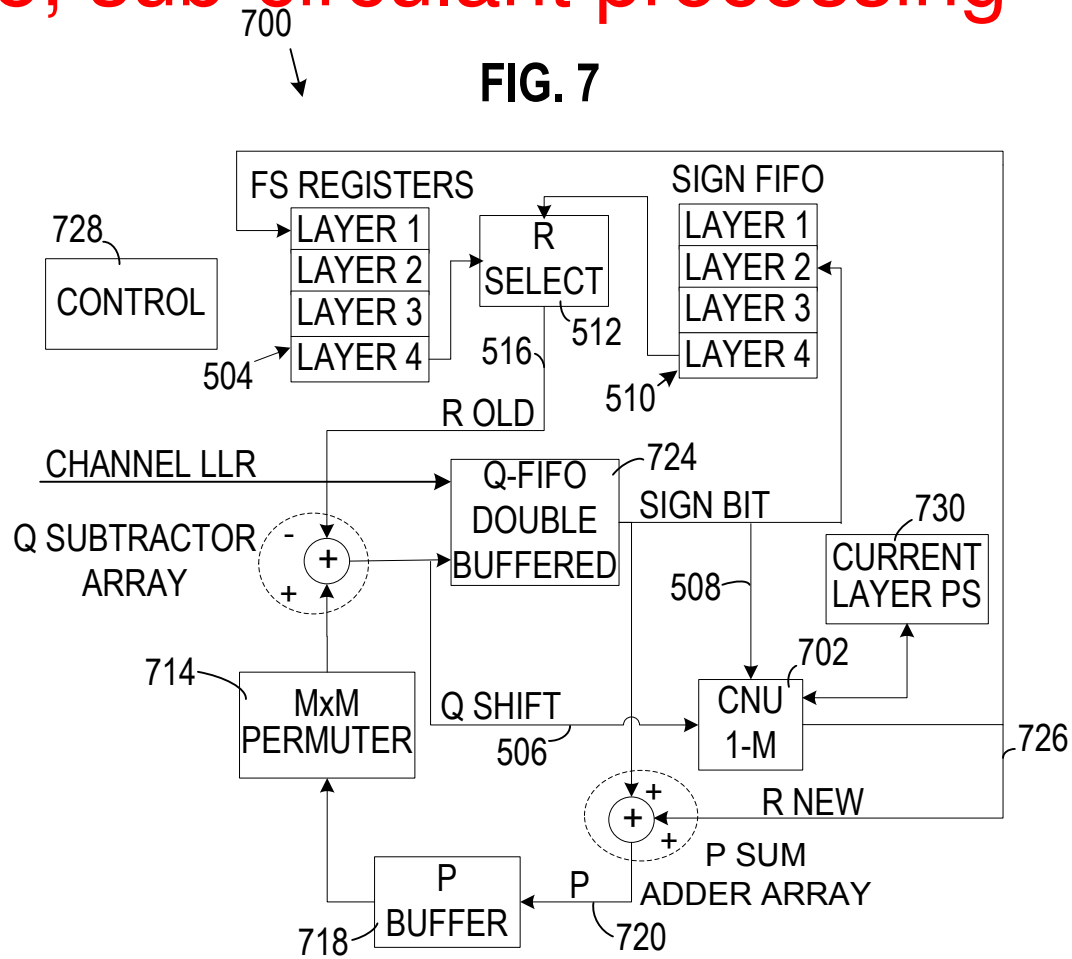
- The design for the decoder based on 2-circulant processing is similar to 1-circulant processing explained in slides 26-33.
- Q memory width is equal to circulant size *8 bits and depth is number of block columns for 1-circulant processing.
- For 2-circulant processing, we divide Q memory into 3 banks. Each bank width is equal to circulant size *8 bits and depth is ceil(number of block columns/3).
- HD memory width is equal to circulant size *1 bits and depth is number of block columns for 1-circulant processing.
- For 2-circulant processing, we divide HD memory into 3 banks. Each bank width is equal to circulant size *1 bits and depth is ceil(number of block columns/3).
- Qsign memory width is equal to circulant size *1 bits and depth is number of non-zero circulants in H-matrix for 1-circulant processing.
- For 2-circulant processing, we divide HD memory into 3 banks. Each bank width is equal to circulant size *1 bits and depth is ceil(number of non-zero circulants in H-matrix/3).

Summary and Key slides

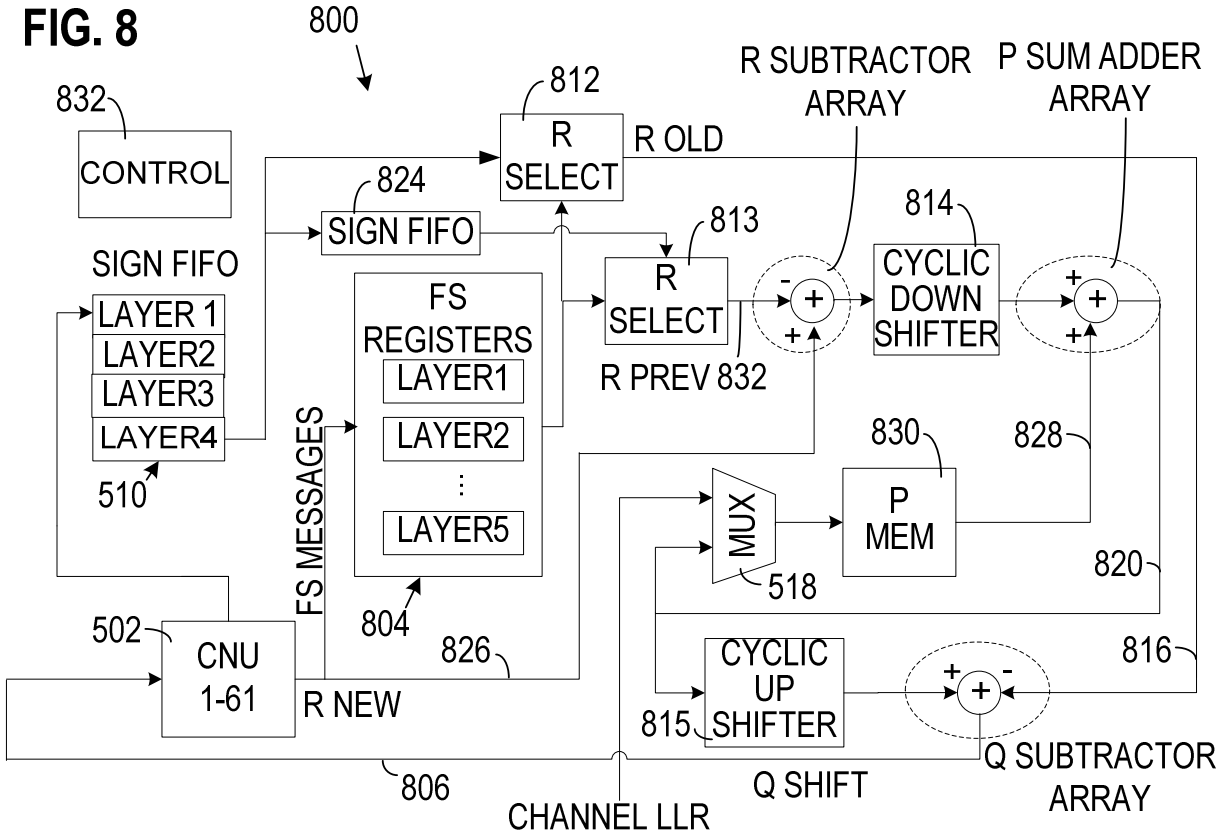
- An area (logic and memory) and power efficient multi-rate architecture for standard message passing decoder (non-layered decoder) of LDPC- **Slide 15**
- An area (logic and memory) and power efficient multi-rate architecture for Layered decoding of regular QC- LDPC – **Slide 20**
- An area (logic and memory) and power efficient multi-rate architecture with efficient scheduling of computations to minimize idle cycles for Layered decoding of irregular QC-LDPC for IEEE 802.11n (Wi-Fi), IEEE 802.16e(Wimax) and storage (HDD read channel and Flash read channel)applications. – **Slide 26**, **slide 41**
- An area (logic and memory) efficient parallel layered decoder for regular LDPC for storage (HDD read channel and Flash read channel) and other applications (IEEE 802.3 10-GB Ethernet) – **Slide 37**
- FPGA prototyping and ASIC design clearly illustrates the advantages of the proposed decoder architectures. – Slides 39-40 and published results listed in the references. Several commercial high-volume designs are based on these architectures as part of speaker's prior industrial work.

Some of architecture variations, 1/5, sub-circulant processing

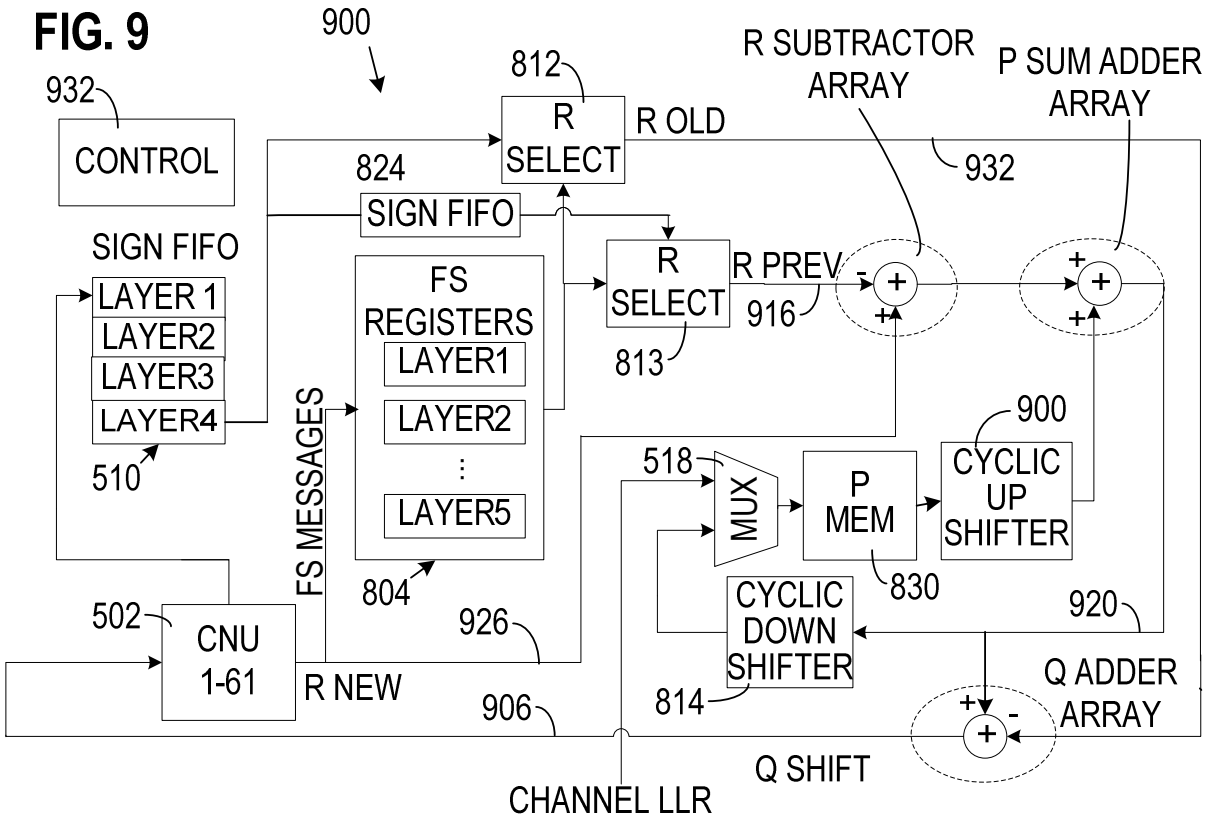
FIG. 7



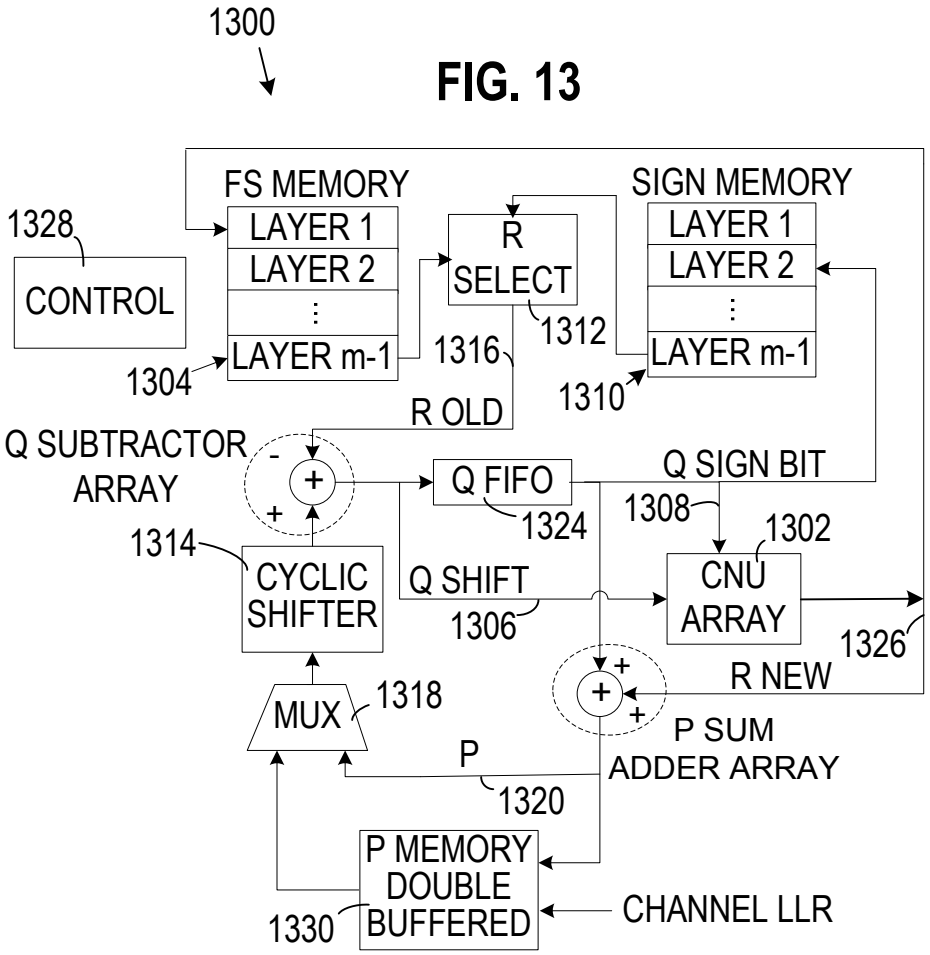
Architecture Variations, 2/5



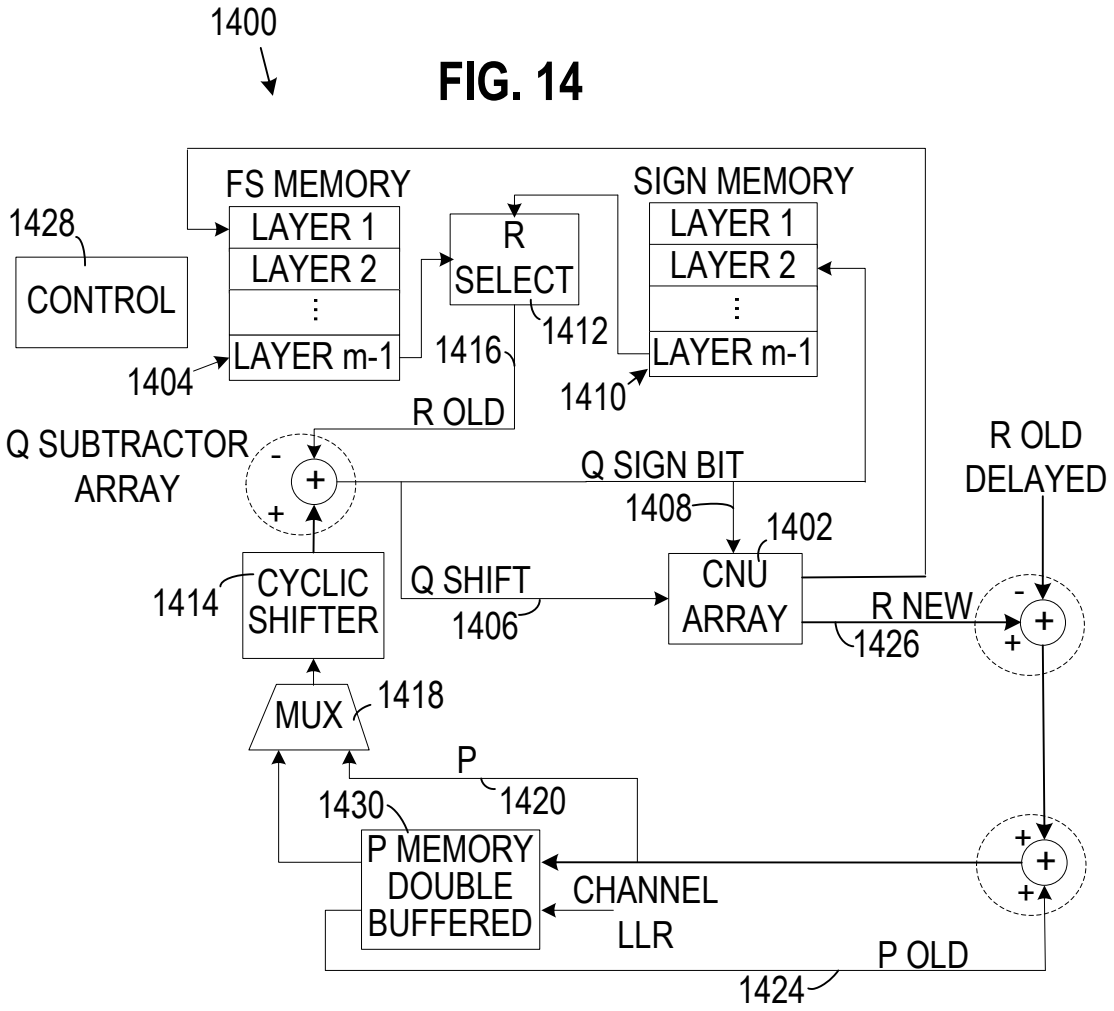
Architecture Variations, 3/5



Architecture Variations, 4/5



Architecture Variations, 5/5



References

- Check http://dropzone.tamu.edu for technical reports.
- 1. Gunnam, KK; Choi, G. S.; Yeary, M. B.; Atiquzzaman, M.; "VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax," Communications, 2007. ICC '07. IEEE International Conference on 24-28 June 2007 Page(s):4542 - 4547
- 2. Gunnam, K.; Gwan Choi; Weihuang Wang; Yeary, M.; "Multi-Rate Layered Decoder Architecture for Block LDPC Codes of the IEEE 802.11n Wireless Standard," Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on 27-30 May 2007 Page(s):1645 – 1648
- 3. Gunnam, K.; Weihuang Wang; Gwan Choi; Yeary, M.; "VLSI Architectures for Turbo Decoding Message Passing Using Min-Sum for Rate-Compatible Array LDPC Codes," Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on 5-7 Feb. 2007
- 4. Gunnam, Kiran K.; Choi, Gwan S.; Wang, Weihuang; Kim, Euncheol; Yeary, Mark B.; "Decoding of Quasi-cyclic LDPC Codes Using an On-the-Fly Computation," Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on Oct.-Nov. 2006 Page(s):1192 - 1199
- 5. Gunnam, K.K.; Choi, G.S.; Yeary, M.B.; "A Parallel VLSI Architecture for Layered Decoding for Array LDPC Codes," VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on Jan. 2007 Page(s):738 – 743
- 6. Gunnam, K.; Gwan Choi; Yeary, M.; "An LDPC decoding schedule for memory access reduction," Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on Volume 5, 17-21 May 2004 Page(s):V - 173-6 vol.5
- 7. GUNNAM, Kiran K., CHOI, Gwan S., and YEARY, Mark B., "Technical Note on Iterative LDPC Solutions for Turbo Equalization," Texas A&M Technical Note, Department of ECE, Texas A&M University, College Station, TX 77843, Report dated July 2006. Available online at http://dropzone.tamu.edu March 2010, Page(s): 1-5.
- 8. K. Gunnam, G. Choi, W. Wang, and M. B. Yeary, "Parallel VLSI Architecture for Layered Decoding ," Texas A&M Technical Report, May 2007. Available online at http://dropzone.tamu.edu
- 9. Kiran K. Gunnam, Gwan S. Choi, Mark B. Yeary, Shaohua Yang and Yuanxing Lee , Next Generation Iterative LDPC Solutions for Magnetic Recording Storage", 42nd Asilomar Conference on Signals, Systems and Computers, 2008, pp. 1148-1152
- 10.. E. LI, K. Gunnam, and D. Declercq, "Trellis based Extended Min-Sum for Decoding Nonbinary LDPC codes," ISWCS'11, Nov. 2011.

References [Contd] & Important Information

- Several features presented in the Module 2 by Kiran Gunnam are covered by the following pending patent applications by Texas A&M University System (TAMUS).

[P1] K. K. Gunnam and G. S. Choi, "Low Density Parity Check Decoder for Regular LDPC Codes," U.S. Patent Application No. 12/113,729, Publication No. US 2008/0276156 A1

[P2] K. K. Gunnam and G. S. Choi, "Low Density Parity Check Decoder for Irregular LDPC Codes," U.S. Patent Application No. 12/113,755, Publication No. US 2008/0301521 A1