Floorplanning and Topology Generation for Application-Specific Network-on-Chip

Bei Yu¹  Sheqin Dong¹  Song Chen²  Satoshi GOTO²

¹Department of Computer Science & Technology
Tsinghua University, Beijing, China

²Graduate School of IPS
Waseda University, Kitakyushu, Japan

2010.01.20
Outline

1. Introduction
   - Previous Works
   - Problem Formulation

2. Topology Synthesis Algorithm
   - Partition Driven Floorplanning
   - Switches and Network Interfaces Insertion
   - Energy Aware Path Allocation

3. Experimental Results
Network-on-Chip

- Solution to global communication challenges
- Alternative to Bus communication architectures
  - Better modularity
  - Lower power consumption
  - Scalability
- Regular NoCs and Application-Specific NoCs
- Network components:
  - Switch
  - Network Interface (NI)
Regular or Application-Specific Topology

- Regular Topology
  - Task Scheduling and Mapping problem
- Application-Specific Topology?
  1. Irregular core sizes
  2. Different communication flow requirements
  3. Reducing energy by reducing hop count and switch count
  4. Possibly higher performance
Regular or Application-Specific Topology

- **Regular Topology**
  - Task Scheduling and Mapping problem

- **Application-Specific Topology?**
  1. Irregular core sizes
  2. Different communication flow requirements
  3. Reducing energy by reducing hop count and switch count
  4. Possibly higher performance

*Focus on Application-Specific Topology Generation!*
Previous Works

– K. Srinivasan et al. TVLSI 06:
  - Used fixed floorplan as optimization starting point
  - Switch at corners of cores

– Murali et al. ICCAD06:
  - Two steps topology generation procedure using min-cut partitioner
  - Greedy based path allocation assignment

– Chan & Parameswaran, ASPDAC08:
  - Iterative refinement strategy
  - Supports both packet-switched networks and point to point connections

– Murali et al. ASPDAC09:
  - Synthesis approach for 3D NoC
  - LP based switch position computation
Motivations

In previous works:
- Partition w/o physical information
- Fail to consider area consumption of NI and Switch

In our works:
- Integrate partition into floorplanning phase
- Consider Switches and NI area consumption
- Min-Cost-Flow algorithm to insert NI
- Effective paths allocation to minimize power consumption
**Problem Formulation**

**Input:**
- a set of $n$ cores $C = \{c_1, c_2, \ldots, c_n\}$.
- switches number $m$.
- core communication graph (CCG).
- network components power model.

**Output:** an NoC topology satisfying
- minimize area consumption.
- minimize the communication energy.

CCG: Core Communication Graph.
Synthesis Algorithm

- Obtain min-cut partitions of CCG
  - Communication Requirement
  - Distances between cores
- Cores in a cluster share a switch
- Switch Communication Graph (SCG)
- Path Allocation on SCG
  - Minimize power consumption
  - Minimize hop-count
  - Satisfy width constraints
Overview of Algorithm

Floorplanning
- Core Size
- CCG
  - Generate new floorplan
  - Partition
    - Stop?
      - No
      - Yes
        - Generate floorplan with partitions.
Post-Floorplanning
- Switches Insertion
- Network Interfaces Insertion
  - Path Allocation
  - Optimized Floorplan
Overview of Algorithm

Floorplanning

- Core Size
- CCG
- Generate new floorplan
- Partition
- Stop?
- No
- Yes

Post-Floorplanning

- Switches Insertion
- Network Interfaces Insertion
- Path Allocation

Optimized Floorplan

Insert Switches.

Bei Yu
Floorplanning & Topology Generation for NoCs
Overview of Algorithm

Floorplanning
- Core Size
- CCG
  - Generate new floorplan
  - Partition
  - Stop?
    - No
    - Yes

Post-Floorplanning
- Switches Insertion
- Network Interfaces Insertion
- Path Allocation

Optimized Floorplan

Insert NI with Min-Cost Flow Algorithm

Bei Yu
Floorplanning & Topology Generation for NoCs
Overview of Algorithm

Floorplanning
- Core Size
- CCG
  - Generate new floorplan
  - Partition
    - Stop? Yes

Post-Floorplanning
- Switches Insertion
- Network Interfaces Insertion
  - Path Allocation

Dynamic Programming based Path Allocation.
Partition Driven Floorplanning

- Traditionally, partition before floorplanning
  - Lose physical information
- In our work
  - Integrate partition into floorplanning
  - Cores with larger communication incline to one cluster
  - Minimize interconnect power consumption
- Define new edge weight $w'_{ij}$ in CCG:

\[
 w'_{ij} = \alpha_w \times \frac{w_{ij}}{\max_w} + \alpha_d \times \frac{\text{mean}\_\text{dis}}{\text{dis}_{ij}}
\]

- Using CBL\(^1\) as topological representation
  - Record white space information

After floorplanning stage

- Each cluster has a minimal bounding box.
- Heuristical method to insert switches:
  1. Switch initially in the center of bounding box.
  2. Partition the white space into grids.
  3. Sort switches.
  4. Insert switches in grids one by one.

- In cluster $p_k$, cost of insert switch $k$ to grid $g$:

\[
Cost_{gk} = \sum_{i,j} w_{ij} \times (\text{dis}_{gi} + \text{dis}_{gj}), \forall e_{ij} \in \bar{E}
\]

Choose free grid with smallest $Cost$. 
Switches Insertion

After floorplanning stage

- Each cluster has a minimal bounding box.
- Heuristical method to insert switches:
  1. Switch initially in the center of bounding box.
  2. Partition the white space into grids.
  3. Sort switches.
  4. Insert switches in grids one by one.

In cluster $p_k$, cost of insert switch $k$ to grid $g$:

$$\text{Cost}_{gk} = \sum_{i,j} w_{ij} \times (\text{dis}_{gi} + \text{dis}_{gj}), \forall e_{ij} \in \bar{E}$$

Choose free grid with smallest $\text{Cost}$. 
Switches Insertion

After floorplanning stage

- Each cluster has a minimal bounding box.
- Heuristical method to insert switches:
  1. Switch initially in the center of bounding box.
  2. Partition the white space into grids.
  3. Sort switches.
  4. Insert switches in grids one by one.

In cluster $p_k$, cost of insert switch $k$ to grid $g$:

$$\text{Cost}_{gk} = \sum_{i,j} w_{ij} \times (\text{dis}_{gi} + \text{dis}_{gj}), \forall e_{ij} \in \bar{E}$$

Choose free grid with smallest Cost.
Switches Insertion

After floorplanning stage

- Each cluster has a minimal bounding box.
- Heuristical method to insert switches:
  1. Switch initially in the center of bounding box.
  2. Partition the white space into grids.
  3. Sort switches.
  4. Insert switches in grids one by one.

In cluster $p_k$, cost of insert switch $k$ to grid $g$:

$$\text{Cost}_{gk} = \sum_{i,j} w_{ij} \times (\text{dis}_{gi} + \text{dis}_{gj}), \forall e_{ij} \in \bar{E}$$

Choose free grid with smallest Cost.
Switches Insertion

After floorplanning stage

- Each cluster has a minimal bounding box.
- Heuristical method to insert switches:
  1. Switch initially in the center of bounding box.
  2. Partition the white space into grids.
  3. Sort switches.
  4. Insert switches in grids one by one.
- In cluster $p_k$, cost of insert switch $k$ to grid $g$:
  \[
  \text{Cost}_{gk} = \sum_{i,j} w_{ij} \times (\text{dis}_{gi} + \text{dis}_{gj}), \forall e_{ij} \in \bar{E}
  \]
- Choose free grid with smallest $\text{Cost}$.
For each core, construct l-bounding box

Insert NI in l-bounding box

Construct network graph $G^* = (V^*, E^*)$:

**Network Graph**

- $V^* = \{s, t\} \cup NI \cup Grids$.
- $E^* = \{(s, ni_k) \mid ni_k \in NI\} \cup \{(ni_k, g_j) \mid \forall g_j \in CG_k\} \cup \{(g_j, t) \mid g_j \in Grids\}$.
- Capacities:
  - $C(s, ni_k) = 1$, $C(ni_k, g_j) = 1$, $C(r_j, t) = 1$.
- Cost:
  - $F(s, ni_k) = 0$, $F(g_j, t) = 0$; $F(ni_k, g_j) = F_{kj}$.

Min-cost flow algorithm, polynomial time.
Network Interfaces Insertion

- For each core, construct $l$-bounding box
- Insert NI in $l$-bounding box
- Construct network graph $G^* = (V^*, E^*)$:

**Network Graph**

- $V^* = \{s, t\} \cup NI \cup Grids$.
- $E^* = \{(s, ni_k)|ni_k \in NI\} \cup \{(ni_k, g_j)|\forall g_j \in CG_k\} \cup \{(g_j, t)|g_j \in Grids\}$.
- Capacities:
  $C(s, ni_k) = 1, C(ni_k, g_j) = 1, C(r_j, t) = 1$.
- Cost:
  $F(s, ni_k) = 0, F(g_j, t) = 0; F(ni_k, g_j) = F_{kj}$.
- Min-cost flow algorithm, polynomial time.

Bei Yu  
Floorplanning & Topology Generation for NoCs
Energy Aware Path Allocation

Solve once is enough? **No!**

- Consider Power Consumption.
- Path with minimal power consumption may change.

Simple example:
- Two flows: \((s1 \rightarrow s3), (s2 \rightarrow s3)\).
- Solve \((s1 \rightarrow s3)\) first.
- First,
  - shortest path from \(s2\) to \(s3\) is \(s1 \rightarrow s3\).
- After flow \((s1 \rightarrow s3)\):
  - shortest path from \(s2\) to \(s3\) is \(s1 \rightarrow s4 \rightarrow s3\).
Energy Aware Path Allocation

Solve once is enough?  **No!**

- Consider Power Consumption.
- Path with minimal power consumption may change.

Simple example:

- Two flows: \((s_1 \rightarrow s_3), (s_2 \rightarrow s_3)\).
- Solve \((s_1 \rightarrow s_3)\) first.
- First,
  - shortest path from \(s_2\) to \(s_3\) is \(s_1 \rightarrow s_3\).
- After flow \((s_1 \rightarrow s_3)\):
  - shortest path from \(s_2\) to \(s_3\) is \(s_1 \rightarrow s_4 \rightarrow s_3\).
Energy Aware Path Allocation

Solve once is enough? **No!**

- Consider Power Consumption.
- Path with minimal power consumption may change.

Simple example:
- Two flows: \((s_1 \rightarrow s_3), (s_2 \rightarrow s_3)\).
- Solve \((s_1 \rightarrow s_3)\) first.
- First,
  - shortest path from \(s_2\) to \(s_3\) is \(s_1 \rightarrow s_3\).
- After flow \((s_1 \rightarrow s_3)\):
  - shortest path from \(s_2\) to \(s_3\) is \(s_1 \rightarrow s_4 \rightarrow s_3\).
Energy Aware Path Allocation

- \( dis_n(i, d) \): distance from node \( i \) to \( d \)
- \( dis_e(i, j, d) \): distance \( i \) to \( d \) using \( e_{ij} \)

DP based method to find paths:

\[
\begin{align*}
    dis_e(i, j, d) &= \begin{cases} 
        t_{id}, & j = d \\
        t_{ij} + dis_n(j, d), & \text{otherwise}
    \end{cases} \\
    dis_n(i, d) &= \begin{cases} 
        0, & i = d \\
        \min_k dis_e(i, k, d), & \text{otherwise}
    \end{cases}
\end{align*}
\]

- Run time is bounded by \( O(|V| \cdot |E|) \)
- If \( dis_e(i, j, d) = dis_n(i, d) \), then \( \text{path}(i, d) = j \).
Update Paths

1: //Update when $t_{ij}$ change to $(t_{ij} + \Delta t)$;
2: $t_{ij} \leftarrow (t_{ij} + \Delta t)$;
3: queue q.push($e_{ij}$);
4: while q is not empty do
5:   $e_{ab} \leftarrow q$.pop();
6:   $\text{dis}_e(a, b, d) \leftarrow t_{ab} + \text{dis}_n(b, d)$;
7:   if $\text{PATH}[a][d] = b$ then
8:     Find $k \in \text{Post}(a)^a$ to minimize $\text{dis}_n(k, d) + t_{ak}$;
9:     $\text{dis}_n(a, d) \leftarrow \text{dis}_n(k, d) + t_{ak}$;
10:    $\text{path}(a, d) \leftarrow k$;
11:   q.push($e_{pa}$), $\forall p \in \text{Pre}(a)^b$;
12: end if
13: end while

\[
^a\text{Post}(a) = \{v_k | \forall v_k \in V \land e_{ak} \in E\} \\
^b\text{Pre}(a) = \{v_k | \forall v_k \in V \land e_{ka} \in E\}
\]
## Experimental Setup

### Power Model:
- **Switch power model**
  - | ports  | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
    |       | (pJ/bit) | 0.22 | 0.33 | 0.44 | 0.55 | 0.66 | 0.78 | 0.90 |

- **Interconnect power model**
  - | length(mm) | 1   | 4   | 8   | 12  | 16  |
    |       | (pJ/bit) | 0.6  | 2.4  | 4.8  | 7.2  | 9.6  |

### Benchmark:
- Bertozzi et al. (G1, G2, G3)
- Srinivasan et al. TVLSI06 (G4, G5, G6)
- Murali et al. ASPDAC09 (G7)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>V#</th>
<th>E#</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1 MPEG4</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>G2 MWD</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>G3 VOPD</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>G4 263decmp3dec</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>G5 263encmp3dec</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>G6 mp3encmp3dec</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>G7 D_38_tvopd</td>
<td>38</td>
<td>47</td>
</tr>
</tbody>
</table>
### Experimental Results

**The Consumption Between the PBF and the PDF:**

<table>
<thead>
<tr>
<th>Part#</th>
<th>Power(mW)</th>
<th>Hops</th>
<th>W.S(%)</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PBF</td>
<td>ours</td>
<td>PBF</td>
<td>ours</td>
</tr>
<tr>
<td>G1</td>
<td>3</td>
<td>25.9</td>
<td>16.0</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>24.3</td>
<td>14.1</td>
<td>1.25</td>
</tr>
<tr>
<td>G2</td>
<td>3</td>
<td>3.05</td>
<td>3.08</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.19</td>
<td>3.02</td>
<td>1.25</td>
</tr>
<tr>
<td>G3</td>
<td>3</td>
<td>7.43</td>
<td>6.12</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7.62</td>
<td>6.59</td>
<td>1.0</td>
</tr>
<tr>
<td>G4</td>
<td>3</td>
<td>4.96</td>
<td>3.92</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7.86</td>
<td>4.35</td>
<td>1.25</td>
</tr>
<tr>
<td>G5</td>
<td>3</td>
<td>24.7</td>
<td>19.2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>58.6</td>
<td>19.2</td>
<td>1.0</td>
</tr>
<tr>
<td>G6</td>
<td>3</td>
<td>8.4</td>
<td>4.4</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>11.2</td>
<td>8.6</td>
<td>1.0</td>
</tr>
<tr>
<td>G7</td>
<td>3</td>
<td>12.7</td>
<td>8.2</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12.3</td>
<td>6.8</td>
<td>1.44</td>
</tr>
<tr>
<td>Avg</td>
<td>-</td>
<td>15.16</td>
<td>8.83</td>
<td>1.14</td>
</tr>
<tr>
<td>Diff</td>
<td>-</td>
<td>-</td>
<td>-41.8%</td>
<td>-</td>
</tr>
</tbody>
</table>

- PBF: similar to Murali ICCAD06, **Partition Before Floorplanning**.
- PDF: our methods, **Partition Driven Floorplanning**.
- Can save 41.8% of power and 2.6% of hops number.
Experimental Results

The Consumption Between the PBF and the PDF:

<table>
<thead>
<tr>
<th>Part#</th>
<th>Power(mW)</th>
<th>Hops</th>
<th>W.S(%)</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PBF</td>
<td>ours</td>
<td>PBF</td>
<td>ours</td>
</tr>
<tr>
<td>G1</td>
<td>3</td>
<td>25.9</td>
<td>16.0</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>24.3</td>
<td>14.1</td>
<td>1.25</td>
</tr>
<tr>
<td>G2</td>
<td>3</td>
<td>3.05</td>
<td>3.08</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.19</td>
<td>3.02</td>
<td>1.25</td>
</tr>
<tr>
<td>G3</td>
<td>3</td>
<td>7.43</td>
<td>6.12</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7.62</td>
<td>6.59</td>
<td>1.0</td>
</tr>
<tr>
<td>G4</td>
<td>3</td>
<td>4.96</td>
<td>3.92</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7.86</td>
<td>4.35</td>
<td>1.25</td>
</tr>
<tr>
<td>G5</td>
<td>3</td>
<td>24.7</td>
<td>19.2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>58.6</td>
<td>19.2</td>
<td>1.0</td>
</tr>
<tr>
<td>G6</td>
<td>3</td>
<td>8.4</td>
<td>4.4</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>11.2</td>
<td>8.6</td>
<td>1.0</td>
</tr>
<tr>
<td>G7</td>
<td>3</td>
<td>12.7</td>
<td>8.2</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12.3</td>
<td>6.8</td>
<td>1.44</td>
</tr>
<tr>
<td>Avg</td>
<td>-</td>
<td>15.16</td>
<td>8.83</td>
<td>1.14</td>
</tr>
<tr>
<td>Diff</td>
<td>-</td>
<td>-41.8%</td>
<td>-</td>
<td>-2.6%</td>
</tr>
</tbody>
</table>

- PBF: similar to Murali ICCAD06, **Partition Before Floorplanning**.
- PDF: our methods, **Partition Driven Floorplanning**.
- Can save 41.8% of power and 2.6% of hops number.
Experimental Results (cont.)

263encmp3dec (4 clusters):

mp3encmp3dec (3 clusters):
Experimental Results (cont.)

263encmp3dec (4 clusters):

mp3encmp3dec (3 clusters):

highly communicating cores places close to each other.
**Experimental Results (cont.)**

- **Effectiveness of Path Update Algorithm:**

<table>
<thead>
<tr>
<th></th>
<th>V#</th>
<th>Flow#</th>
<th>Update#</th>
<th>Run Time(s)</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSP</td>
<td>ours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_01</td>
<td>20</td>
<td>34</td>
<td>20</td>
<td>0.024</td>
<td>0.008</td>
</tr>
<tr>
<td>t_02</td>
<td>100</td>
<td>130</td>
<td>30</td>
<td>0.604</td>
<td>0.016</td>
</tr>
<tr>
<td>t_03</td>
<td>300</td>
<td>457</td>
<td>50</td>
<td>20.35</td>
<td>0.08</td>
</tr>
</tbody>
</table>

- **DSP:** re-solves all distances by **Dijkstra’s Shortest Path Algorithm.**
- **Ours:** effective path update algorithm.
- **Larger graph, more effective.**
Conclusion

In our works:

- Integrate partition into floorplanning phase
- Consider Switches and NI area consumption
- Min-Cost-Flow algorithm to insert NI
- Effective paths allocation to minimize power consumption
Thank You!