

LDPC Decoder Implementation using NoC

Shaishav Yogeshkumar Shah (133070050)

Supervisor: Prof. Sachin B. Patkar,
PhD Guide: Pinalkumar Engineer.
Dept. of EE,
IIT Bombay

May 26, 2015

Acknowledgement

- ▶ **Prof. Sachin B. Patkar Sir** for giving me opportunity to present this idea and his consistent suggestion.
- ▶ **Pinal J. Engineer**, PhD student for his good attention, idea suggestion, feasibility and troubleshooting..

LDPC Introduction

NoC

Why NoC for LDPC

LDPC 7×7 Implementation of NoC

LDPC 73×73 Implementation using NoC Mesh 4×4

Conclusion and Future Work

Introduction

- ▶ Found by Robert G. Gallager in his PhD thesis from 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 Decoding Algorithm- Sum Product Algorithm (SPA)

The SPA works as follow:

1. Initialize all $L(u_{ij}) = L(c_i)$
2. At check-node j calculate $L(v_{ji})$ as

$$L(v_{ji}) = (\prod_{i' \in v_{j1}} \alpha_{ij}) \cdot \phi(\sum_{i' \in v_{j1}} \phi(\beta_{i'j}))$$
 Also,

$$\phi(x) = \log\left(\frac{e^x + 1}{e^x - 1}\right)$$
3. At bit-node i , calculate $L(u_{ij})$ as

$$L(u_{ij}) = L(C_i) + \sum_{j' \in C_{ij}} L(v_{j'i})$$
4. Calculate $L(U_i) = L(c_i) + \sum_{j' \in C_{ij}} L(v_{ji})$
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 iteration exceeds a threshold, End
 else go to step (2)

Modification to SPA- Min Sum Approximation

- ▶ $\phi(\sum_{i' \in V_{j1}} \phi(\beta_{i'j})) \approx \phi(\phi(\min_{i' \in V_{j1}} \beta_{i'j})) = \min_{i' \in V_{j1}} \beta_{i'j}$
because $\phi(\phi(x)) \approx x$
- ▶ consider 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 $2^{2m} + 2^m + 1 = 73$.
- ▶ 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 $\phi(\sum_{i' \in V_{j1}} \phi(\beta_{i'j})) \approx (\min_{i' \in V_{j1}} \beta_{i'j} - \beta_0)$

NoC Introduction

- ▶ On a single chip, multiple processing units, memory, peripherals want to communicate with each-other.
- ▶ Trade-off between high speed more complex- point to point transmission and Low speed simpler- Shared bus data transmission.
- ▶ Ease of verification and logic error determination

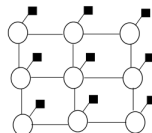
NoC Topologies and Routing strategies

NoC Topologies

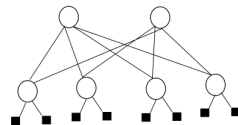
- ▶ Mesh
- ▶ Fat-Tree
- ▶ Ring, Bus etc

NoC Routing Strategies

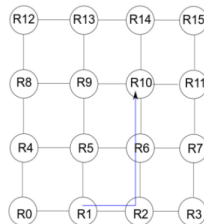
- ▶ XY Routing
- ▶ Adaptive Routing



Mesh Noc 3x3



Fat Tree Noc



XY Routing

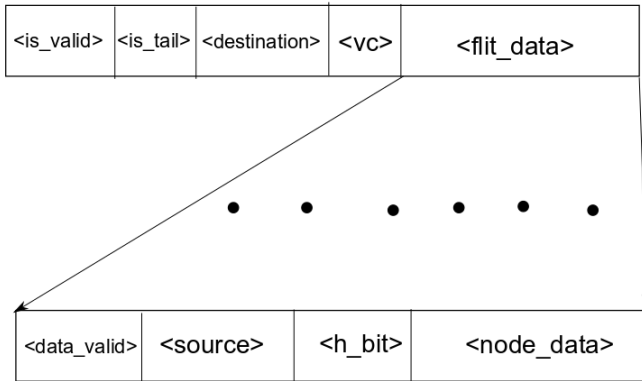
CONfigurable NEtwork Creation Tool

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 cycle and +1 cycle per additional router distance.
- ▶ Provision of custom parameter selection i.e. Data width, number of I/O buffers, Routing strategies.
 Enables the selection of optimum NoC for our configuration

Parameter	Value
Network Topology	
Topology ⓘ	Mesh ▾
Routers per Row	4 ▾
Routers per Column	4 ▾
Expose Edge Ports ⓘ	<input type="checkbox"/>
Network and Router Options	
Router Type ⓘ	Virtual Channel (VC) ▾
Number of VCs ⓘ	2 ▾
Flow Control Type ⚠	Credit-Based Flow Control ▾
Flit Data Width ⓘ	64 ▾
▼ Advanced Options (click to expand)	
Flit Buffer Depth ⓘ	8 ▾
Allocator ⓘ	Separable Input-First Round-Robin ▾
Pipeline Router Core ⓘ	<input type="checkbox"/>
Pipeline Allocator ⓘ	<input type="checkbox"/>
Pipeline Links ⓘ	<input type="checkbox"/>
Use Virtual Links ⓘ	<input type="checkbox"/>
Debug Symbols ⚠	None ▾
Contact and Delivery Info	
Name	First Last
Affiliation	
Email ⓘ	Valid email required
<input type="checkbox"/> I have read, understood, and I agree to the license terms	
<input type="button" value="Generate Network"/> ← click here to generate network	

Flit-Data structure

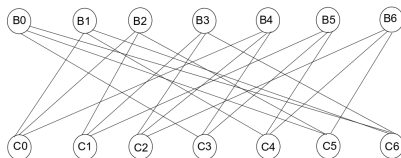


Why NoC for LDPC

- ▶ For LDPC 73×73 - each bit node is connected to 9 check-nodes and vice versa.
Total interconnection between bit-nodes and check-nodes are
(Nos. of Bit-nodes \times Degree of Bit-node) + (Nos. of Check-nodes \times Degree of Check-node)) \times Datawidth
 $= ((73 \times 9) + (73 \times 9)) \times 5$
 $= 6570$
- ▶ Implementation issues for larger systems.
- ▶ the communication issues and flow control are taken care by data routing mechanism.
- ▶ Caution: In practice selection of NoC is very critical work because larger NoC may create more overhead for system.

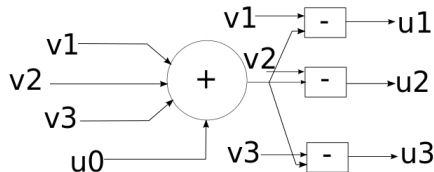
Tanner Graph for LDPC 7×7

- ▶ (7,4) PG-LDPC code which is obtained by a projective geometry over $GF(2^m)$, $m=1$.
- ▶ The degree of total bit-node and check-node is $2^m + 1 = 3$.
- ▶ Cyclic Code
- ▶ In my design, 5 bit precision :1 bit for sign, 3 bit of integer precision and 1 bit of fraction precision



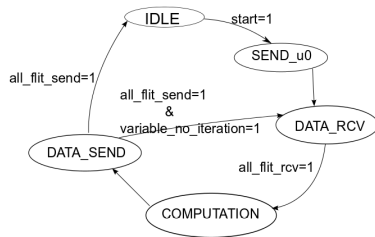
Bit-Node Architecture for LDPC 7×7

- ▶ Since the Algorithm is Min- sum, Bit-node performs the summation of adjacent data and allows to pass.



Bit-Node FSM for LDPC 7×7

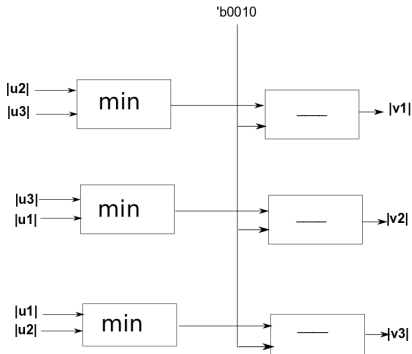
- ▶ At the starting of the iteration bit-node sends data to router (3 cycle)
- ▶ When data from check-node arrives, Bit-node performs the computation in one cycle.
- ▶ Nos. of Clock Cycle Required for Bit-node computation + communication overhead
 1 Iteration = 5 cycles
 n iteration = $5 + 4 \times (n-1)$



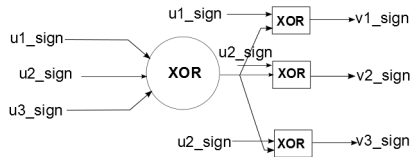
Note: The Same FSM for Check-node except SEND_u0 state

Check-Node Architecture for LDPC 7×7

Magnitude Calculation

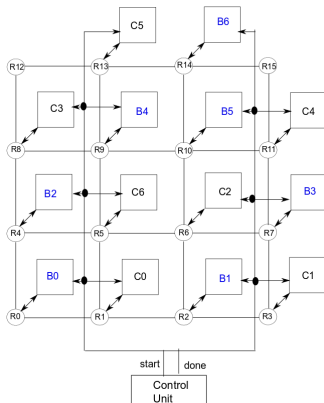


Sign Calculation

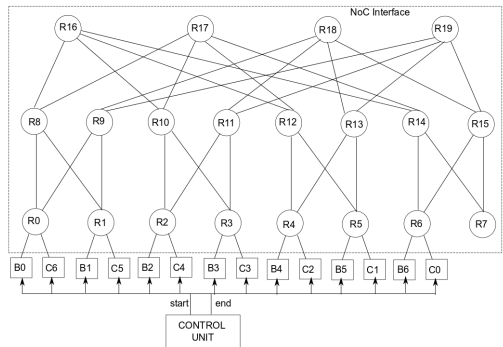


Final Implementation of LDPC 7×7

Implementation on 4×4 Mesh NoC



Implementation on Fat Tree NoC



Final Implementation of LDPC 7×7

Parameters	LDPC on 4x4 Mesh NoC	LDPC on Fat-Tree-16 NoC
Clock-cycle (1 iteration)	18	16
Clock-cycle (n iteration)	18	16
Total cycle for n complete iteration	$18 + (n-1)*12$	$18 + (n-1)*11$

Table : Results for LDPC 7x7 on NoC

Synthesis Report of LDPC 7×7 for Xilinx Virtex-5: XC5VLX110FT

	LDPC 7x7 (4x4 Mesh NoC)	LDPC 7x7 (Fat- Tree NoC 16)
Number of Slice Registers	5578 (8%)	5539 (8%)
Number of Slice LUTs	10886 (15%)	12347 (17%)
Maximum Operating Frequency	36.327MHz	38.279MHz
Nos. of clock cycles	18	16
Data Rate	14.12 mega bits/s	16.74 mega bits/s

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

Comments

- ▶ Area occupancy for both of the cases are nearer to the same
- ▶ The data-rate is higher in Fat-tree NoC
- ▶ Fat Tree seems to serve as the good choice

Comments

- ▶ Area occupancy for both of the cases are nearer to the same
- ▶ The data-rate is higher in Fat-tree NoC
- ▶ Fat Tree seems to serve as the good choice

But,

- ▶ It's not.
- ▶ As nos of PE increases beyond 16, nos. of routers increases exponentially
- ▶ Maximum operating Frequency also decreases.

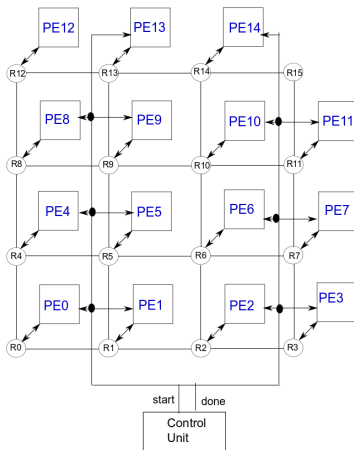
NoC Performance Comparison and Resource Utilization for Virtex-5: XC5VLX110FT

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%)

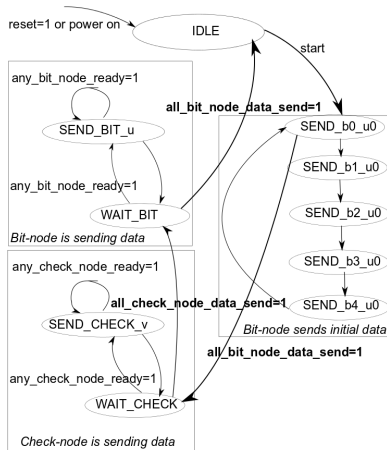
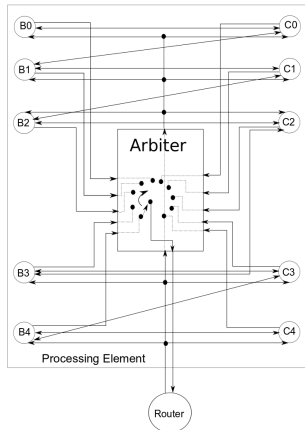
How to implement

- ▶ The approach of connecting bit-node and check-node to individual router has lack of scalability due to the overhead of NoC complexity.
- ▶ One idea is the trade-off between increased complexity, good latency- PG LDPC without NoC and increased latency, moderate complexity- Full connected NoC
- ▶ Use Partially Routed LDPC using NoC.
- ▶ LDPC 73×73 Decoder is implemented on Mesh NoC 4×4.
- ▶ 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.

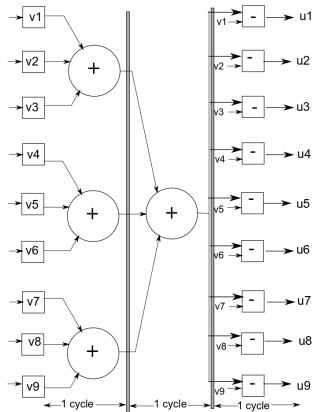
System Implementation



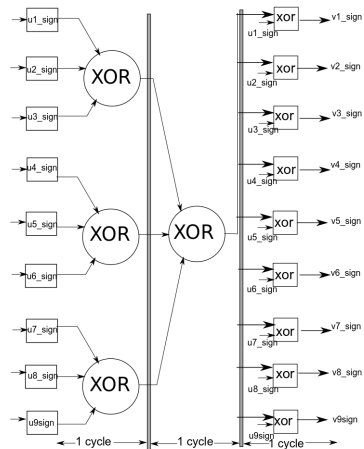
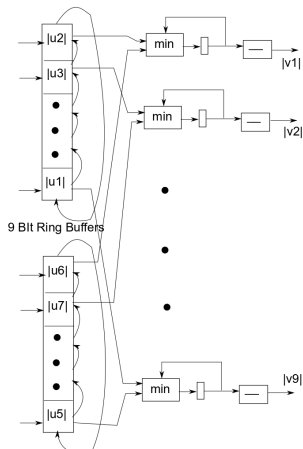
Processing Element Architecture



Bit-node Architecture for LDPC 73×73



Check-node Magnitude and Sign Calculation



Timing measurements of Bit-node and Check-node

Parameters	Results
Minimum Clock-cycle (bit-node)	38 + sending and receiving flit overhead
Minimum clockcycle (check-node)	40 + sending and receiving 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





	LDPC 73x73 (directly connected)	LDPC 73x73 on 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 mega bits/s	14.8 mega bits/s

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

Conclusion and Future Work

- ▶ Since LDPC is a complex system, NoC can be used for data routing mechanism.
- ▶ Connecting single bit-node and single check-node to individual router doesn't give good performance
- ▶ For less complexity, we can connect multiple bit-nodes and check-nodes to a single router and partial arbitration can be performed.
- ▶ By communicating between multiple FPGAs, larger systems can be implemented

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*
-  T. Theocharides, G. Link, N. Vijaykrishnan, M. J. Irwin, “Implementing LDPC Decoding on Network-On-Chip“, Penn State University
-  Yatish Turakhiya “Multi-FPGA Hardware Acceleration of Boolean Matrix Vector Multiplication using Network-on-Chip“, *DDP Thesis- Dept of Electrical Engineering, IIT Bombay*
-  Michael K. Papamichael, “Fast Scalable FPGA-Based Network-on-Chip Simulation Models“ *Computer Science Department, Carnegie Mellon University*

Thank You