SYNTHESIS OF NOC INTERCONNECTS FOR CUSTOM MPSoC ARCHITECTURES

GUL N. KHAN AND ANITA TINO
Department of Electrical and Computer Engineering
Ryerson University
Toronto, ON Canada

NOCS 2012
OVERVIEW

• Introduction to NoCs – Power and Performance
• NoC Topology Design Flow
• Concept Overviews
  • TABU Search
  • Layered Queuing Networks (LQN)
• Topology Generation Technique
  • Initial Solution
  • Successive Filter Strategy, neighborhood selection
  • Contention Analysis
• Experimental Results
• Final Remarks
NOC INTRODUCTION

- Network-on-Chips (NoC)
  - Application Specific
- Power
  - Automated Technique
  - Models: Static, Dynamic, Leakage
- Performance
  - Contention Modeling
  - Deadlock Avoidance
- Trade-offs
Input directed graph $G(V,E)$

- Each vertex $v_i \in V$ represents a core within the graph.
- The communication between vertex $i$ and $j$ represent a directed edge $(v_i, v_j)$, expressed as $e_{i,j} \in E$.
- The weight found on an edge $e_{i,j}$ denoted by $b(e_{i,j})$ characterizes the bandwidth.
- A destination vertex (core) $d_x$, where $d_x \in V$ may have 1 to many sources cores $s_x$.
- The source vertex $s_x \in V$, and $x = \{1...N\}$.
- $N$ represents the number of cores in the core graph.
NOC DESIGN FLOW

Floor Planner

Core Graph G(V,E)

Power Models

Design Constraints

Topology Generator

LQN Model

LQNS

Performance Constraints

Layout

NOC

Topology
NOC DESIGN FLOW

Output: Topology & Floorplan
Meta-heuristic optimization method designed to escape the trap of local optiums.
Start with initial feasible solution
Iterates through a neighborhood of possible solutions until optimal solution is found.
Tabu List $TL(s)$ – memory list of non-optimal/last optimal solution.
Aspiration list $AL(s)$ – list which will allow a Tabu move in list if results of a solution are better as compared to the current solution.
Two types of memory used to provide quality solution and multiple objectives:

- **Explicit Memory** – Directs search towards influential/quality-based solution
  - \( TL(s) \) and \( AL(s) \)
  - *Candidate List CL(s)* – generates list of possible moves
- **Attributive Memory** – Long term memory (a.k.a FR-Memory)
  - Diversification
  - Recency of vertices, frequency of moves within each neighbourhood area
- **Candidate List Strategies** – Strategies which narrow the examination of solutions in the \( CL(s) \)
Simulated Annealing (SA): Min-cut Partitioning
  • Limited to cost function
  • Single objective
  • Problems determining global optimaums

Genetic Algorithms (GA): Chromosome Strings Generation
  • Fitness function
  • Random generations

Both GA and SA techniques invoke Pareto curve technique
i.e. almost SINGLE OBJECTIVE MAPPING

ILP: Single-objective Limitations
  Experiments show it takes lot of time to converge

ANOC: Recursive Slicing ree
  • Heuristic with low complexity
  • Limited to small design spaces (not suitable for MPSoCs)
TS & Topology Generation

• Supports **multiple** objectives by memory functions
  Not limited to cost function
• Base solutions are **priori information**
• Able to keep track of **optimal/non-optimal** solutions
  ▪ Less computation to find **global optimum solutions**
  ▪ Shorter runtimes
• Candidate list strategies – narrow down the solution space
  ▪ Search for solutions that fit various constraints

**Limitation**
Solved during topology synthesis
**TS TOPOLOGY GENERATION**

**N(s, f, P)** - current feasible NoC Topology solution $s$ consuming power $P$ at a frequency $f$

**N(s)** - new possible solution $s'$ within the neighbourhood set.

**TL(s)** – Tabu List - contains non-optimal solutions

**AL(s)** – Aspiration List - responsible for consulting $TL(s)$ to ensure that $s'$ is optimal and more desirable than the previous encountered solutions.

---

**Algorithm 2** Tabu Search Topology Generation Algorithm

1: Generate initial topology solution $N(s,f,P)$
2: Evaluate $N(s,f,P)$ for initial frequency $f$ and power $P$
3: $TL(s) = \emptyset$ //ensure empty Tabu List
4: WHILE performance and power constraints NOT met DO
5:     Identify $s' = N(s)$
6:     Move to the temporary solution $s'$
7:     Evaluate $s'$ solution for $f'$ and $P'$ using Orion models
8:     $ConstraintCheck(s', AL(s), TL(s))$
9:     IF solution meets $AL(s)$, $TL(s)$, $f'$, and $P'$ constraints
10:        Create LQN models for sub-networks
11:        Invoke LQNS tool for performance analysis
12:        Place solution as last optimal $TL(s)$ entry
13:        Update current solution, $N(s,f,P) = s'$
14:        Check for deadlock, contention, utilization
15:        $VCInsertion(); (T_{arb})$
16:        Determine $f'$ based on updated $T_{arb}$
17:        IF $f' \leq f_{max}$ THEN
18:            Run through system-level floorplanner
19:        ELSE
20:            Go to line 22
21:        END
22:        IF power/ perf constraints & router ports satisfied
23:            $N(s,f,P)$ and EXIT
24:        END
25:    ELSE
26:        Place as a non-optimal $TL(s)$ entry
27:    END
28: END
Initial NoC Topology

Initial topological solution
- Crossbar approach
- Poor solution
  - Power
  - Performance
NoC Solution Evaluation

Core Graph

TSG

LRG

VC Insertion
NoC Solution Evaluation

\[
T_{\text{lat}} = T_{pk} + \sum_{i=0}^{y} T_{arb} + \sum_{i=0}^{y} T_{\text{blk}} + T_{dpk}
\]

\[
\Gamma_{r,p} = \sum_{s_x} \sum_{d_x} \Lambda(s_x, d_x, r, p)
\]

\[
T_{\text{arb},r} = \sum_{r=0}^{v_R} \left[ \sum_{p=0}^{v_{p,r}} \frac{\Gamma_{r,p} \left( \theta(\Lambda(s_x, d_x, r, p), i, j) \right)}{\sum_{p=0}^{v_{p,r}} \Gamma_{r,p}} \right] T_{\text{arb},\text{init}}
\]

\[
\theta(\Lambda(s_x, d_x, r, p), i, j) = \begin{cases} 
N_{i\rightarrow j} & \text{if } \exists \Lambda \text{ from port } i \text{ to port } j \\
0 & \text{otherwise}
\end{cases}
\]

**Deadlock**

\[
E = \{ (l_i, l_j) \mid CNX(l_i, V) = l_j, \; v \in V \}
\]

\[
\sigma_{l_i} = \sum_{k=0}^{E_{l_i}} e_k \\
\text{Max}^o = \max(\sigma_{l_i}) \; \forall \; i \in L
\]

**Contention**

\[
\mu_{r, p_z} = \max(\theta(\Lambda(s_x, d_x, r, p), i, j))
\]

\[
\delta = \{ \text{Max}^o \cup \mu_{r, p_z} \cup l_i \}
\]

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDG</td>
<td>Topology Dot Graph is a directed graph ( G ) where ( TDG = G(V, L) ). ( V ) is a set of vertices which represent the resources (cores and routers) and ( L ) is a set of edges denoting the links within the topology.</td>
</tr>
<tr>
<td>CNX</td>
<td>Connection function, which connects the link ( l_i ) to the next communicating link ( l_j ), in order for ( s_k ) to reach ( d_v ).</td>
</tr>
<tr>
<td>LRG</td>
<td>Link Reliance Graph is a directed graph ( G ) where ( LRG = G(V, E) ). ( E ) is now a set of edges signifying links, and ( E ) is a set of edges denoting the pairs of links connected through ( \text{CNX} ).</td>
</tr>
<tr>
<td>( s_k )</td>
<td>A source core, or source vertex in the ( TDG ).</td>
</tr>
<tr>
<td>( d_v )</td>
<td>A destination core, which source ( s_k ) in the ( TDG ).</td>
</tr>
<tr>
<td>( T_{arb} )</td>
<td>Arbitration delay.</td>
</tr>
<tr>
<td>( T_{lat} )</td>
<td>Overall latency delay.</td>
</tr>
<tr>
<td>( T_{\text{pack}} )</td>
<td>Packing or de-packing delay of the NIs.</td>
</tr>
<tr>
<td>( T_{\text{trans}} )</td>
<td>Number of transactions expected in or out port ( p ) of router ( r ).</td>
</tr>
<tr>
<td>( \theta(\Lambda(s_x, d_x, r, p), i, j) )</td>
<td>Overall traffic flow from port ( i ) to port ( j ).</td>
</tr>
<tr>
<td>( \phi )</td>
<td>Deadlock communication link.</td>
</tr>
<tr>
<td>( \sigma_{l_i} )</td>
<td>Degree of a vertex link ( l_i ).</td>
</tr>
<tr>
<td>( \text{Max}^a )</td>
<td>Heavily utilized links of ( LRG ).</td>
</tr>
<tr>
<td>( \text{Trans}_{r, p_z} )</td>
<td>Amount of transactions incurred by router ( r ), port ( p_z ).</td>
</tr>
<tr>
<td>( \mu_{r, p_z} )</td>
<td>Highly utilized ports of ( TDG ).</td>
</tr>
<tr>
<td>( \delta )</td>
<td>An expected contention point within the topology.</td>
</tr>
<tr>
<td>( \Omega[] )</td>
<td>1D array which holds all contention and deadlock points to be modeled with LQNs.</td>
</tr>
</tbody>
</table>
Successive Filter Strategy (SFS):

- $N$ is the total number of vertices/cores in the core graph, where $n = \{1, 2, \ldots, N\}$
- $\pi \in \Pi$, where $\Pi$ is a set of positions in the search space.
- $\pi(j)$ represent core $j$ attaining the head candidate position.
- $\Omega(s)$ denote the set of possible moves that core $j$ can have, when core $j$ has occupied the position $\pi(j)$.
- $m$ signify all possible combination of moves formed by the SFS.
- $\Omega(s)$ be divided into subsets $\Omega(1, s), \Omega(2, s), \ldots, \Omega(m, s)$, where $\Omega(1, s)$ denotes the 1st subset move in the possible set of total moves generated by the topology generator etc.

Given a vertex/core $n$:

- $N$ is the total number of vertices/cores in the core graph, where $n = \{1, 2, \ldots, N\}$
- $N_{tr}$ is the number of source, $s_{xt}$ and/or destination, $d_{xt}$, transactions that the vertex $V_n$ is expected to incur.
- $X$ is the total amount of sources or destinations for the respective core $n$, where $x = \{1, 2, \ldots, X\}$.
- $V_n(f)$ represents vertex $n$ and its expected total number of transactions $f$.
Successive Filter Strategy (SFS):

- \( N \) is the total number of vertices/cores in the core graph, where \( n = \{1, 2, \ldots, N\} \)
- \( \pi \in \Pi \), where \( \Pi \) is a set of positions in the search space.
- \( \pi(j) \) represent core \( j \) attaining the head candidate position.
- \( \Omega(s) \) denote the set of possible moves that core \( j \) can have, when core \( j \) has occupied the position \( \pi(j) \).
- \( m \) signify all possible combination of moves formed by the SFS.
- \( \Omega(s) \) be divided into subsets \( \Omega(1, s), \Omega(2, s), \ldots, \Omega(m, s) \), where \( \Omega(1, s) \) denotes the 1st subset move in the possible set of total moves generated by the topology generator etc.
Layered Queuing Network

LQN and NoC Contention in NoC

Table 1: LQN/Contention Model Conversion

<table>
<thead>
<tr>
<th>LQN Element</th>
<th>Contention Model Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Task</td>
<td>Source Core</td>
</tr>
<tr>
<td>Non-Reference Task</td>
<td>Router/ Destination Core</td>
</tr>
<tr>
<td>Thinking Time (Z)</td>
<td>Packetization Delay</td>
</tr>
<tr>
<td>Execution Time of Task</td>
<td>Reference Task Packetization Delay</td>
</tr>
<tr>
<td></td>
<td>Non-Reference Task De-packetization Delay</td>
</tr>
<tr>
<td></td>
<td>Router Arbitration Delays</td>
</tr>
<tr>
<td>Number of Transactions</td>
<td>Number of packets sent from component i to j</td>
</tr>
</tbody>
</table>
Different methods include:

- Adaptive routing
- Task rescheduling
- Router buffer space allocation
- Virtual Channel (VC) insertion

VC Insertion

- Power
- Performance
The performance improvement $\Phi_j$ is greater than the extra power dissipation $P_j$ that the on-chip network will experience.

- There are enough VC resources for the insertion to take place.
- The new frequency of operation will not exceed the maximum allowable frequency.
A Solution Space of $N$ Cores

- For a move within the TS given the constraints imposed by the SFS yields $N(N-1)/2$ moves $= O(N^2)$
- Swaps needed to place the cores in the new topological arrangement results in a complexity of $O(1)$
- $O(N)$ time is needed to evaluate the $N$ cores.
- For a total of $k$ iterations within the search
- Average $TL(s)$ search time of $i$
- Overall Complexity of the method can be expressed as:
  $$O(k*[N^2+N+ i])$$

It can be further simplified as $O(N^2 + N)$, assuming $k$ and $i$ are much smaller than $N$
---- EXPERIMENTAL RESULTS ----
**BENCHMARKS**

- MPEG4 Decoder – 12 cores
- Network Communication System (NCS) – 15 cores
- Multi-Window Display (MWD) – 15 cores
- Set Top Box – 25 cores
- Audio Video (A/V) – 21 cores
- D26_media – 26 cores
- Layer-3 Switch – 12 cores

**TEST SET UP**

- 1 GB RAM, 1.66 GHz Pentium based Linux System
- Network Interfaces = 0.2 mm²
- Routers modeled as individual components in floorplanner (VCs also considered)
- Fully specified temporal pattern traffic generation
- Buffer sizes = 4 flits per port, max 6 ports/router and frequency of 2 GHz
All topologies redesigned using 65 nm technology

Comparable on-chip area overhead

< 50 sec to generate and analyze topologies

33.1% less power with 33.6% increase in throughput (flits/sec)


Experimental Results

Genetic Algorithm based Topology [21]

Tabu search based

Mesh based

NOCS-12
## Experimental Results

- Deadlock/Contention Analyzer – accurate within a 19.8% error margin
- Many contention points also occurred at deadlock cycles
- Deadlock comparison to resource ordering technique of Dally: able to save 84.8% resource saving, in turn using 9.35 times less power with a slight performance improvement of 4%

### Table: VC Resource Comparison

<table>
<thead>
<tr>
<th>Topology</th>
<th>Sub-Network</th>
<th>Performance (%)</th>
<th>Power (%)</th>
<th>Total VCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>SDRAM2</td>
<td>0.3215</td>
<td>6.37</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SDRAM</td>
<td>11.5646</td>
<td>6.37</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>RISC</td>
<td>12.01</td>
<td>14.57</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>12.24</td>
<td>5.87</td>
<td>2</td>
</tr>
<tr>
<td>NCS</td>
<td>ITC</td>
<td>19.10</td>
<td>10.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARM926</td>
<td>4.18</td>
<td>4.70</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Router 0 to</td>
<td>5.877</td>
<td>4.71</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Router 3</td>
<td>9.04</td>
<td>8.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>36.157</td>
<td>6.390</td>
<td>2</td>
</tr>
<tr>
<td>Layer-3 Switch</td>
<td>DMA_1 Master</td>
<td>5.7</td>
<td>2.00</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>DMA_1 Master</td>
<td>36.157</td>
<td>6.390</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Master</td>
<td>37.023</td>
<td>37.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MEM1</td>
<td>1.483</td>
<td>8.945</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>30.00</td>
<td>12.70</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>21.99</td>
<td>12.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMEM2</td>
<td>0.6</td>
<td>12.7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>18.2</td>
<td>13.99</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ASIC2 Slave</td>
<td>19.48</td>
<td>12.70</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>CMEM3</td>
<td>20.95</td>
<td>20.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>18.3</td>
<td>15.2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>MEM1</td>
<td>25.1</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>21.99</td>
<td>12.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU2</td>
<td>20.95</td>
<td>20.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>18.3</td>
<td>15.2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ARM CPU</td>
<td>23.4</td>
<td>13.22</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Actual</td>
<td>25.1</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D26_Media</td>
<td>18.3</td>
<td>24.8</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram: VC Resource Comparison

- Power & Performance Efficient
- Resource Ordering Method

---

NOCS-12
Summary

• Proposed a methodology to produce efficient performance and power optimization in NoC design using Tabu Search optimization method
• New approach to contention relief using Layered Queuing Networks (LQN) and a power and performance tradeoff
• 33.1% less power dissipation (on average) as compared to previous works
• Average performance improvement (including contention relief) of 33.6% (flits/cycle)
• Deadlock and Contention VC insertion technique allowing upto 84.8% resource savings with lower power