Design of Parallel LDPC Interleaver Architecture: A Bipartite Edge Coloring Approach

Awais. H. SANI, P. Coussy, C Chavet, E. Martin
University Bretagne-Sud / Lab-STICC
awais-hussain.sani@univ-ubs.fr

December 14, 2010

Outline

• Problem Formulation
• Modeling
• Algorithm
• Conclusion

Problem Formulation

• Set of \( K \) data elements \( \{d_0, d_1, \ldots, d_{K-1}\} \)
• Set of \( P \) processing elements \( \{PE_0, PE_1, \ldots, PE_{P-1}\} \)
• Set of \( B = P \) memory banks \( \{b_0, b_1, \ldots, b_{P-1}\} \)
• Size of each memory bank \( M = K/P \)
• Set of \( N \) time instances \( \{t_0, t_1, \ldots, t_N\} \) in which \( P \) processing elements process \( K \) data elements.
Problem Formulation

- Set of $K$ data elements \{$d_0, d_1, \ldots, d_{K-1}$\}
- Set of $P$ processing elements \{$PE_0, PE_1, \ldots, PE_{P-1}$\}
- Set of $B = P$ memory banks \{$b_0, b_1, \ldots, b_{P-1}$\}
- Size of each memory bank $M = K/P$
- Set of $N$ time instances \{$t_1, t_2, \ldots, t_N$\}

Mapping problem: Store $K$ data elements in $B$ memory banks in such a manner that $P$ processing elements can access $B$ memory banks in parallel at each time instance for first reading and then writing $B$ data elements without any conflict.

Example

- Set of $K$ data elements \{$d_0, d_1, \ldots, d_{K-1}$\}
- Set of $P$ processing elements \{$PE_0, PE_1, \ldots, PE_{P-1}$\}
- Set of $B = P$ memory banks \{$b_0, b_1, \ldots, b_{P-1}$\}
- Size of each memory bank $M = K/P$
- Set of $N$ time instances \{$t_1, t_2, \ldots, t_N$\}

$K = 6$, $P = B = 3$, $M = 2$, $N = 6$

Conflict Access Graph

Traditional Approach

- Set of $K$ data elements \{$d_0, d_1, \ldots, d_{K-1}$\}
- Set of $P$ processing elements \{$PE_0, PE_1, \ldots, PE_{P-1}$\}
- Set of $B = P$ memory banks \{$b_0, b_1, \ldots, b_{P-1}$\}
- Size of each memory bank $M = K/P$
- Set of $N$ time instances \{$t_1, t_2, \ldots, t_N$\}

$K = 6$, $P = B = 3$, $M = 2$, $N = 6$
Traditional Approach

Nodes connected with the same edge should be of different color and each color represents memory bank.

We need 5 memory banks to access 6 data elements.
New Approach

To tackle this problem, we introduce the concept of multiple read and multiple write access Concept.

Data Access Matrix

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mapping Matrix

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints

- At each time instance, all the memory banks in the read column (resp. in the write column) of the mapping matrix must be different.
Constraints

Mapping Constraints:
• At each time instance, all the memory banks in the read column (resp. in the write column) of the mapping matrix must be different.
• The bank of the last write access to a data must be the same as the bank of its first read access.

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mapping Matrix

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints

Mapping Constraints:
• At each time instance, all the memory banks in the read column (resp. in the write column) of the mapping matrix must be different.
• The bank of the last write access to a data must be the same as the bank of its first read access.

Preparing Bipartite Graph

<table>
<thead>
<tr>
<th>Time Nodes</th>
<th>Data Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>● 1</td>
</tr>
<tr>
<td>t₂</td>
<td>● 2</td>
</tr>
<tr>
<td>t₃</td>
<td>● 3</td>
</tr>
<tr>
<td>t₄</td>
<td>● 4</td>
</tr>
<tr>
<td>t₅</td>
<td>● 5</td>
</tr>
<tr>
<td>t₆</td>
<td>● 6</td>
</tr>
</tbody>
</table>
Preparing Bipartite Graph

1st Read/Write Access

2nd Read/Write Access

Last Read/Write Access

Data Node Representation

Where

$e_{0l,R}$ and $e_{(n-1)l,R}$ represent first and last read access of the data $l$ respectively.

$e_{0l,W}$ and $e_{(n-1)l,W}$ represent first and last write access of the data $l$ respectively.

Placement Property:

\[ i_{th} \text{ write access} = \text{ modulo}_{\text{degree}}(i + 1) \text{th read access} \]

This placement property is used to search edges during algorithm.

The read access of the $i_{th}$ write access is called direct edge whereas the corresponding write access is called induced edge next in this presentation.

For (degree $l$) = 3

\[ i_{th} \text{ write access} = \text{ modulo}_3(i + 1) \text{th read access} \]

Induced edges

Direct edges

Dotted edges

Bold edges

Time Nodes

Data Nodes

Time Nodes

Data Nodes
Outline

- Problem Formulation
- Modeling
  - Preparing Bipartite Graph
- Algorithm
  - Partitioning
  - Coloring
- Conclusion

Partitioning Algorithm

Proper Partition:
Proper partition is the subgraph in which all the time vertices have degree 2.

Proper partition is the subgraph in which all the time vertices have degree 1.

Partitioning Constraint:
No more than 2 read or write accesses have to be done at each time instance in a proper partition.
Partitioning Algorithm

At each node: Traverse the path & remove the edges which do not follow the constraints.

Yes

Path is completed

No

Partition is completed

Yes

Remove the partition.

No

Graph is traversed

Yes

Algorithm is completed.

Partitioning Algorithm:

• Process of Traversal
  1. Randomly selects the edge available at the current data and time vertex.
  2. Records the induced edge of the selected edge.

• Process of Elimination
  1. Removes the edges from current partition, the selection of which makes the construction of proper partition following "partitioning constraint" impossible.
At each node: Traverse the path & remove the edges which do not follow the constraints.

**Path is completed.**

**Partition is completed.**

Yes

No

Path is not completed. Start adding other edges to the path.

**Partitioning Algorithm**

**Algorithm invokes the process of traversal to add the edge** \((t_1, t_2)\) **into the path** \(p_1\) **as shown with bold line. Induced edge** \((t_1, t_6)\) **of the** \((3, t_5)\) **is also recorded with dotted line.**

\[ p_1 = \{(1, t_1), (t_1,3)\} \]

**Algorithm invokes the process of traversal to add the edge** \((t_1, t_6)\) **into the path** \(p_1\) **as shown with bold line. Induced edge** \((1, t_6)\) **of the** \((t_3, t_4)\) **is also recorded with dotted line.**

\[ p_1 = \{(1, t_1), (t_1,3)\} \]
Partitioning Algorithm

At each node, traverse the path & remove the edges which do not follow the constraints.

Path is completed
Partition is completed

Yes
No

Algorithm is completed

Process of elimination is invoked and found that two read accesses are done at $t_1$, other read accesses at $t_i$ is deleted as shown with large and small dotted line.

$P_1 = \{(1,t_1), (t_1,3)\}$

Algorithm invokes the process of traversal to add the edge $(3, t_2)$ into the path $P_1$. Since $(3, t_2)$ is already a induced edge so this edge is now bold and dotted.

$P_1 = \{(1,t_1), (t_1,3), (3, t_2)\}$

Traversal continues until $P_1$ reaches at $t_4$.

$P_1 = \{(1,t_1), (t_1,3), (3, t_2), (t_5, 5), (5, t_4)\}$
At each node: Traverse the path & remove the edges which do not follow the constraints.

Path is completed
Partition is completed

Yes
Yes
No
No

Partitioning Algorithm
Lab-STICC

Time Nodes
Data Nodes

Process of elimination finds that 2 write accesses is done at \( t_2 \) as shown through induced edges. So other edges, \((t_3, 6)\) in this case, which have write accesses at \( t_2 \) is deleted.

Path \( p_1 \) is completed as shown with bold lines.

But partition \( sg_1 \) is not completed.

Algorithm traverses another path \( p_2 \) to complete \( sg_1 \).

\[ p_1 = \{(1, t_1), (t_1, 3), (3, t_3), (t_3, 5), (5, t_5), (t_5, 6)\} \]

\[ p_2 = \{(1, t_1), (t_1, 3), (3, t_3), (t_3, 4), (4, t_4), (t_4, 1)\} \]
At each node: Traverse the path & remove the edges which do not follow the constraints.

Path is completed
Partition is completed

Yes
Yes

No
No

Remove the partition.

Partitioning Algorithm

Lab-STICC

www.univ-ubs.fr

Remove $sg_1$ from graph $G$.

$G' = G - sg_1$

Graph is completely traversed.

Partioning algorithm is completed.

$p'_1 = \{(2, t_1)\}$,
$p'_2 = \{(2, t_4)\}$,
$p'_3 = \{(2, t_5)\}$,
$p'_4 = \{(4, t_6)\}$,
$p'_5 = \{(5, t_2)\}$,
$p'_6 = \{(6, t_3)\}$

sg

1

= $p_1 + p_2$

sg

2

= $p_3 + p_4 + p_5 + p_6$

At each node: Traverse the path & remove the edges which do not follow the constraints.

Path is completed
Partition is completed

Yes
Yes

No
No

Remove the partition.

Partitioning Algorithm

Lab-STICC

www.univ-ubs.fr

Remove $sg_2$ from graph $G'$.

Graph is completely traversed.

Partioning algorithm is completed.

$p'_1 = \{(2, t_1)\}$,
$p'_2 = \{(2, t_4)\}$,
$p'_3 = \{(2, t_5)\}$,
$p'_4 = \{(4, t_6)\}$,
$p'_5 = \{(5, t_2)\}$,
$p'_6 = \{(6, t_3)\}$

sg

1

= $p_1 + p_2$

sg

2

= $p_3 + p_4 + p_5 + p_6$
Problem Formulation

Modeling
  - Preparing Bipartite Graph

Algorithm
  - Partitioning
  - Coloring

Conclusion
Each partition is colored with 2 colors. Alternately remove the read and write access conflict for each edge.

Partition is completed

Graph is traversed

Yes

No

Search for the time node in which induced edge of the current colored edge exits. \( t_6 \) is the required node.

\[ e_{0l,W} = e_{1l,R} \]
\[ e_{1l,W} = e_{2l,R} \]
\[ e_{2l,W} = e_{0l,R} \]

Remove the read access conflict by giving different color to the edge at \( t_2 \) and give it 2nd color. In that way, we remove the write access conflict.

\[ e_{0l,W} = e_{1l,R} \]
\[ e_{1l,W} = e_{2l,R} \]
\[ e_{2l,W} = e_{0l,R} \]
Each partition is colored with 2 colors. Alternately remove the read and write access conflict for each edge.

Partition is completed.

Graph is traversed

Yes

No

No

Yes

Algorithm is completed.

Coloring Algorithm

Graph is not completely colored.

Give one color to partition $s_2$.
Each partition is colored with 2 colors. Alternately remove the read and write access conflict for each edge.

Graph is completely colored. Coloring algorithm is completed.

Outline

- Problem Formulation
- Modeling
  - Preparing Bipartite Graph
- Algorithm
  - Partitioning
  - Coloring
- Conclusion

Conclusion

- Concept of multiple read/write approach is presented.
- Modified bipartite edge coloring approach is introduced to find conflict free memory mapping for any type of parallel iterative decoding for LDPC.
Future Perspectives

- Additional constraints will be added in order to find mapping following the targeted interconnection network.
- Complexity of the interconnection network is considered in the future development of the mapping algorithm.