Area and Energy Efficient VLSI Architectures for Low-Density Parity-Check Decoders using an On-the-fly Computation*

Kiran Gunnam

LSI Corporation

kgunnam@ieee.org

Feb 21 2008

* Work done as part of PhD research at Texas A&M University.

Outline

- Introduction to LDPC
- Problem Statement
- On-the-fly computation for QC-LDPC
- Block Serial standard message passing Decoder
- Block Serial Layered Decoder
- Block Parallel Layered Decoder
- Results and Performance comparison

Introduction to LDPC

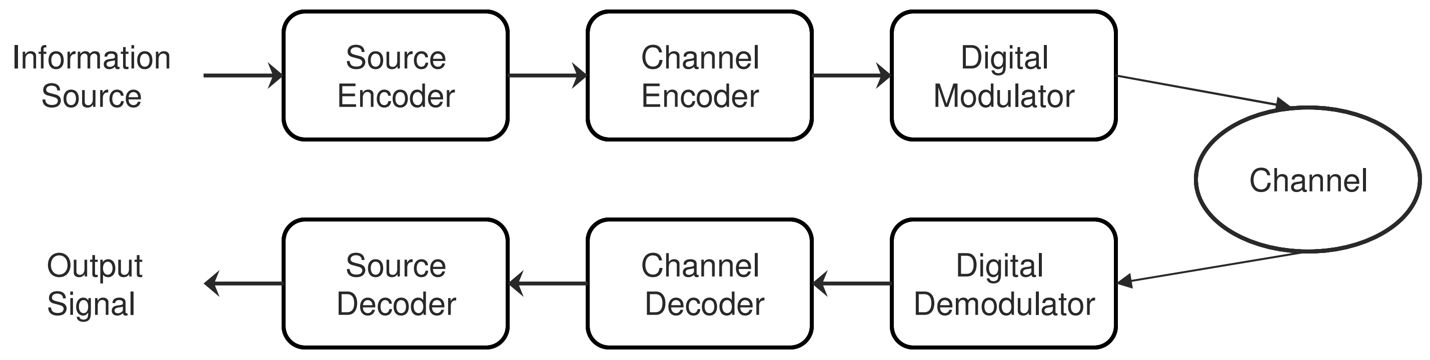


Figure 1: Block Diagram of Communication System

- Channel Encoder introduces redundancy in transmitted bit stream
- Channel decoder use the redundancy to correct the errors due to channel impairments and noise.
- Low-Density Parity-Check (LDPC) Code is the best available error correction code.

Progress in Error Correction Systems

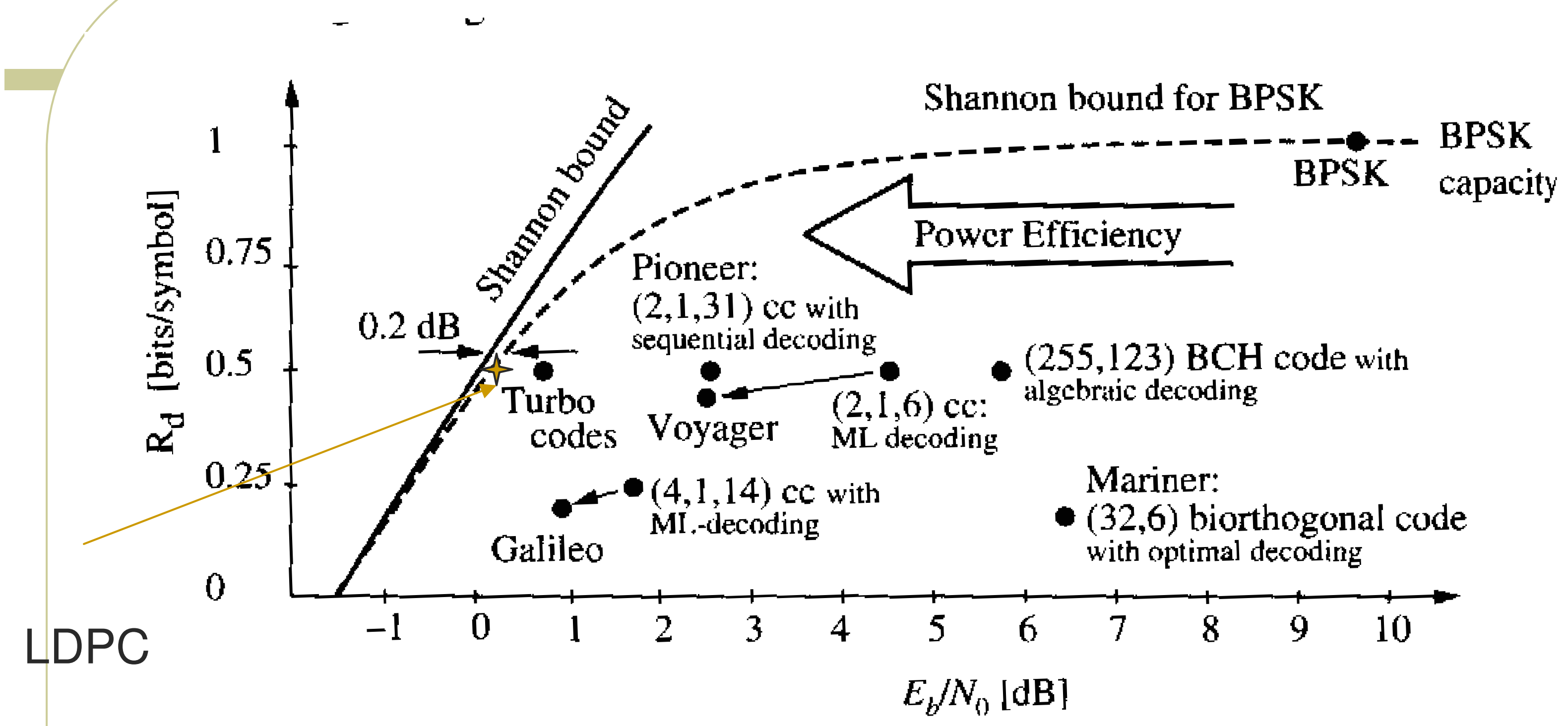


Figure 1.7 Milestones in the drive toward channel capacity achieved by the space systems which evolved over the past 40 years as an answer to the Shannon capacity challenge.

- Source :Trellis coding by C. Schlegel, IEEE Press

Some Notation and Terminology

- k- number of input (information) bits.
  n – number of output (coded) bits

- Code is a set of 2^k vectors of length n

- Encoder is a specific mapping of 2^k inputs to codewords

- Decoder tries to recover the information bits from the received code word that is corrupted with channel impairments

- d_{min} – minimum distance of the code
  Smallest Hamming distance between any two codewords.

  - from Dr. Krishna Narayanan (Texas A&M)'s presentation

Shannon’s Idea in the proof

- Pick 2^k codewords of length n at random
- Code is guaranteed to be good as k ! 1
- Problem: How to decode this?
- Brute force: we require storage of 2^k codewords
- There are only about 10^{79} (~ 2^{250}) atoms in the universe
    - from Dr. Krishna Narayanan (Texas A&M)’s presentation

Good Code Requirements

- As k ! 1 code must correct up to the Shannon limit
- Encoding/Decoding complexities don’t increase drastically with k
- Storage does not increase drastically with k
- Randomness Vs Structure
  - Random codes are good
  - But structure is needed to make it practical

- from Dr. Krishna Narayanan (Texas A&M)’s presentation

Coding Theory Advances

- There are two kinds of codes: Block Codes and Convolutional codes
- In an (n,k) block code, k bits are encoded in to n bits. Block code is specified by k x n generator matrix G or an (n-k) x n parity check matrix H
- Block Codes: Hamming, BCH, Reed Solomon Codes. Hard decision decoding is used. Soft decoding possible- but complex.
- Convolutional codes: Can encode infinite sequence of bits using shift registers. Soft decision decoding such as viterbi can achieve optimal maximum likelihood decoding performance.
- Turbo Codes (1993): Parallel concatenated convolutional. Codes
- Rediscovery: **LDPC** Block code**(1962, 1981, 1998).** Near shannon limit code, Efficient soft decoding (message passing) and with iterations.

Coding Theory Advances

- LDPC codes were invented by Robert Gallager [R1] in his PhD thesis.
- Soon after their invention, they were largely forgotten, and reinvented several times for the next 30 years.
- Their comeback is one of the most intriguing aspects of their history, since different communities reinvented codes similar to Gallager's LDPC codes, but for entirely different reasons.

[R1] R. Gallager, "Low-density parity-check codes," IRE Trans. Information Theory, pp. 21, 28, January 1962.

[R2] M. R. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, vol. 27, pp. 533–547, 1981.

[R3] D. MacKay, "Good error correcting codes based on very sparse matrices," IEEE Trans. Information Theory, pp. 399-431, March 1999.

- from "LDPC Codes: An Introduction" Amin Shokrnollahi

LDPC Characteristics

- ❑ Excellent Bit Error Rate performance!
- ❑ Computationally easier decoding than Turbo Codes
- ❑ Moderate storage requirements

Table 1: BER performance for different codes

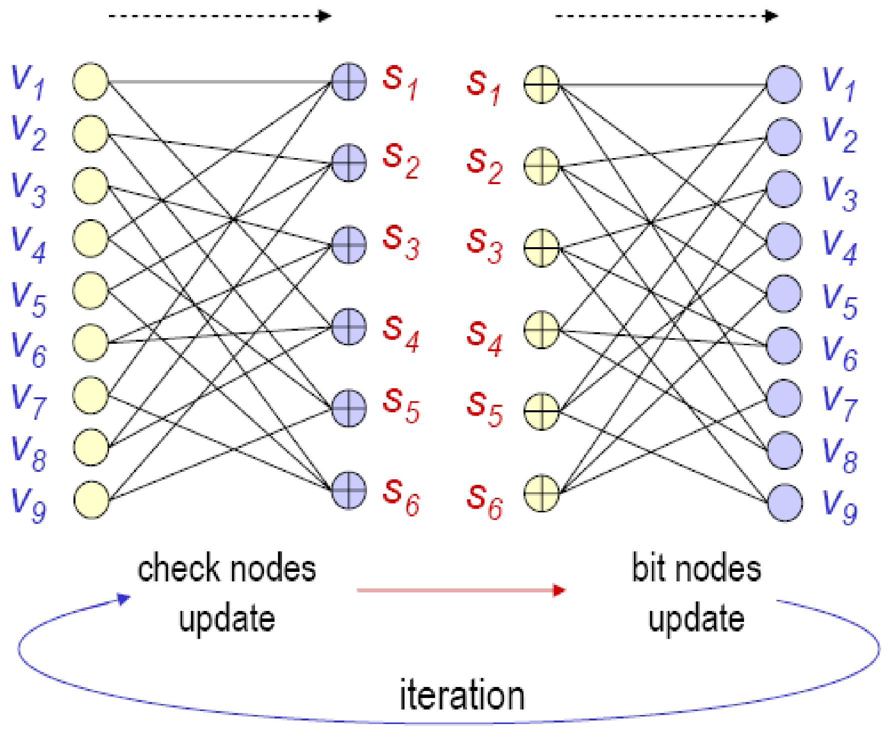
Rate ½ Code	SNR required for BER <1e-5
Shannon, Random Code	0 dB
(255,123) BCH	5.4 dB
Convolutional Code	4.5 dB
Iterative Code Turbo	0.7 dB
Iterative Code LDPC	0.0045 dB

Example LDPC Code

check nodes

$$H=\begin{array}{c c}\left(\begin{array}{ccccccccc}1&0&0&1&0&0&1&0&0\\0&1&0&0&1&0&0&1&0\\0&0&1&0&0&1&0&0&1\\1&0&0&0&0&1&0&1&0\\0&1&0&1&0&0&0&0&1\\0&0&1&0&1&0&1&0&0\end{array}\right) & \begin{array}{c}s_1\\s_2\\s_3\\s_4\\s_5\\s_6\end{array}\\ v_1v_2v_3v_4v_5v_6v_7v_8v_9 & \end{array}$$

variable nodes



v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9
s_1 s_2 s_3 s_4 s_5 s_6
check nodes update
bit nodes update
iteration

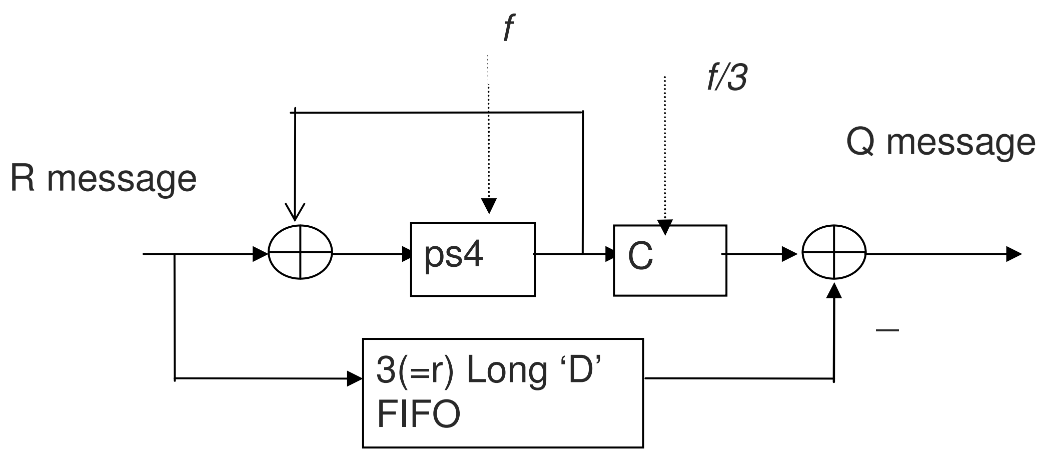
Variable nodes correspond to the soft information of received bits.

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).

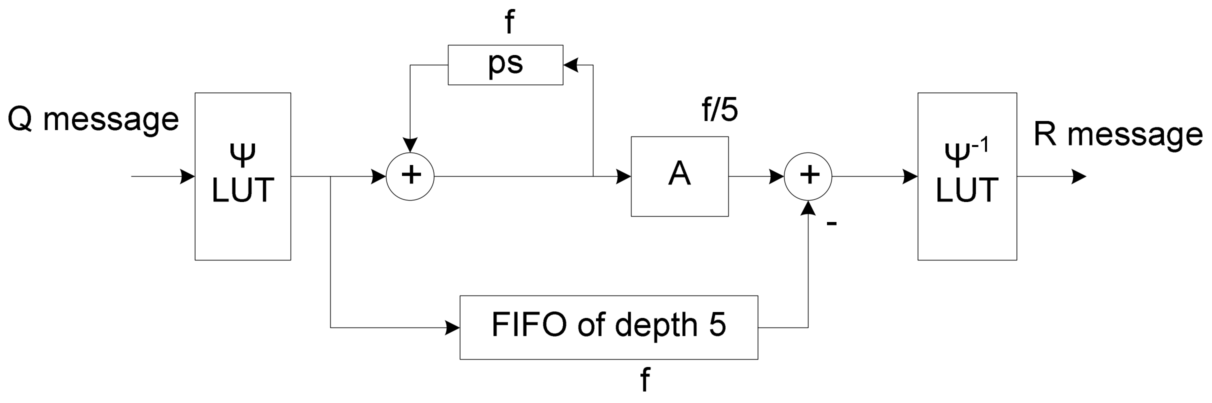
Micro-Architecture for VNU



Serial Variable Node Unit (VNU) to compute the variable node messages (Q) from check node messages

$$Q_{bi,cj} = \left(\sum_{j'=Col[bi][1]}^{Col[bi][r]} R_{j',bi} \right) - R_{cj,bi} + \wedge(bi)$$

Micro-Architecture for CNU



A serial Check Node Unit (CNU) for Sum of Products algorithm.

$$R_{cj,bi} = \psi^{-1}\left[\left(\sum_{i'=Row[cj][1]}^{Row[cj][c]} \psi\left(Q_{i',cj}\right)\right) - \psi\left(Q_{bi,cj}\right)\right].\delta(cj,bi)$$

Decoder Architectures

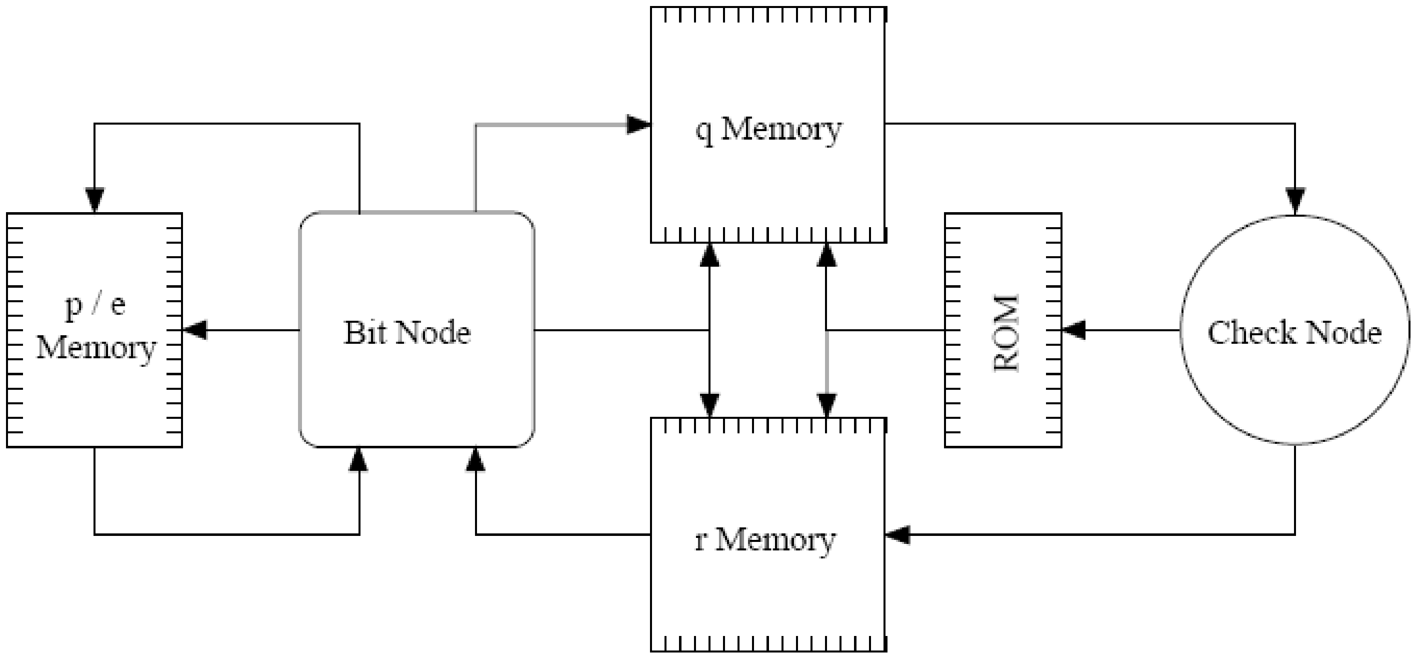
- Fully Parallel Architecture:
  - All the check updates in one clock cycle and all the bit updates in one more clock cycle.
  - Huge Hardware resources and routing congestion.

Decoder Architectures, Serial

Serial Architecture[1-2]

Check updates and bit updates in a serial fashion.

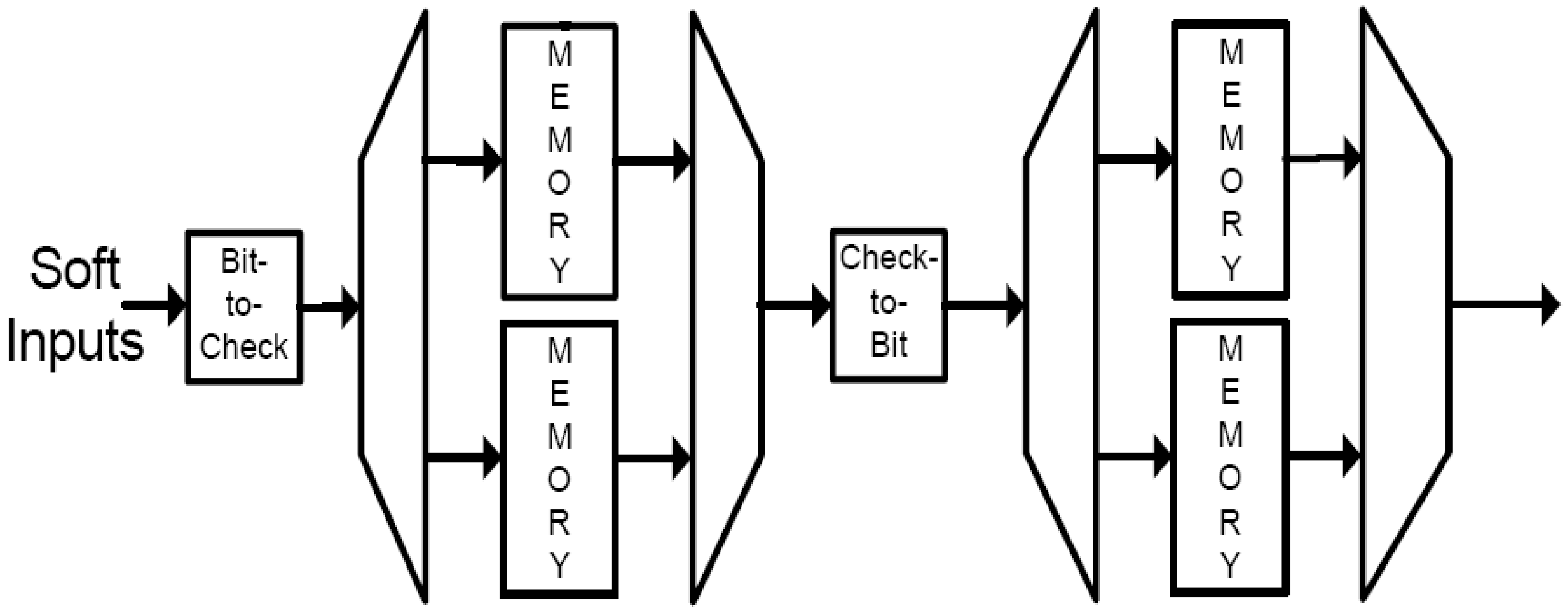
Huge Memory requirement. Memory in critical path.



Serial Architecture

[1] Levine,.etal Implementation of near Shannon limit error-correcting codes using reconfigurable hardware IEEE Field-Programmable Custom Computing Machines, 2000

Decoder Architectures, Serial



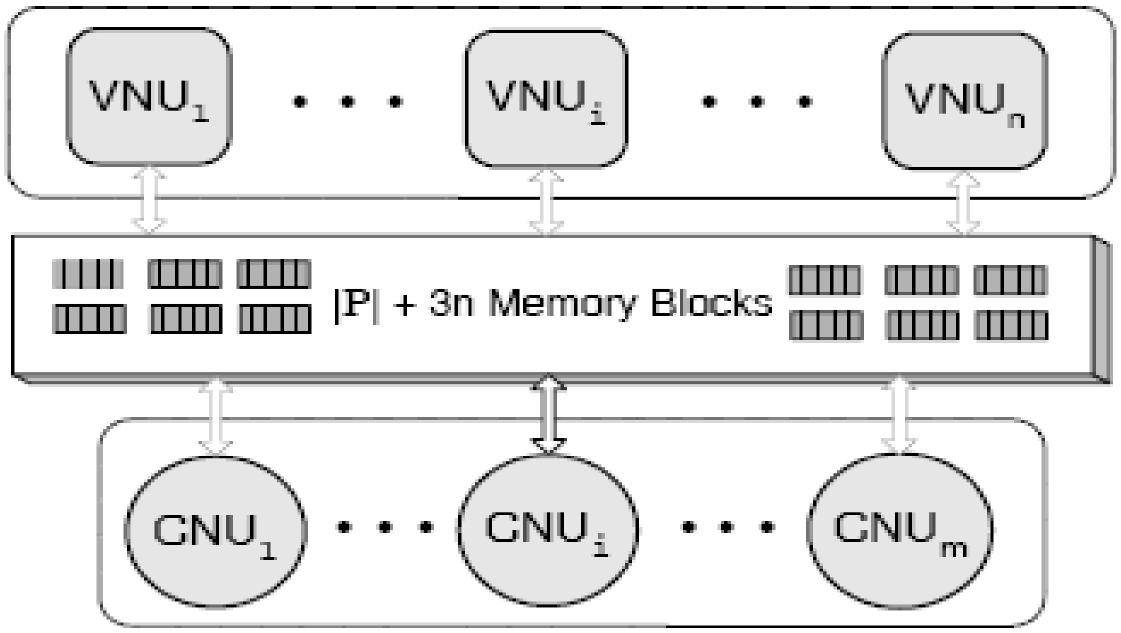
Serialized and fully pipelined implementation requires two memory buffers per stage, alternating between read/write.

Serial Architecture.

[2] E. Yeo, "VLSI architectures for iterative decoders in magnetic recording channels,“ IEEE Trans. Magnetics, vol.37, no.2, pp. 748-55, March 2001.

Semi-Parallel Architecture

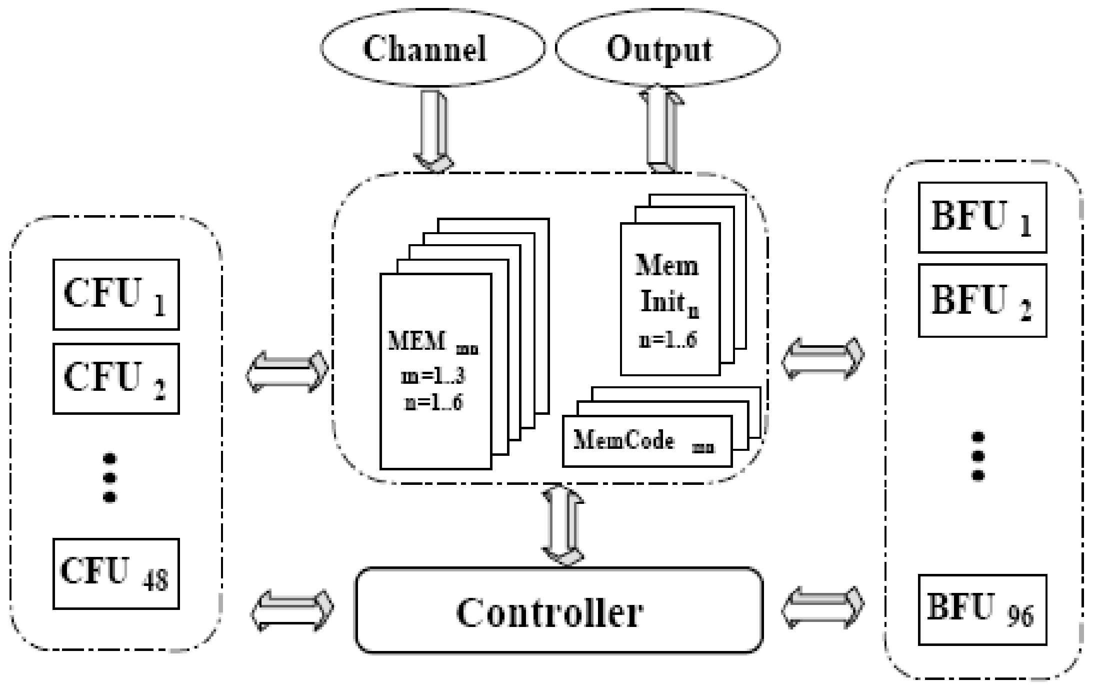
Check updates and bit updates using several units.
Partitioned memory by imposing structure on H matrix.
Practical solution for most of the applications.
Complexity differs based on architecture and scheduling



Semi Parallel Architecture

[3] T. Zhang and K. K. Parhi, Joint (3,k)-Regular LDPC Code and Decoder/Encoder Design, IEEE Transactions on Signal Processing April, 2004.

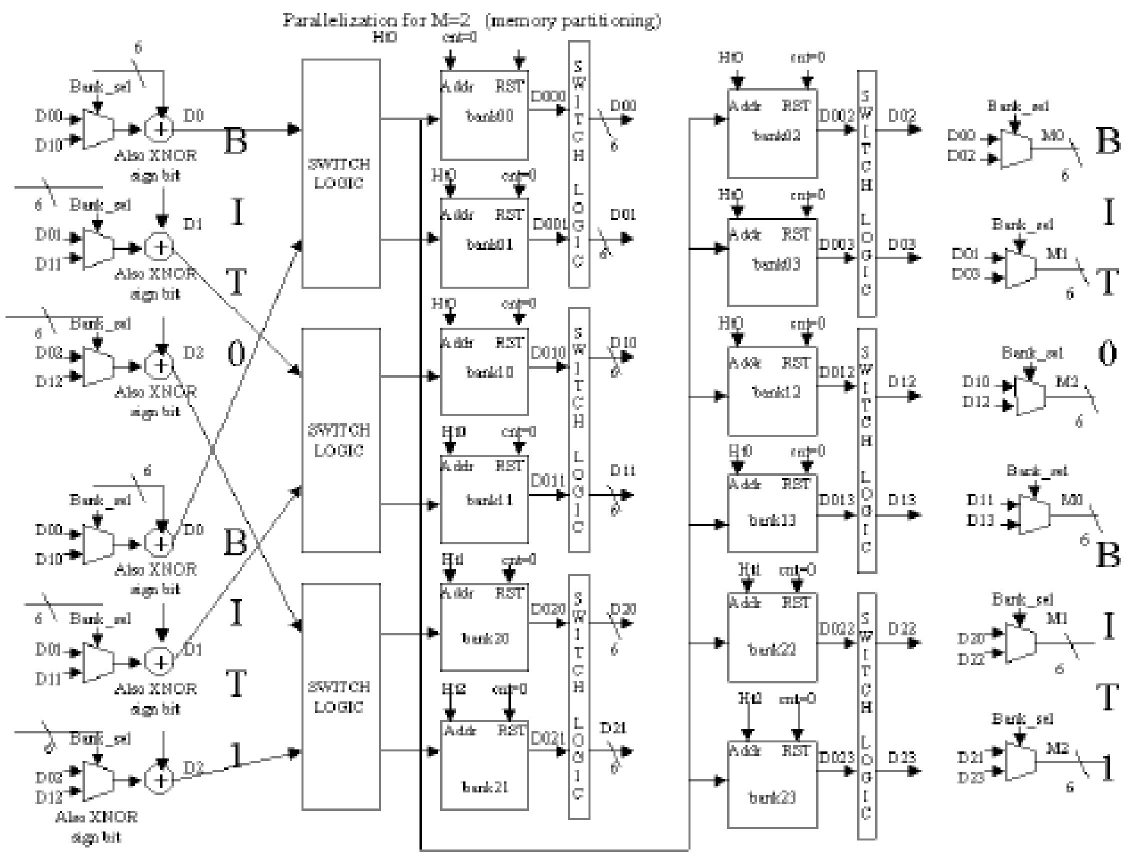
Semi-Parallel Architecture



Semi Parallel Architecture

[4] Karkooti etal. Semi-parallel reconfigurable architectures for real-time LDPC decoding ; Proceedings. ITCC 2004.

Semi-Parallel Architecture



Semi Parallel Architecture

[5] A. Selvarathinam, G.Choi, K. Narayanan, A.Prabhakar, E. Kim, "A Massively Scalable Decoder Architecture for Low-Density Parity-Check Codes", in proceedings of ISCAS'2003

Problem Statement

- The authors in [6] reported that 95% of power consumption of the decoder chip developed in [2] results from memory accesses.
- The authors in [7] reported that 50% of their decoder power is from memory accesses.
- Memory access is a bottleneck in preventing full utilization of units.
- Efficient implementations for the irregular codes is a hard problem

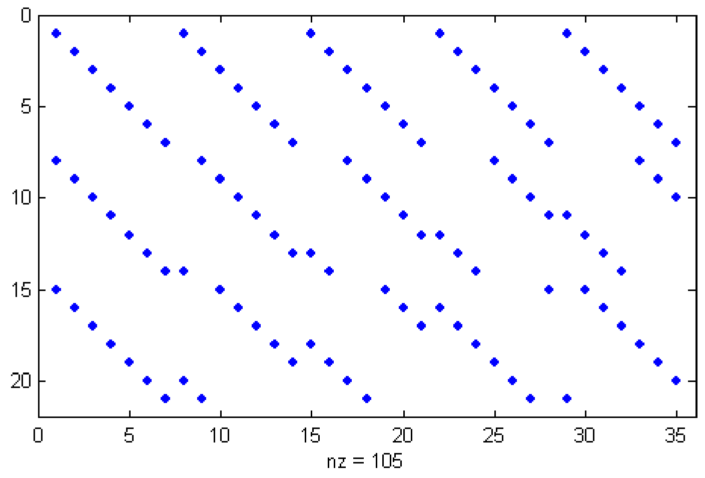
[6] Yijun Li et al, "Power efficient architecture for (3,6)-regular low-density parity-check code decoder," IEEE ISCAS 2004

[7] Mansour et al "A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip"-IEEE Journal of Solid-State Circuits, March 2006

Structured LDPC Codes

Array Codes

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



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

Example H Matrix

r row/ check node degree=5

c columns/variable node degree=3

P=7

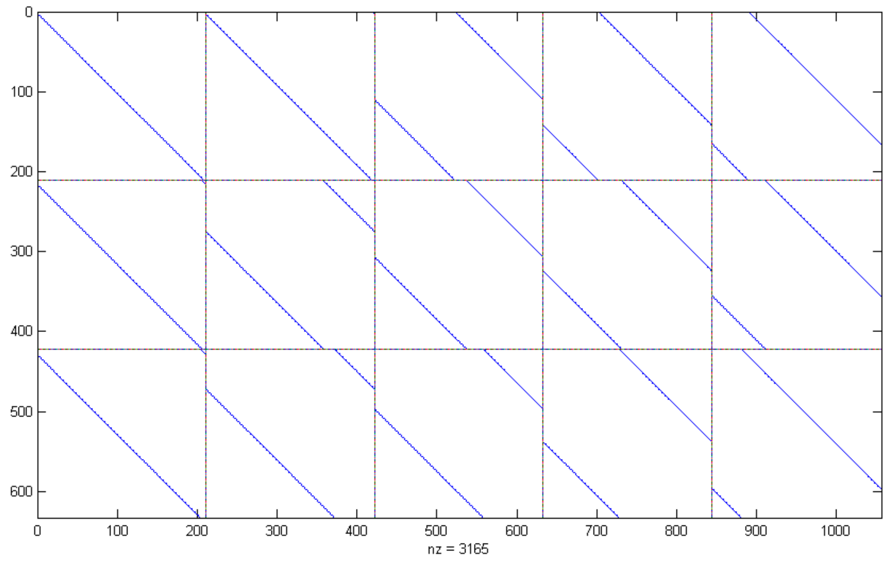
*N=P*r=35*

[8] J. L. Fan, "Array-codes as low-density parity-check codes", In Proc. TPP

Structured LDPC Codes

Cyclotomic Cosets

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



Example H Matrix

r row degree=5

c column degree =3

P=211

*N=P*r=1055*

On-the-fly computation

This research introduces the following concepts to LDPC decoder implementation

[ICASSP'04,Asilomar'06,VLSI'07,ISWPC'07,ISCAS'07,ICC'07]

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.

[Items in bold are covered in more detail for this presentation]

Decoder Architectures utilizing On-the-fly Computation

- **Block Serial standard message passing Decoder**
- **Block Serial Layered Decoder**
- **Block Parallel Layered Decoder**

Block Serial Standard Message Passing Decoder

Key Observations and Results

Represent R and Q messages by the following matrices (similar to physical message storage employed in other architectures) except that these matrices are not really stored in the proposed architecture [9].

$$Rm = \begin{bmatrix} R_{1,Row[1][1]} & R_{1,Row[1][2]} & \cdots & R_{1,Row[1][r]} \\ R_{2,Row[2][1]} & R_{2,Row[2][2]} & \cdots & R_{2,Row[2][r]} \\ \vdots & \vdots & \vdots & \vdots \\ R_{cp,Row[cp][1]} & R_{cp,Row[cp][1]} & \cdots & R_{cp,Row[cp][r]} \end{bmatrix}$$

$$Qm = \begin{bmatrix} Q_{1,Col[1][1]} & Q_{2,Col[2][1]} & \cdots & Q_{rp,Col[rp][1]} \\ Q_{1,Col[1][2]} & Q_{2,Col[2][2]} & \cdots & Q_{rp,Col[rp][2]} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{1,Col[1][c]} & Q_{2,Col[2][c]} & \cdots & Q_{kp,Col[rp][c]} \end{bmatrix}$$

R –Check Node to Variable Node messages

Q- Variable Node to Check Node messages

*Rm->(p*r)*c=>21x5*

*Qm->N*c->35x3*

[9] K. Gunnam, G. Choi and M. B. Yeary, "An LDPC Decoding Schedule for Memory Access Reduction", IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)

Block Independence Property

The variable nodes in each block column has support only on the one block column edges of the check nodes.

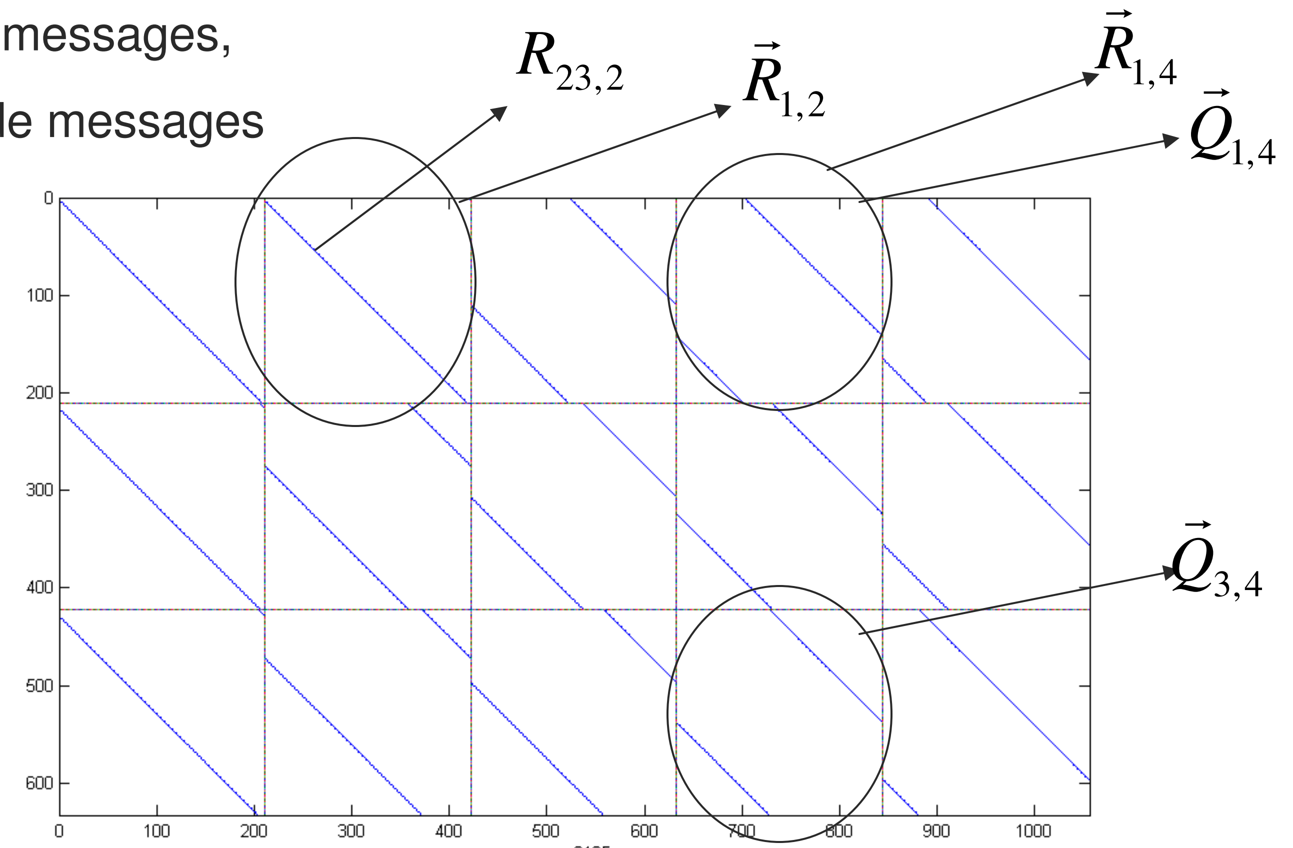
R check node messages,

Q variable node messages

Block Row 1

Block Row 2

Block Row 3



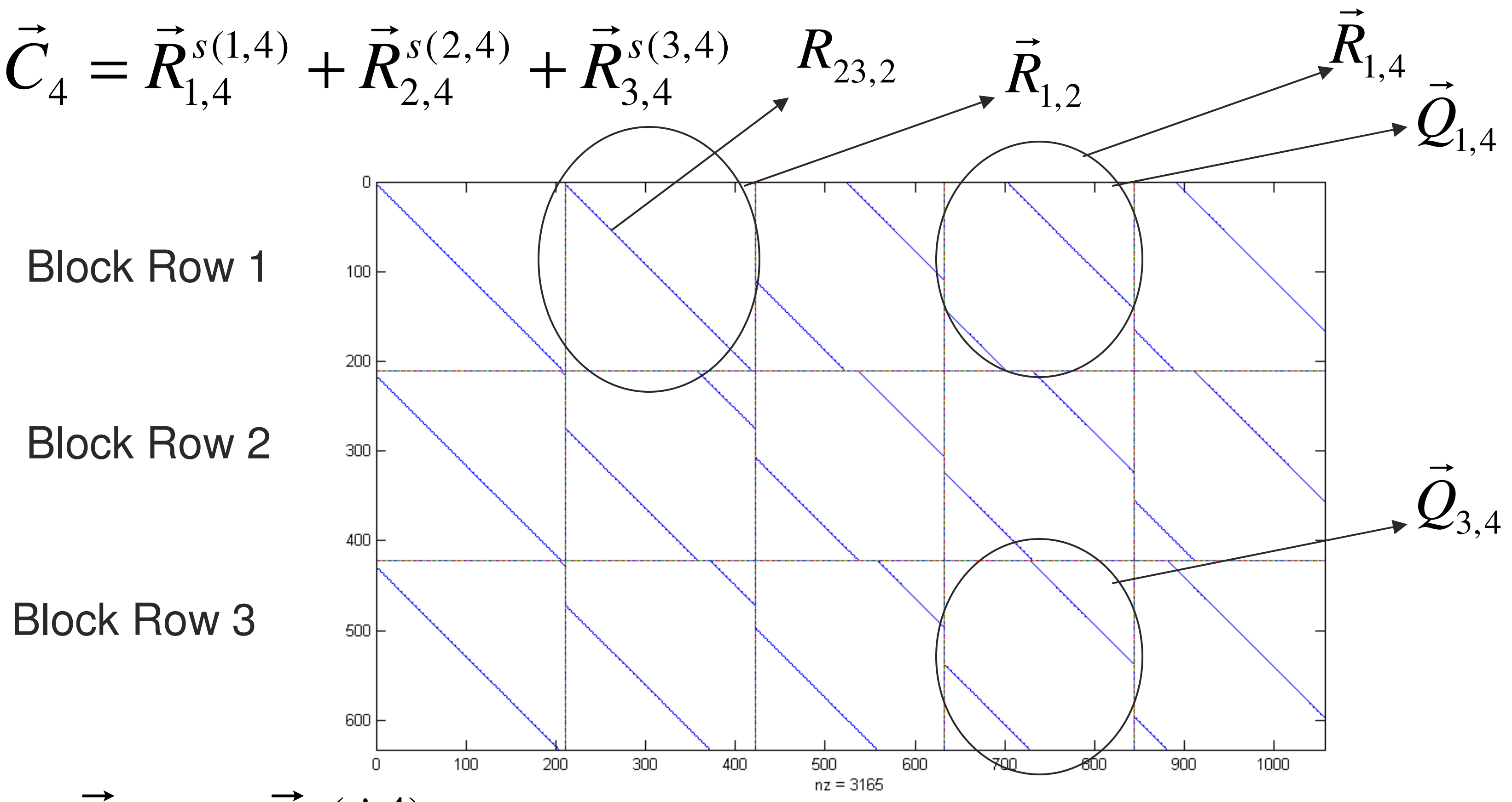
$$\vec{R}_{j,k} = \left[Rm_{1+(j-1)p,k}, \dots, Rm_{l+(j-1)p,k}, \dots, Rm_{p+(j-1)p,k}\right]^T$$

$$\vec{Q}_{j,k} = \left[Qm_{j,1+(k-1)p}, \dots, Qm_{j,l+(k-1)p}, \dots, Qm_{j,p+(k-1)p}\right]^T$$

Q Vector Computation

The bit nodes in each block column has support only on the one block column edges of the check nodes.

$$\vec{C}_4 = \vec{R}_{1,4}^{s(1,4)} + \vec{R}_{2,4}^{s(2,4)} + \vec{R}_{3,4}^{s(3,4)}$$



,

$$\vec{D}_{j,4} = \vec{R}_{j,4}^{s(j,4)}$$

$$\vec{Q}_{j,4} = \vec{C}_k - \vec{D}_{j,4} + \vec{\lambda}_k$$

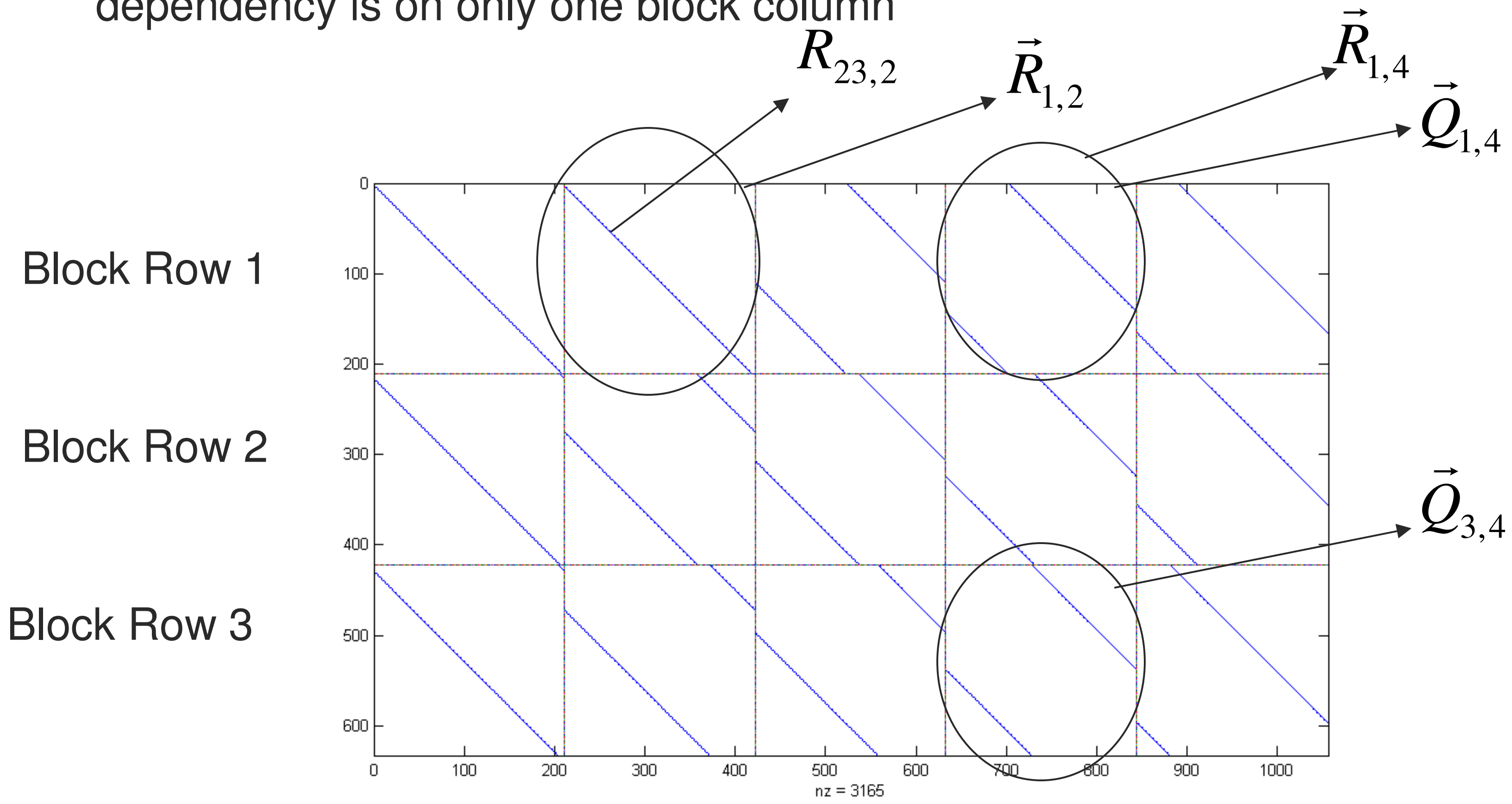
Check Node Processing

- *Partial State Computation of Block row of R messages is only dependent on the block column of Q messages if Block Serial Processing of multiple block columns of Q messages is used.*

- *Final State is only one vector for each Block row of R messages.*

R Vector Computation

The check nodes has support on all block columns. Once the computation is split into accumulate and subtraction stages, the dependency is on only one block column



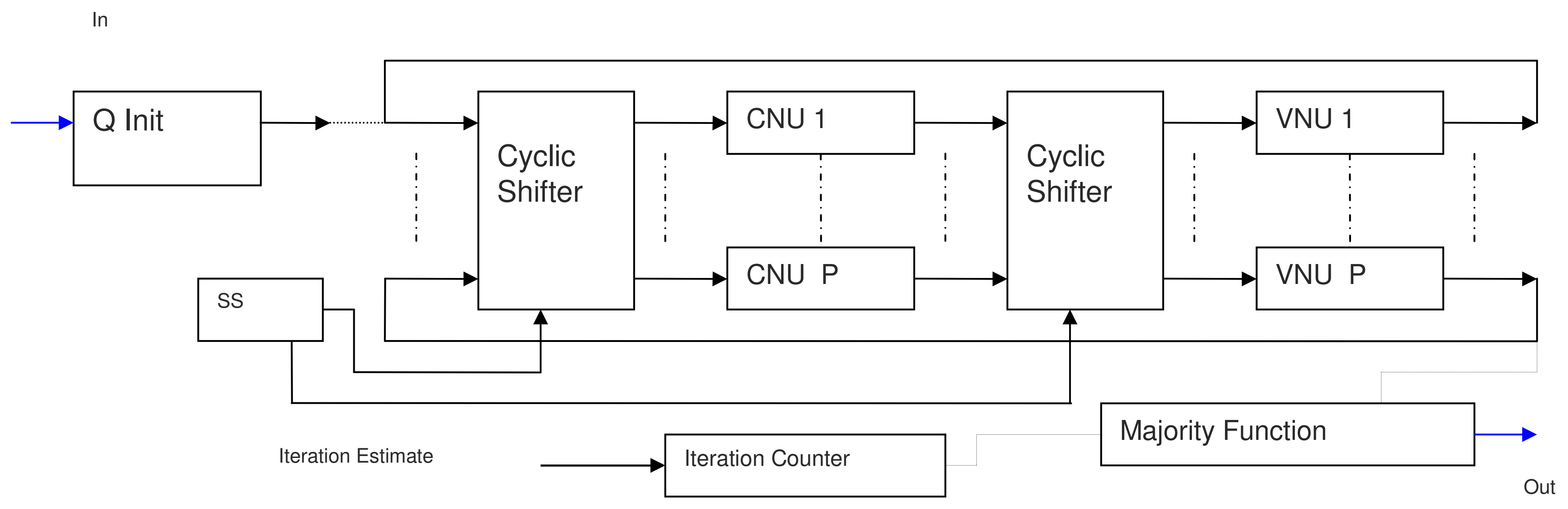
,

Proposed Scheduling schemes for no Message Passing Memory

- *Key Finding: Doing Computations in Block Column fashion and Block row serial fashion*
- *P serial CNU and VNU along with PxP Cyclic shifters [On-the-Fly Type 1].*
- *P*c serial CNU and P Parallel VNU: [On-the-Fly Type 2].*
- *Transpose to Key Finding: Doing Computations in Block row fashion and Block column serial fashion . P parallel CNU and P*r serial VNU*

  [On-the-Fly Type 3].

Proposed Architecture for Memory Less Decoding



Decoder for (3, 5) – structured LDPC code of length 1055. No message communication memory is needed. Possible due to structured property and scheduling.

[9] K. Gunnam, G. Choi and M. B. Yeary, "An LDPC Decoding Schedule for Memory Access Reduction", IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)

Proposed Micro Architecture for CNU min-sum

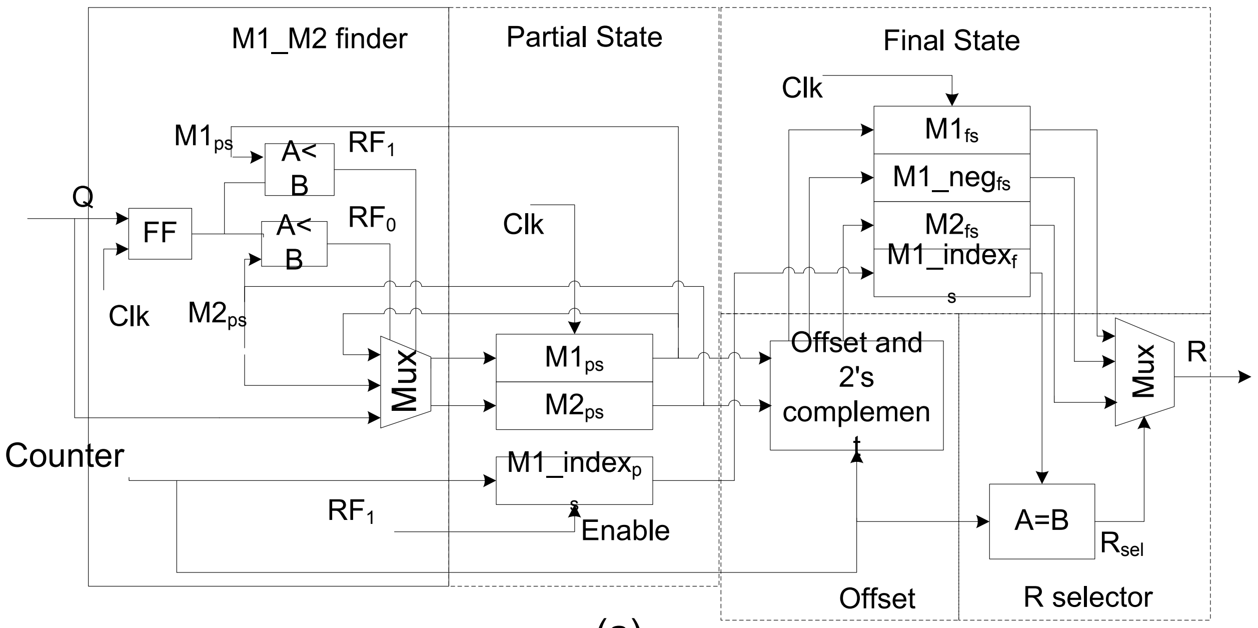
$$R_{cj,bi} = \psi^{-1}\left[\left(\sum_{i'=Row[cj][1]}^{Row[cj][c]} \psi\left(Q_{i',cj}\right)\right) - \psi\left(Q_{bi,cj}\right)\right].\delta(cj,bi)$$

$$R_{cj,bi} = \delta_{cj,bi}\kappa_{cj,bi}$$

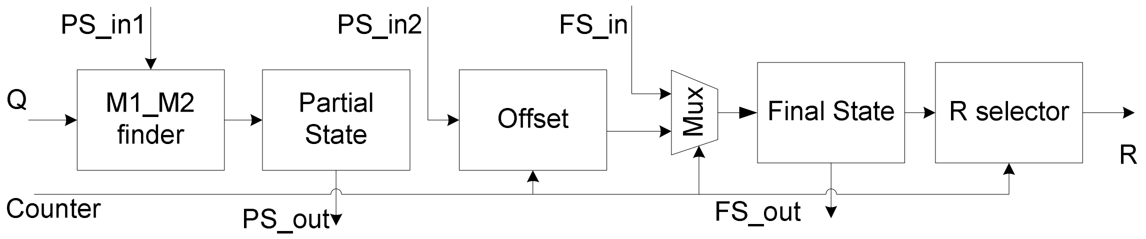
$$\kappa_{cj,bi} = \min_{i' \in Row[cj][bi] \setminus i} \left|Q_{i',cj}\right|.$$

- Simplifies the number of comparisons required as well as the memory needed to store CNU outputs.
- 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



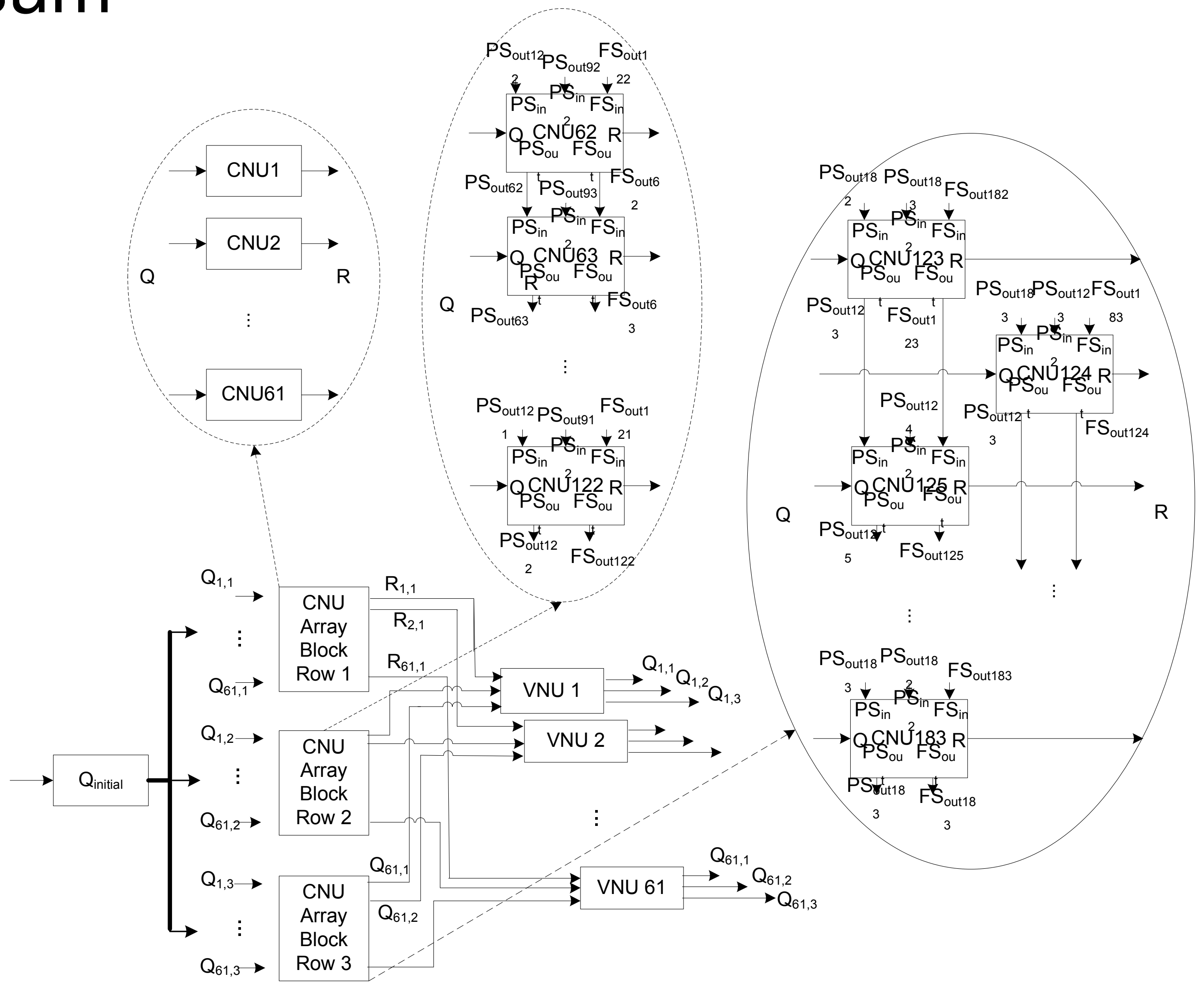
(a)



(b)

New micro architecture a) Static unit b) Dynamic Unit

On-the-fly Type 2 Architecture for min-sum



Block Serial Layered Decoder

Block Serial Layered Decoder

- The Block Serial Standard Message Passing Decoder architecture discussed in the previous slides (proposed in 2003) is used in high throughput applications using short length and regular LDPC codes.

- Block Serial Standard Message Passing Decoder is the one that is recently adapted in the industry (for read channel in hard disk drive electronics, for custom radio applications for NASA and DoD).

- Our recent work, Block Serial Layered Decoder and Block Parallel Layered Decoder are more efficient architectures for long length LDPC codes(for read channels in holographic storage, flash storage and hard disk drive electronics) and irregular LDPC codes (IEEE 802.11n, IEEE 802.16e) are discussed now!

Irregular QC-LDPC codes

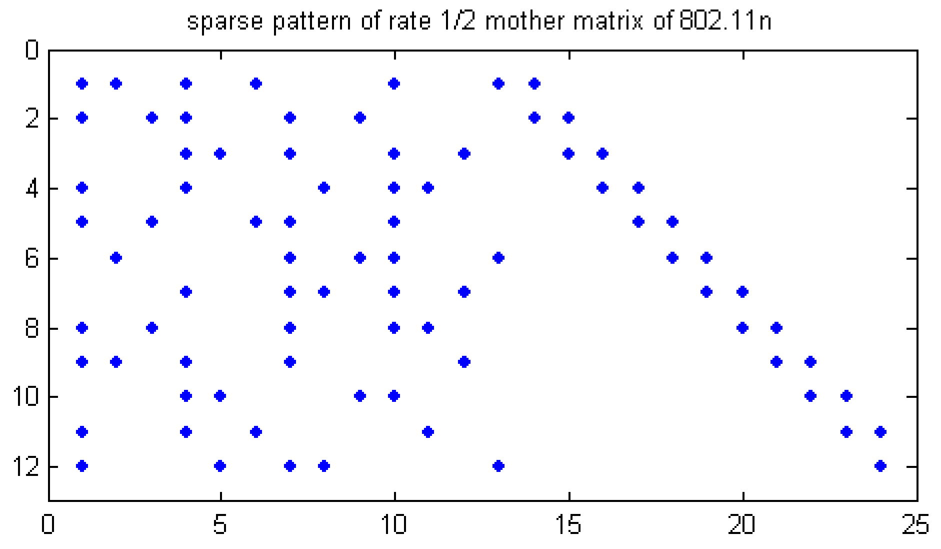
$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,n_b-2} & \mathbf{P}_{1,n_b-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2}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

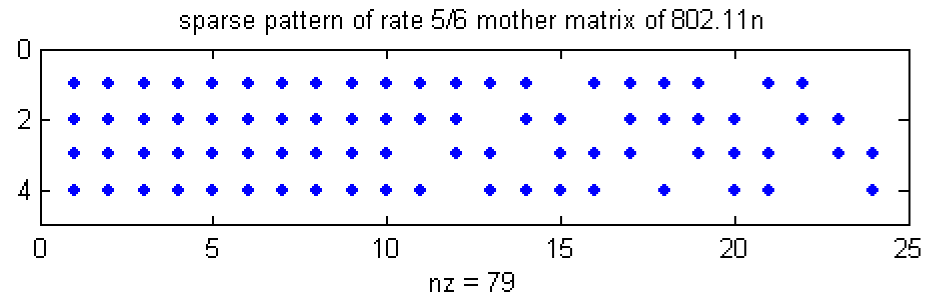
Different base matrices to support different rates.

Different expansion factors (z) to support multiple lengths.

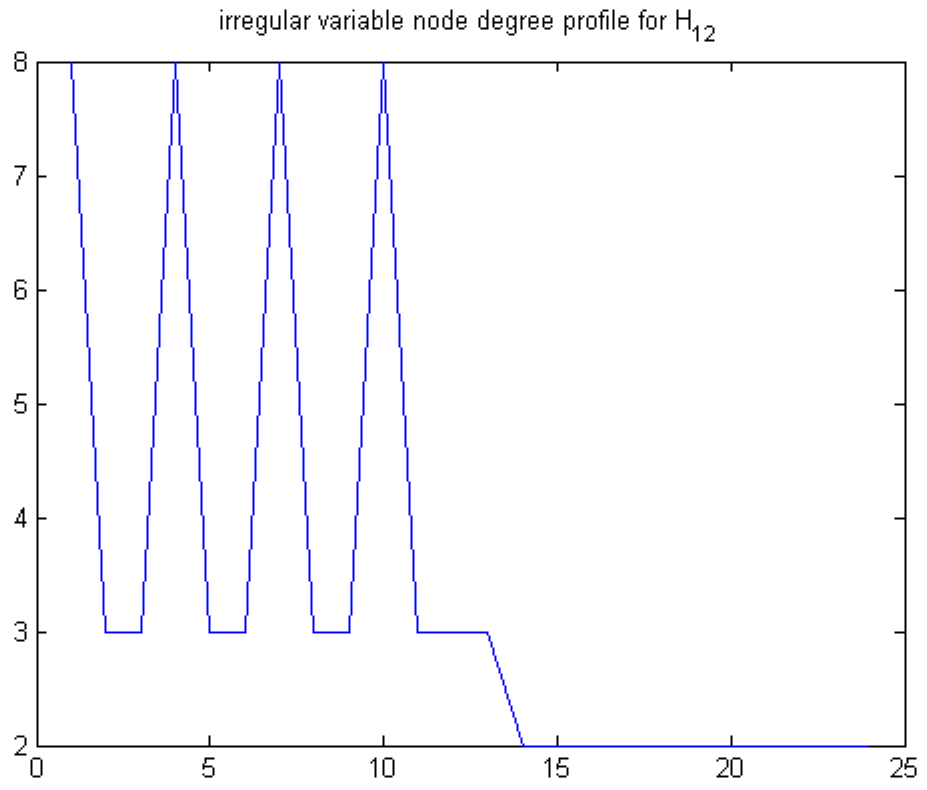
All the shift coefficients for different codes for a given rate are obtained from the same base matrix using modulo arithmetic

Irregular LDPC codes

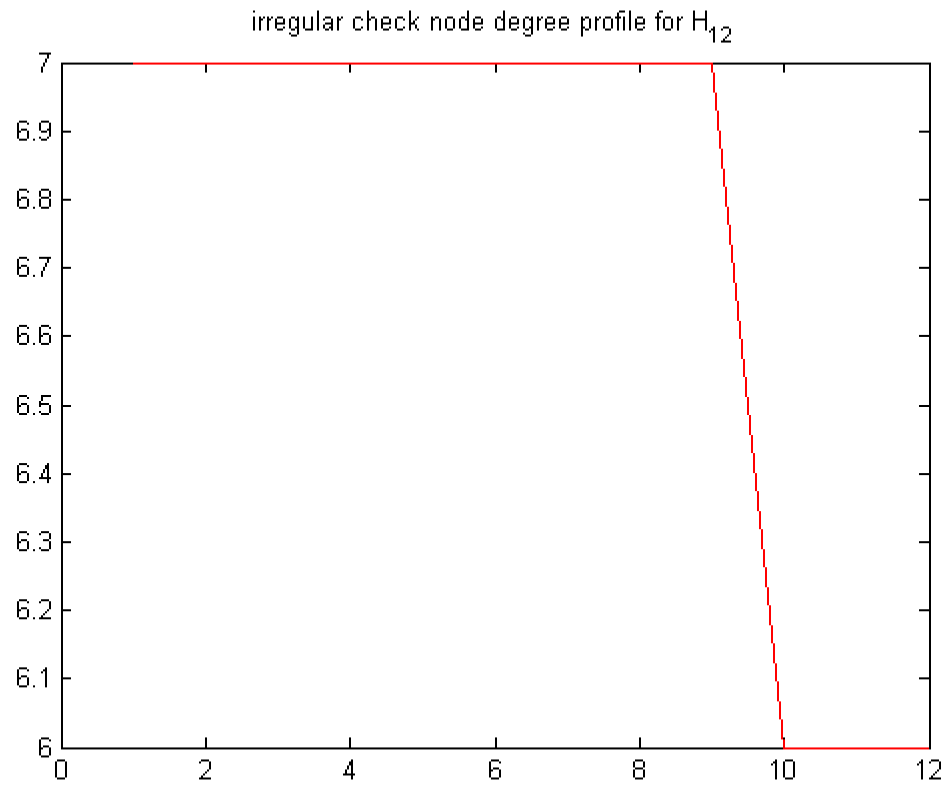




Irregular LDPC codes



Irregular LDPC codes



Irregular LDPC codes

- Existing implementations [12] 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.

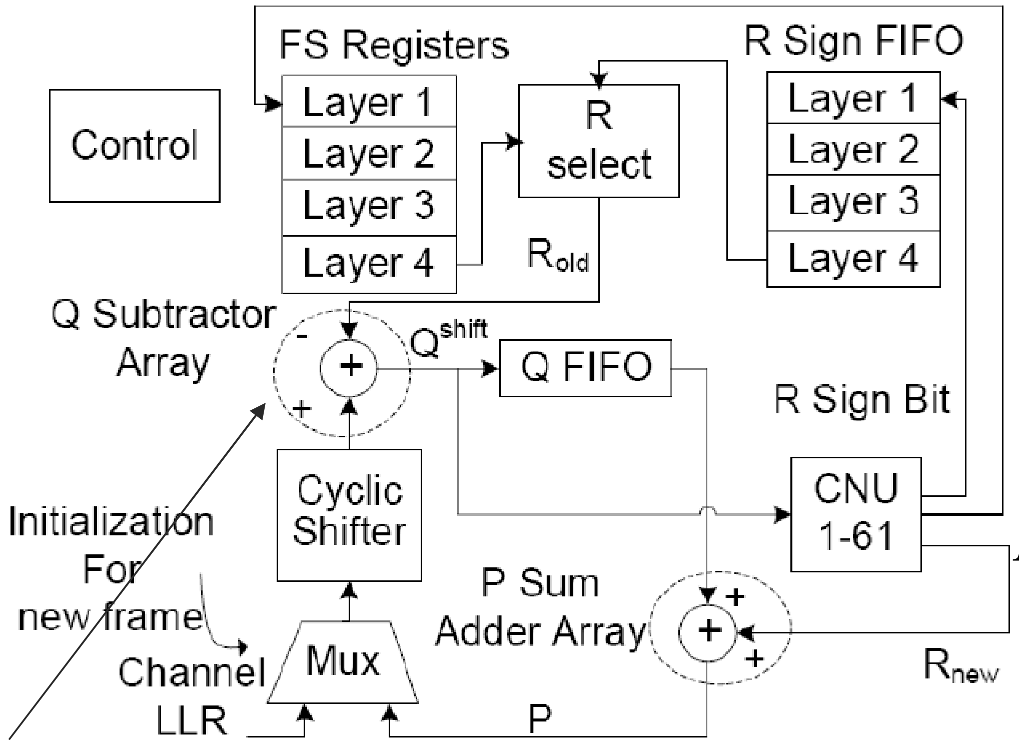
- It is anticipated that these codes will be the default choice for most of the standards.

- 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

[12] 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

New Dataflow Graph for Layered Decoding for regular mother matrices

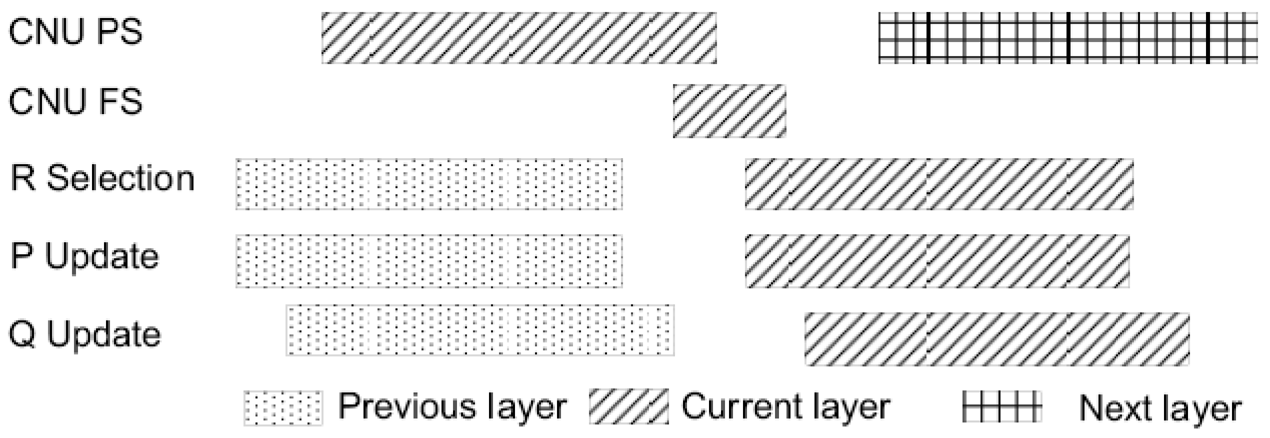


$$\vec{R}_{l,n}^{(i)} = f\left(\left[\vec{Q}_{l,n'}^{(i)}\right]^{S(l,n')}, \forall n' = 1,2,\cdots,k \right)$$

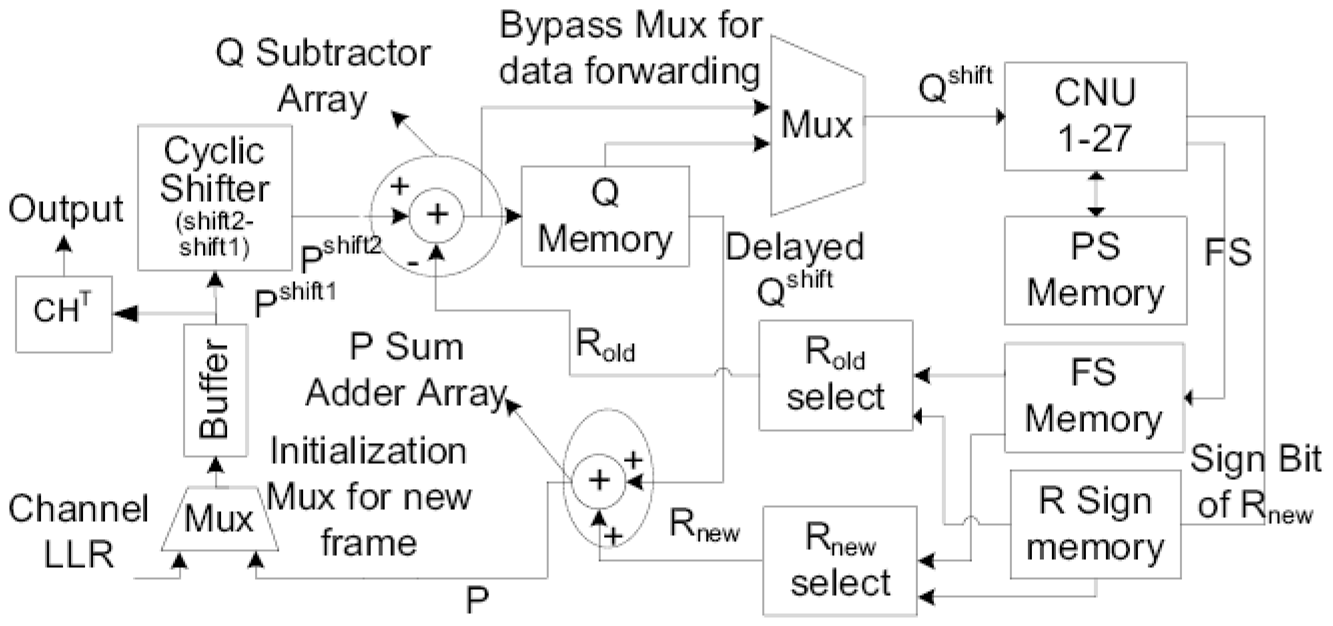
$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)}$$

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

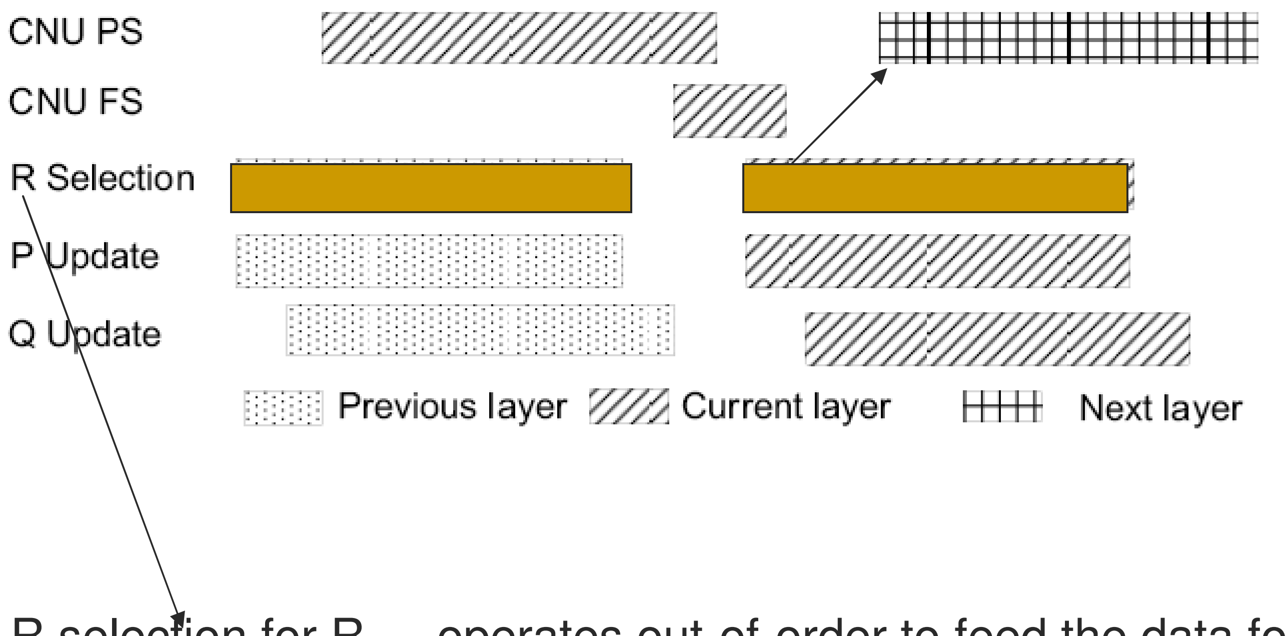
Decoder operation



New Dataflow Graph for Layered Decoding for irregular mother matrices

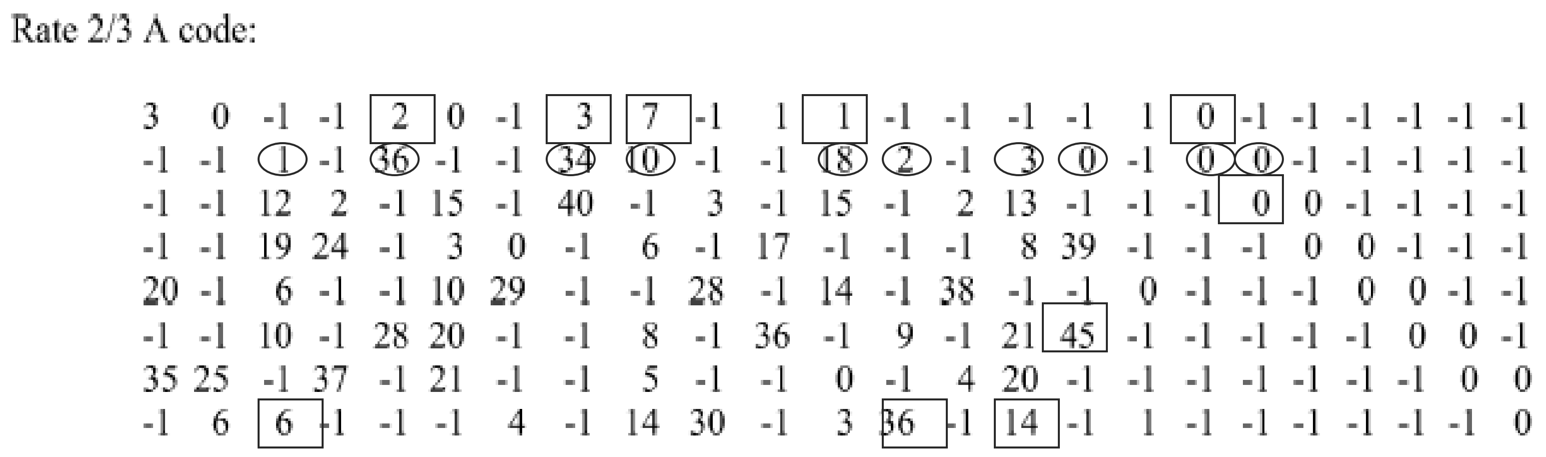


Pipeline for Irregular codes



R selection for R_{new} operates out-of-order to feed the data for PS processing of next layer

Out-of-order layer processing for R Selection



Rate 2/3 A code:

3	0	-1	-1	$\boxed{2}$	0	-1	$\boxed{3}$	$\boxed{7}$	-1	1	$\boxed{1}$	-1	-1	-1	-1	1	$\boxed{0}$	-1	-1	-1	-1	-1	-1
-1	-1	$\enclose{circle}{1}$	-1	$\enclose{circle}{36}$	-1	-1	$\enclose{circle}{34}$	$\enclose{circle}{10}$	-1	-1	$\enclose{circle}{18}$	$\enclose{circle}{2}$	-1	$\enclose{circle}{3}$	$\enclose{circle}{0}$	-1	$\enclose{circle}{0}$	$\enclose{circle}{0}$	-1	-1	-1	-1	-1
-1	-1	12	2	-1	15	-1	40	-1	3	-1	15	-1	2	13	-1	-1	-1	$\boxed{0}$	0	-1	-1	-1	-1
-1	-1	19	24	-1	3	0	-1	6	-1	17	-1	-1	-1	8	39	-1	-1	-1	0	0	-1	-1	-1
20	-1	6	-1	-1	10	29	-1	-1	28	-1	14	-1	38	-1	-1	0	-1	-1	-1	0	0	-1	-1
-1	-1	10	-1	28	20	-1	-1	8	-1	36	-1	9	-1	21	$\boxed{45}$	-1	-1	-1	-1	-1	0	0	-1
35	25	-1	37	-1	21	-1	-1	5	-1	-1	0	-1	4	20	-1	-1	-1	-1	-1	-1	-1	0	0
-1	6	$\boxed{6}$	-1	-1	-1	4	-1	14	30	-1	3	$\boxed{36}$	-1	$\boxed{14}$	-1	1	-1	-1	-1	-1	-1	-1	0

◯ PS processing   □ R selection

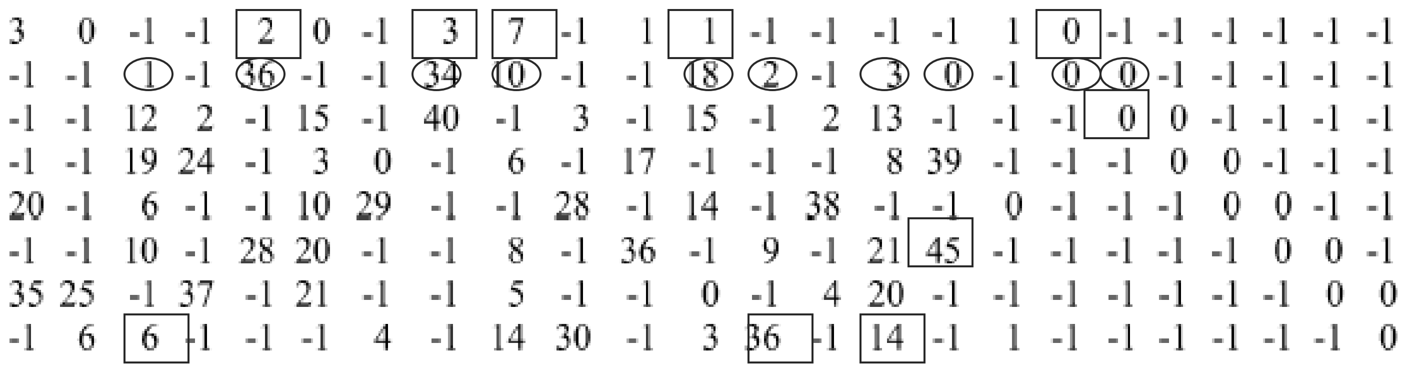
R selection is out-of-order so that it can feed the data required for the PS processing of the second layer.

So here we decoupled the execution of R new messages with the execution of CNU processing.

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

Out-of-order block processing for Partial State

Rate 2/3 A code:



3	0	-1	-1	$\boxed{2}$	0	-1	$\boxed{3}$	$\boxed{7}$	-1	1	$\boxed{1}$	-1	-1	-1	-1	1	$\boxed{0}$	-1	-1	-1	-1	-1	-1
-1	-1	$\textcircled{1}$	-1	$\textcircled{36}$	-1	-1	$\textcircled{34}$	$\textcircled{10}$	-1	-1	$\textcircled{18}$	$\textcircled{2}$	-1	$\textcircled{3}$	$\textcircled{0}$	-1	$\textcircled{0}$	$\textcircled{0}$	-1	-1	-1	-1	-1
-1	-1	12	2	-1	15	-1	40	-1	3	-1	15	-1	2	13	-1	-1	-1	$\boxed{0}$	0	-1	-1	-1	-1
-1	-1	19	24	-1	3	0	-1	6	-1	17	-1	-1	-1	8	39	-1	-1	-1	0	0	-1	-1	-1
20	-1	6	-1	-1	10	29	-1	-1	28	-1	14	-1	38	-1	-1	0	-1	-1	-1	0	0	-1	-1
-1	-1	10	-1	28	20	-1	-1	8	-1	36	-1	9	-1	21	$\boxed{45}$	-1	-1	-1	-1	-1	0	0	-1
35	25	-1	37	-1	21	-1	-1	5	-1	-1	0	-1	4	20	-1	-1	-1	-1	-1	-1	-1	0	0
-1	6	$\boxed{6}$	-1	-1	-1	4	-1	14	30	-1	3	$\boxed{36}$	-1	$\boxed{14}$	-1	1	-1	-1	-1	-1	-1	-1	0

◯ PS processing   □ R selection

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.

Block Parallel Layered Decoder

Block Parallel Layered Decoder

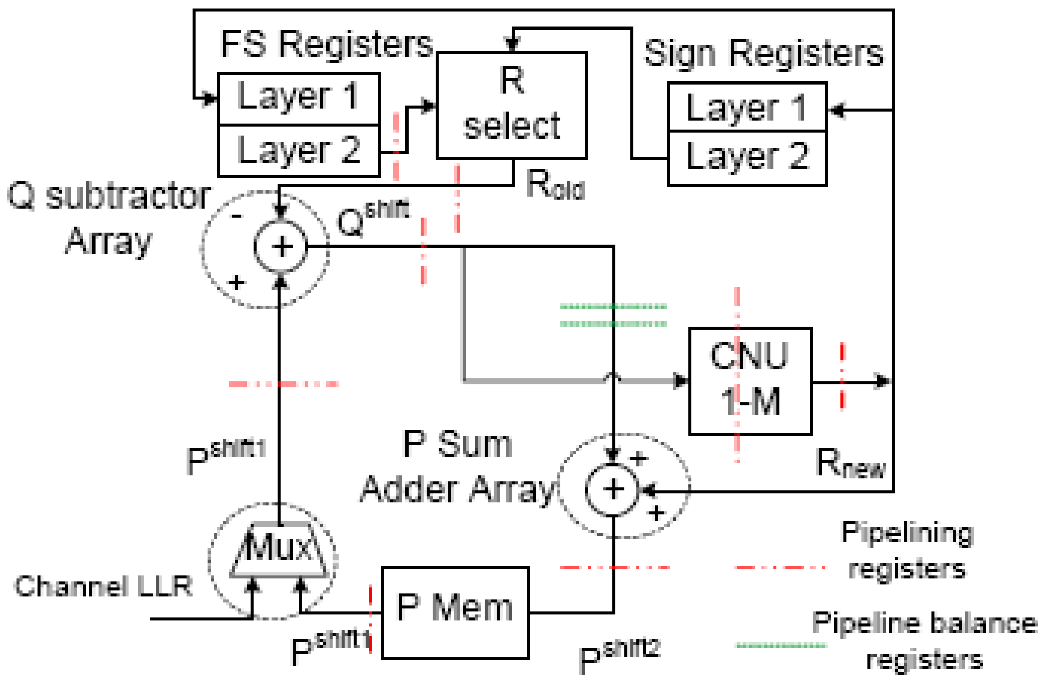
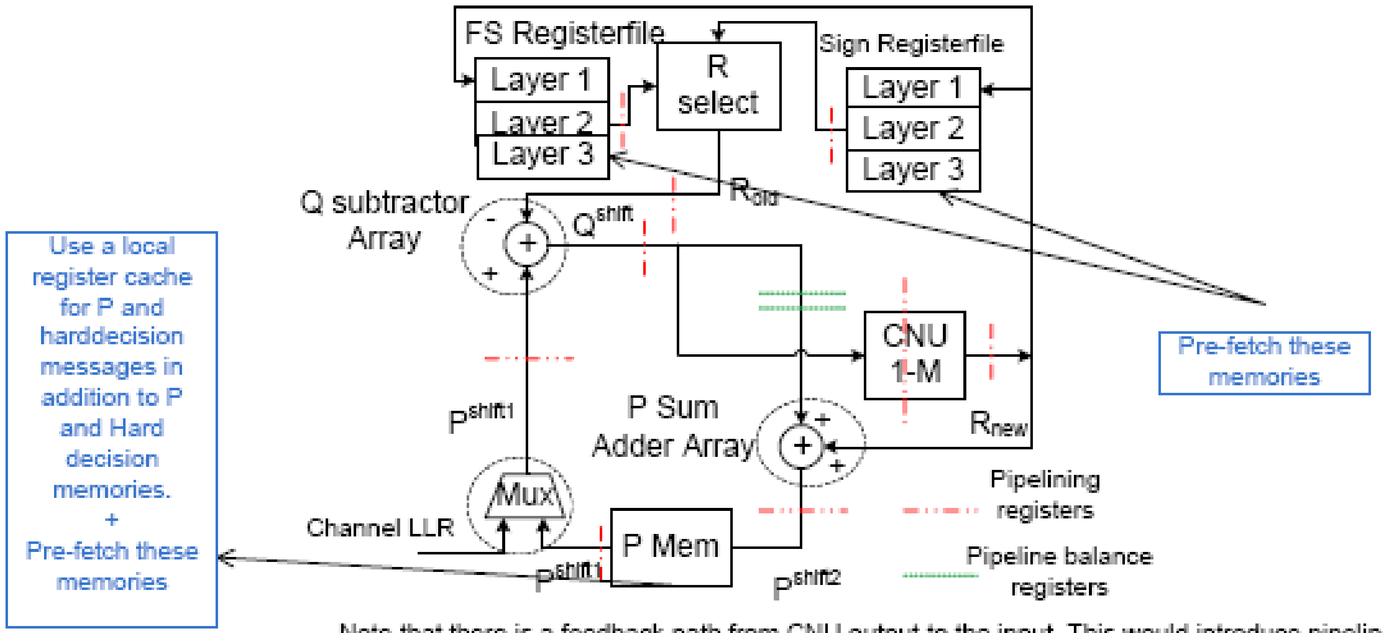


Figure.14: Semi-Parallel architecture for layered decoder. (Simply referred as parallel architecture)

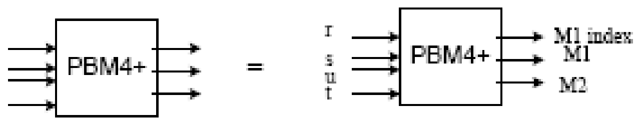


Note that there is a feedback path from CNU output to the input. This would introduce pipeline penalty if there are pipeline stages in the hardware. We propose the solution to handle the pipeline issue in a 3-way solution:

1. Pre-execution of Rold messages/Pre-fetching of FS memory and sign registers to enable timely calculation of Rold messages
2. Caching of P message and Harddecision messages in local register cache near the logic

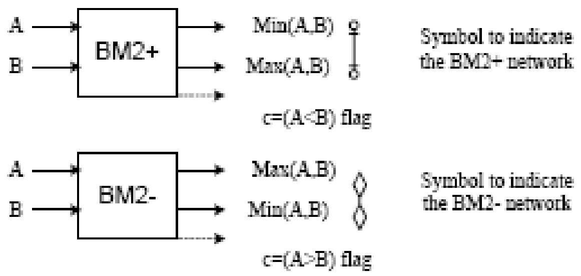
3. Enforcing independence: 3.1. If the shift difference for each circulant in the present layer with the circulant in the previous layer is more than N_p, then the last N_p rows and first N_p rows of adjacent rows are independent.
3.2. Processing of the N_p independent rows in the current layer which does not depend on the last N_p rows of the previous row. The selection of these independent rows can be obtained by off-line row re-ordering of the H matrix.
3.3. Processing of rows in partial manner thus leading to out-of-order processing. If there is a row with check node degree of d_c, we can do the CNU processing for that row with what ever information available. Possible dependencies will be resolved after the pipeline latency of the previous layer is accounted for. Essentially, now instead of using a CNU with d_c inputs we try to use a CNU with less inputs, say w. Note that in this case, the control is more complex.

Figure 15: a) Semi-Parallel Architecture, Pipeline solutions



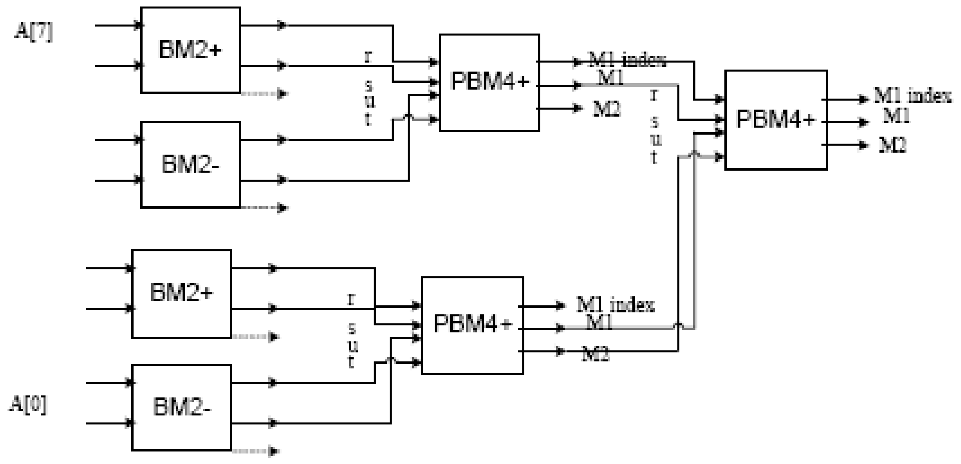
PBM4+ defined and symbol

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).
The Partial Bitonic Merge(PBM4+) circuit outputs min1(M1) and min2(M2) along with the min1 index (M1 index). Note that r has a index of 3, s has a index of 2, u has a index 1 and t has a index of 0.



A
B
BM2+
Min(A,B)
Max(A,B)
c=(A<B) flag
Symbol to indicate the BM2+ network
A
B
BM2-
Max(A,B)
Min(A,B)
c=(A>B) flag
Symbol to indicate the BM2- network

A)Basic building blocks, PBM4+, BM2+,BM2-



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

Figure 11a: Min1-Min2 finder using PBM4+

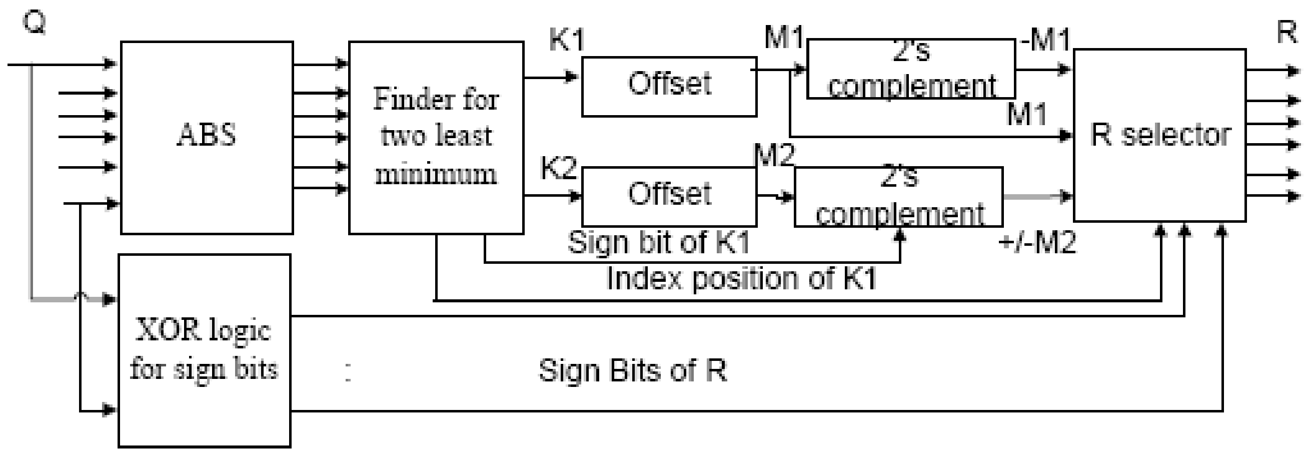
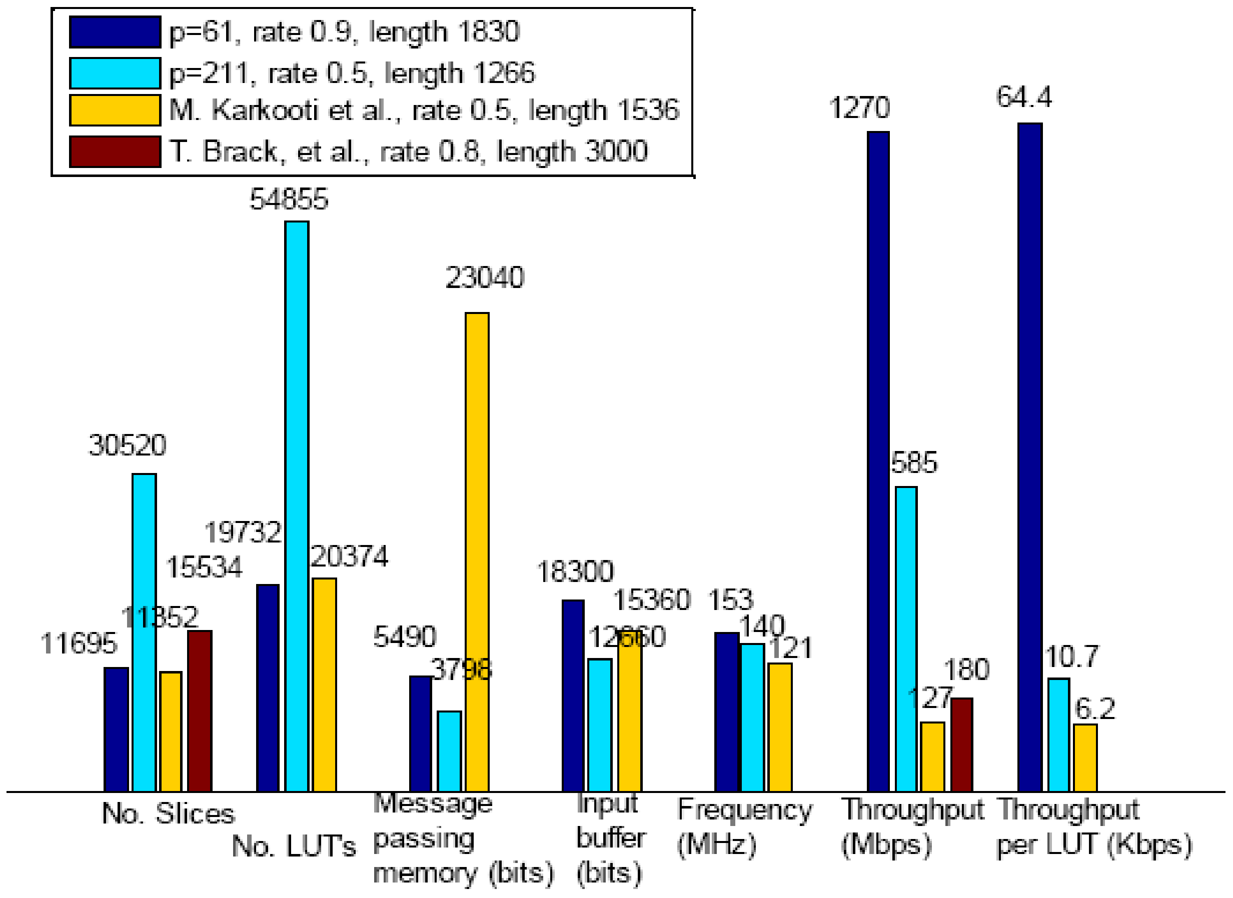


Figure.12: Parallel CNU based on Value-reuse property of OMS.

Results

FPGA Results for Block Serial Standard Message Passing Decoder



[4] Karkooti etal. Semi-parallel reconfigurable architectures for real-time LDPC decoding ; Proceedings. ITCC 2004.

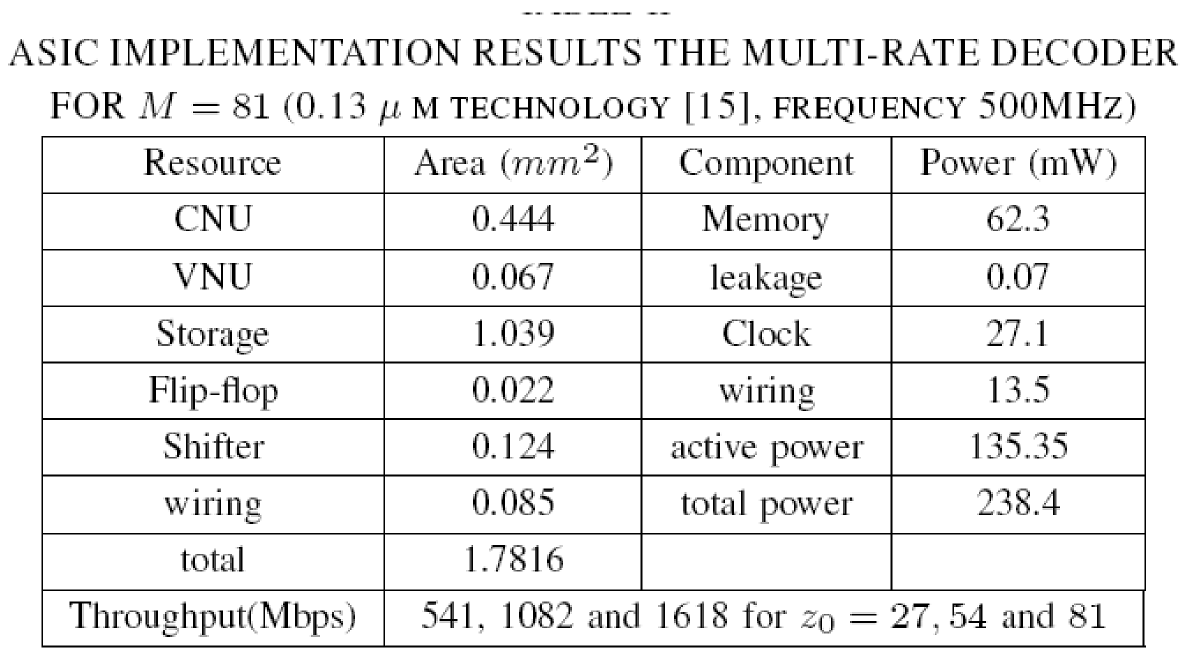
[10] T. Brack etal. "Disclosing the LDPC Code Decoder Design Space" Design, Automation and Test in Europe (DATE) Conference 2006, March 2006, Munich, Germany

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 for a throughput of 1.6 Gbps

[13] takes 375 K logic gates and 88452 RAM bits for memory for a throughput of 940 Mbps

[14] takes 195 K logic gates for pipelined implementation, plus 77, 760 bits memories. for a throughput of 1 Gbps

[13] Rovini, M.; L'Insalata, N.E.; Rossi, F.; Fanucci, L., "VLSI design of a high-throughput multi-rate decoder for structured LDPC codes," Digital System Design, 2005. Proceedings. 8th Euromicro Conference on , vol., no.pp. 202- 209, 30 Aug.-3 Sept. 2005
[14] Y.Sun, M. Karkooti and J. R. Cavallaro, "High Throughput, Parallel, Scalable LDPC Encoder/Decoder Architecture for OFDM Systems" Fifth IEEE Dallas Circuits and Systems Workshop: Design, Application, Integration and Software. Oct 2006, Dallas.

Contributions

- An area (logic and memory) and power efficient multi-rate architecture for standard message passing decoder of LDPC
- An area (logic and memory) and power efficient multi-rate architecture for Layered decoding of regular QC- LDPC (IEEE 802.3 10-GB Ethernet)
- An area (logic and memory) and power efficient multi-rate architecture for Layered decoding of irregular QC- LDPC for IEEE 802.11n (Wi-Fi), IEEE 802.16e(Wimax) and storage applications.
- An area (logic and memory) efficient parallel layered decoder for regular LDPC for storage and other applications
- FPGA prototyping and ASIC design clearly illustrates the advantages of the proposed decoder architectures

More Information

- 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

Acknowledgements

Research Collaborators:

Dr. Gwan Choi , Professor at Texas A&M

Dr. Mark Yeary, Professor at University of Oklahoma

Mr. Weihuang Wang, MS student at Texas A&M

Research Sponsors

- NASA
- ONR
- DoD
- Texas Instruments
- Intel
- Star Vision
- Schlumberger

- Thank you !