Formalize On-chip Network

CS 6115 Big Project Proposal

Yao Wang

On-chip network is a programmable system that transports data between different components of a System-on-Chip (SoC). It has evolved from the simple bus network to more complicated topologies, such as mesh network, to provide efficient on-chip data transfer for the scaling systems. In this project, I plan to formalize several on-chip networks in Coq and prove some interesting properties about them.

On-chip networks:
I plan to formalize four kinds of topologies: bus, ring, mesh and torus. The complexity of these topologies scales up. The simple bus network has been used widely in SoC design due to its simplicity in the last couple of decades. However, as the SoC gets more complicated and with the advent of multicore processor, people starts to use more complicated and efficient topologies such as mesh network.

The basic approach to model these on-chip networks is treating the network as nodes and links. Each node is identified by an address (e.g. (0, 0) in a mesh network) and each link is identified by two nodes that it connects to. Note the implementation can be different for each network topology.

[Optional] More topologies.

Functions:
Given an on-chip network model, we can define various functions on top. The functions are described below:

- **Distance**: given a source node and a destination node, the distance returns the minimal number of hops from the source node to the destination node.

- **Data Flow**: a data flow defines a communication flow from a source node to a destination node. It also comes with an average data rate. So it's defined by a tuple (source, dest, rate).

- **Link capacity**: each link has a capacity which defines the maximum data rate allowed.

- **Routes**: a route is the path from the source node to the destination node for a data flow. It will be implemented as a list in which every list element is an intermediate node in the path. For example, in a mesh network, a route between (0, 0) and (1,1) will be a list: [(0,0), (0,1), (1,1)].

- **Power**: the dynamic power is defined as the number of hops times the data rate.

- **Routing Algorithm**: a routing algorithm determines the routes for a given data flow. Routing algorithm can be divided into two categories: minimal routing and non-minimal routing. Minimal routing ensures the number of hops is equal to the distance while non-minimal routing does not.

- [Optional] **Non-deterministic Routing**: a data flow can take multiple routes from source to destination.

Properties:
We can prove some interesting properties after formalizing the on-chip networks.

- **Distance commutativity**: Distance(A, B) = Distance(B, A)

- **Step Property**: in the list that represents route, each adjacent element is only differed by one (except at the border)

- **Minimal Routing**: prove some routing algorithms (X-Y routing or Y-X routing) are minimal routing algorithms.

- **Minimal Power**: Power(minimal routing) < Power(non-minimal routing)

- [Optional] **Mapping:** Given a set of data flows and link capacity, find the routes for every data flow while yielding the minimum power. This is a NP-hard problem, I'll implement some heuristics that yield a mapping (not necessarily produces the minimum power)

- **Mapping Minimal Power**: If each route for each data flow is minimal routing, then the mapping yields the minimal power.

- [Optional] **Non-deterministic Sum**: the sum of data rates in each route should be equal to the data flow's data rate.