

Min Li (Penn. State University)

2nd Annual North American School of Information Theory, August 10-13, 2009

Problem Statement and Motivation

- Random LDPC codes are widely studied for the past few years, which are shown to have powerful error-correcting capability, i.e., only 0.0045 dB away from Shannon limit at rate $\frac{1}{2}$;
- Structured Irregular-Repeat-Accumulate (SIRA) LDPC codes have lower implementation complexity, which makes them more attractive for applications in practice than random LDPC codes;
- Two problems are solved here:
 - To construct multi-rate SIRA (MR-SIRA) LDPC codes with hardware-constraint;
 - To design an efficient **reconfigurable decoder** for multi-rate block-structure LDPC codes.

Key Tools and Rules

- Code Construction
 - Node Degree Distribution Optimization: Gaussian Approximation (GA) with Constraint;
 - Binary Mother Matrix Optimization: Progressive Edge Generation (PEG);
 - Objective Block Matrix Optimization: Optimal Cycle Distribution Rule
- Reconfigurable Decoder Design
 - Programmable processing modules
 - Programmable logic control modules
 - Minimum memory usage rule
 - Minimum logic gates rule
 - Scalable

SIRA LDPC Codes

- SIRA Codes
 - Simple structure
 - Composed of small blocks of circulant permutation matrix
 - Own double-diagonal property
 - Ensure fast encoding through accumulator
 - Allow low-complexity decoding scheme in practice
- Question:** How to fill in shifting index P_{ij} to obtain an H matrix with the optimal performance?
- $$E_H = \begin{bmatrix} P_{10} & P_{11} & \dots & P_{1,M-1} & 0 & \dots & -1 & -1 \\ P_{20} & P_{21} & \dots & P_{2,M-1} & 0 & 0 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{M-2,0} & P_{M-2,1} & \dots & P_{M-2,M-1} & \dots & \dots & \dots & \dots \\ P_{M-1,0} & P_{M-1,1} & \dots & P_{M-1,M-1} & -1 & -1 & \dots & 0 \end{bmatrix}_{M \times M}$$
- $$H_0 = [H_{00} \ H_{01} \ H_{02}]_{M \times N}$$

Hardware-constraint and Multi-rate Consideration

- Hardware-constraint Consideration
 - Carefully design node degree distribution to avoid very large degree of variable nodes, i.e., greater than 20, which will reduce the complexity of message updating; Carefully place non-zero shifting index to avoid too much overlapping at the same column of consecutive rows to facilitate the partially parallel decoding schedule;
- Multi-rate Consideration
 - Maintain the double-diagonal property for each rate; To construct the high rate code, it's a priority to fill in the same element as the lower rate code at the same position, which makes the new matrix look like a sub matrix of the previous one and ensures they share some common decoding architecture.

Code Construction Algorithm

- Key Steps**
- Given the objective code parameter such as rate, find the optimal degree distribution using Gaussian Approximation with hardware-constraint;
 - Given the node degree distribution, construct a binary mother matrix $M(H_0)$ to fix the non-zero positions in the matrix. Apply the improved PEG concept during this process in order to obtain an optimal mother matrix;
 - Construction of objective matrix $E(H)$ based on $M(H_0)$; this step is composed of several sub-steps, which will be detailed later;
 - Given $E(H)$, do cycle distribution analysis. Repeat Step 2 and Step 3 until the optimal objective matrix is found in a given number of trials.

Code Construction Algorithm (Cont.)

Construction of $E(H)$ based on $M(H_0)$:

```

* Initialization:
Fill in the double-diagonal elements with identity matrix;
MaxGirth = Girth = 0; MinNumOfCycles = NumOfCycles = 0;
* While (n < N*M)
j = Index of column with minimum weight among the remaining columns;
For i = 1: M
    If (M(i,j) != 0)
        For Pk = 0:L-1
            Update Girth and NumOfCycles by graph expansion;
            If (Girth > MaxGirth)
                Survive_Pk = Pk;
            else if (NumOfCycles < MinNumOfCycles)
                Survive_Pk = Pk;
            else
                Survive_Pk = Survive_Pk;
        end
    end
end
Pj = Survive_Pk; n = n + 1;
end
    
```

An Example

Code Parameters

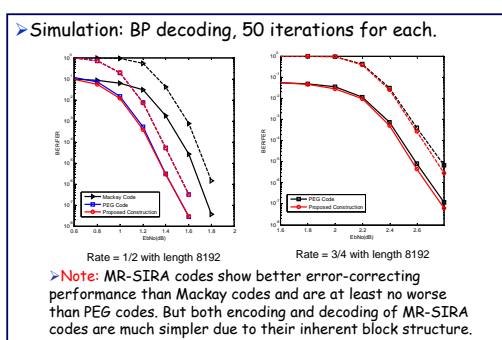
- $R = \frac{1}{2}$, $M=16$, $N=32$, $L=256$;
- Node degree distribution:
 - V node: $\lambda(x) = 0.2655x + 0.2389x^2 + 0.4956x^6$
 - C node: $\rho(x) = 0.9292x^6 + 0.0708x^7$

Construction Result

Cycle Length	6	8	10	12
Opt. Cyc. Dis.	6.53	42.46	50.91	0.1
PEG	0	77.97	22.63	0
Our Code	0	53.13	46.87	0

(8192, 4096) S-IRA LDPC Code Example

Performance of Our Codes



Efficient Decoding for Our Codes

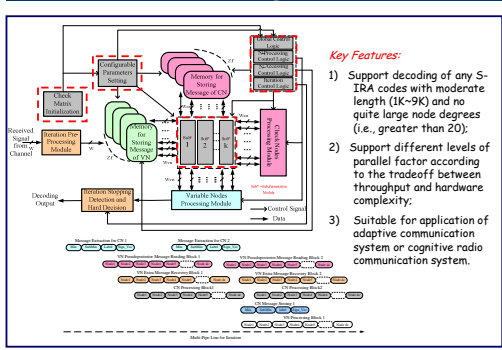
Layered-Shuffle BP-Based Decoding

Algorithm Description

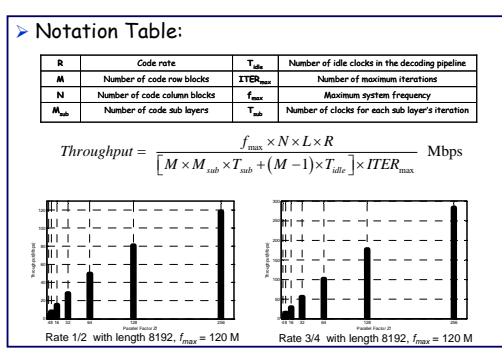
- Initialize C_{sub} .
- Initialize C_{sub} to $(M \times N)$ matrix C_{sub} .
- For $n = 1: M$
 - For $i = 1: N$
 - Update $C_{sub}(i, n)$ to $C_{sub}(i, n) + (C_{sub}(i, n) - 1) \times R_{sub}(i, n)$.
- For each sub-layer k to M
 - For each sub-layer k to M
 - Update $C_{sub}(k, n)$ to $C_{sub}(k, n) + (C_{sub}(k, n) - 1) \times R_{sub}(k, n)$.
- For each sub-layer k to M
 - Update $C_{sub}(k, n)$ to $C_{sub}(k, n) + (C_{sub}(k, n) - 1) \times R_{sub}(k, n)$.

Sub-layers (layers) are scheduled sequentially; Once each sub-layer finishes updating, the variable nodes connected with this layer are updated immediately so that the latest information from check nodes is effectively propagated.

Reconfigurable Decoder Design



Throughput Evaluation



Conclusions

- Conclusions:
- The proposed MR-SIRA codes and reconfigurable decoder design are attractive for applications that require both excellent code performance and low implementation complexity;
 - Possible Work: (1) Structured codes and implementation issues of decoding for LDPC on $Gf(q)$; (2) Application of BP to compressive sensing; (3) Extended application of LDPC codes, i.e., cooperative network coding scenario.
- (Note: This work was my previous research which had been done in early 2008 before I joined in WCAN.)