LDPC Decoder Implementation using NoC

Shaishav Yogeshkumar Shah (133070050)

Supervisor: Prof. Sachin B. Patkar,
Dept. of EE,
IIT Bombay

June 1, 2015

Acknowledgement

- **Prof. Sachin B. Patkar** sir for giving me opportunity to present this idea and his consistent suggestion.
- **Pinal J. Engineer**, PhD student Electrical Engineering working under Prof. Rajbabu Velmurugan and Prof. Sachin Patkar. The core idea has been inspired from Pinalkumar Engineer's research work, and the work done is also closely associated with Pinalkumar's doctoral research.

Aim

- The core aim of this work is straightly associated with Pinalkumar Engineer's doctoral research.
- To overcome the limitation of interconnect bottleneck for projective geometry based low density parity check (PG-LDPC) [1] decoder.
- To provide flexibility and scalability over complex PG-LDPC system.
- The idea may cost reduced data rate and resource overhead.

1. Yu Kou, Shu Lin, and Marc P. C. Fossorier. Low-density parity-check codes based on finite geometries: A rediscovery and new results. IEEE Transactions on Information Theory

Introduction

- PG-LDPC nodes contain bit nodes and check nodes as computing nodes.
- Parity Check matrix consists of row circulant property.
- The computing nodes can be decoupled from PG structure and clusters can be formed
- These **decoupled clusters** are attached with the network interface (NI) of NoC.
- The **circulant property** of Parity check matrix is utilized to leverage **folding**.
- The folded architecture of PG-LDPC (73,45) with fold factor of 5 is implemented using Mesh 4×4 NoC.
- The architecture to decoding multiple data sequences is implemented.
- The implemented architectures have been synthesized on Xilinx Virtex 5 and their parameters are extracted.

Why NoC for LDPC

- For PG-LDPC, interconnect is the bottleneck.
- For PG-LDPC (73,45)- each bit node is connected to 9 check-nodes and vice versa
- Total interconnection between bit-nodes and check-nodes are calculated as below:
  (Nos. of Bit-nodes $\times$ Degree of Bit-node) + (Nos. of Check-nodes $\times$ Degree of Check-node)) $\times$ Data-width
  $=((73\times9) + (73\times9)) \times 5$
  $=6570$
- Implementation issues for larger systems.
- By connecting computing nodes to NI of NoC, routing congestion is taken care by NoC.
- It can be useful to design flexible and scalable architecture implementation

Tanner Graph representation of PG-LDPC (73,45)

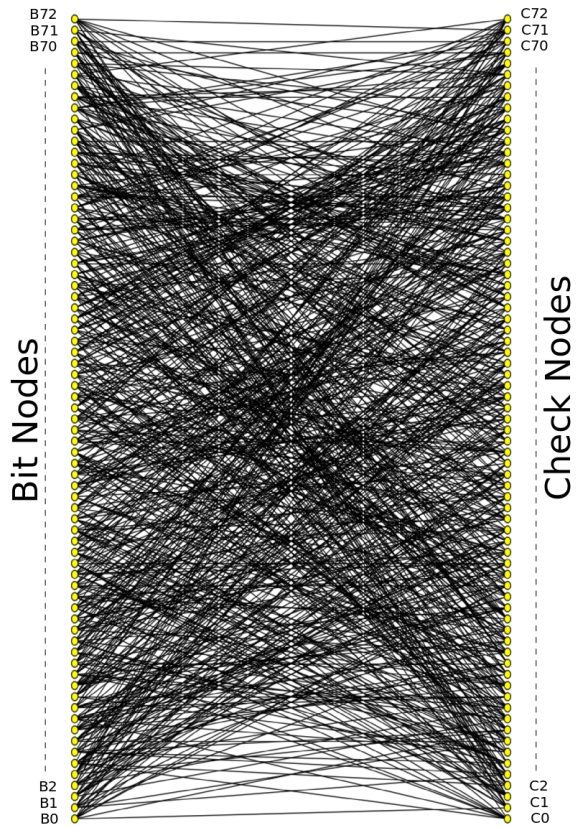


Figure : Tanner Graph representation of PG-LDPC (73,45)

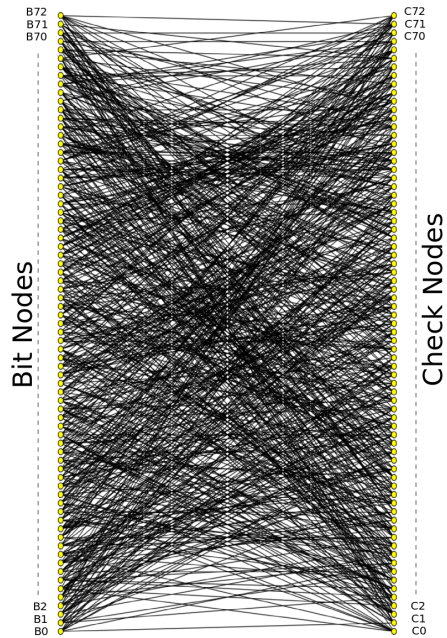
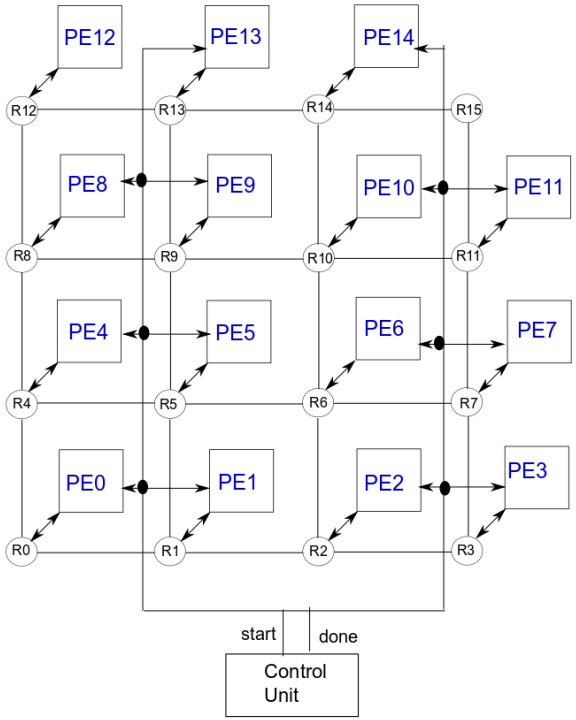


Figure : PG-LDPC (73,45) using NoC



Decoupled clusters of bit nodes and check nodes should be mapped to processing elements (PE) of NoC

LDPC Introduction

- Found by Robert G. Gallager in his PhD thesis fom MIT, USA
- Linear Error Correcting Codes
- Used in Wireless Mobile Communication, Disk Drives
- Can Achieve near shannon's limit Channel Capacity
- Decoding can be done in parallel.

LDPC Definition

Low Density Parity Checking Codes

[2] *LDPC code is defined as null space of parity-check matrix H, which satisfies the following structural properties:*

1. *Each row consists of ρ $1's$*
2. *Each column consists of γ $1's$*
3. *Let λ be the number of 1's in common between any two columns, then $\lambda \not> 1$ ($i.e.$ $\lambda = 0$ or $\lambda = 1$)*
4. *ρ and γ, both are small compared with the length of the code and the number of rows in H.*

2. Shu Lin and Daniel J. Costello. Error Control Coding, Second Edition. Prentice-Hall, Inc

Important Definition

- **Density of Parity Check Matrix, r**

  The ratio of total number of $1's$ in H to total number of entries in H is defined as density r of parity-check matrix H.

  If $J =$ The number of rows in H, then

  $$r = \frac{\rho}{n} = \frac{\gamma}{J}$$

- **Cyclic Code**

  Let C be an (n, k) linear code, then if every cyclic shift of codeword in C is also a codeword in C, then C is called Cyclic Code.

- **Regular LDPC Code**

  If all the columns and all the rows of parity check matrix, H have the same weight, an LDPC code is said to be Regular LDPC.

Important Definition

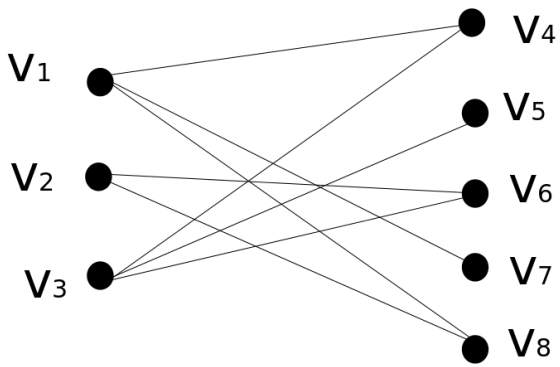


Figure : Bipartite Graph

- **Bipartite Graph**

  A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called Bipartite graph if its vertex set $\mathcal{V}$ can be partitioned into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ such that each edge in $\mathcal{E}$ joins the vertex in $\mathcal{V}_1$ with a vertex in $\mathcal{V}_2$, and no two vertices in either are connected.

- **Node**

  The vertices of graph are called as nodes.

2D PG-LDPC Parameters [3]

- Length $n = 2^{2s} + 2^s + 1$
- Number of parity bits $n - k = 3^s + 1$
- Dimension $k = 2^{2s} + 2^s - 3^s$
- Minimum Distance $d_{min} = 2^s + 2$
- Density $r = \frac{2^s}{2^{2s}+2^s+1}$

s = Any Prime Number.

3. Yu Kou, Shu Lin, and Marc P. C. Fossorier. Low-density parity-check codes based on finite geometries: A rediscovery and new results. IEEE Transactions on Information Theory

PG-LDPC Parameters

Table : Two-Dimensional PG-LDPC code Parameters

s	Length (n)	Dimension (k)	Minimum Distance (d_{min})	Row Weight (ρ)	Column Weight (γ)	Density (r)
1	7	3	4	3	3	0.4286
2	21	11	6	5	5	0.2381
3	73	45	10	9	9	0.1233
4	273	191	18	17	17	0.0623
5	1057	813	34	33	33	0.0312
6	4161	3431	66	65	65	0.0156
7	16513	14325	130	129	129	0.0078

1. $\mathbf{PG-LDPC\ (7,3)}$ with $\mathbf{s = 1}$
2. $\mathbf{PG-LDPC\ (73,45)}$ with $\mathbf{s = 3}$

Tanner Graph

- Consider a linear block code C of length n specified by a parity check matrix H. It has J rows, $h_1, h_2, h_3, ..., h_J$. Construct a graph $\mathcal{G}_T$ that consists of two sets of vertices $\mathcal{V}_1$ and $\mathcal{V}_2$.

- Bit node

  The first set $\mathcal{V}_1$ consists of n vertices, that represents n **code bits** *of the code. These vertices are denoted by $u_0, u_1, ..., u_{n-1}$ and are called as variable nodes or* **Bit nodes**.

- Check node

  The second set $\mathcal{V}_2$ consists of J vertices, that represents J **party-check sums (or equations)** *of the code. These vertices are denoted by $v_0, v_1, ..., v_{J-1}$ and are called as check-sum vertices or* **Check nodes**

Tanner Graph

- A bit node vertex u_l is connected to check node vertex v_j by an edge (u_l, v_j) if and only if the code bit u_l is contained in the parity-check sum v_j.
- No two bit-nodes are connected and no two check-nodes are connected.
- Tanner Graph $\mathcal{G}_T$ is a Bipartite Graph.
- The degree of a bit node u_l is equal to the number of parity-check sum that contain u_l. And the degree of a check node u_l is equal to the number of bit nodes that contain v_j
- Regular LDPC code can be represented by regular Tanner Graph.
  Bit node degree $= \gamma$.
  Check node degree $= \rho$.

Parity Check Matrix for PG-LDPC (7,3)

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

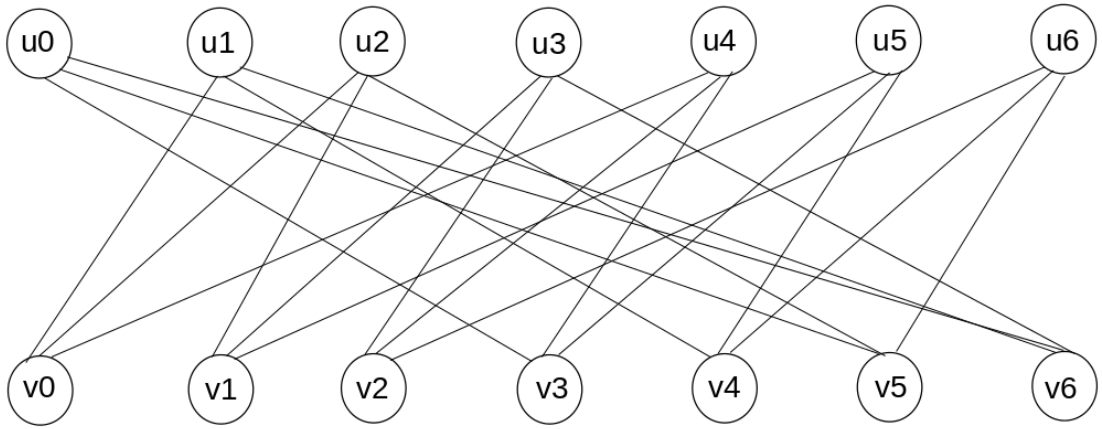


Figure : Tanner Graph representation

Modified Sum Product Algorithm (SPA)

The Modified SPA works as follow:

1. Initialize all $L(u_{ij}) = L(c_i)$
2. At check-node j calculate $L(v_{ji})$ as
   $L(v_{ji}) = (\prod_{i' \in v_{j'}} \alpha_{ij}).\phi(\sum_{i' \in V_{j'}} \phi(\beta_{i'j}))$ Also,
   $\phi(x) = log(\frac{e^x+1}{e^x-1})$
3. At bit-node i, calculate $L(u_{ij})$ as
   $L(u_{ij}) = L(C_i) + \sum_{j' \in C_{i'}} L(v_{j'i})$
4. Calculate $L(U_i) = L(c_i) + \sum_{j' \in C_{i'}} L(v_{ji})$
5. $c_i = 0, if\ \ L(U_i) > 0$
   $c_i = 1, if\ \ L(U_i) \leq 0$
   If these bits satisfy all the parity check equation or if the total nos. of itertion exceeds a threshold, End
   else go to step (2)

Modification to SPA- Min Sum Approximation

- $\phi(\sum_{i' \in V_{j1}} \phi(\beta_{i'j})) \approx \phi(\phi(min_{i' \in V_{j1}} \beta_{i'j})) = min_{i' \in V_{j1}} \beta_{i'j}$ [4] because $\phi(\phi(x)) \approx x$
- condider the code (73,45) PG-LDPC code which is obtained by a projective geometry over GF(2^m).
- (73,45) code has m=3 and thus the code length is $2^{2s} + 2^s + 1 = 73$.
- The degree of total bit-node and check-node is $2^m + 1 = 9$.
- The high degree of check-nodes result in a significant error in the min-sum approximation.
- Better approximation of the min-sum can be obtained as $\phi(\sum_{i' \in V_{j1}} \phi(\beta_{i'j})) \approx (min_{i' \in V_{j1}} \beta_{i'j} - \beta_0)$

4. Subhasis Das, Hrishikesh Sharma, Sachin Patkar. A min-sum based 800 Mbps LDPC decoder on FPGA. Technical report

NoC Introduction

- For System on Chip (SoC), multiple processing units, memory, peripherals want to communicate with each-other.
- Trade-off between
  1. Point to point transmission - high speed and complex
  2. Shared bus data transmission - Low speed and simpler
- Ease of verification and logic error determination

NoC Building Blocks

- **Links**
  The elements which physically connect the nodes and actually implement the communication are called as Links.
- **Router**
  The decentralized logic which implements the communication protocol is called Router.
- **Network Adapter (NA) or Network Interface (NI)**
  The block which makes the logic connection between IP cores and network is called as Network Interface or Network Adapter.

Router and Flit

Routers

- The critical part of NoC.
- Receives the data packet from the adjacent router or from the attached IP-core.
- Redirects packet to the IP-core or the next adjacent router.

Flit

- The physical unit, which is the minimum amount of data that is transmitted in one link transaction.
- Atomic unit that forms *packets* and *streams*.

Flow Control: Characterizes the packet movement across NoC channel.

- *Centralized Control:* Routing decisions made globally and applied to all nodes.
- *Distributed Control:* Each router makes decision locally.

Routing Algorithm: Logic that selects one output port to forward a packet that arrives at the router input.

- *Deterministic Routing:* A packet always uses the same path between any set of nodes.
- *Adaptive Routing:* Alternate path between two nodes may be used if the original path or local link is congested.
- *Static Routing:* Paths between cores are defined at compilation time.
- *Dynamic Routing:* Paths between cores are defined during run-time.

Arbitration: Logic that selects one input port, when multiple packets are arrive at the router simultaneously requesting the same output port.

- *static* (fixed), or *dynamic* (variable)
- *distributed* (one per port) or *centralized* (one per router).

Buffering: The strategy used to store information in the router during congestion in the network, and the packet cannot be forwarded immediately to the correct port **Routing Algorithm:** Logic that selects one output port to forward a packet that arrives at the router input.

- *Single Buffer per Router*
- *Single Buffer per port*

- **Deadlock:** The condition when network resources are fully occupied and waiting for each other to be released for communication.
- **Virtual Channel (VC):** VC uses the concept of multiplexing a single physical channel over numbers of logically separate channels with individual and independent buffer queues.
- The NoC used involves *Distributed Flow Control* approach and *Virtual Channels*. NoC uses *XY routing*, which is *deterministic* and *static* routing, *Round Robin* arbitration, which is *distributed* and *dynamic* arbitration approach, and multiple buffers (typically 8) at each input port

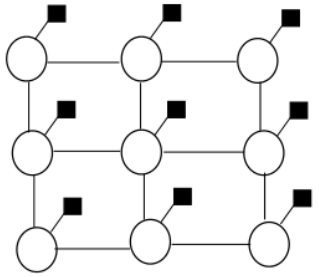
NoC Topologies and Routing strategies

NoC Topologies

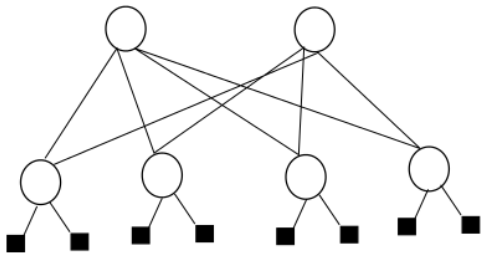
- Mesh
- Fat-Tree
- Ring, Bus etc

NoC Routing Strategies

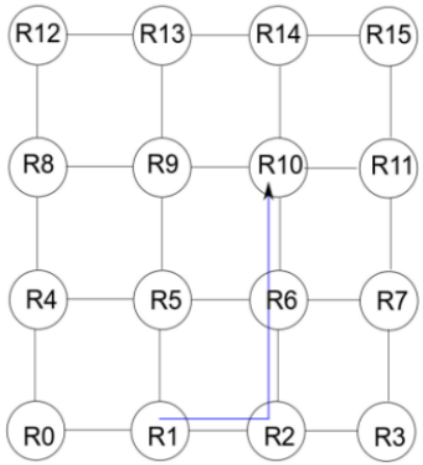
- XY Routing



Mesh Noc 3x3



Fat Tree Noc

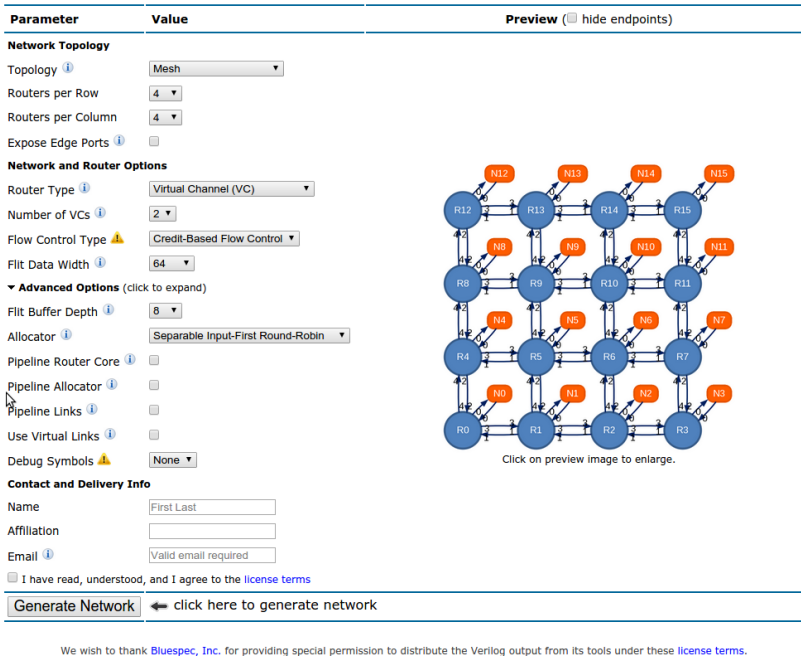


XY Routing

*CON*figurable *NE*twork *C*reation *T*ool [5]

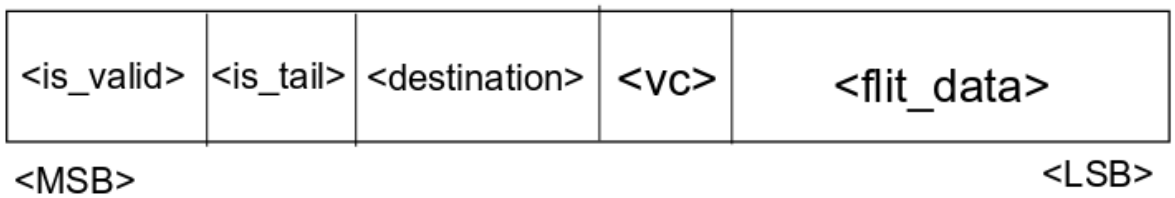
CONNECT NoC

- Developed by Michael K. Papamichael, Carnegie Mellon University
- Generates fully synthesizable verilog code.
- Latency between two adjacent Processing Element under no traffic is 2 cyle and +1 cycle per additional router distance.
- Provision of custom parameter selection i.e. Data width, number of I/O buffers, Routing strategies.



5. Michael Papamichael. Fast scalable FPGA-based network-on-chip simulation models. In MEMOCODE,

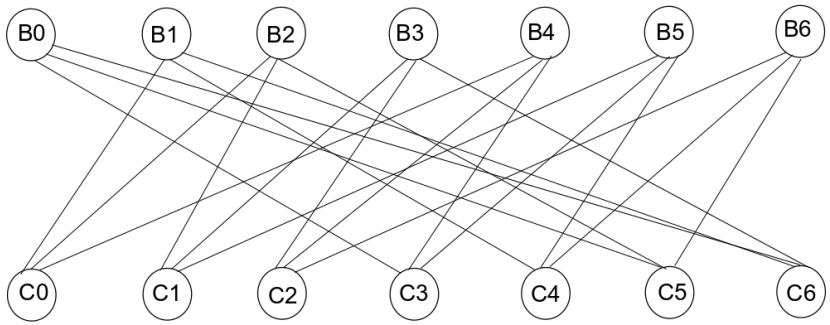
Flit-Data structure and NoC Parameters



Topology	Mesh
Routers per Row	4
Routers per Column	4
Router Type	Virtual Channel (VC)
Number of VCs	2
Flow Control Type	Credit-Based Flow Control
Flit Buffer Depth	8
Allocator	Separate-Input First Round-Robin

Tanner Graph for PG-LDPC (7,3)

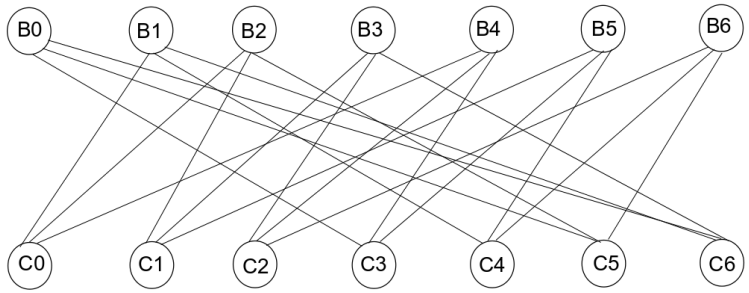
- PG-LDPC (7,3) code which is obtained by Projective Geometry over $GF(2^s)$, $s = 1$.
- The degree of total bit-node and check-node is $2^s + 1 = 3$.
- Cyclic Code
- In this design, 5 bit precision :1 bit for sign, 3 bit of integer precision and 1 bit of fraction precision



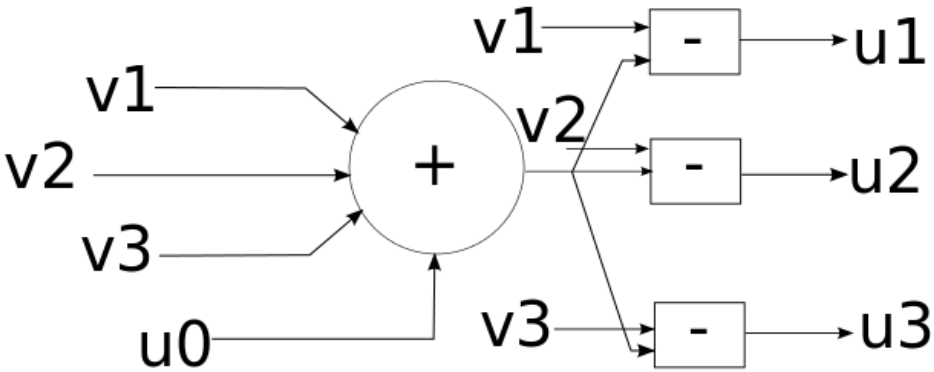
Bn: Bit node n
Cn: Check Node n

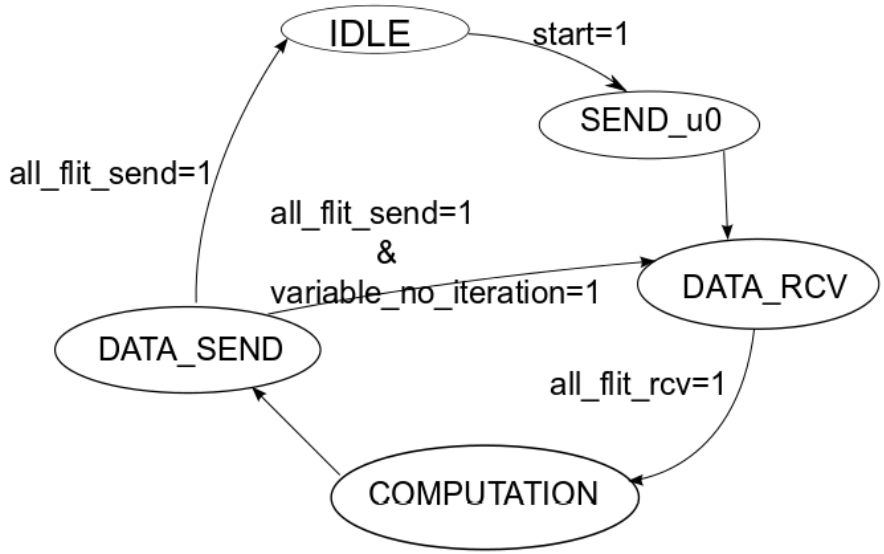
How LDPC works?

1. Initialize all bit nodes by Log Likelihood Ratio (LLR).
2. LLR propagates from bit nodes to check nodes via edges.
3. Check node performs computation.
4. Check node sends the computed LLR back to Bit node via edges.
5. Bit node checks whether the minimum required error is achieved or not.
6. If yes, sends the decoded data out, or Else go to step (2).



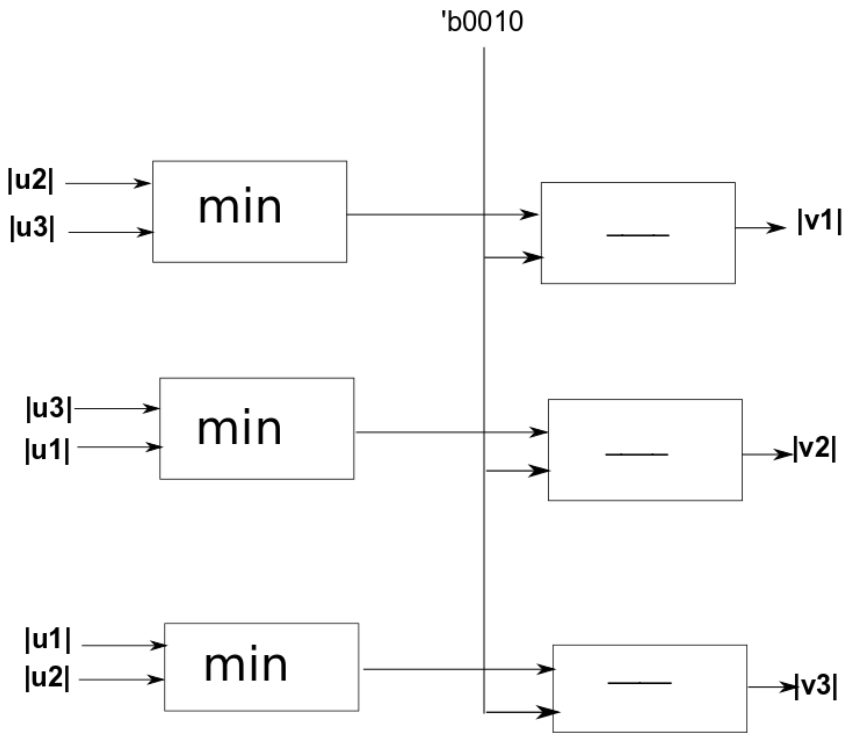
Bit-Node Architecture for PG-LDPC (7,3)



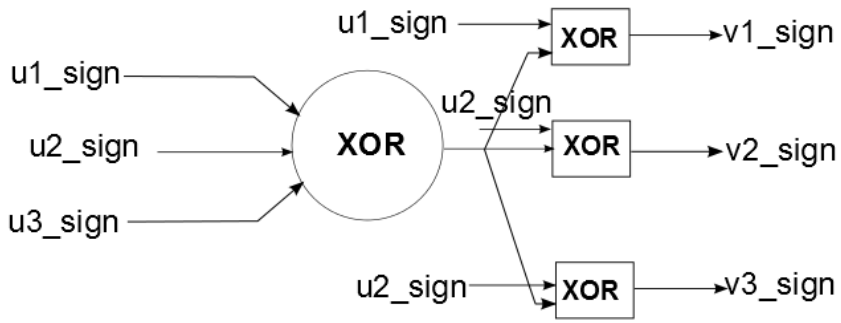


Check-Node Architecture for PG-LDPC (7,3)

Magnitude Calculation

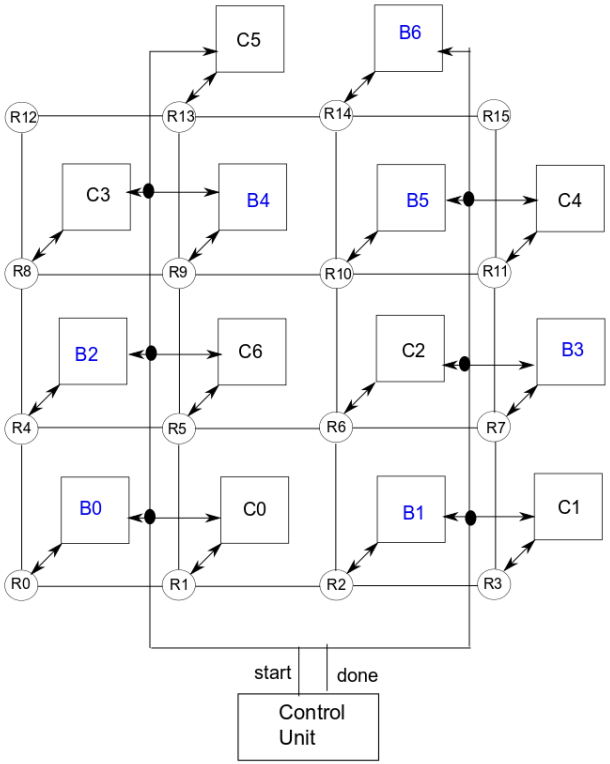


Sign Calculation

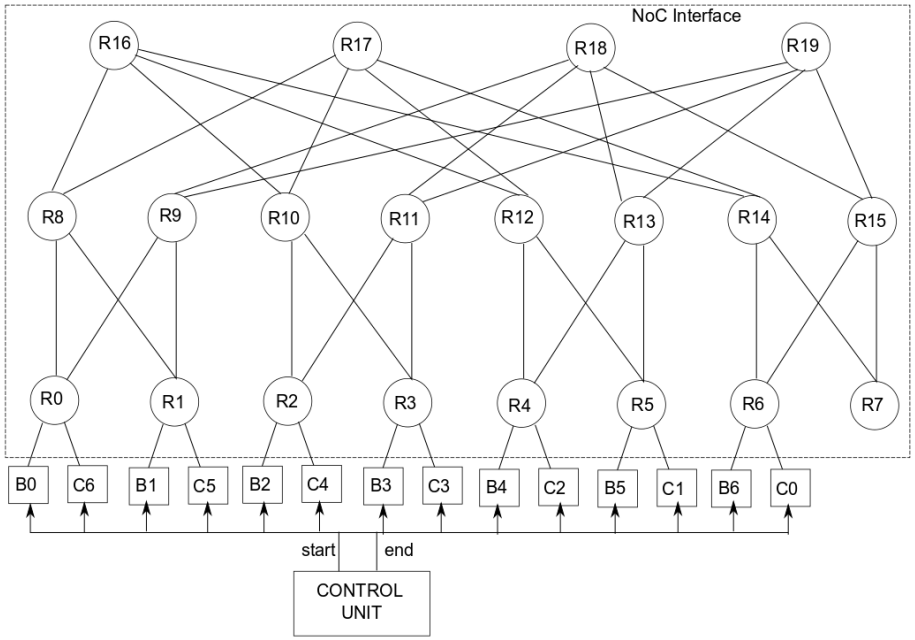


System Implementation of PG-LDPC (7,3) using NoC

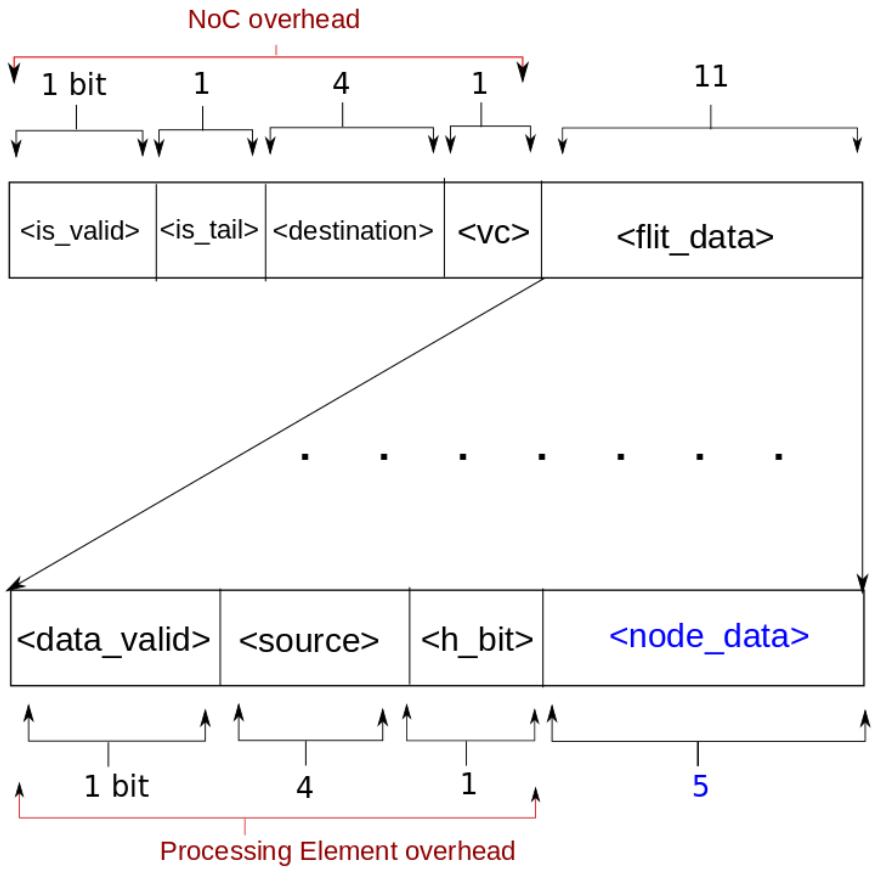
Implementation on Mesh 4x4 NoC



Implementation on Fat Tree NoC



NoC Flit Structure



Results for Implementation of PG-LDPC (7,3) on NoC

Parameters	PG-LDPC (7,3) (Mesh 4×4 NoC)	PG-LDPC (7,3) (Fat-Tree NoC 16)
Clock-cycle (1 iteration)	18	16
Clock-cycle (n iteration)	18	16
Total cycle for n complete iteration	18 + (n-1)×12	16 + (n-1)×11
Number of Slice Registers Number of Slice LUTs	5578 (8%) 10886 (15%)	5539 (8%) 12347 (17%)
Maximum Operating Frequency Nos. of clock cycles Data Rate (mega bits/s)	36.327MHz 18 14.12 Mbps	38.279MHz 16 16.74 Mbps

$$DataRate = \frac{Maximum\ \ frequency \times Datawidth}{Nos\ \ of\ \ Iteration \times Nos\ \ of\ \ clock\ \ cycles}.$$

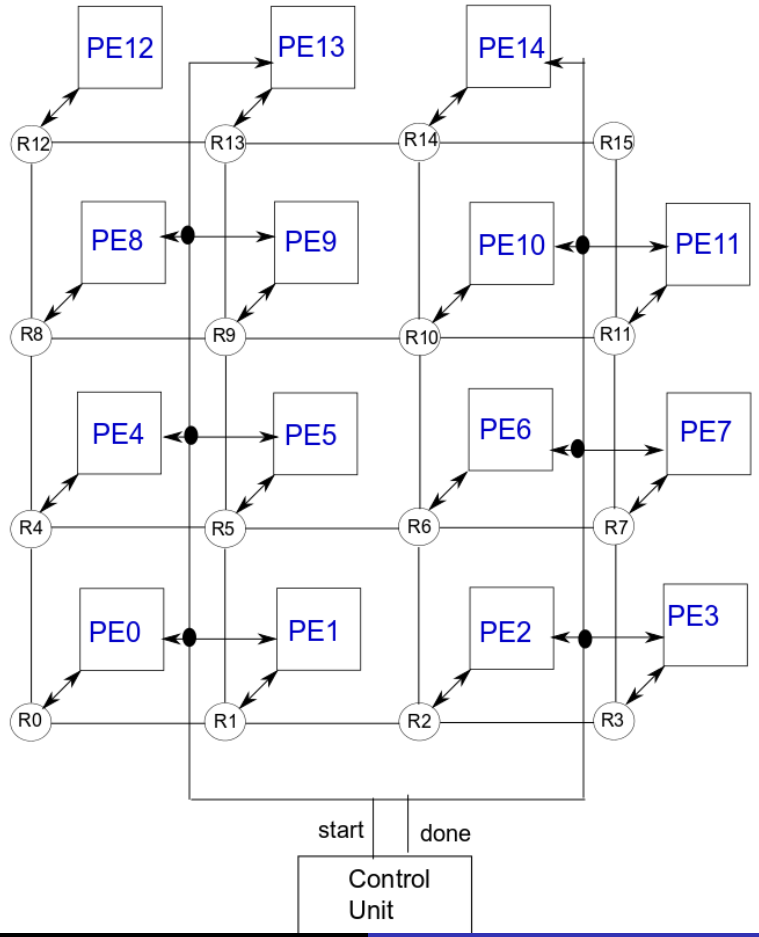
NoC Performance Comparison and Resource Utilization for Virtex-5: XC5VLX110FT

Parameters	Mesh NoC	Fat-Tree NoC 16
Number of Routers		
16 Processing Elements	16	22
32 Processing Elements	32	56
64 Processing Elements	64	144
Nos of slice LUT		
32 Processing Elements	35451 (51%)	44937 (65%)
64 Processing Elements	73866 (106%)	120552 (174%)
Nos of slice Registers		
32 Processing Elements	14472 (20%)	16576 (23%)
64 Processing Elements	26656 (38%)	42624 (61%)

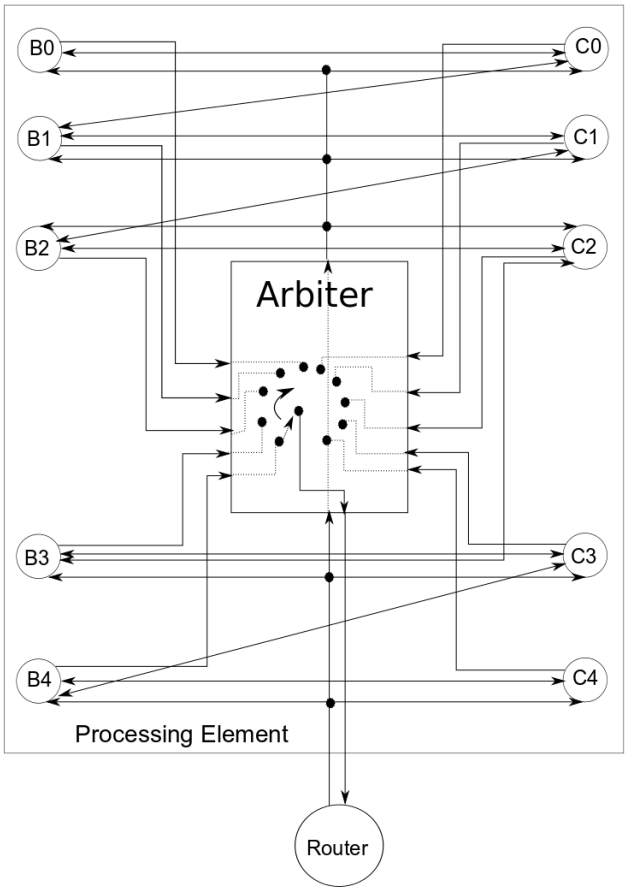
How to implement PG-LDPC (73,45) Implementation using Mesh 4x4 NoC- Partial Interconnect

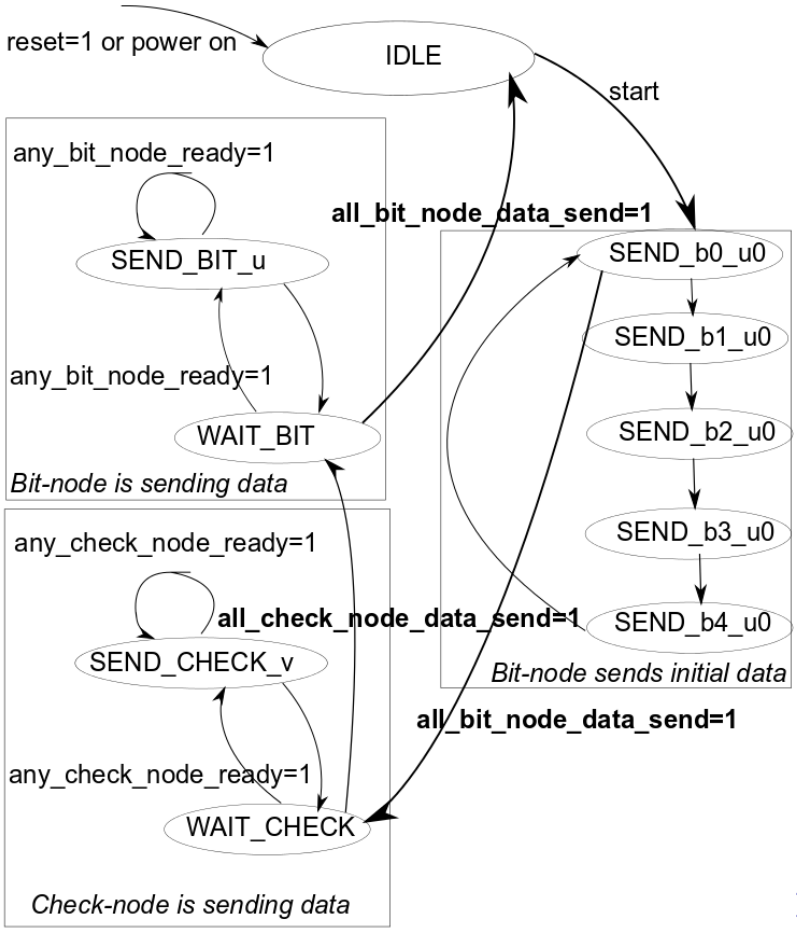
- PG-LDPC (73,45) Decoder is implemented on Mesh 4×4 NoC.
- Use Partially Routed LDPC using NoC.
- Since we have 146 nodes and 16 routers,
  Average occupancy on each router$= \frac{146}{16} = 9.125$
  Let's place 10 nodes in each PE.

System Implementation

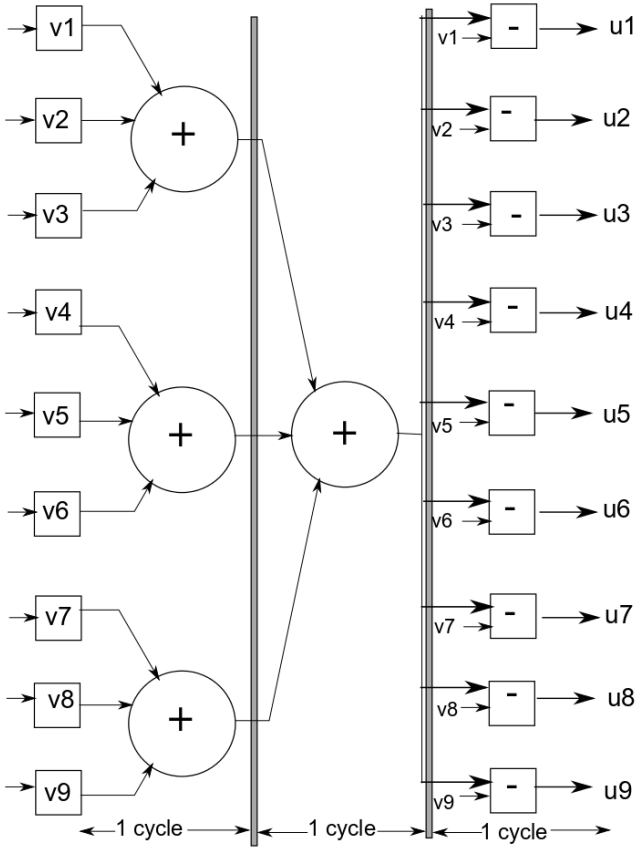


Processing Element Architecture

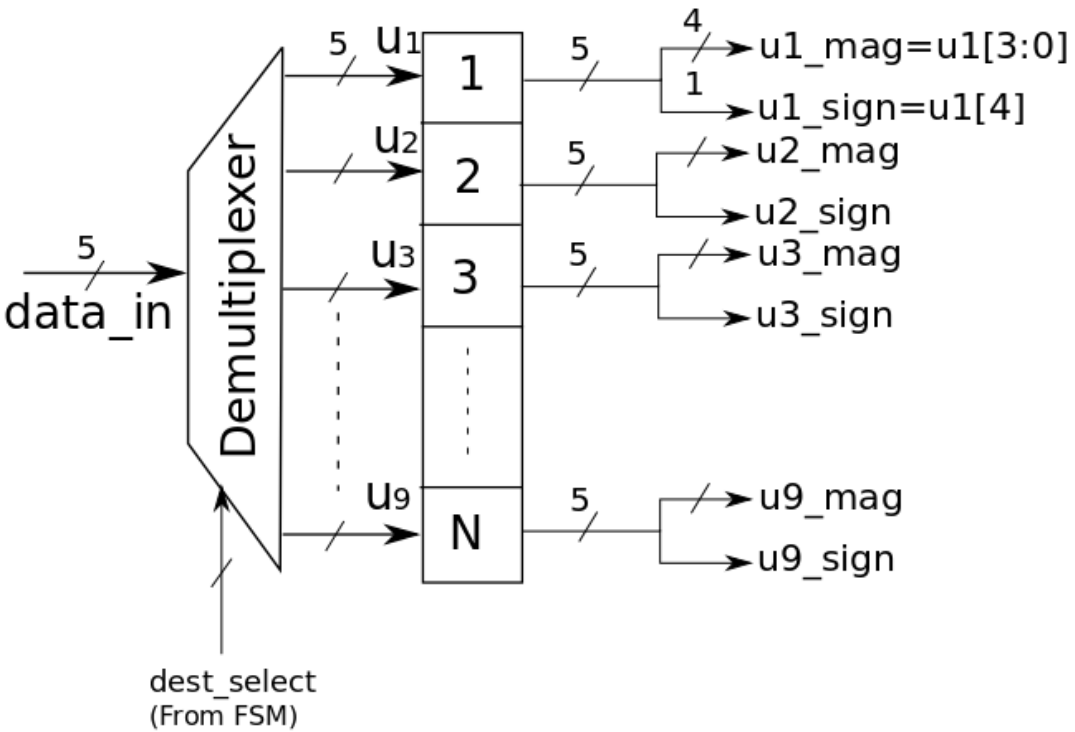




Bit-node Architecture for PG-LDPC (73,45)

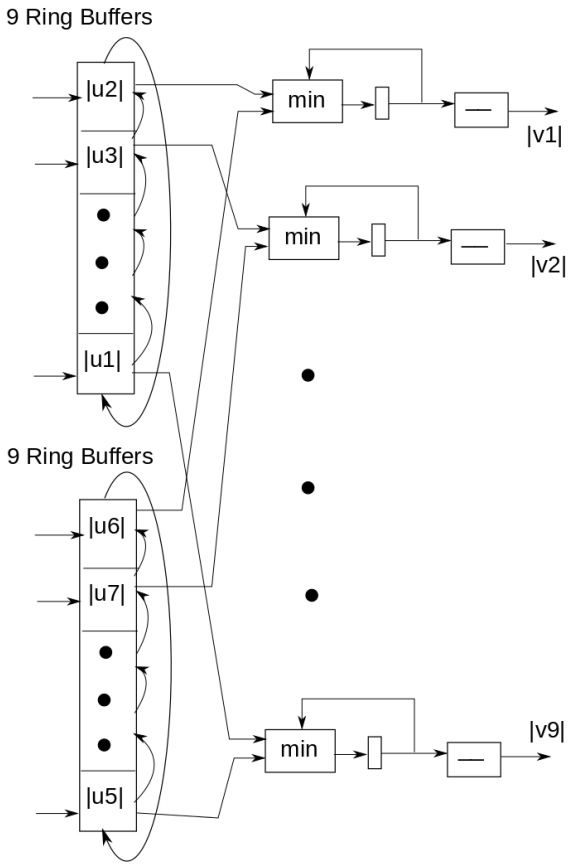


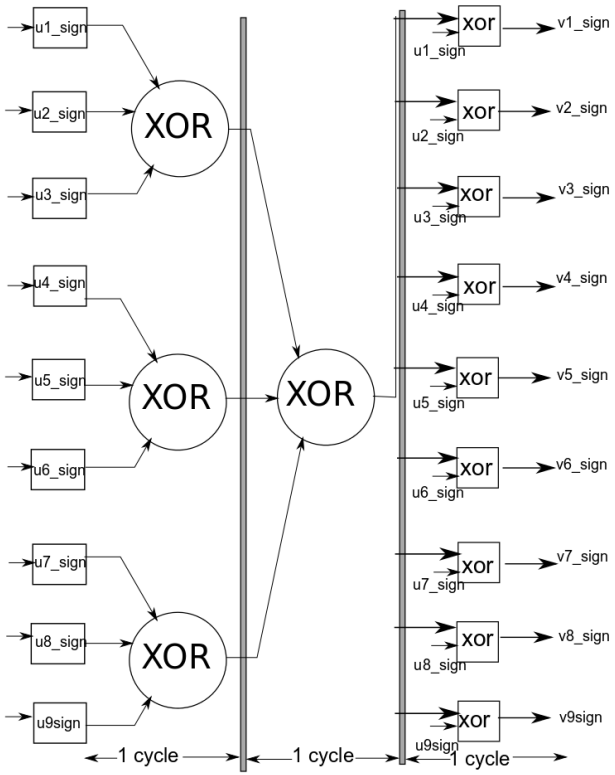
Check-node Input Unit



- Check node seperates sign and magnitude parts and operate on them individually.
- After completion of operation, 2's complement data is sent via output unit.

Check-node Magnitude and Sign Calculation





Timing measurements of Bit-node and Check-node

Parameters	Results
Minimum Clock-cycle (bit-node)	38 + sending and receiveing flit overhead
Minimum clockcycle (check-node)	40 + sending and receiveing flit overhead
Total cycle for one complete iteration	136
Total cycle for n complete iteration	136 + (n-1) $\times$ 100 ....(approximately)

Performance Measurement for Virtex-5: XC5VLX110FT

	PG-LDPC (73,45) (directly connected)	PG-LDPC 73×73 using Mesh 4x4 NoC using partial routing
Number of Slice Registers	23556 (34%)	27278 (39%)
Number of Slice LUTs	32601 (47%)	62418 (90%)
Maximum Operating Frequency	128 MHz	44.528 MHz
Nos. of clock cycles	15	135
Data Rate	623 Mbps	14.8 Mbps

Note: Results shown are not for comparison but to indicate that the design is fit within the board and works correctly.

Comments

- Large Number of bit nodes and check nodes are connected to single router.
- As the length of LDPC increases, number of bit nodes and check nodes in a processing element increases.
- The overall system becomes more complex.
- One way is to apply the idea of **Folding Architecture**.

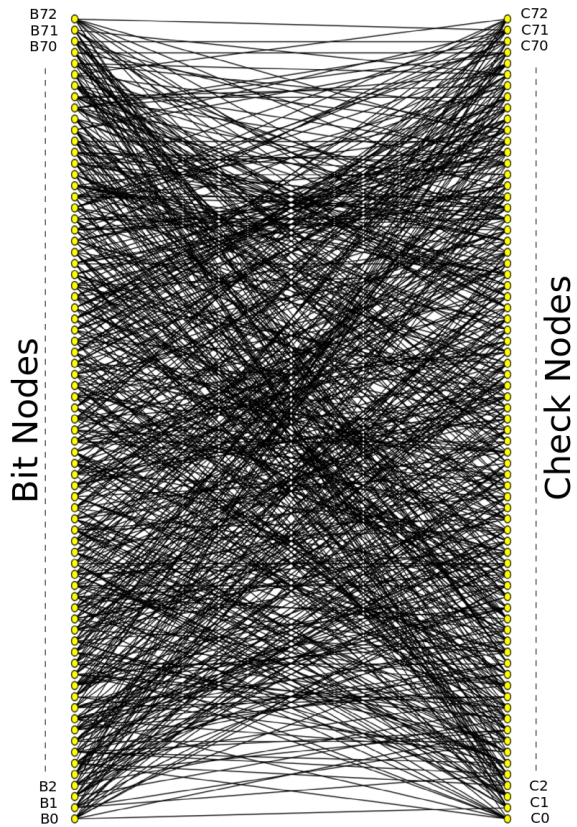
What is Folding

- Partition the vertex set of Bipartite Graph
- Overlay these vertices on available physical processing element [6].
- Need not to overlay set of edges onto each other.
- Hence, the design interconnects need to be reconfigured.

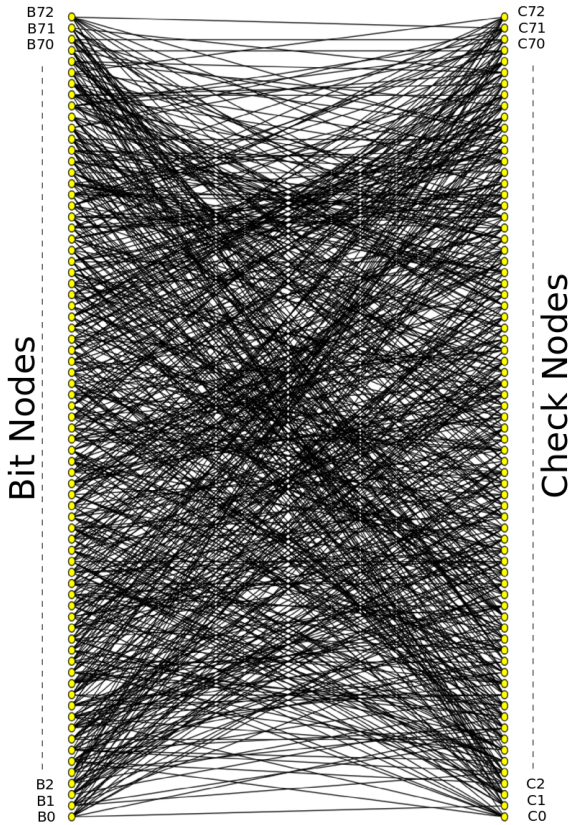
Note: Folding is completely different than Partition.

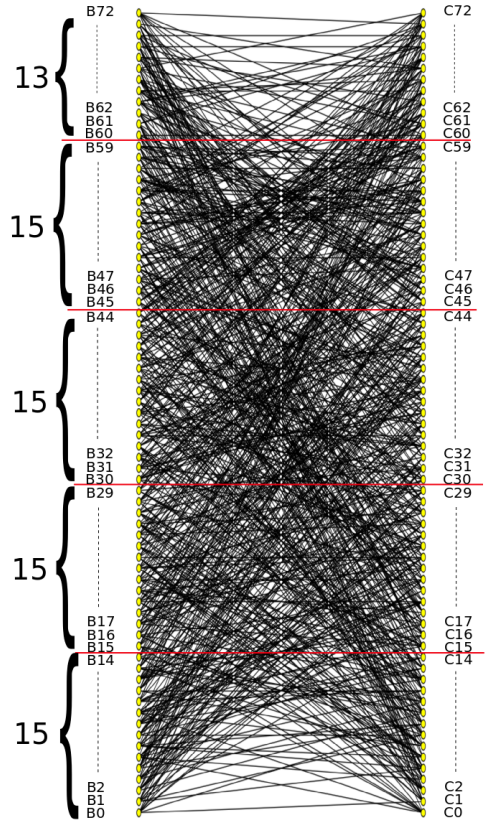
6. Hrishikesh Sharma. Exploration of Projective Geometry-based New Communication Methods for Many-core VLSI Systems. PhD thesis, IIT Bombay

Tanner Graph representation of PG-LDPC (73,45)

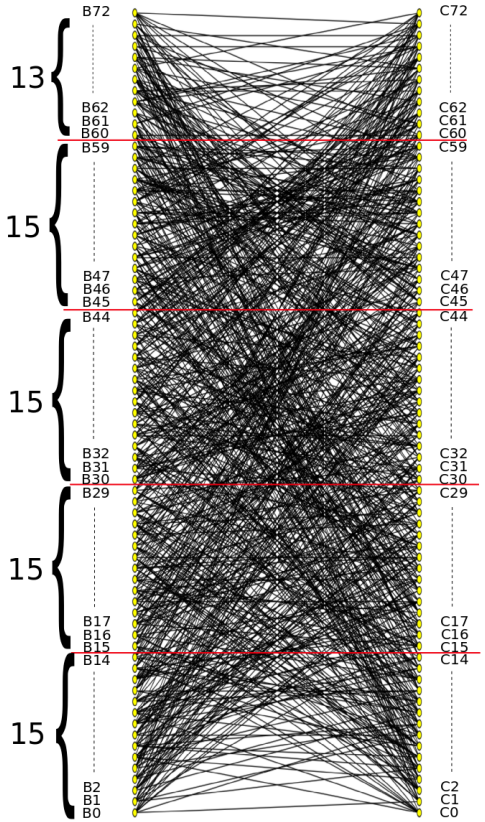


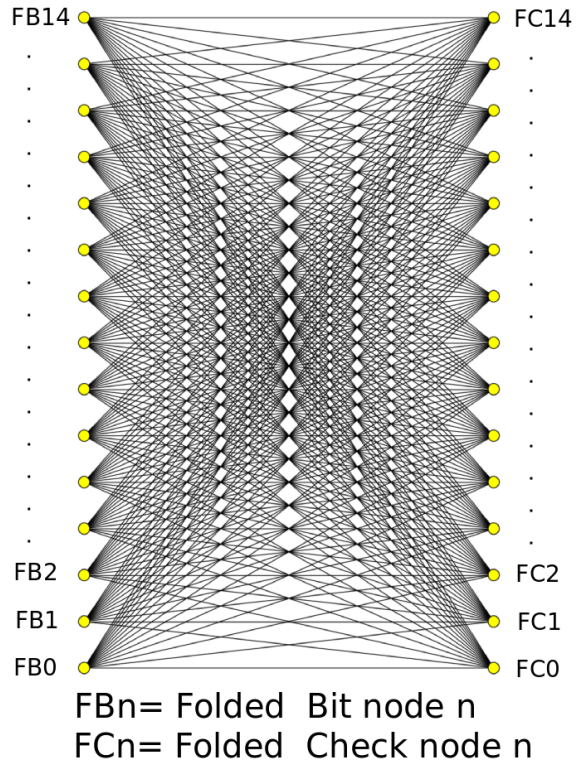
Partition of Bipartite Graph (number of parts=5)



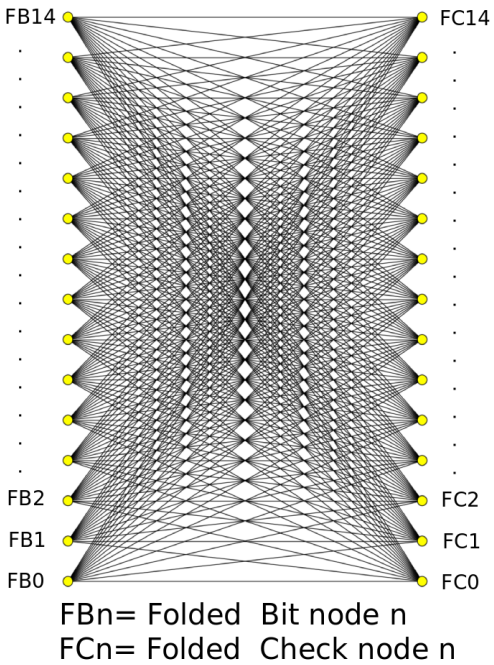


Applying Folding to partitioned graph (Fold Factor=5)



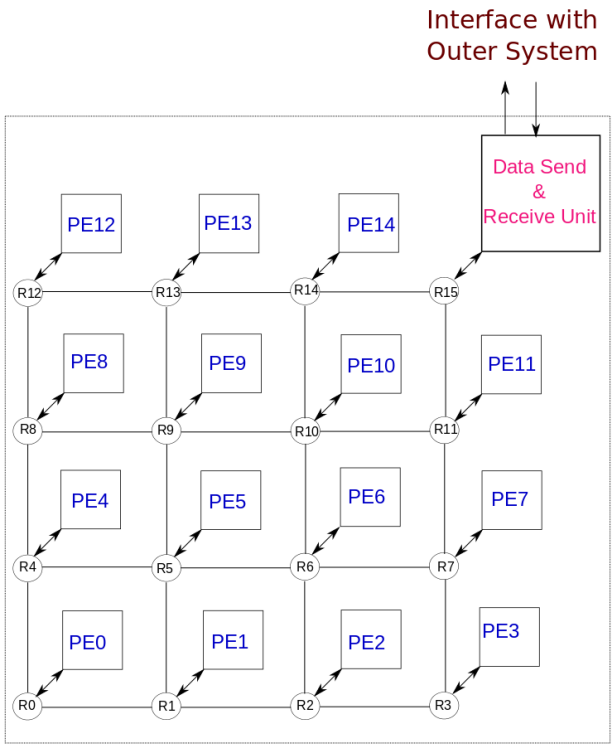


Divide PG-LDPC architecture into Decoupled Clusters



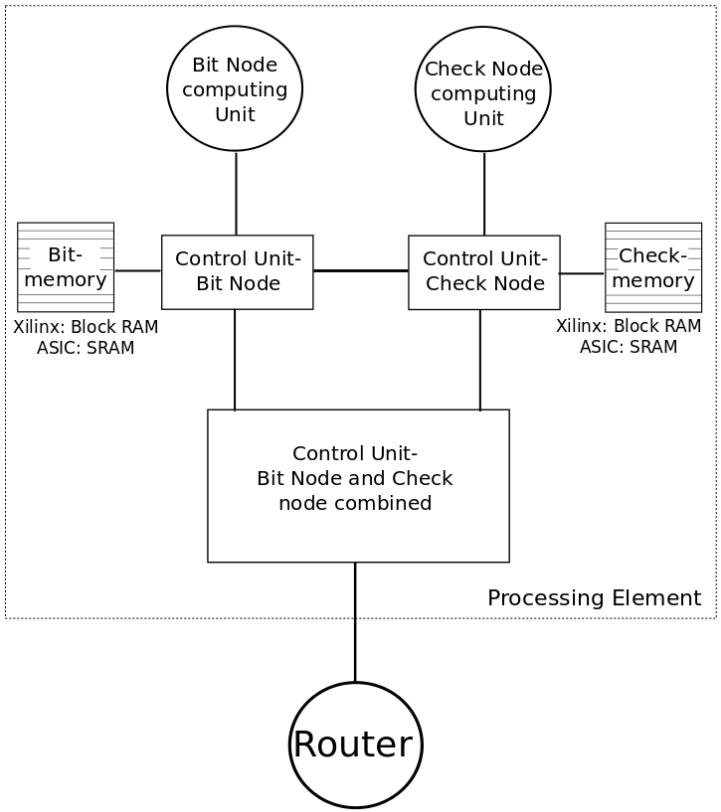
- Fold Factor=5
- Interconnects are reconfigured.
- One bit-node will operate as 5 bit-nodes, and one check-node as 5 check-nodes
- Additional memory will be required for each vertex to keep track of reconfigured edges.
- The resource requirement of bit-node or check-node is not increased.
- The functionality is increased by **serializing the operations**.
- Reconfigured Bit nodes are called as **Folded Bit nodes (FB)** and reconfigured Check nodes are called as **Folded Check nodes (FC)**,
- Each decoupled cluster of computing node will contain one FB and one FC.

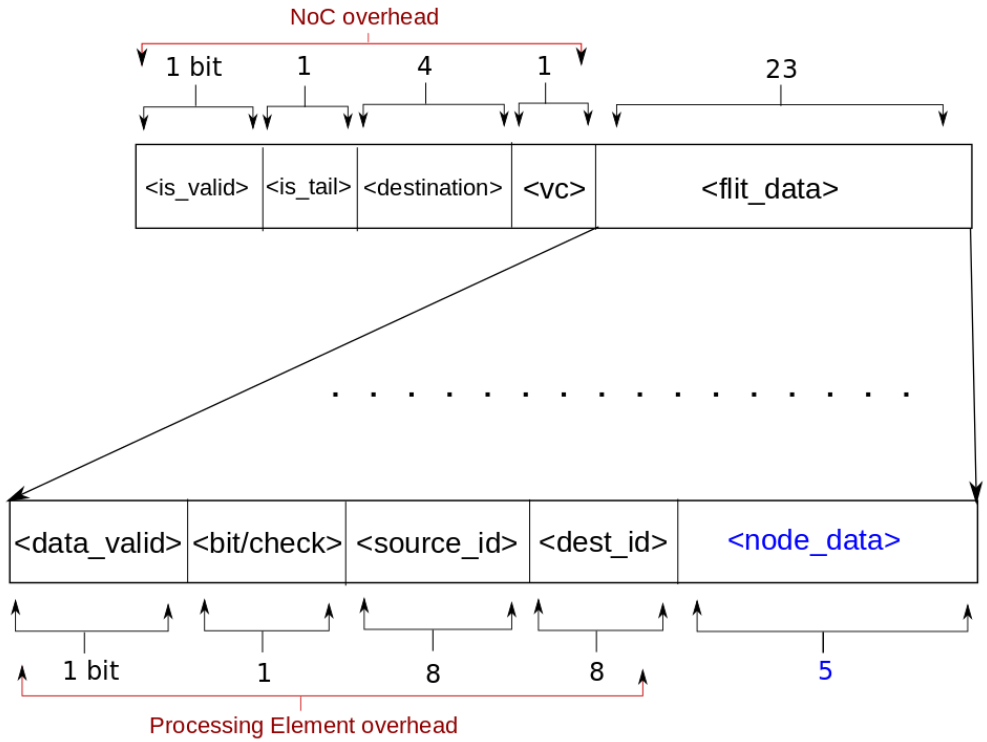
PG-LDPC (73,45) (Fold Factor=5) Implementation on Mesh 4×4 NoC



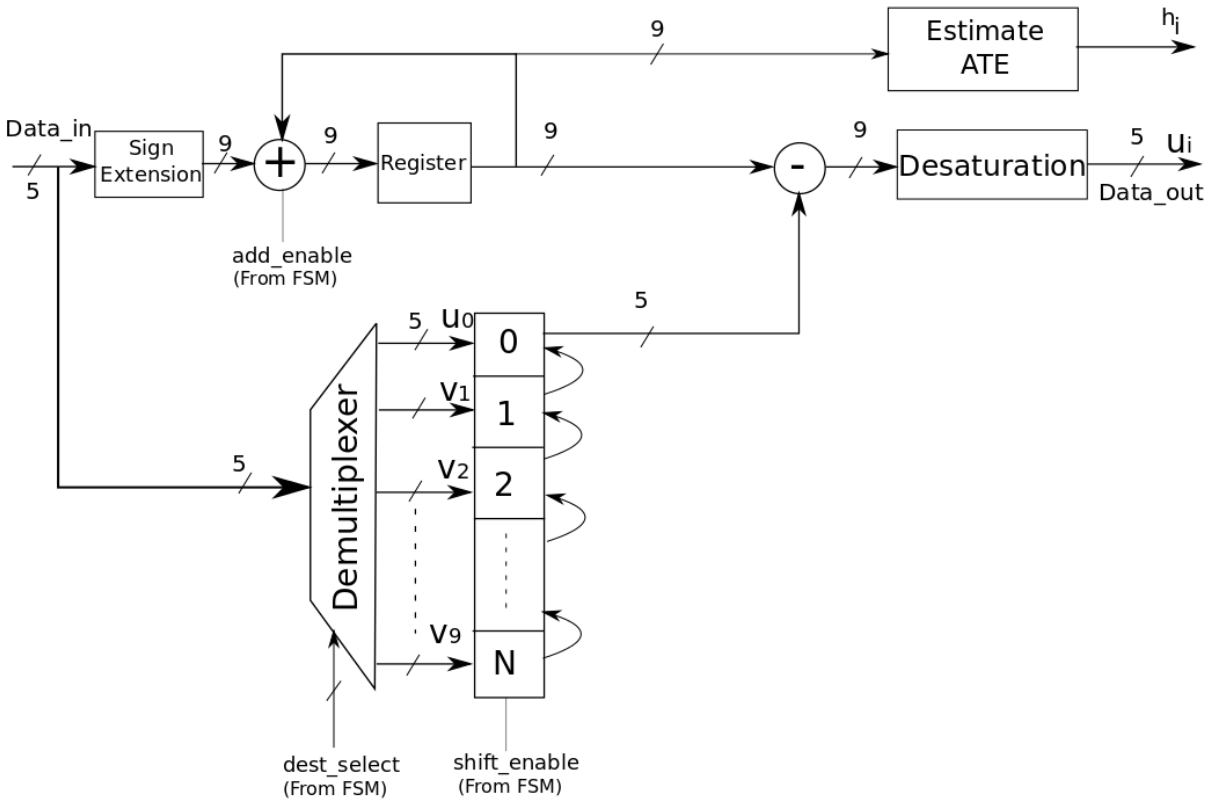
PEn: Processing Element no. n
Rn: Router no. n

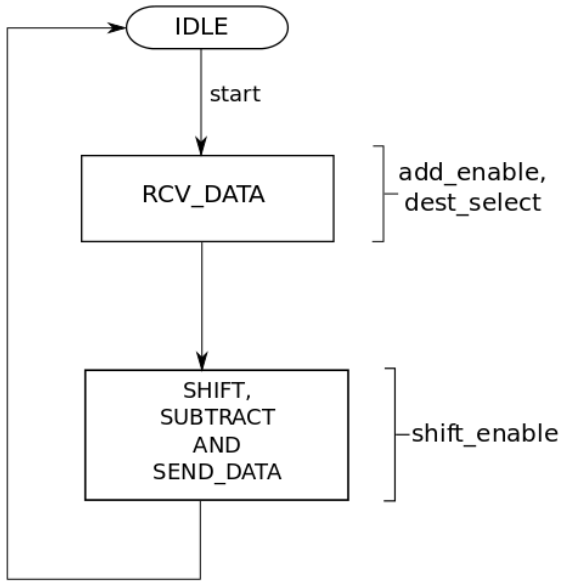
Processing Element and Flit Structure





Bit Node Computing Unit Data-path and FSM





Check Node Computing Unit Data-path

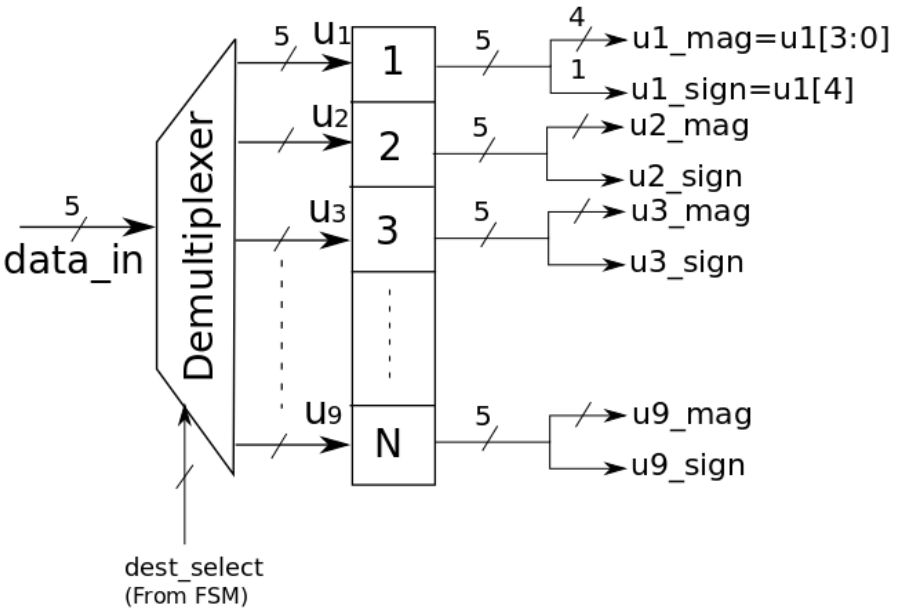
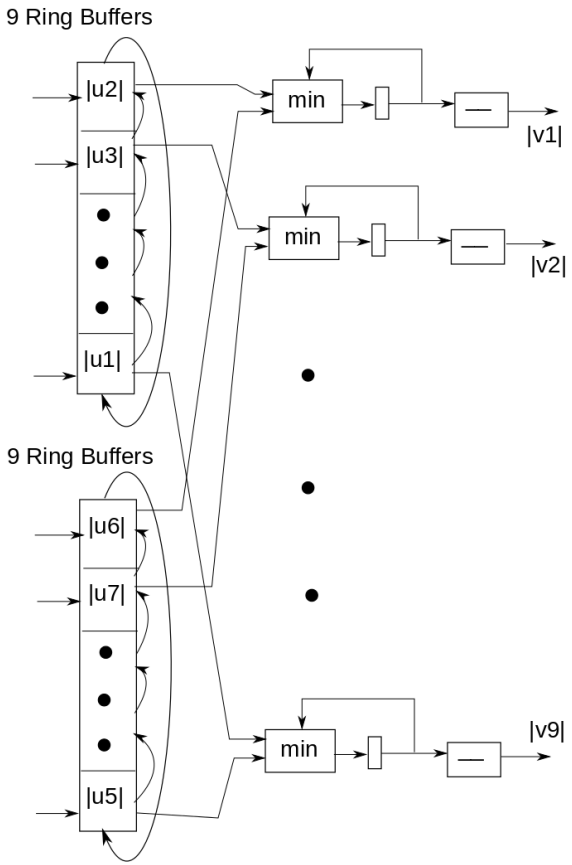
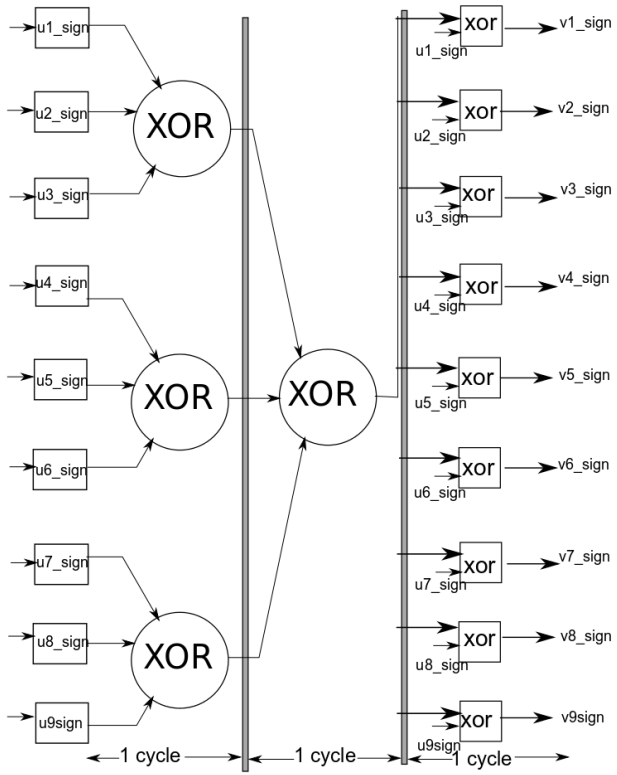


Figure : Input Data Path

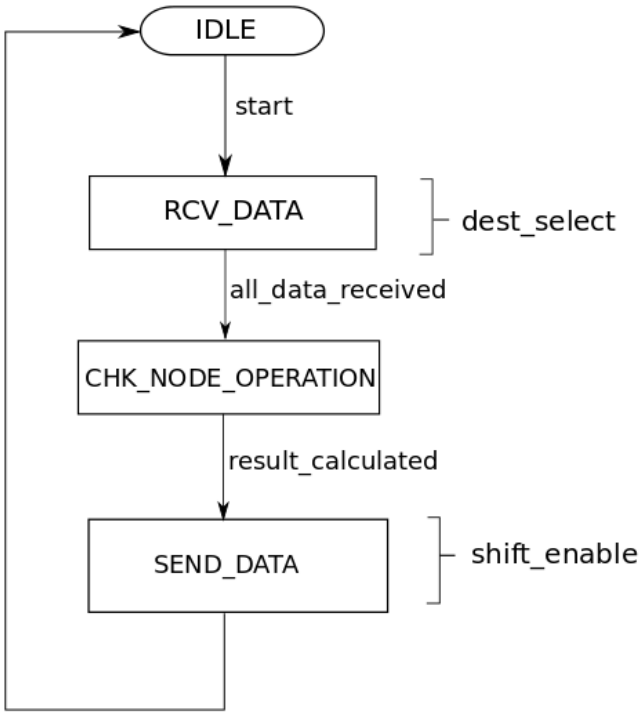
- Check node Computing Unit Data-Path is same as that was used in Unfolded Architecture.
- Check node output unit is different.
- Check node FSM is different.

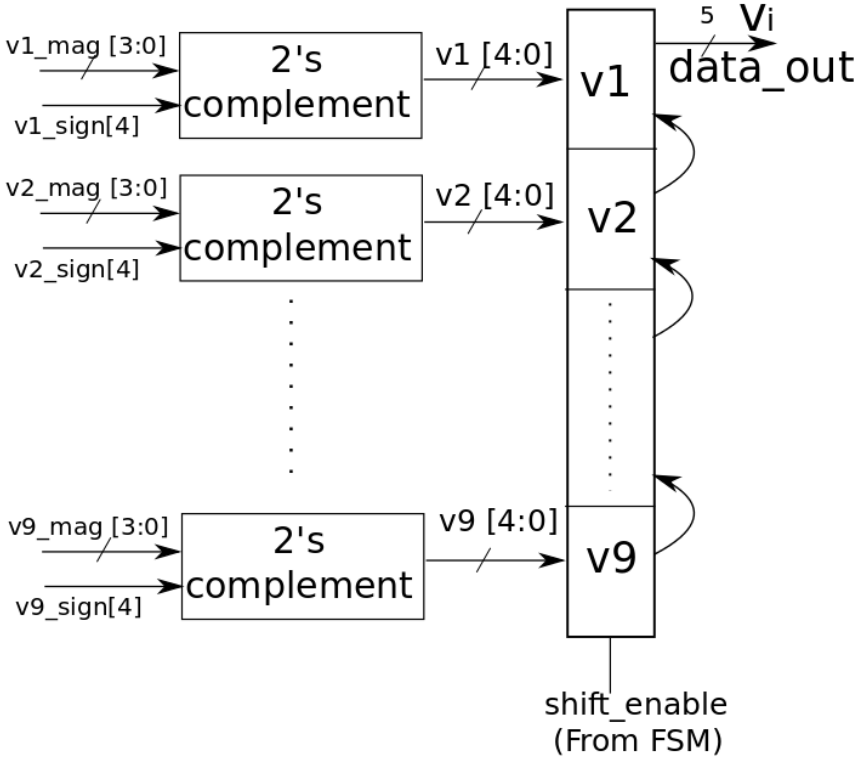
Check-node Magnitude and Sign Calculation





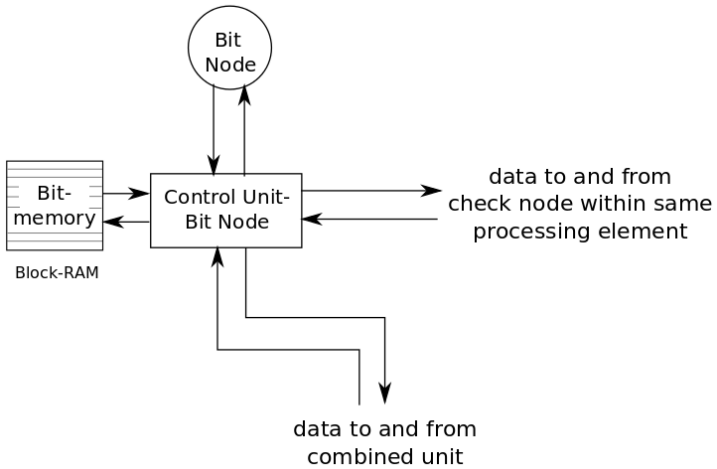
Check Node FSM and Output Unit





Control Unit- Bit node Functionality

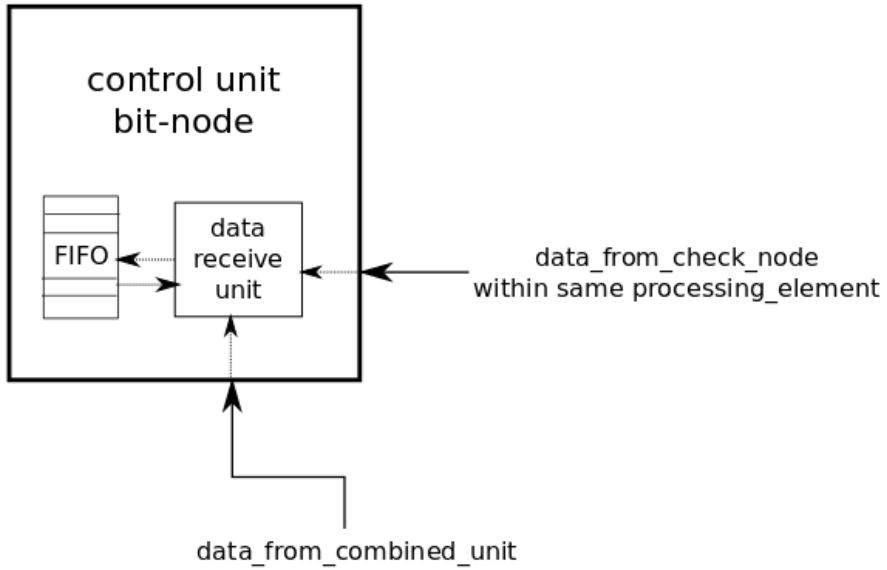
1. Receive data from Router and check for validity
2. Store data to proper location in the memory
3. Keep track of all the data is received or not.
4. As soon as all the data required for any one bit-node is received, send this data to bit-node and allow bit-node for completion of its operation. Set corresponding flag bit for that *bit_node_id*.
5. Send the data to proper destination by appending additional information.
6. Wait for all (5) bit-node within the PE to complete its operation.



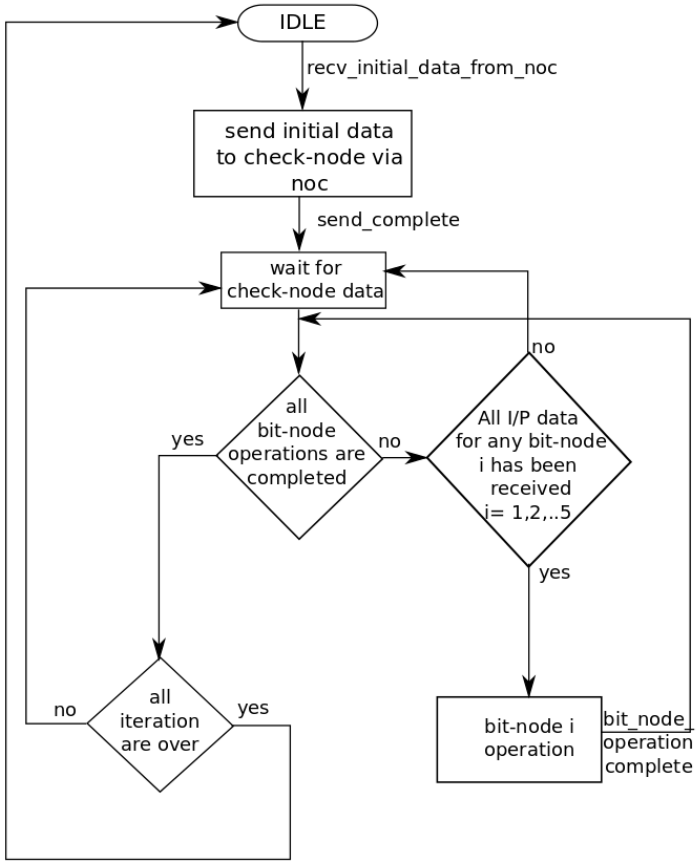
CU-bit node

Control Unit for bit node is divided into 3 units

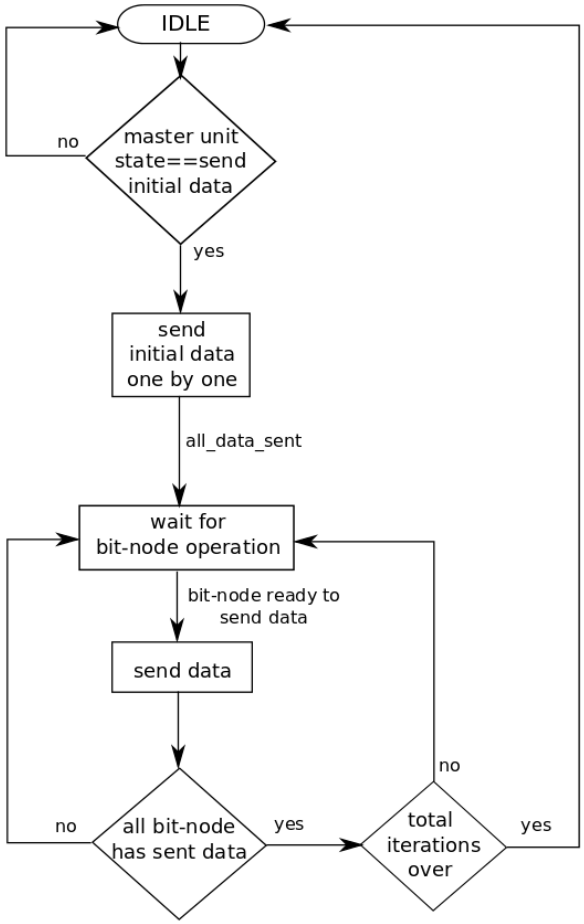
1. Data Receive Unit
2. Master Unit
3. Data Send Unit



1. Data Receive Unit



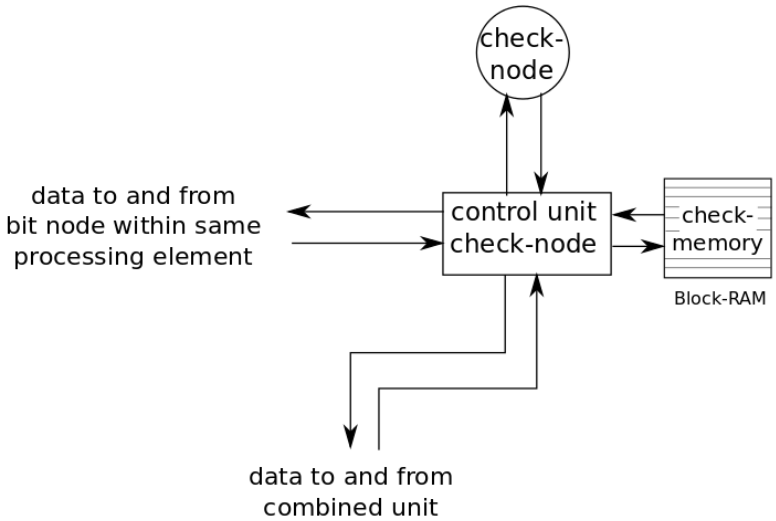
2. Master Unit Flow-chart



3. Data Send Unit Flowchart

Control Unit- Check node Functionality

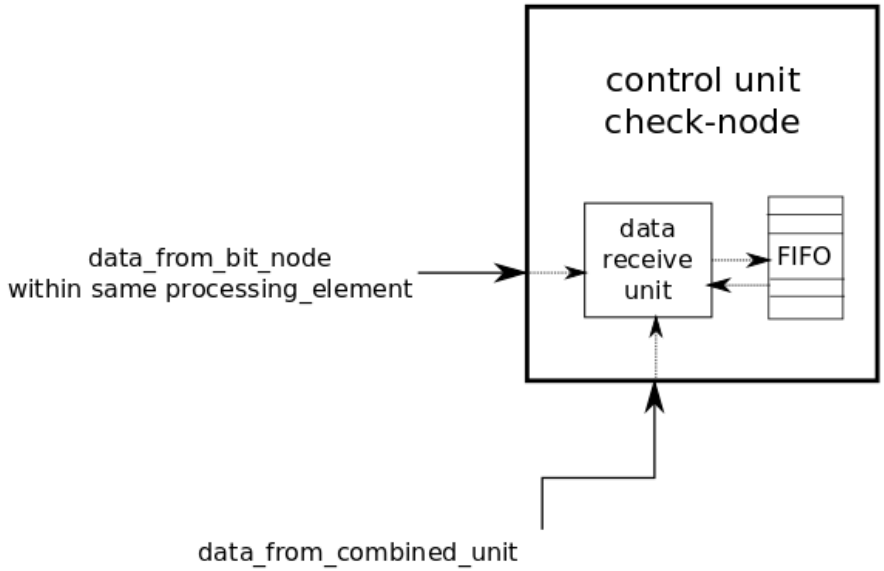
- Control Unit Check Node functionality is same as that of Control Unit Bit node.



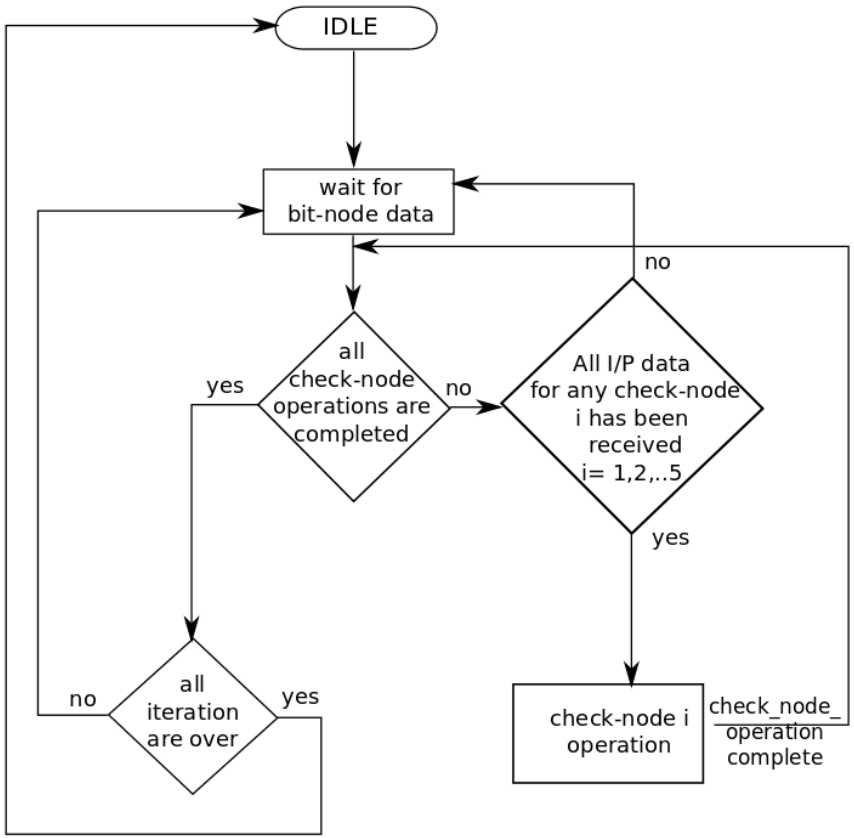
CU-check node

Control Unit for Check node is divided into 3 units

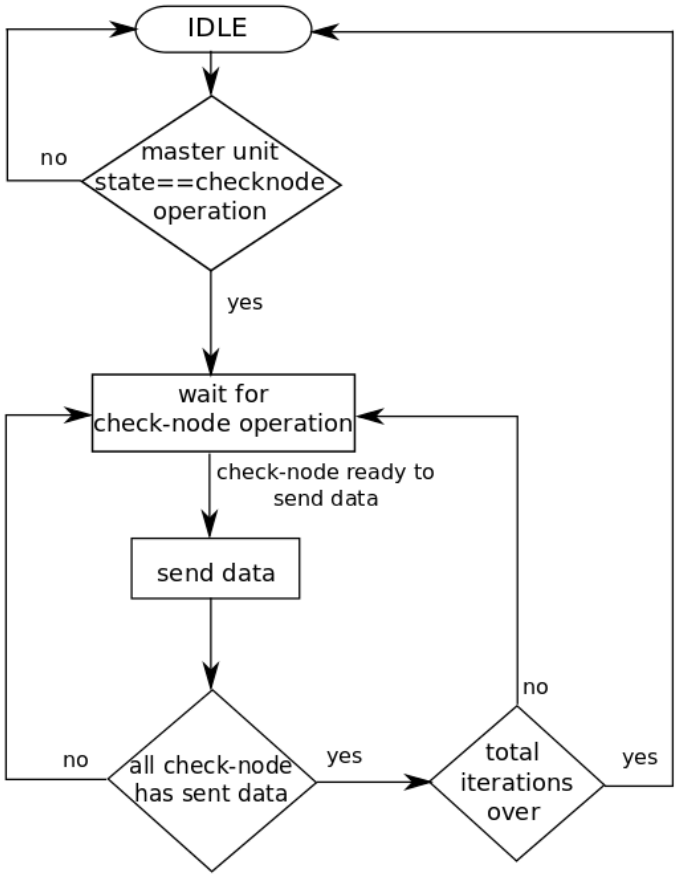
1. Data Receive Unit
2. Master Unit
3. Data Send Unit



1. Data Receive Unit



2. Master Unit Flow-chart



3. Data Send Unit Flow-chart

Bit node and check node combined Unit

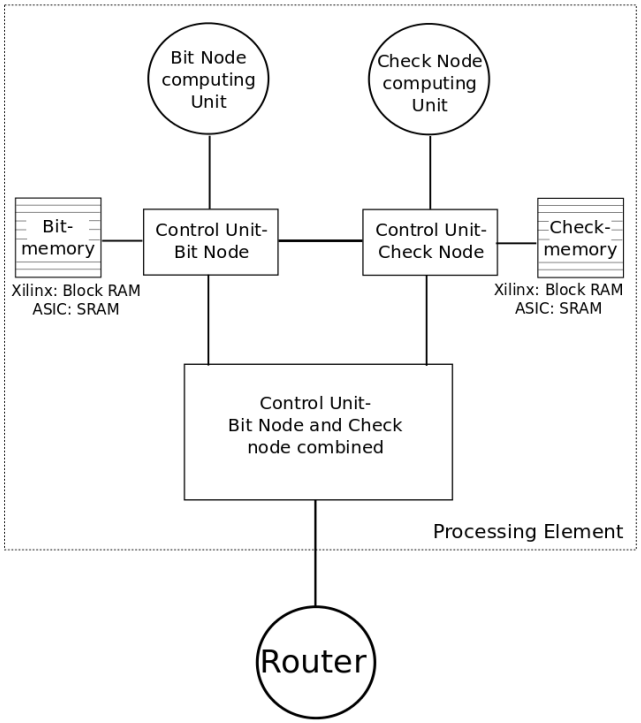


Figure : Processing Element

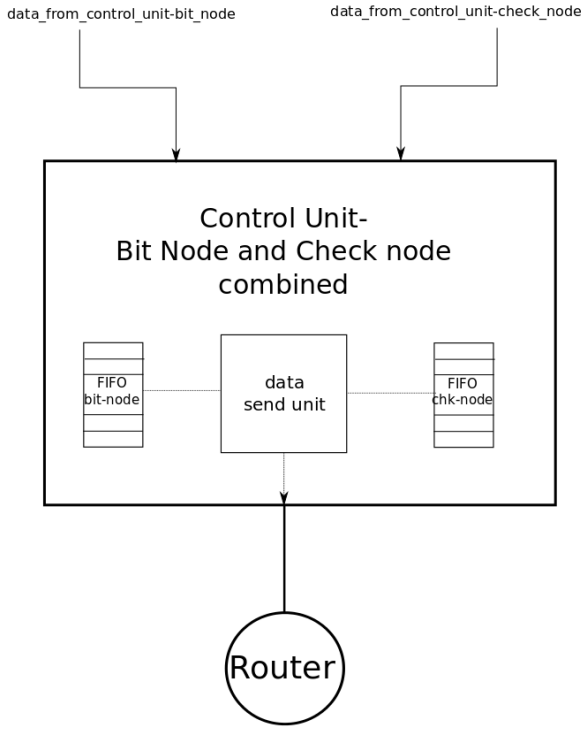
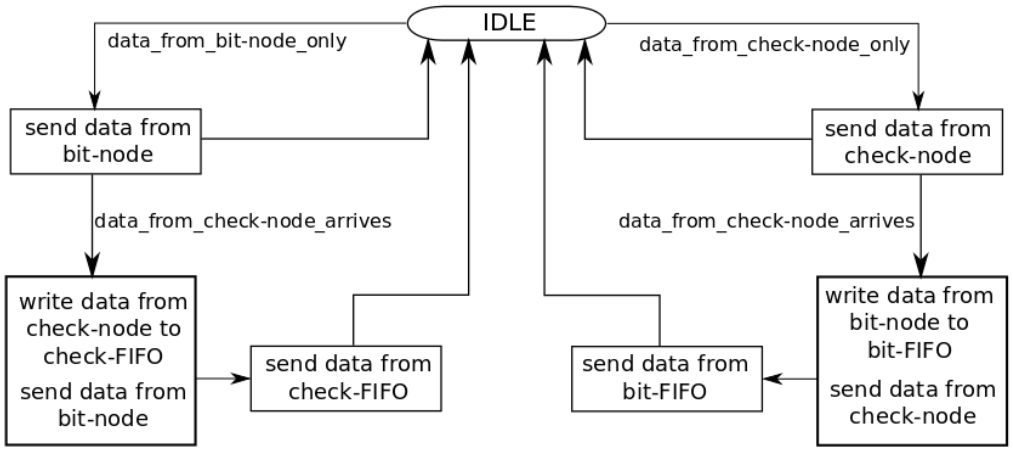


Figure : Block Diagram

Data collision Handling



1. When data is arrived from one source only, channel is provided.
2. While one node is sending data, if the other node data requests the channel, the current channel is allowed to complete its data transmission.
3. The requesting node which is ready stores its data in a FIFO.
4. After the current node completes its transmission, the data stored in the FIFO is sent.

Results for Xillinx Virtex-5 :XC5VLX110FT

Parameters	Results
Minimum Clock-cycles (bit-node)	(28×5) + Control Unit Operation + Sending and Receiveing flit overhead
Minimum Clock-cycles (check-node)	(30×5) + Control Unit Operation + Sending and Receiveing flit overhead
Total cycle for one complete iteration	423

	Available Resources	PG-LDPC (73,45) (Fold Factor=5) using Mesh 4×4 NoC	PG-LDPC (73,45) using Mesh 4×4 NoC using partial routing
Number of Slice Registers	69120	35433 (51%)	27278 (39%)
Number of Slice LUTs	69120	48270 (69%)	62418 (90%)
Number of Block RAM/FIFO	148	23 (15%)	0 (0%)
Number of BUFG/BUFGCTRLs	32	16 (50%)	0 (0%)
Maximum Operating Frequency	-	67.203 MHz	44.528 MHz
Nos. of clock cycles	-	423	135
Data Rate	-	11.6 Mbps	14.8 Mbps

ASIC Implementation

Technology	UMC 65 nm
Gate area	0.96 μm^2
Number of Gates	1085239
Number of Cells	258241
Chip Area	1041830.1 μm^2
NoC maximum operating frequency	140 MHz
Processing Element maximum frequency	300 MHz
Complete System maximum frequency	80 MHz
Number of clock cycles	423
Data Rate	13.81 Mbps

Observations

- The complete structure can be fitted within Virtex-5 FPGA board
- Bit node and check node computing units are simplified by applying folding approach, we can expect resource utilization should be reduced by 5 (Folding Factor), but area is reduced by a factor of < 2.
- Even if the Bit-node and Check-node architecture have been simplified, the Control Units become more complex than Bit-node and Check-node computing unit.
- Comparison shows that the resources required for Control Unit-bit node and Control Unit-check node is 3 to 10 times more than their respective computing units.
- Since Each Folded Bit-node requires one Control Unit and each Folded Check-node requires one Control Unit, the overall resource utilization.

Table : Resource Utilization of Bit-node Control Unit and Computing Unit

Resource	Bit-Node Computing Unit	Bit-node Control Unit
Number of Slice Registers	83	689
Number of Slice LUTs	98	918
Number of Block RAM/FIFO	0	1
Number of BUFG/BUFGCTRLs	1	2

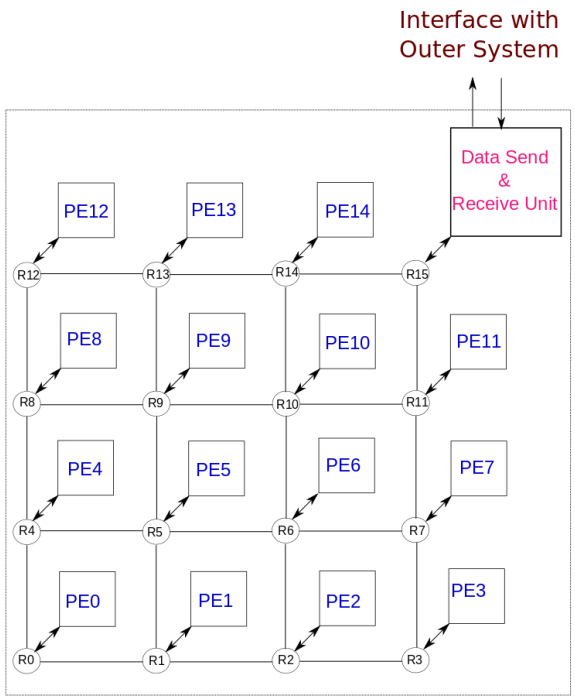
Table : Resource Utilization of Check-node and Computing Unit and Control Unit

Resource	Check-Node Computing Unit	Check-node Control Unit
Number of Slice Registers	266	766
Number of Slice LUTs	361	1016
Number of Block RAM/FIFO	0	1
Number of BUFG/BUFGCTRLs	1	2

Remedy

- Control Units for both the bit-node and check-node is the bottleneck.
- One possible way is to **decode multiple data sequences simultaneously**.
- Doing this will keep the control unit as it is, only add multiple computing units for bit node and check node.

PG-LDPC (73,45) (Fold Factor=5) Implementation on Mesh 4×4 NoC using Multiple Computing Units



PEn: Processing Element no. n
Rn: Router no. n

- LDPC Decoder for multiple data sequence is implemented in the same way as it is implemented for single data sequence.

System Implementation

For multiple data to decode simultaneously (for different n values),

1. Bit node Computing Unit and Check node Computing Unit are replicated n times.
2. Each Computing Unit implementation is identical
3. Irrespective of number of computing units, Control Unit for Bit-node=1, Control Unit for Check-node=1.
4. Number of bit-memory=1 and number of check-memory=1, Their depth is the same (i.e. number of words=50), but the width changes linearly (i.e. $5 \times n$)
5. Flit data overhead remains invariant irrespective of number of computing elements used.

Processing Element

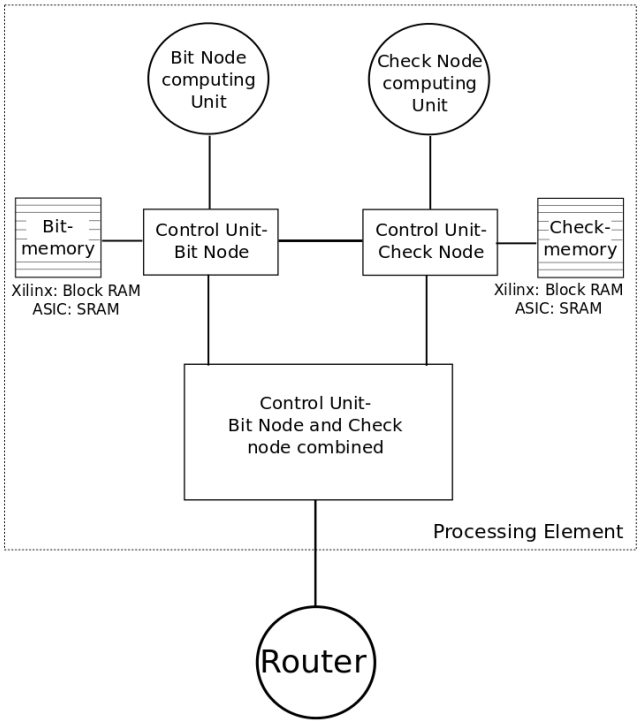


Figure : Single Computing Unit

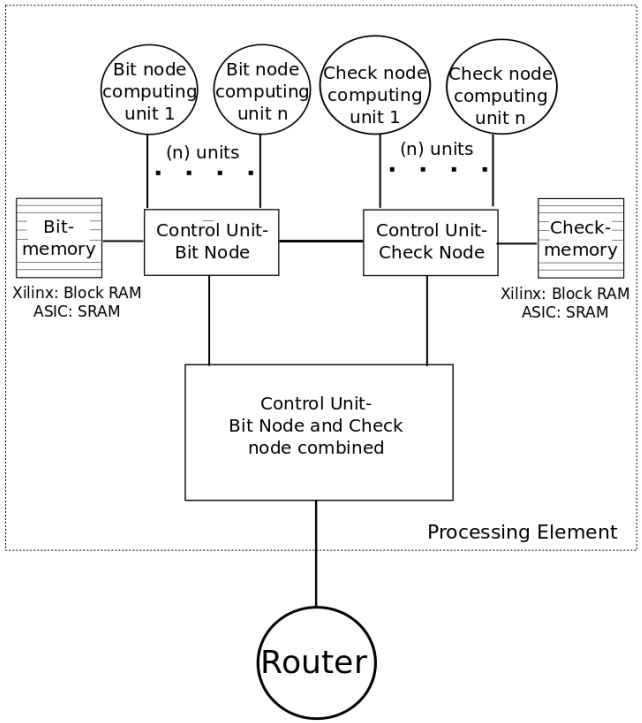


Figure : Multiple Computing Units

Flit structure

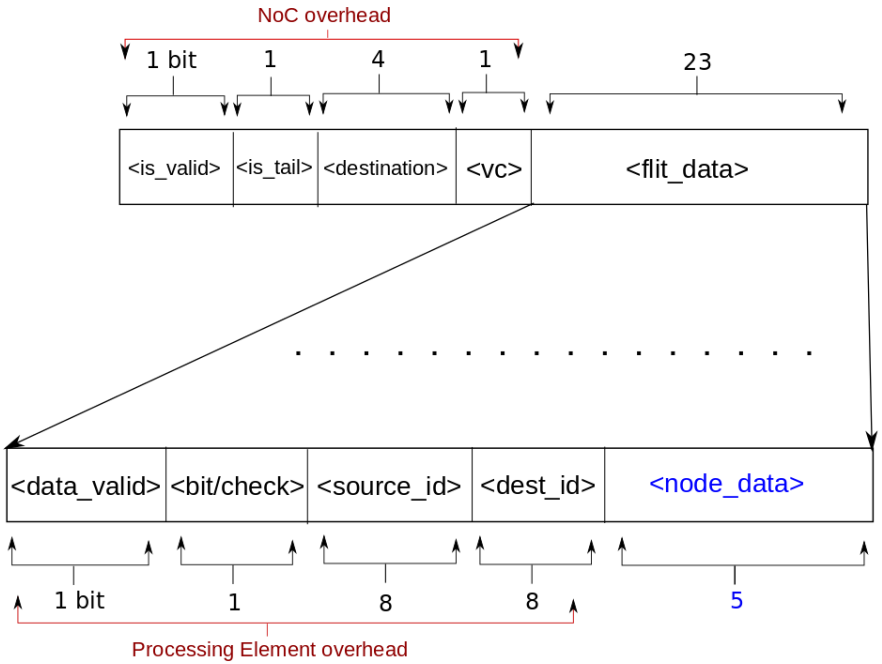


Figure : Single Computing Unit

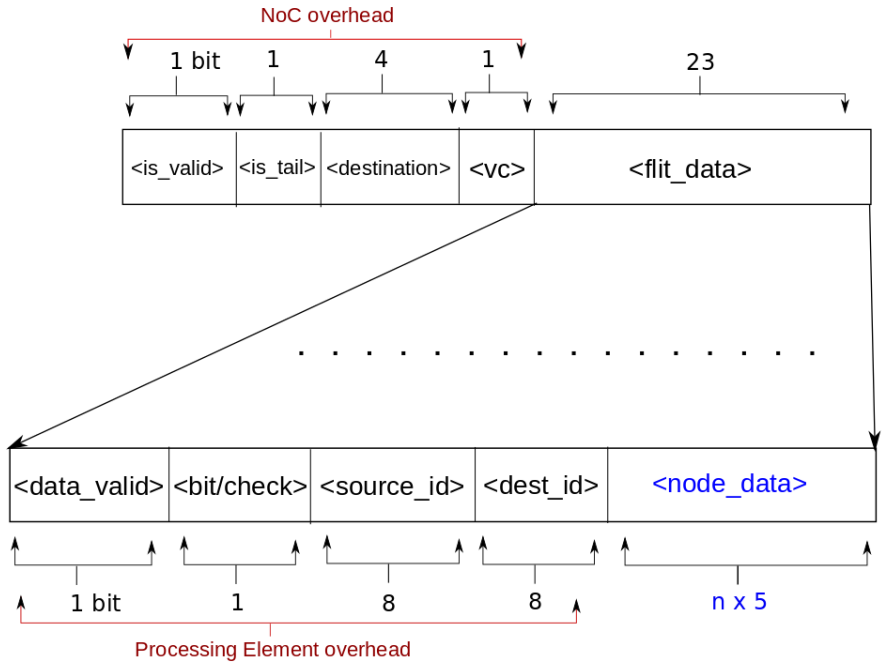


Figure : Multiple Computing Units

Results

Number of Computing Unit	Available Resources	1	2	3	4
Data width (bits)	-	5	10	15	20
Flit overhead (bits)	-	18	18	18	18
Flit width= Data width+ Flit overhead (bits)	-	23	28	33	39
Flit utilization ($\frac{Data\ width}{Flit\ width}$)	-	22%	36%	45%	51%
Slice Registers	69120	35928 (51%)	46097 (66%)	55926 (80%)	66395 (96%)
Slice LUTs	69120	49219 (71%)	60993 (88%)	64064 (92%)	74469 (107%)
Bonded IOBs	640	82 (12%)	160 (25%)	238 (37%)	316 (49%)
Block RAM	148	15 (10%)	15 (10%)	15 (10%)	15 (10%)
Normalized Resources	-	1	1.37	1.69	2.05
Maximum Operating Frequency	-	67.203 MHz	69.132 MHz	68.454 MHz	66.862 MHz
Nos. of clock cycles	-	423	423	423	423
Data Rate (megabits/sec)	-	11.597 Mbps	23.86 Mbps	35.44 Mbps	46.15 Mbps

Observation

- Data rate increases by the factor of n.
- Resources increase approximately 33%.
- Reason is that Flit overhead and memory depth remains invariant.
- Control Unit for bit node and check node each, which are the most complex parts of the system, remains of of fixed size irrespective of the data width.

Summary of work done

- LDPC is a complex system, NoC can be used for data routing mechanism.
- NoC has been used for data routing media, which takes care of data routing congestion.
- Bit nodes and check nodes are decoupled from PG structure and the clusters has been formed.
- These decoupled clusters can be connected to NI of NoC, which reduces interconnect complexity.
- PG-LDPC (73,45) has been implemented using this approach.

Summary of work done (contd.)

- The concept of folding of PG-LDPC has been explored. The circulant property of folding allows to exploit folding.
- PG-LDPC (73,45) with Fold Factor of 5 has been implemented using Mesh 4×4 NoC,
- The control unit resource requirements and timing parameters are complex than computing unit.
- To overcome control unit bottleneck, multiple data sequences are being decoded simultaneously.

Conclusion

- PG-LDPC structure has been folded and this folded PG-LDPC structure has been decoupled into cluster of computing nodes.
- These clusters have been connected to network interface of NoC.
- The idea provides flexibility for connecting the clusters to any kind of network interface of NoC.
- This idea provides scalability to implement larger complex system like PG-LDPC (1057,813).
- The flexibility and scalability has been achieved at the cost of reduced data rate.

Future Work

- This concept can be generalized to design larger LDPC decoder of PG-LDPC (1057,813) using multiple FPGA (NoC partitioning and multiple data sequence decoding.)
- CONNECT NoC used is very generalized. Custom NoC can be built which occupies less resource overhead yet good latency.
- Control Unit for bit node and Check nodes need to be optimized.
- Degree of Bit node and check node is large. The architecture with smaller degree can be utilized
- Alternate approach other than PG can be explored.

Reference


Subhasis Das, Hrishikesh Sharma, Sachin Patkar, "A Min-Sum based 800 Mbps LDPC decoder on FPGA", *Technical Report-Dept of Electrical Engineering, IIT Bombay*


Yu Kou, Shu Lin, and Marc P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results", *EEE Transactions on Information Theory, pages 27112736, 2001*


Shu Lin and Daniel J. Costello *Error Control Coding, Second Edition. Prentice-Hall,Inc.*


Michael K. Papamichael, "Fast Scalable FPGA-Based Network-on-Chip Simulation Models" *Computer Science Department, Carnegie Mellon University*

Reference


Yatish Turakhiya "Multi-FPGA Hardware Acceleration of Boolean Matrix Vector Multiplication using Network-on-Chip", *DDP Thesis- Dept of Electrical Engineering, IIT Bombay*


Shu Lin and Daniel J. Costello *Error Control Coding, Second Edition. Prentice-Hall,Inc.*


Hrishikesh Sharma "Exploration of Projective Geometry-based New Communication Methods for Many-core VLSI Systems", *PhD thesis, IIT Bombay, India, 2012*

Thank You