**Problem Formulation**

An interleaver is a component that shuffles data from a given input data order (natural order: e.g. Nat = 0, 1, 2, 3, 4, …) to a different output order (interleaved order: e.g. Int = 1, 5, 2, 6, 9, …)

The problem is to find an in-place memory mapping avoiding memory access conflicts in both natural and interleaved order.

**Proposed Design Flow**

In the mapping matrix each element is filled with a memory bank (i.e. the corresponding data will be stored in this memory bank)

This memory mapping is done according to:

- **Structural constraints**
  - A data must be mapped in the same memory bank in natural and interleaved order (i.e. in matrix MAPNat and matrix MAPInt)
  - In any column of MAPNat and MAPInt each memory has to be used only one time

- **Architectural objectives**
  - The memory mapping in any column has to respect the rules of the steering components that compose the network

**Related Works**

<table>
<thead>
<tr>
<th>Standard Interleave</th>
<th>Architectural complexity</th>
<th>Algorithm complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>YES</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>YES</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>YES</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>YES</td>
<td>Low / Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

The data exchanges in natural and interleaved order are formalized through two distinct matrices

**Natural access matrix**

<table>
<thead>
<tr>
<th>PE1</th>
<th>PE2</th>
<th>Mem1</th>
<th>Mem2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

**Interleaved access matrix**

<table>
<thead>
<tr>
<th>PE1</th>
<th>PE2</th>
<th>Mem1</th>
<th>Mem2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>

**Algorithm Iterations**

1. **Step I**
   - Select the most constrained column c in the matrix
   - MapNat

2. **Step II**
   - Generation of the valid mapping list l for c
   - MapNat

3. **Step III**
   - Validate the map matrix
   - MapNat

4. **Step IV**
   - Mapping of column c
   - MapNat

5. **Step V**
   - Update the memory mapping in the other matrix
   - MapNat

6. **Step VI**
   - Update the memory mapping in the other matrix
   - MapNat

7. **Step VII**
   - Update the memory mapping in the other matrix
   - MapNat

**Conclusion**

SAGE is a memory mapping methodology dedicated to design parallel interleaver architecture

This approach allows to generate a valid memory mapping in any case, and if the interleaving law enables it, then the resulting memory mapping will respect a targeted interconnection network

Patent in France n°0754793 and patent pending in USA n°20090031094

C. Chavet, P. Coussy, P. Urard, E. Martin, “Apparatus for data interleaving algorithm”