Fault Tolerant Projective Geometry Based Low Density Parity Check Code Decoder

B.Venkateshulu[†], Dr.B.Suryanarayana Adiga[††] and Dr.B.C.Jinaga[†††]
ven66@rediffmail.com , bs.adiga@gmail.com , jinagabc@gmail.com

[†]Associate Prof.,Dept. of ECE, G.N.I.T.S,Shaikpet, (J.N.T.U.H affiliated), Hyderabad, 500 008, A.P.,India
[††]Principal Scientist, Innovation labs, TCS. White field, Bangalore,Karnataka,India
[†††]Ex-Rector, J.N.T.U.H,.Kukatpally, Hyderabad, A.P., India

Summary

Low Density Parity Check (LDPC) codes are a special case of error correcting codes having high throughput, good decoding performance, low implementation complexity, low decoding latency, as well as no error floors at high SNR's. However the implementation of the fully parallel LDPC decoder is impeded by the complexity of the interconnection network. The Perfect Difference Network (PDN) based interconnection schemes have been suggested to negotiate many of the interconnection problems to accommodate regular/irregular LDPC codes in a network of diameter 2 . When a single node/link failure occurs the diameter of the network increases to 3 and thus latency of communication increases. In this paper we describe a modified version the PDN ,namely 0-free PDN, which does not suffer from this problem. The paper contains detailed analysis of an example of 0-free PDN and its application to an LDPC decoder.

Key words:

LDPC Decoder , Perfect Difference Set, Projective Geometry, Perfect Difference Network (PDN), 0—free PDN.

1. Introduction

Low Density Parity Check (LDPC) codes are a special case of error correcting codes that have been recently receiving a lot of attention because of their high throughput and good decoding performance. Inherent parallelism of the message passing decoding algorithm for LDPC codes makes them suitable for hardware implementation. Also they have low implementation complexity, low decoding latency, as well as no error floors at high SNR's. However the implementation of the fully parallel LDPC decoder is impeded by the complexity of the interconnection network. In [1] Perfect Difference Network (PDN) based interconnection schemes have been suggested to negotiate many of the interconnect problems faced by earlier decoders. Some salient features of the PDN based decoders are:

1. The design is independent of LDPC code specifications. Decoders are flexible enough to accommodate regular/irregular LDPC codes.
2. The diameter of the interconnection network is 2.
3. The network is d-regular with d << total number of nodes (processors) in the network. This implies that for a given node the set of neighbors is small, thus making interconnection topology less complex.
4. All communications among nodes (processors) are conflict-free.
5. Features 2, 3 and 4 keep communication latency very low.

Perfect difference networks are based on the mathematical notion of perfect difference sets (PDS). A PDS is a set $\{s_0, s_1, s_2, s_3 \ldots s_m\}$ of m+1 integers having the property that their m^2+ m differences $s_i - s_j$, 0<= i not equal to j <= m, are congruent to modulo m^2+ m +1, to the integers 1,2,3…… m^2+ m in some order. Singer [2] has given explicit construction of PDSs for m a prime or prime power using incidence relation between points and lines in 2-dimensional projective space. There always exists a PDS of the form $\{0, 1, s_2, s_3 \ldots\ldots\ldots s_m\}$ for a given m which is called canonical PDS. A Perfect Difference Network is constructed using the canonical PDS. There are N = m^2+m+1 nodes in the network numbered from 0 to N-1. Node i is connected to nodes i+/-1 and nodes i+/-s_j, 2<=j<=m and i = 0 to N-1. "+" leads to forward link connections and "–" leads to reverse link connections. Some of the properties of PDN are:

1. The diameter of the network is 2.
2. One-to-all, all-to-all and personalized communications can be executed in 2m time steps with nodes having single port facility or in 2 time steps with multi port facility. This is proved in [3];
3. All communications are conflict-free. This is proved in [3]

In [4] an LDPC decoder based on the concepts of PDN is described. However when a single node/link failure occurs, the diameter of the network increases to 3 and thus latency of communication increases. In this paper we describe a

modified version the PDN namely 0-free PDN which does not suffer from this problem. The paper contains detailed analysis of an example 0-free PDN and its application to an LDPC decoder that is essentially a Co-processor attached to Central Processing Unit (CPU) with a strategy to decode received LDPC code vector in the presence of single node/link failure in side the Co-processor.

2. Soft Decoding of LDPC codes

For each received bit x(n), n = 1 to code length N of an LDPC received vector, an LDPC decoder excepts as input the log–likelihood ratio $\boldsymbol{\alpha_n}$ of probability of possible values for x(n) as defined in [5]

$$\alpha_n = \ln\left[\frac{p_r(x(n))=1)}{p_r(x(n))=0}\right] \quad (1)$$

LDPC codes are represented as bipartite graphs made up of two families of nodes, variable nodes and check nodes. Variable nodes represent transmitted bits in the code including both information and parity bits. Each variable node is connected to a sparse set of check nodes through a sparse array of edges. Each check node represents a parity check constraint on the neighboring variable nodes.

LDPC decoders implement a message passing algorithm which specifies the computation of messages and their communication between variable nodes and check nodes as defined by the edges of the graph. An iteration of LDPC decoding consists of a round of message passing from each variable node to all its neighboring check nodes, followed by another round of message passing from each check node to its neighboring variable nodes. Decoding is achieved through iterations of message passing with some stopping criterion.

Let H be the m-by-n parity check matrix of an LDPC code having 'n' variable nodes and 'm' check nodes. The set v(m) = {n: $H_{m,n}$ = 1} defines the variable nodes that are neighbors of check node m. Similarly the set μ(n) = {m: $H_{m,n}$ = 1} defines the check nodes that are neighbors of variable node n. $\mathbf{Q_{n,m}}$ and $\mathbf{R_{m,n}}$ are computed variable node and check node messages.

Message from variable node n to check node m:

$$Q_{n,m} = \alpha_n + \left[\sum_{\mathbf{m?}\in\mu(\mathbf{n})\ \setminus\ \mathbf{m}} R_{m'n}\right] - R_{mn} \quad (2)$$

The notation $m' \in \mu(n) \setminus m$ simply means the indices $m'(1 <= m' <= n)$ of all bits in row $n(1 <= n <= m)$ which have value 1, not including the current bit index *m*

Message from check node m to variable node n:

$$R_{mn} = \Phi^{-1}\left\{\left[\sum_{n'\in v(m)\setminus n}\Phi(Q_{n'm})\right] - \Phi(Q_{nm})\right\} * \left\{sgn(Q_{nm}) * \prod_{n'\in v(m)\setminus n} sgn(Q_{n'm})\right\} * (-1)^{|v(m)|} \quad (3)$$

$$\Phi(x) = -\log\{\tanh(\frac{1}{2}abs(x))\} = \Phi^{-1}(x); x \geq 0 \quad (4)$$

In order to simplify the hardware for the computation of first term in (3), a lookup table is used. The second term is computed by applying an ex-or function on the most significant bits of the input messages with messages being represented as signed magnitude values. Further in the paper we focus on the interconnection scheme of the decoder and its robustness properties with a view to make it tolerant to single link/node failure. We do not delve on computational unit

3. Definition of 0-free PDS and 0-free PDN

A 0-free PDS does not contain 0 as an element. Many 0-free PDSs can be derived from a canonical PDS {0 1 s_1 s_2 s_3 …… s_m} For example adding 1 to all the elements of a canonical PDS yields a canonical 0-free PDS. A perfect difference network based on a 0-free PDS is a 0-free PDN.

The diameter of a network derived by removing one link/node from a 0-free PDN remains 2. This property is direct consequence of there being at least two edge-node-disjoint paths of length 2 between any two nodes, including those that are directly linked. Thus for a single node node/link failure, a 0-free PDN provides a stronger fault tolerance than an ordinary PDN whose diameter may increase to 3. 0-free PDN is d-regular graph with d = 2m+2.

As the network grows in size, probability of multiple failures increases. However PDNs used for LDPC decoding applications are of moderate size and therefore we restrict ourselves to single node/link failure. The fault diameter of a PDN is no longer greater than 4. Thus when multiple faults occur we should tolerate the latency introduced because of enhanced diameter which can be up to a maximum of 4.

In the following we describe a 0-free PDN with an example and give full details of the data communications during one-to-all, all-to-all and personalized communications. This is with a view to get feel of the network design.

3.1 Specifications for an example 0-free PDN

For illustration we consider a PDN with the following specifications:

m = p = 3; Number of nodes = m^2+m+1 = N = 13; Normal PDS = {0 1 3 9}. Fig.1 Shows the PDN for the above case.

To build a 0-free PDN, we consider the canonical 0-free PDS {1 2 4 10}. The 0-free PDN is constructed by establishing the following forward and reverse link connections

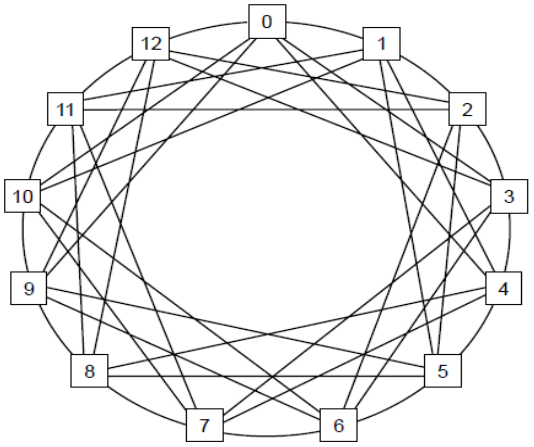

Fig. 1 A PDN with Number of Nodes =13

x +/- s_j (mod N) for nodes x = 0 to N-1. s_j for j = 1to 4 are the elements of the PDS. The node degree of the graph of PDN is 2m+2 and the graph is 2m+2 regular.

3.2 Interconnection Network for the 0-free PDN

The interconnection network for the 0-free PDN of the example is depicted in Table1. The two neighboring sets for each node in the Table are for forward and reverse links respectively. Thus the set of neighboring nodes of node 0 is {1 2 3 4 9 10 11 12} which is the union of the sets {1 2 4 10} and {12 11 9 3} in Table 1. The node degree is 2m+2 = 8. We note that neighboring node sets for the forward and reverse link connections for the nodes 1 to N-1 are derived from the sets for the node 0 as:

Forward:
{x+1 (mod N) x+2 (mod N) x+4 (mod N) x+10 (mod N)}

Reverse:
{x+12 (mod N) x+11 (mod N) x+9 (mod N) x+3 (mod N)}

3.3 Analysis of one-to-all broadcast in the presence of one node/ /link failure

Assume node 4 is the broadcasting node and one of the neighboring nodes of node 4 say node 8 has failed. During Phase1 node 4 establishes the reverse link connections given by eq.(5)

$$4 - s_j \pmod N \tag{5}$$

s_j for j = 1to 4 are the elements of the PDS. This means node 4 gets connected to the set of nodes {3 2 0 7} during Phase1 and sends its data to the nodes contained in the set. Thus at the end of Phase1 set of nodes {0 2 3 7} has the information from node 4.

During Phase2, the following forward link connections are established as given in eq,(6)

$$i + s_j \pmod N \tag{6}$$

i = 0, 2, 3 and 7; s_j for j = 1to 4 are the elements of the PDS. This means node 0 gets connected to the set of nodes {1 2 4 10} and then transmits node 4 data obtained during phase 1 to all the elements of the set. At the same time step node 2 gets connected to the set of nodes (3 4 6 12}, node 3 gets connected to set of nodes {4 5 7 0} and node 7 gets connected to set of nodes {8, 9, 11, 4}. All of them then transmit node 4 data obtained during phase 1 to the elements of the respective sets. The union of these 4 sets along with set {3 2 0 7} which was established during Phase1 is the set {0 1 2 3 4 5 6 7 8 9 10 11 12}. This implies node 4 can complete broadcasting its message in two phases.

Table 1: Interconnection Network of 0-free PDN

Nodes	Neighboring Nodes
0	[{ 1 2 4 10} { 12 11 9 3 }]
1	[{ 2 3 5 11} { 0 12 10 4 }]
2	[{ 3 4 6 12} {1 0 11 5 }]
3	[{ 4 5 7 0 } { 2 1 12 6 }]
4	[{ 5 6 8 1 } { 3 2 0 7 }]
5	[{ 6 7 9 2 } { 4 3 1 8 }]
6	[{ 7 8 10 3 } { 5 4 2 9 }]
7	[{ 8 9 11 4 } { 6 5 3 10}]
8	[{ 9 10 12 5 } { 7 6 4 11}]
9	[{10 11 0 6 } { 8 7 5 12}]
10	[{11 12 1 7 } { 9 8 6 0 }]
11	[{12 0 2 8 } { 10 9 7 1 }]
12	[{ 0 1 3 9 } { 11 10 8 2 }]

With a single port communication model where a node can send only one message in each time step the total broadcast time is 2m+2 time steps. Multi-port communication on the other hand leads to 2 time steps.

During one to all broadcasting of messages we take note of the following observations.

1. If there is a connection between a node and failed node communication is ineffective
2. If there is a connection between a node and node of message origin (in the present case node 4) the communication is ineffective.
3. If a node receives messages from two different nodes it obliges only one of them

3.4Analysis of All -to- All Broadcasting

All –to- all broadcasting involves each node sending a message to all other nodes in a network; hence, N distinct messages must be sent with each one going to N-1 destinations. But since one node is a failed node in the present analysis there are only N-1 distinct messages with N-2 distinct destinations for each one of them. Each node follows algorithm 1 with single port communication, independently. During Phase 1, the broadcast message of a node x is sent to all its neighbors x- s_j in m+1 steps. This action is conflict-free with other nodes because the neighbors u- s_j and v- s_j for distinct nodes u and v are distinct. At the end of this phase, each node except the ones connected to the failed node would have received m+1 broadcast messages from the m+1 neighboring nodes. On the other hand if a node is connected to the failed node it would have received only m broadcast messages from the m active neighboring nodes. In fact m^2-1 nodes would have received m+1 messages each and m+1 nodes would have received m messages each. At the end of Phase 1 each node would have received m or m+1 broadcast messages out of the expected N-2 broadcast messages. In the present example set of nodes {12 11 9 3) would have received broadcast message from node 0, set of nodes {0 12 10 4) would have

received broadcast message from node 1 and so on. In general set of nodes {12+x (mod N) 11+x (mod N) 9+x (mod N) 3+x (mod N)} would have received broadcast message from node x. This relation is because of the cyclic property of the network. The complete list of message transfer is given in table 2.

Note: x stands for failed message transfer because of node/link failure.

From the list of Table 2 we can prepare the list of broadcast messages received by N-1 different nodes at the end of Phase 1. This is given in table 3. For example node 0 receives broadcast messages from set of nodes {1 2 4 10}, node 1 receives broadcast messages from set of nodes {2 3 5 11}.

In general node x receives broadcast messages from set of nodes {x+1 (mod N) x+2 (mod (N) x+4 (mod N) x+10 (mod N)}.It should be noted that the elements of the set are all distinct. Refer Table 3. In a PDN with single port nodes, Phase 1 is completed in m+1 time steps .

Table2. Phase 1 of transfer of broadcast messages

Receiving Nodes	Sending Node
(12 11 9 3}	0
(0 12 10 4}	1
(1 0 11 5}	2
(2 1 12 6}	3
(3 2 0 7}	4
(4 3 1 x}	5
(5 4 2 9}	6
(6 5 3 10}	7
(x x x x}	8
(x 7 5 12}	9
(9 x 6 0}	10
{10 9 7 1}	11
{11 10 x 2}	12

In Phase 2 each node x establishes the forward link connections x+ s_j and transmits its own message and the messages it received from other nodes during Phase 1 to its neighbors. This means node 0 transmits the message set {0 1 2 4 10} to the node set {1 2 4 10}, node 1 transmits the message set {1 2 3 5 11} to the node set {2 3 5 11} and so on. In general node "x" transmits message set {x x+1 (mod N) x+2 (mod N) x+4 mod N x+10 mod N} to the node set (x+1 mod N) x+2 (mod N) x+4 (mod N) x+10 (mod N). This transaction is depicted in table 4. These transactions are completed in m+1 time steps in a PDN with single port facility . Also these transactions are conflict-free because the neighbors u + s_j and v + s_j for distinct nodes u and v are distinct.

Having Table 4 we can prepare list of broadcast messages received by each node at the end of Phase 2. This is depicted in Table 5. From table 5 we observe that union of message sets received by a node is the set {0 1 2 3 4 5 6 7 9 10 11 12}.

Thus all-to-all transmission is executed in 2 phases involving 2(m+1) steps with single port facility for each node. It should be noted that the set does not contain message 8 obviously because node 8 has failed.

During all-to-all broadcasting of messages we take note of the following observations.

1. If there is a connection between a node and failed node communication is ineffective
2. If there is a connection between a node and node of message origin the communication is ineffective.

3. If a node receives messages from two different nodes it obliges only one of them.

Table 3. Phase 1 transactions

Receiving Nodes	Set of Broadcasting Node
0	{ 1 2 4 10 }
1	{ 2 3 5 11 }
2	{ 3 4 6 1 2}
3	{ 4 5 7 0 }
4	{ 5 6 x 1 }
5	{ 6 7 9 2 }
6	{ 7 x 10 3 }
7	{ x 9 11 4 }
8	{ x x x x }
9	{10 11 0 6 }
10	{11 12 1 7}
11	{12 0 2 x}
12	{ 0 1 3 9 }

Table 4. Phase2 transactions.

Broadcast Message Set	**Set of nodes to which Broadcast Message Set is sent**
{0 1 2 4 10} from node 0	{ 1 2 4 10}
{1 2 3 5 11} from node 1	{ 2 3 5 11}
{2 3 4 6 12} from node 2	{ 3 4 6 12}
{3 4 5 7 0} from node 3	{ 4 5 7 0 }
{4 5 6 x 1 } from node 4	{ 5 6 x 1 }
{5 6 7 9 2 } from node 5	{6 7 9 2}
{6 7 x 10 3} from node 6	{7 x 10 3}
{7 x 9 11 4}from node 7	{x 9 11 4}
{x x x x x} from node 8	{x x x x}
{9 10 11 0 6}from node 9	{10 11 0 6}
{10 11 12 1 7}from node 10	{11 12 1 7}
{11 12 0 2 x }from node11	{12 0 2 x}
{12 0 1 3 9 } from node 12	{0 1 3 9}

Table 5. Status at the end of phase 2

Nodes	4 Sets of Broadcast Messages received
0	[(12 0 1 3 9 } {11 12 0 2 x } { 9 10 11 0 6} { 3 4 5 7 0}]
1	[(0 1 2 4 10} {12 0 1 3 9 } (10 11 12 1 7} { 4 5 6 x 1}]
2	[(1 2 3 5 11} { 0 1 2 4 10} (11 12 0 2 x} { 5 6 7 9 2}]
3	[(2 3 4 6 12} { 1 2 3 5 11} (12 0 1 3 9} { 6 7 x 10 3}]
4	[(3 4 5 7 0 } { 2 3 4 6 12} { 0 1 2 4 10} (7 x 9 11 4}]
5	[(4 5 6 x 1 } { 3 4 5 7 0 } { 1 2 3 5 11} (x 9 10 12 5}]
6	[(5 6 7 9 2 } { 4 5 6 x 1 } { 2 3 4 6 12} (9 10 11 0 6]
7	[(6 7 x 10 3 } { 5 6 7 9 2 } { 3 4 5 7 0 } (10 11 12 1 7}]
8	[(x x x x x } { 6 7 x 10 3 } { 4 5 6 8 1 } (11 12 0 2 x}]
9	[(x 9 10 12 5 } { 7 8 9 11 4} { 5 6 7 9 2 } (12 0 1 3 9}]
10	[(9 10 11 0 6 } { x 9 10 12 5} { 6 7 x 10 3 } (0 1 2 4 10}]
11	[(10 11 12 1 7 } { 9 10 11 0 6 } { 7 x 9 11 4 } (1 2 3 5 11}]
12	[(11 12 0 2 x } {10 11 12 1 7 } { x 9 10 12 5 } (2 3 4 6 12}]

3.5.Personalized Communications

Analysis similar to the all-to-all communications can also be done for personalized communications. It can be shown that all transactions can be executed in 2(m+1) steps in the presence of single node/link failure.

Thus the 0-free PDN still has diameter 2 in spite of the fact that it has a single node/link failure.

4. LDPC Decoding in the presence of single node/link failure

To illustrate our strategy we consider a projective geometry based LDPC code as given in [6] with the following specifications.This approach has the advantage of being quasi cyclic, lends itself to the generic structure while ,the other approaches do not provide this generic approach(irrespective of regular or irregular LDPC codes) The example LDPC code is highly structured. However the strategy envisaged can be employed for any LDPC codes including irregular LDPC codes. The LDPC decoder runs on the 0-free PDN explained in the previous section.

4.1 Code specifications:

p = 2; s = 3; Code length = $(p^s)^2 + p^s+1 = (2^3)^2+2^2+1 = 73$; Minimum distance of the code = $(p^s)+1 = (2^3)+1 = 9$; No of variable nodes = No of check nodes = $(p^s)^2 + p^s+1 = (2^3)^2+2^2+1 = 73$; Each variable node has a set of distinct = $(2^3)+1 = 9$ neighboring check nodes Each check node has a set of distinct $(2^3)+1 = 9$ neighboring variable nodes.

From the lines and points incidence relationship of the underlying 2-dimensional projective space, we can determine the neighbors (associated check nodes) of the 0^{th} variable node and they are the elements of the set {0 1 3 7 15 31 36 54 63}. Then the neighbors (associated check nodes) of an arbitrary variable node "x" are given by the check node set{x x+1(mod 73) x+3(mod 73) x+7(mod 73) x+15(mod 73) x+31(mod 73) x+36(mod 73) x+54(mod 73) x+63(mod 73)} for x = 1to 72 . Because the code is symmetric, the neighbors (associated variable nodes) of 0^{th} check node are given by the variable node elements of the set {0 1 3 7 15 31 36 54 63}.

Each node of the 0-free PDN is an arithmetic processor. Any given node(processor) is assigned with task of processing of a set of variable/check nodes. This is a departure from the general design strategy where there are distinct processors for variable node and check node processing. The assignment is heuristic keeping in mind uniform computational and communication load balancing among all the processors. The assignment algorithm we adopt is cyclic distribution of variable/check nodes on processors. This is given by the relation.

Processor i, for i = 0 to 12 gets the set of variable/check nodes j congruent i (mod 13) for j = 0 to 72. The assignment is shown in Table 6

The variable nodes associated with processor 0 are {0 13 26 39 52 65}. During variable node processing, 0^{th} processor updates variable node data(log-likelihood ratio) of the nodes {0 13 26 39 42 65}either in 6 sequential steps using single variable node processing unit or does it in parallel using 6 identical variable node processing units. This updation is based on the check node information obtained by the variable nodes from their respective check node neighbors in the previous iteration step.

After completion of the variable node processing the processor 0 constructs a personalized packet containing information related to each variable node owned by the processor. This information is available as a set of m+1 packets. The contents of each packet have the structure described in Figure 2. The consolidated personalized packet formed by node 0 is shown in Table 7. In Table 7 there are 6 columns each column having a set of 9 packets arranged in rows. The columns represent the packets belonging to variable nodes owned by the processor 0. The variable node is identified by the first entry in each packet. "x" represents the data to be communicated to the respective neighboring check node.

Table 6. Variable / Check node updation

Processors	Variable/check nodes
0	{ 0 13 26 39 52 65}
1	{ 1 14 27 40 53 66}
2	{ 2 15 28 41 54 67}
3	{ 3 16 29 42 55 68}
4	{ 4 17 30 43 56 69}
5	{ 5 18 31 44 57 70}
6	{ 6 19 32 45 58 71}
7	{ 7 20 33 46 59 72}
8	{ 8 21 34 47 60 x }
9	{ 9 22 35 48 61 x }
10	{10 23 36 49 62 x }
11	{11 24 37 50 63 x }
12	{12 25 38 51 64 x }

Variable Node	Owner Processor for the variable node	Neighboring Check Node	Destination Processor of Neighboring Check Node	Data

Fig. 2 Packet Structure

These packets are bundled together to form one consolidated personalized packet and sent to the set of processors which is the union of the 4^{th} column of packet information of all (m+1)*6 packets. Note: 4^{th} column of the packet information has destination processor information.

Though there are many ways of communicating this information, the present strategy is preferred to maintain balance in communication load

Table 7.Packets of variable nodes of processor 0

0 0 0 0 x	13 0 13 0 x	26 0 26 0 x	39 0 39 0 x	52 0 52 0 x	65 0 65 0 x
0 0 1 1 x	13 0 14 1 x	26 0 27 1 x	39 0 40 1 x	52 0 53 1 x	65 0 66 1 x
0 0 3 3 x	13 0 16 3 x	26 0 29 3 x	39 0 42 3 x	52 0 55 3 x	65 0 68 3 x
0 0 7 7 x	13 0 20 7 x	26 0 33 7 x	39 0 46 7 x	52 0 59 7 x	65 0 72 7 x
0 0 15 2 x	13 0 28 2 x	26 0 41 2 x	39 0 54 2 x	52 0 67 2 x	65 0 7 7 x
0 0 31 5 x	13 0 44 5 x	26 0 57 5 x	39 0 70 5 x	52 0 10 10 x	65 0 23 10 x
0 0 36 10 x	13 0 49 10 x	26 0 62 10 x	39 0 2 2 x	52 0 15 2 x	65 0 28 2 x
0 0 54 2 x	13 0 67 2 x	26 0 7 7 x	39 0 20 7 x	52 0 33 7 x	65 0 46 7 x
0 0 63 11 x	13 0 3 3 x	26 0 16 3 x	39 0 29 3 x	52 0 42 3 x	65 0 55 3 x

Now we describe the functioning of the decoder.

4.1.LDPC Decoder with single node/link failure

We make the following assumptions:

1. LDPC Decoder is a Co-processor attached to a Central Processing Unit (CPU) whose sole function is to decode LDPC data received from the CPU. The Co-processor has a set of arithmetic processors which can do variable/check node processing. The processors are connected by an interconnection network which is essentially a 0-free PDN. Number of processors is equal to m^2+m+1. $m = p^t$ where t is a +ve integer. Each processor has its own local memory to store check node updates, variable node updates and also to cater miscellaneous memory requirements during decoder execution.

2. When we say the decoder is fault tolerant, it means it can only tolerate single node/link failure

in side the Co-processor. It does not deal with the failure of the system (CPU) of which it is a part.

3. At a time the decoder can work in one of the 3-modes: Init mode, Communication mode or Computation mode. Computation mode involves either check node processing or variable node processing depending on the iteration step presently being executed.

4. All operations in different modes are synchronous and are controlled by the system clock (CPU). Though data flow architectures can also be considered we leave it for future research.

5. Each node (processor) is having single port facility. This means Communication mode operations take 2m+2 time steps. Each time step depends on the maximum size of the packets being transmitted between a node and its neighboring nodes.

In the Init mode the CPU examines whether there are any node/link failures in the decoder. If node/ link failures are greater than 1 the decoder is considered as faulty. Else CPU distributes the variable nodes and check nodes on the available number of processors using some heuristics say cyclic assignment and initiates decoding process.

In the Init mode the nodes of the decoder will be acquiring LDPC received data supplied by the CPU. In the computation mode, depending on iteration step, either variable node processing or check node processing is executed. This is done in parallel on all the available processors. At the end of Computation mode personalized packets are constructed by each processor for dispatching to the respective neighboring processors that own check/variable nodes.
In the Communication mode variable/check node information will be communicated to the respective processors via the 0-free communication network. If any failure of node/link is observed information is sent to CPU to reinitiate the decoding process which calls for fresh assignment of variable/check nodes on the available processors if possible.

5. Conclusion

The design of an LDPC decoder that has fault tolerance to single node/link failure has been demonstrated. This is based on a new approach called 0- free PDN which is a modified version of existing PDN based on 2-dimensional projective geometry. This concept of decoder design has applications in the areas of aero-space communications which are safety critical.

Future Scope

Currently, the single node/link failure has been made fault secure .Fault tolerance to multiple node/links failures is worth attempting.

References

[1] B. Parhami and M. Rakov, "Perfect Difference Networks and Related Interconnection Structures for Parallel and Distributed Systems," *IEEE Trans. Parallel and Distributed Systems*, Vol. 16, No. 8, pp. 714-724, August 2005.

[2] Singer, J.,"A Theorem in Finite Projective Geometry and Some Applications to Number Theory", *Trans. American Mathematical Society*, Vol.43, pp. 377-385, 1938.

[3] B. Parhami and M. Rakov, "Application of Perfect Difference Sets to the Design of Efficient and Robust Interconnection Networks," *Proc. Int'l Conf. Communications in Computing*, Las Vegas, NV, 27-30 June 2005, pp. 207-213.

[4] F.Guilloud, "Generic Architecture for LDPC Codes Decoding", *Ph.DThesis under SPRING project*, July 2004.

[5] Sarah J.Johnson, "Introducing Low Density Parity - Check Codes", *The University of Newcastle*, sigpromu.org /sarah / SJohns on LDPCintro.pdf

[6] Harihara S.G, M. Girish Chandra, Taraka praveen Uppalapati, B.S. Adiga, "Decoding Architectures for Projective Geometry Based LDPC Codes", *IFIP Wireless Days*, Dubai, Nov.2008.

B.Venkateshulu, received B.Tech and M.Tech degrees in1989 and 1994 from JNTU, SVU respectively. He is a faculty at GNITS, Hyderabad ,A.P, He has published paper in reputed international journal .Currently he is doing PhD. His interests include Wireless Communication, Channel coding techniques.

Dr.B.S.Adiga obtained his BE (Electrical Engg.) and M.Tech (Industrial Electronics) degrees from Karnataka Regional Engineering College, Surathkal, India. He obtained his PhD in Computer Science from the Indian Institute of Science, Bangalore, India. He worked as a scientist at National Aerospace Laboratories, Bangalore, India, for nearly 20 years. Later on, he was with Motorola India Electronics

Limited and Philips Innovation Labs, Bangalore, India. Presently, he is a Principal Scientist at the Innovation Labs, Tata Consultancy Services, Bangalore, India. His interests are in the broad areas of Signal Processing, Communications and Computing including, Error Control Coding, Compressive Sensing, High-Performance Computing and Multimedia Signal Processing.

Dr. B.C. Jinaga graduated in Engineering from Karnatak University, Dharwad and did his M.Tech from Regional Engineering College, Warangal and Ph.D. from Indian Institute of Technology, Delhi. For two years he was on the faculty of Karnatak Regional Engineering College, Surathkal and for a short period he worked with a research group at Indian Institute of Science, Bangalore. He was with Jawaharlal Nehru Technological University Hyderabad for over 34 years. Dr. Jinaga has several publications in International and National Journals and Conferences. His research interests are in the area of Digital Signal Processing, Image Processing, Neural Networks, Digital Circuits, Embedded Systems and Digital Communications. He was awarded Best Teacher award in 2002 from the Government of Andhra Pradesh, India.