Software for Simulation and Performance Evaluation of the Reconfigurable ONoC

Malik Malone (Undergraduate), Tim Travitz (Undergraduate), Dr. Lei Zhang (Faculty Mentor)
Depts. of Mech. & Computer Engineering , University of Dayton, Dayton, USA
Dept. of Elect. & Computer Engineering, Case Western Reserve University, Cleveland, USA
Dept. of Computer Engineering, University of Maryland Eastern Shore, Princess Anne, USA

Introduction

As technology continues to increase in complexity, computers and processors are expected to stay reliable by increasing in speed and adhering to strict power limits. Recently, processing clock speeds have begun to plateau. Network on chip (NoC) architectures have been considered as a possible solution. Unfortunately, the metallic interconnections' between the cores introduces issues such as high power consumption, high latency, and increasing complexity that inhibit this approach's effectiveness. A promising solution in response to these issues is the Optical Network on Chip (ONoC); these networks utilize optical interconnections, instead of metallic wiring. ONoCs have attracted attention due to their potential for low power consumption, high bandwidth, and low latency architectures.

Problem Description

NoCs with optical interconnections can be further improved by rearranging the cores on the network to respond to the activity of the application that is currently running on the network. We have developed software tool that can be used to simulate and test the performance of a multi-core Ouroboros ONoC in order to find an improved solution of core arrangements for a given application data log. In this poster we present some of the results we have obtained utilizing our software given a Barnes workload from the SPLASH-2 benchmark.

Future Work

We plan to use the tools to find optimized core arrangements for other SPLASH-2 benchmarks. We also hope to find and identify a pattern based on sections of the data log and utilize it in our model. This would allow us to dynamically find the best configuration for the position of the nodes in a network during simulation time.

Acknowledgements

This research was funded by the NSF under grant no. CCF-1460900, and hosted by Salisbury University for the EXERCISE (Explore Emerging Computing in Science and Engineering) program.

https://github.com/timfoil/PhotonicCoreSimulator

ONoC Software

Log Analysis

The log analysis tool takes real-life NoC log output, and can perform various experiments to gain specific insights on the read log data. This analysis tool can allow us to analyze the log or certain sections of the log for insights on the cores that send/receive the most data, the pairs of cores that send/receive the most data, (see fig 1.) and determining sections of the log where particular core sends large amounts or bursts of data. These functions help to provide insights on how to reorganize node configurations near each other. Thus optimal reconfiguration solutions can be found that have low latency values.

Simulator

To test the theoretical results of an optical network on chip (ONoC) architecture we developed a tool which allowed us to simulate this theoretical setup. Our program allows us to feed it actual log data files from an electrical NoC architecture to simulate the same events on a theoretical Ouroboros ONoC architecture. When simulating a cyclic architecture we can configure our simulation to tell it where to place specific IP nodes in the layout. The software also allows for parameters like flit packet size, number of cycles for teardown time, and number of nodes in a specific architecture to be set and configured.This allows us to easily setup and test specific node configurations with our real-life log data. Fig. 3 shows a basic overview of our simulator structure

Genetic Algorithm

We created a genetic algorithm to provide insights on what specific node configurations would perform better than others. Some results are shown in fig. 5 and 6. These tools and applications have been developed in Java and have been extensively documented using Java's Javadoc documentation generator. This will hopefully allow for our program to be utilized or extended upon by researchers interested in this topic in the future.. The code is currently publicly available on github and can be viewed and downloaded by using the link or QR code in the bottom left corner.

Results

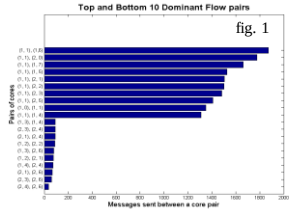

fig. 1

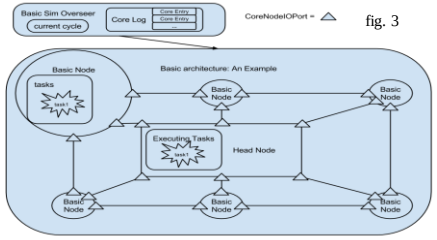

fig. 3

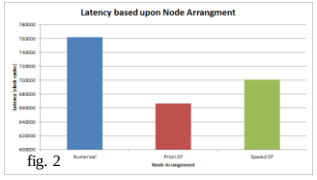

fig. 2

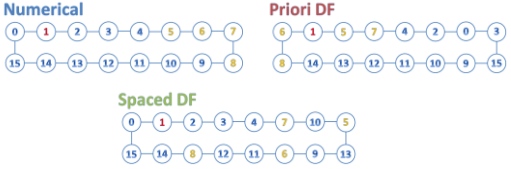

fig. 4

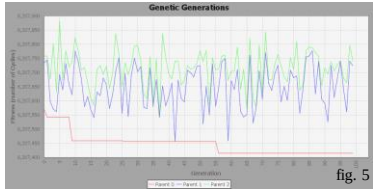

fig. 5

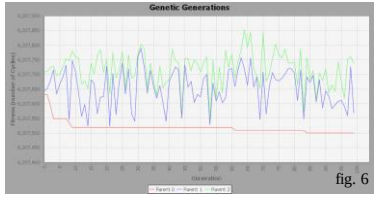

fig. 6

The data analyzer has a feature that enumerates the dominant flow pairs from greatest to least (fig. 1). One can see core (1,1) is clearly the most active node of the network. Using the simulator, we attempted to exploit the dominant flow pairs of core (1,1) by reconfiguring the network into the 'Priori DF' and 'Spaced DF' arrangements, The Priori arrangement, (seen in fig. 4), core (1,1) is labeled as a red 1 and its corresponding most dominant flow partner core is highlighted in gold. The latency comparison of the arrangements in fig. 4, can be seen in fig. 2.

fig5. and fig.6

The y axis represents the fitness of a generation (where a lower fitness number is preferred) and the x axis represents the generation number. Each line represents a parent chosen from that generation (the best 2 individuals of that generation plus the all-time best configuration).
Figure 5 shows the results from 100 generations of 30 individuals per population for the genetic algorithm. Figure 6 can be seen as the control.