Minimizing Clock Domain Crossing in Network on Chip Interconnect

Parag Kulkarni\textsuperscript{1}, Puneet Gupta\textsuperscript{2}, Rudy Beraha\textsuperscript{3}

\textsuperscript{1}Synopsys
\textsuperscript{2}UCLA
\textsuperscript{3}Qualcomm Corp. R&D
Outline

• Motivation
• The Router Coloring Problem
• Approaches
• Results and Summary
Clock Domain Crossings in NoCs

- A “Clock Adapter” is needed every time a packet has to cross from one clock-domain to another. For asynchronous clocks, this entails using a FIFO and gray-code synchronization of the FIFO pointers (“async-fifo”).

![Diagram of clock domain crossings]
Motivation

• These “Clock Adapters” are expensive, reducing CDCs in the NoC fabric:
  – Improves Latency: avoiding synchronization delays can result in better end to end latency
  – Reduces Area: saving silicon real-estate
  – Helps Verification: each CDC has to be verified, effort is needed on both simulation and formal verification fronts
  – Simplifies Physical Design: the skew of bits crossing CDC boundaries has to be carefully managed, requiring manual intervention and checks
Motivation

• Existing topology tools generate topology optimizing for many metrics, but require manual assignment of clock-domains to constituent routers
• Modern SoCs: upwards of 40 Masters + Slaves, NoCs: 4-10 routers
• Even 3-4 Clock Domains large set of router frequency assignments!
• Objective:
  • Assign frequency/clock-domains to routers to minimize CDCs in NoC
  • Model cost of CDCs and add to cost-function during topology generation
Outline

• Motivation
• The Router Coloring Problem
  • Formulation
  • NP-Hardness
• Approaches
• Results and Summary
Given a set of cores (both masters and slaves) \( C = \{c_1, c_2, ..., c_n\} \), each with a frequency \( f_i \in F = \{f_1, f_2, ..., f_k\} \) and a set of routers \( R = \{r_1, r_2, ..., r_m\} \) that connect them in a topology \( T \). Assign a frequency \( f_j \in F \) to each router \( r_j \in R \) such that the number of CDCs in \( T \) are minimized.

Given a graph of interconnected nodes, some colored (cores) and some uncolored (routers). Color the uncolored nodes such that the number of edges that connect two nodes of different colors is minimized.
NP-Hardness

• Router Coloring is a variation of the multi-terminal cut problem:
  – **Given** a graph $G = (V,E)$, a set $S = \{s_1, s_2, \ldots, s_k\}$ of $k$ specified vertices or terminals, and a positive weight $w(e)$ for each edge $e \in E$, **find** a minimum weight set of edges $E' \subseteq E$ such that the removal of $E'$ from $E$ disconnects each terminal from all the others.

• When the number $k$ of terminals is two, this is simply the min-cut, max-flow problem.

• Becomes **NP-hard for $k \geq 3$ for arbitrary graphs**\(^1\). Polynomial time for planar graphs for fixed $k$, but exponential in $k$: $O((4k)^k n^{2k-1} \log n)$

Outline

• Motivation
• The Router Coloring Problem
• Approaches
  • ILP
  • Approximate Heuristic
  • Comparison
• Results and Summary
ILP : Formulation

• Can be setup and solved as an ILP problem
  – Colors: $C = \{c_1, c_2, \ldots, c_m\}$
  – Routers: $R = \{r_1, r_2, \ldots, r_n\}$
  – Variables in ILP: $X = \{x_{11}, x_{12}, \ldots, x_{nm}\}$
  – $x_{ij}$ represents whether or not router $r_i$ is colored with $c_j$

• Each router can be colored with exactly one color, resulting in:
  – Each $x_{ij} \in [0,1]$
  – For each router $r_i$ with variables $X_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$, $\sum_{j=1}^{m} x_{ij} = 1$
ILP : Example

\[ CDC_{R0} = 2 \times X_{0Y} + 1 \times X_{OR} \]

\[ CDC_{R1} = 2 \times X_{1Y} + 0 \times X_{1R} \]

\[ CDC_{ROR1} = \frac{1}{2} \times (|X_{OR} - X_{1R}| + |X_{0Y} - X_{1Y}|) \]

- \[ CDC_{NoC} = CDC_{R0} + CDC_{R1} + CDC_{R0R1} \]
- **Minimize**: \[ CDC_{NoC} \]
- **s.t.**
  - \{\(X_{OR}, X_{0Y}, X_{1R}, X_{1Y}\) \} \in [0,1]
  - \(X_{OR} + X_{0Y} = 1\)
  - \(X_{1R} + X_{1Y} = 1\)
Approximate Heuristic: Motivation

- Easy to color when more is known about connections:
  - $r.\text{knownInformation} = \frac{r.\text{numColoredConnections}}{r.\text{numTotalConnections}}$

- Maintain the routers in a queue sorted by known information

- Complexity $O(n^2 + nk)$

- At each iteration:
  - Color router at front of the queue
  - Update known information for connected routers
Approximate Heuristic: Example

- **Break Ties with more common color**

- **Priority Queue at each stage**

- **Routers with more “known information” colored first**

- **Router with least “known information” colored last**
Optimality

- Tested a combination of random and common NoC topologies, found:
  - ILP is always optimal
  - Kway-Partitioning optimal for all but few topologies, attributed to partition balancing
  - Brute Force takes hours for numRouters $\geq 10$
Runtime

- KWay/ILP take 20-30ms for networks with ~10 routers
- Too slow for inner-most loop of Annealing
- Approximate Heuristic is 200X faster

Average sub-optimality for heuristic varies with network size, ~10% for up to 20 routers
Outline

• Motivation
• The Router Coloring Problem
• Approaches
• Results and Summary
Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Cores</th>
<th>Network Type</th>
<th>Original Flop Count</th>
<th>Optimized Flop Count</th>
<th>Change</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip1</td>
<td>21</td>
<td>Response</td>
<td>5624</td>
<td>3424</td>
<td>-39%</td>
<td>+12%</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>Request</td>
<td>5568</td>
<td>3616</td>
<td>-35%</td>
<td>+9%</td>
</tr>
<tr>
<td>Chip2</td>
<td>45</td>
<td>Response</td>
<td>10432</td>
<td>7424</td>
<td>-29%</td>
<td>+14%</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>Request</td>
<td>7680</td>
<td>7290</td>
<td>-5.3%</td>
<td>+16.6%</td>
</tr>
</tbody>
</table>

- We add CDC cost to the cost-function of the existing SA based topology tool
- Observe a 5-39% reduction in flop-count, while satisfying original latency constraints

3: “Leveraging application-level requirements in the design of a NoC for a 4G SoC.” Rudy Beraha et. Al. DATE 2010.
Summary

- We propose a method to model and optimize CDCs during NoC topology generation
- Prove that the underlying problem is NP-hard
- Present and compare multiple approaches to model the CDC cost, including a novel fast heuristic
  - 200X faster and within 10% of optimal ILP solution
- Applied within an existing topology generation tool our approach results in a 5%–39% reduction in flop count of the resulting interconnect while still satisfying the original communication/performance constraints
Thank you for your attention!

Questions?
Backup Slides
The Router Coloring Problem

• Given a set of cores (both masters and slaves) \( C = \{c_1, c_2, ..., c_n\} \), each with a frequency \( f_i \in F = \{f_1, f_2, ..., c_k\} \) and a set of routers \( R = \{r_1, r_2, ..., r_m\} \) that connect them in a topology \( T \). Assign a frequency \( f_j \in F \) to each router \( r_j \in R \) such that the number of CDCs in \( T \) are minimized.

• By representing each clock domain by a unique color the problem can also be reformulated as the Router Coloring

• Given a graph of interconnected nodes, some colored (cores) and some uncolored (routers). Color the uncolored nodes such that the number of edges that connect two nodes of different colors is minimized.
Example
NP-Hardness : Proof

Each terminal becomes a core with a unique color in the NoC.

All other vertices are routers and all edges remain the same.

Solving router-coloring on the thus created network assigns a color to each router.

Since router coloring minimizes the number of edges between different colors $E''$, the multi-terminal cut $E'$

Hence multi-terminal cut is polynomial time Turing reducible to router coloring, making router coloring NP-hard for arbitrary graphs.
NP-Hard Proof Continued

• In real world applications **direct connections** between cores (nodes of fixed/pre-assigned color) are **unlikely** (possibly prohibited) in the NoC. The router coloring problem is **NP-hard even with such a constraint**

• A solver for router-coloring with such a restriction can still be used to solve multi-terminal cut once all edges that connect any two terminals are removed (by definition part of the final cut)

• Any edges connecting terminals can be removed in $O(n^2)$ ensuring that multi-terminal cut is **still polynomial time Turing reducible to router coloring** even in the presence of such a constraint
K-Way Partitioning

- Router-Coloring can be modeled as k-way partitioning:
  - Each color corresponds to a unique partition
  - Cores are nodes with pre-assigned fixed partition
  - Router can move and be assigned partitions
- We used hMETIS\(^2\) to solve the partitioning problem we setup for router coloring
- Complexity of the approach is \(O(n + m + k \log k)\)