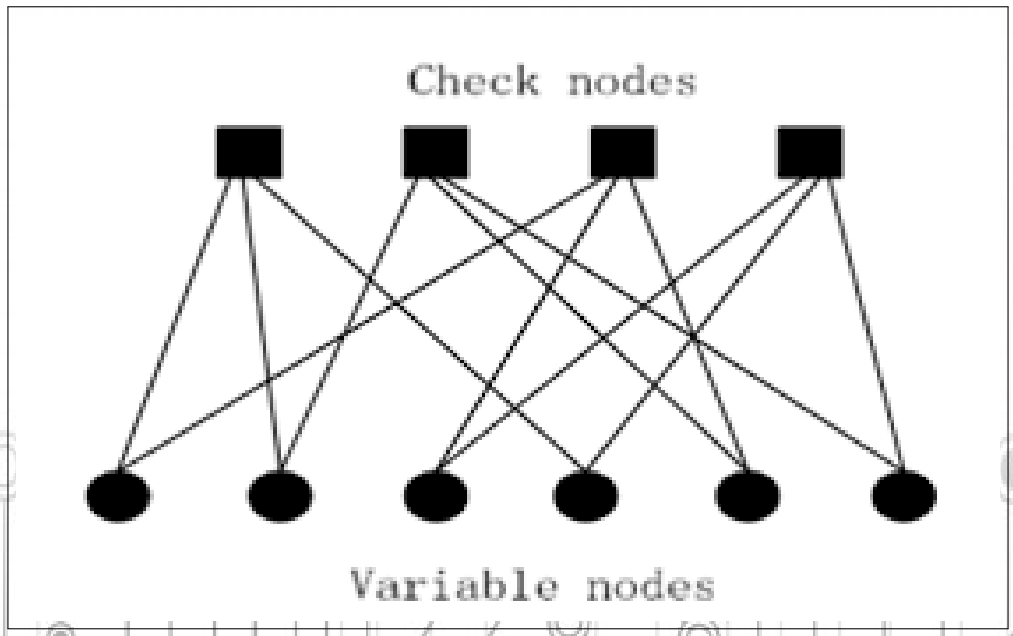
Science and Technology Council
Indian Institute of Technology, Kanpur

An FPGA Implementation of (3, 6)-Regular Low-Density Parity-Check Code Decoder



Final Report

July 18, 2020

Contents

1 Abstract

Because of their excellent error-correcting performance, low-density parity-check (LDPC) codes have recently attracted a lot of attention. In this project, we are interested in the practical LDPC code decoder hardware implementations. The direct fully parallel decoder implementation usually incurs too high hardware complexity for many real applications, thus partly parallel decoder design approaches that can achieve appropriate trade-offs between hardware complexity and decoding throughput are highly desirable. Applying a joint code and decoder design methodology, we develop a high-speed (3, k)-regular LDPC code partly parallel decoder architecture based on which we implement a 9216-bit, rate-1/2 (3, 6)-regular LDPC code decoder on Xilinx FPGA device. We have only simulated and tested the entire codes using the testbench files, since we were not able to test it on real hardware due to lockdown constraints.

2 Acknowledgements

Our team would like to express their gratitude to Mr. Mudit Agrawal, General Secretary, Science and Technology Council, Student's Gymkhana, Project Mentor Mr. Vaibhav Thakkar and to the Electronics Club giving us invaluable guidance and support in doing this project.

3 Introduction

Low-density parity-check (LDPC) codes are a class of linear block codes. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's. Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore are they suited for implementations that make heavy use of parallelism. They have been widely considered as next-generation error-correcting codes for telecommunication and magnetic storage.

Defined as the null space of a very sparse M×N parity-check matrix H, an LDPC code is typically represented by a bipartite graph, usually called Tanner graph, in which one set of N variable nodes corresponds to the set of codeword, another set of M check nodes corresponds to the set of parity-check constraints and each edge corresponds to a nonzero entry in the parity-check matrix H. (A bipartite graph is one in which the nodes can be partitioned into two sets, X and Y, so that the only edges of the graph are between the nodes in X and the nodes in Y.) An LDPC code is known as (j, k)-regular LDPC code if each variable node has the degree of j and each check node has the degree of k, or in its parity-check matrix each column and each row have j and k nonzero entries, respectively. The code rate of a (j, k)-regular LDPC code is 1-j/k provided that the parity check matrix has full rank. LDPC codes can be effectively decoded by the iterative belief-propagation (BP) algorithm.

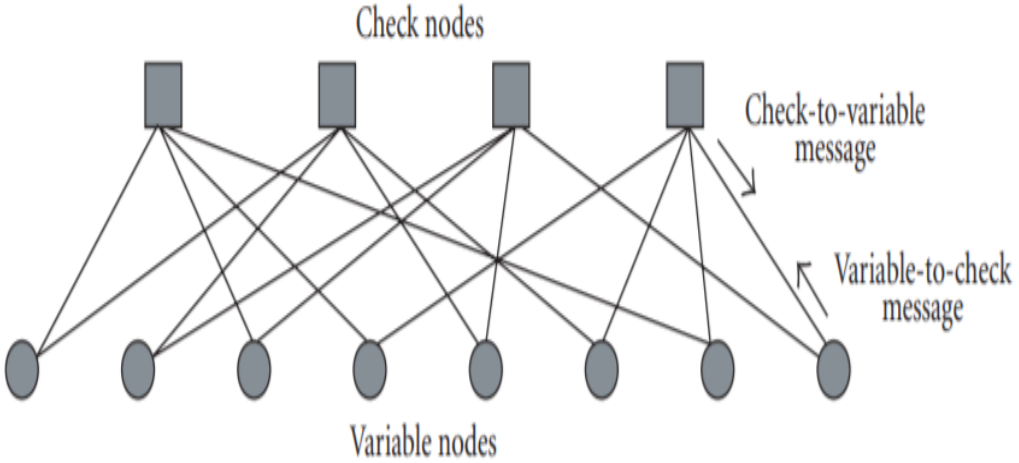
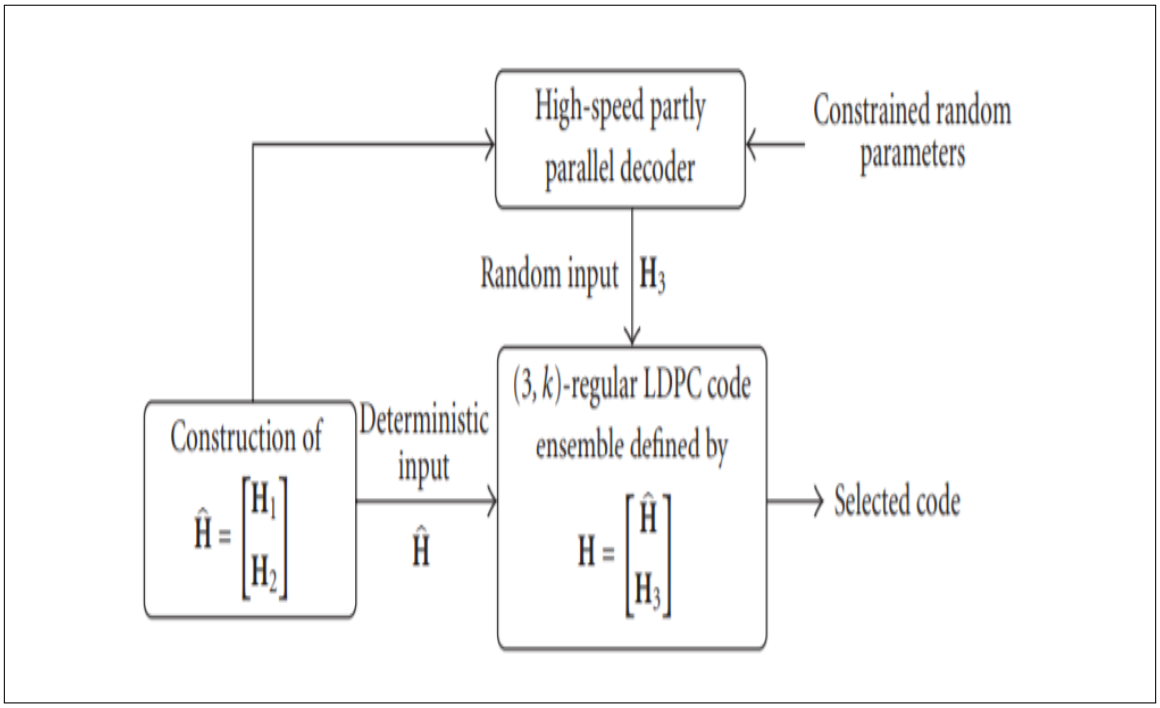


FIGURE 1: Tanner graph representation of an LDPC code and the decoding messages flow.

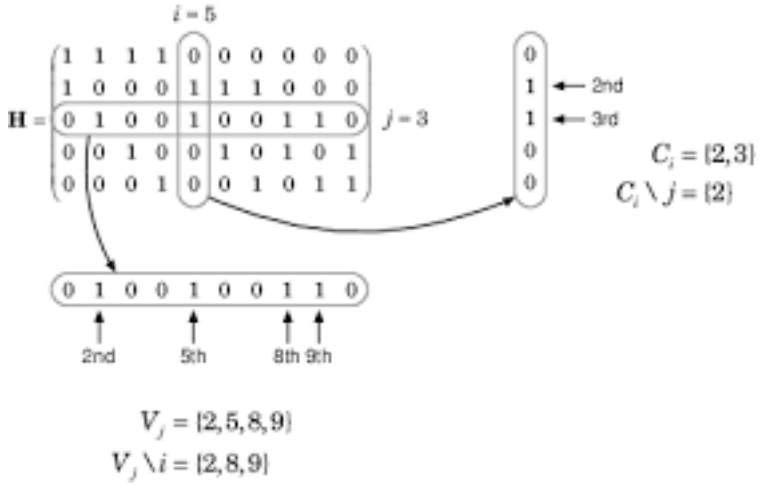
For any given LDPC code, due to the randomness of its Tanner graph, it is nearly impossible to directly develop a high-speed partly parallel decoder architecture. To circumvent this difficulty, we have used a decoder-first code design methodology: instead of trying to conceive the high-speed partly parallel decoder for any given random LDPC code, use an available high-speed partly parallel decoder to define a constrained random LDPC code. We may consider it as an application of the well-known "Think in the reverse direction" methodology. Inspired by the decoder-first code design methodology, we have used a joint code and decoder design methodology in [14] for (3, k)-regular LDPC code partly parallel decoder design. By jointly conceiving the code construction and partly parallel decoder architecture design, we presented a (3, k)-regular LDPC code partly parallel decoder structure in [14], which not only defines very good (3, k)-regular LDPC codes but also could potentially achieve high-speed partly parallel decoding.

Applying the joint code and decoder design methodology, we develop an elaborate (3, k)-regular LDPC code high-speed partly parallel decoder architecture based on which we implement a 9216-bit, rate-1/2 (3, 6)-regular LDPC code decoder using Xilinx Virtex FPGA (Field Programmable Gate Array) device.



4 Decoding Algorithms: Belief Propagation

These algorithms are called message passing algorithms, and are iterative algorithms. The reason for their name is that in each round of the algorithms, messages are passed from variable nodes to check nodes, and from check nodes back to variable nodes. The messages from variable nodes to check nodes are computed based on the observed value of the variable node and some of the messages passed from the neighboring check nodes to that message node. An important aspect is that the message that is sent from a variable node v to a check node c must NOT take into account the message sent in the previous round from c to v. The same is true for messages passed from check nodes to message nodes.



Before the description of Log-BP decoding algorithm, we introduce some definitions as follows. Let H denote the M×N sparse parity-check matrix of the LDPC code and

$$H_{(i,j)}$$

denote the entry of H at the position (i, j). We define the set of bits n that participate in parity-check m as

$$N(m) = n : H_{(m,n)} = 1,$$

and the set of parity-checks m in which bit n participates as

$$M(n) = m : H_{(m,n)} = 1$$

We denote the set N(m) with bit n excluded by N(m)\n, and the set M(n) with parity-check m excluded by M(n)\m.

Input: The prior probabilities $p_n^0 = P(x_n = 0)$ *and* $p_n^1 = P(x_n = 1) = 1 - p_n^0$, $n = 1, \cdots, N$;

Output: Hard decision $\hat{\mathbf{x}} = \{\hat{x}_1, \cdots, \hat{x}_N\}$;

Procedure:

1. *Initialization: For each* n, *compute the intrinsic (or channel) information* $\gamma_n = \log \frac{p_n^0}{p_n^1}$ *and for each* $(m, n) \in \{(i, j) | H_{i,j} = 1\}$, *compute*

$$\alpha_{m,n} = sign(\gamma_n) \log\left(\frac{1 + e^{-|\gamma_n|}}{1 - e^{-|\gamma_n|}}\right), \ where \ \ sign(\gamma_n) = \begin{cases} +1 & \gamma_n \geq 0 \\ -1 & \gamma_n < 0 \end{cases}.$$

2. *Iterative Decoding*

   - *Horizontal (or check node processing) step: For each* m, n, *compute*

$$\beta_{m,n} = \log\left(\frac{1 + e^{-\alpha}}{1 - e^{-\alpha}}\right) \prod_{n' \in \mathcal{N}(m) \backslash n} sign(\alpha_{m,n'}), \tag{1}$$

     where $\alpha = \sum_{n' \in \mathcal{N}(m) \backslash n} |\alpha_{m,n'}|$.

   - *Vertical (or variable node processing) step: For each* m, n, *compute*

$$\alpha_{m,n} = sign(\gamma_{m,n}) \log\left(\frac{1 + e^{-|\gamma_{m,n}|}}{1 - e^{-|\gamma_{m,n}|}}\right), \tag{2}$$

     where $\gamma_{m,n} = \gamma_n + \sum_{m' \in \mathcal{M}(n) \backslash m} \beta_{m',n}$. *For each* n, *update the "pseudo-posterior log-likelihood ratio (LLR)"* λ_n *as:*

$$\lambda_n = \gamma_n + \sum_{m \in \mathcal{M}(n)} \beta_{m,n}. \tag{3}$$

- *Decision step.*

   (a) *Perform hard decision on* $\{\lambda_1, \cdots, \lambda_N\}$ *to obtain* $\hat{\mathbf{x}} = \{\hat{x}_1, \cdots, \hat{x}_N\}$ *such that* $\hat{x}_n = 0$ *if* $\lambda_n > 0$ *and* $\hat{x}_n = 1$ *if* $\lambda \leq 0$;

   (b) *If* $\mathbf{H} \cdot \hat{\mathbf{x}} = 0$, *then algorithm terminates, else go to Horizontal step. A failure will be declared if pre-set maximum number of iterations occurs without successfully decoding.* ■

5 Our current approach

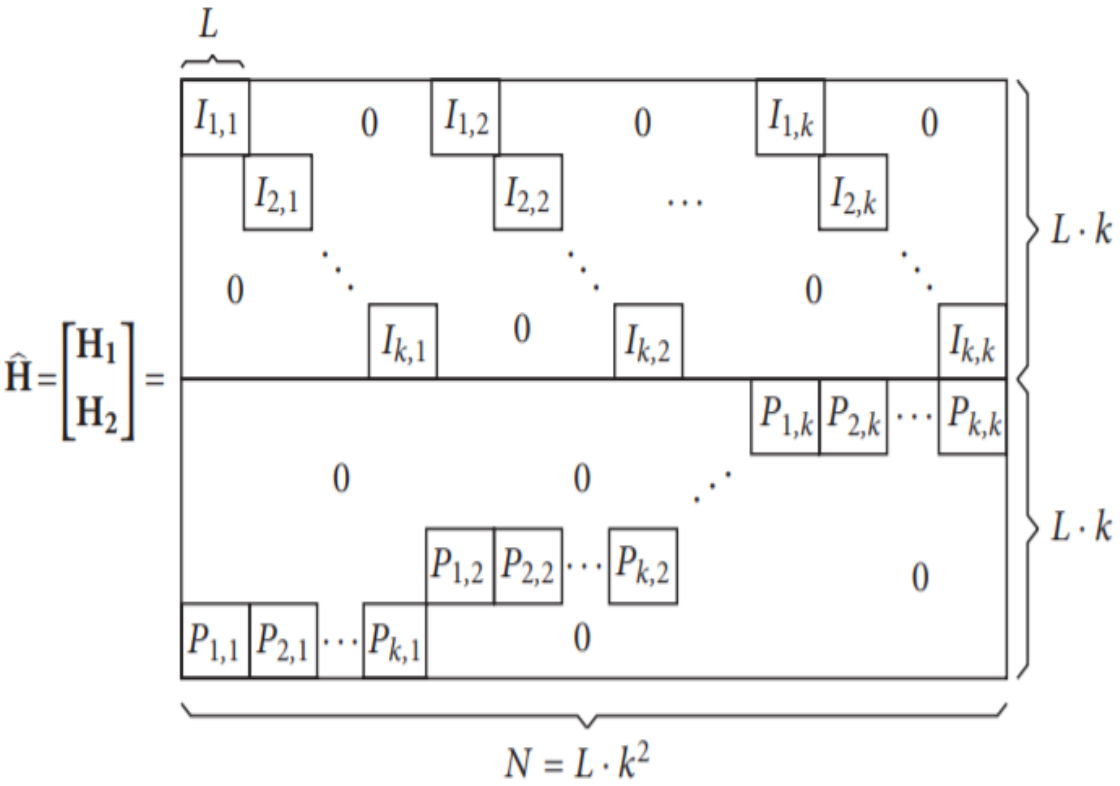
5.1 Joint (3,k)-Regular LDPC Code and Decoder/Encoder Design

We first explicitly construct a high-girth (2,k)-regular LDPC code that exactly fits to a high-speed partly parallel (2,k) - regular LDPC decoder, then extend this decoder to a (3,k)-regular LDPC decoder that is configured by a set of constrained random parameters and defines a (3,k)-regular LDPC code ensemble. Each code in such code ensemble is essentially constructed by randomly inserting certain check nodes into the deterministic high-girth (3,k)-regular LDPC code under the constraint specified by the decoder. Thus it is reasonable to expect that the codes in this ensemble most likely don't contain too many short cycles and the selected code is prone to assume good performance.

1. Explicitly construct two matrices H_1 and H_2 in such a way that

   $$\hat{H} = [H_1^T, H_2^T]^T$$

   defines a (2, k)-regular LDPC code C_2 whose Tanner graph has the girth of 12.

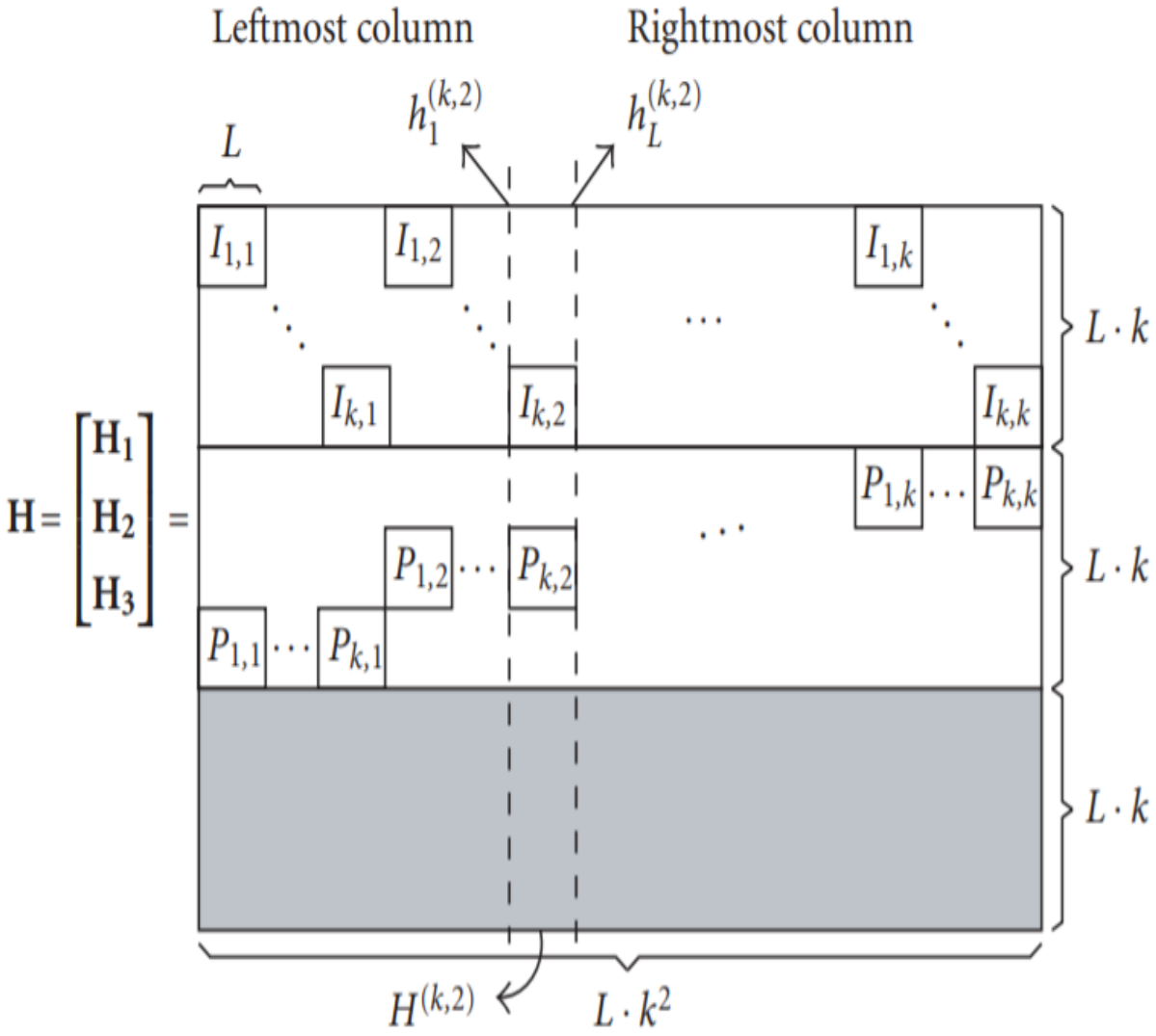


2. Develop a partly parallel decoder that is configured by a set of constrained random parameters and defines a (3, k)-regular LDPC code ensemble, in which each code is a subcode of C_2 and has the parity-check matrix

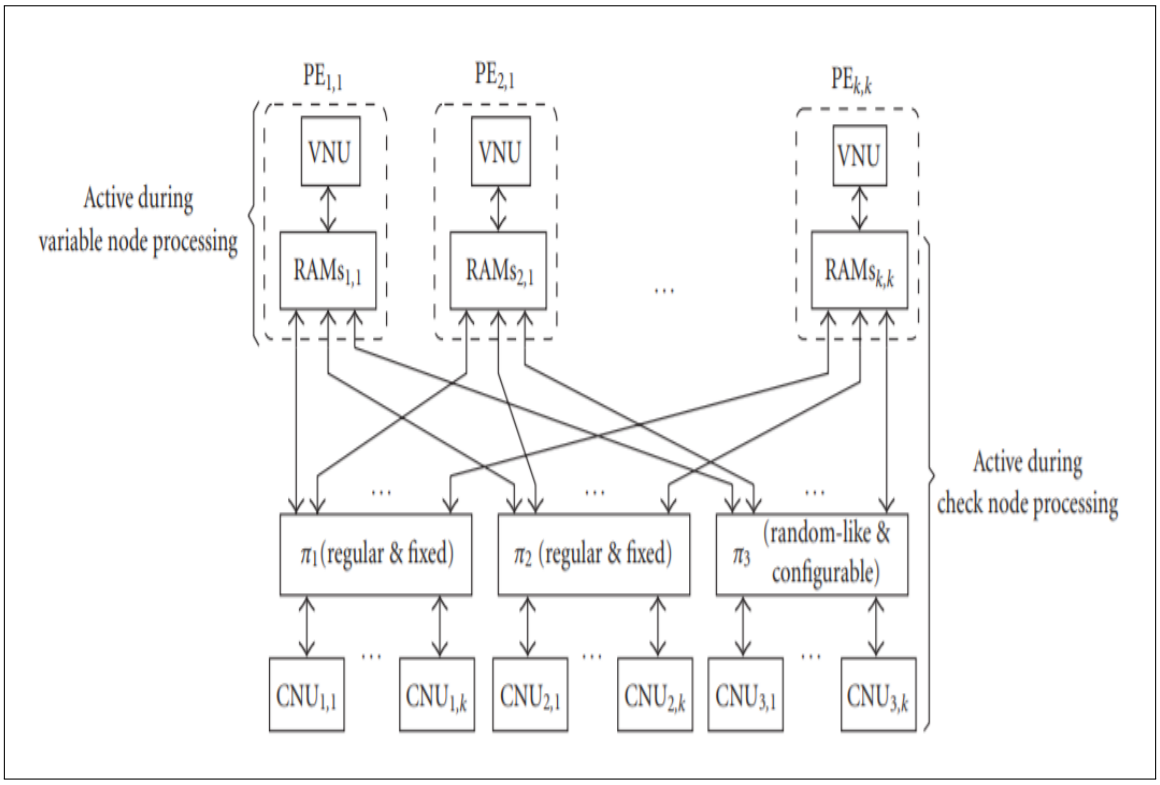
   $$H = [\hat{H}^T, H_3^T]^T$$

.

3. Select a good (3, k)-regular LDPC code from the code ensemble based on the criteria of large Tanner graph average cycle length and computer simulations. Typically the parity-check matrix of the selected code has only few redundant checks, so we may assume that the code rate is always 1-3/k.



5.2 Architecture



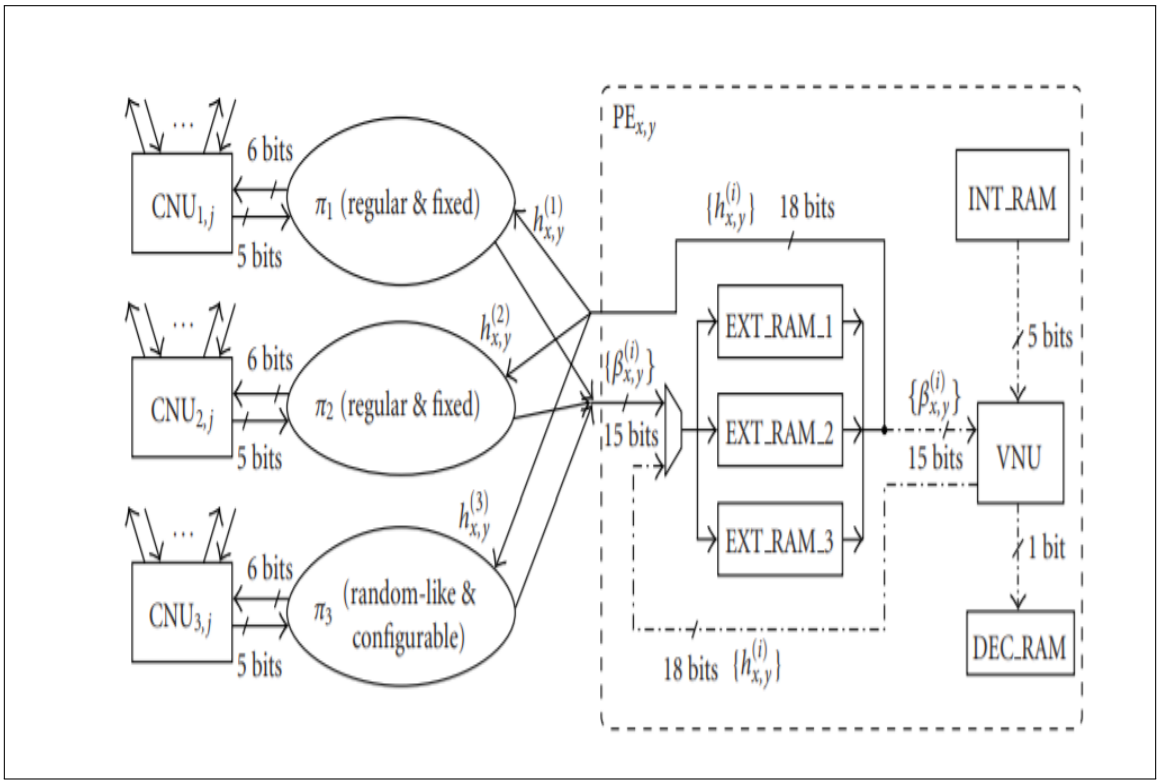
1. Memory Banks: Each PE block $PE_{x,y}$ contains five RAM blocks: EXT-RAM-i for i = 1, 2, 3, INT-RAM, and DEC-RAM. Each EXT-RAM-i has L memory locations and the location with the address d-1 ($1 \geq d \geq L$) contains the extrinsic messages exchanged between the variable node ${v^{x,y}}_d$ in $VG_{x,y}$ and its neighboring check node in CG_i. The INT-RAM and DEC-RAM store the intrinsic message and hard decision associated with node ${v^{x,y}}_d$ at the memory location with the address d-1 ($1 \geq d \geq L$).

2. Address Generator: One address generator ${AG^{(i)}}_{x,y}$ associates with one EXT-RAM-i in each $PE_{x,y}$. In the check node processing, ${AG^{(i)}}_{x,y}$ generates the address for reading hybrid data and, due to the five-stage pipelining of datapath loop, the address for writing back the check-to-variable message is obtained via delaying the read address by five clock cycles.

3. Shuffle Network: A bi-directional shuffle network π_i is implemented to realize the parity-check matrix H with the address generators ${AG^{(i)}}_{x,y}$.

4. CNU: Each CNU carries out the operations of one check node, including the parity check and computation of check-to-variable extrinsic messages.

5. VNU: Each VNU generates the hard decision and all the variable-to-check extrinsic messages associated with one variable node.

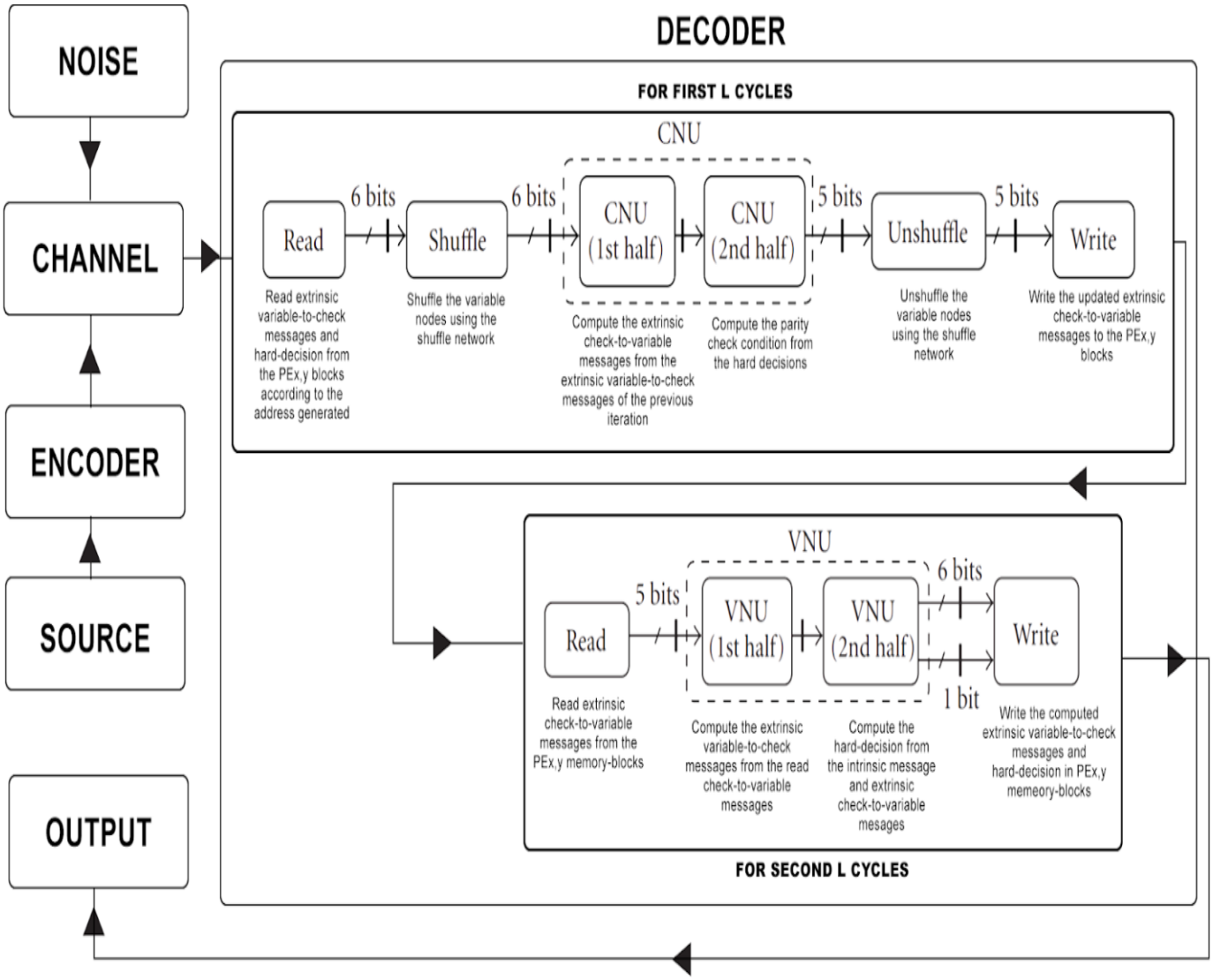


6 Setup

6.1 Software Used

We have mainly used Verilog, which is a hardware description language(HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. HDL simulators are software packages that simulate expressions written in one of the hardware description languages, such as VHDL, Verilog or SystemVerilog. We have also used python to write few codes. For compilation and simulation, we have used Xilinx. For creating noise in the input image, we have used python libraries and opencv.

6.2 Block Diagram



6.3 Implementation

We have used a greyscale image as our input and have done image processing on it so as to add the noise. The image is then filtered by using the LDPC code so as to give the original image as the final output.

- We implemented a (3, 6)-regular LDPC code partly parallel decoder for L = 256 using Xilinx Virtex-E XCV2600E device with the package FG1156. The corresponding LDPC code length is $N = L \cdot k^2 = 256 \cdot 62 = 9216$ and the corresponding code rate is 1/2. We obtain the constrained random parameter set for implementing π_3 and each $AG^{(3)}_{x,y}$ as follows: first generate a large number of parameter sets from which we find few sets leading to relatively high Tanner graph average cycle length, then we select one set leading to the best performance based on computer simulations.
- This partly parallel decoder works simultaneously on three consecutive code frames in two-stage pipelining mode: while one frame is being iteratively decoded, the next frame is loaded into the decoder, and the hard decisions of the previous frame are read out from the decoder.
- Thus each INT RAM contains two RAM blocks to store the intrinsic messages of both current and next frames. Similarly, each DEC RAM contains two RAM blocks to store the hard decisions of both current and previous frames.

7 Experiments and Results

8 Scope of Future Work

- LDPC codes could fulfill the high throughput demand (around 50Mbps) of 5G New Radio(NR) networks. The implementation of the LDPC code decoder using FPGA would reduce the fabrication time and enable a faster implementation of 5G networks world-wide.
- Usage of FPGAs for hardware implementation enables its reconfiguration infinite number of times and hence many companies like Microsoft Azure have their networks setup based on FPGA powered-software reconfigured networking.

9 References

- Tong Zhang* and Keshab K. Parhi ECE Dept.University of Minnesota.
- Elements of information theory by CoverT.M(2ed, wiley,2006) .
- Amin Shokrollahi Digital Fountain, Inc. 39141Civic Center Drive, Fremont, CA 94538

10 Team

10.1 Project Mentors

- Vaibhav Thakkar
- Anshul Rai
- Afzal Rao
- Utkarsh Gupta
- Netravat Pendsey

10.2 Team Members

- Prakhar Maheshwari
- Naiza Singla
- Harsh Vardhan Arya
- Shivam Malhotra
- Aditya Gupta
- Malkurthi Sreekar
- Shubham Korde
- Prateek Gupta
- Pranab Pandey