Reconfigurable Interconnection Networks for SDR Applications

Tim Stabrawa (stabrawa@stanford.edu)

EE382C – Interconnection Networks
Department of Electrical Engineering
Stanford University

Abstract

In this paper, we review methods for on-line reconfiguration of interconnection networks with the intention of evaluating their suitability for use in a Software-Defined Radio. We first look at a static reconfiguration strategy, then consider dynamic strategies based upon deadlock avoidance and deadlock detection / recovery. We conclude with a summary of which methods would be appropriate for an SDR-system and under what conditions.

1. Introduction

As software-defined radio (SDR) platforms begin to deliver on their promise of a dynamically reconfigurable radio interface, we will be faced with the challenge of how to meet the diverse interconnection needs of a large multi-channel SDR-based system. Unlike traditional radio systems, where the bandwidth demands are both homogeneous and known in advance, an SDR radio could be expected to simultaneously maintain RF connections with any combination of GPS, GSM, UMTS, WiMAX, APCO-25, TETRA, 802.11, Bluetooth, etc. Each air-interface has its own diverse requirements on bandwidth and latency that must be obeyed for correct operation, yet the designers of future SDR systems would like to accommodate all of them in an interchangeable manner.

A traditional approach for interconnecting the line-cards in a wireless system would be to determine the I/O allocations needed to support your expected traffic patterns, then route these channels across the system backplane to other line-cards and/or to a central controller/uplink card. However, using this technique in the backplane of an SDR system is likely to overprovision the links between some line-cards (such as those running standard GSM voice channels), while falling short of the needs for more bandwidth intensive air-interfaces (such as UMTS or WiMAX). The easiest way to alleviate this problem would be to designate different line-cards to different classes, and specify which air-interfaces can be supported by each line-card class. However, full interchangeability can be preserved if we allow for some dynamic reconfiguration of our I/O's into one or more network channels.

One potential topology that can support partial dynamic channel reconfiguration is shown in Figure 1. In this arrangement, each line card has an FPGA which is responsible for collecting and distributing A/D samples between its radio interface(s) and designated

signal-processing node(s). The connections to other nodes can be either hard-wired, reconfigurable, or both depending on the needs of the application. With such a flexible interconnection network, we now have the capability to allocate addition bandwidth between specific nodes as needed. The challenge that remains is how to manage channel reconfiguration in order to meet new bandwidth requirements while not violating the quality of service requirements of existing connections during the reconfiguration.

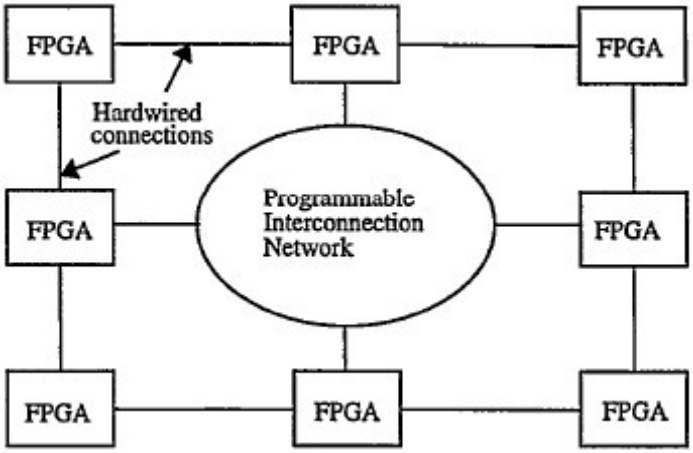

Figure 1 - A generic multi-FPGA system with programmable interconnect [4]

In this paper, we will analyze three techniques for managing channel reconfiguration in interconnection networks and will evaluate what role, if any, that each technique could play in a multi-channel SDR-based system. In section 2, we will introduce a static reconfiguration mechanism proposed by [1]. In section 3, we will cover a dynamic reconfiguration mechanism from [2] that guarantees deadlock avoidance. In section 4, we will consider the possibility for a dynamic reconfiguration mechanism that allows deadlock to occur. [3] Finally, in section 5, we will offer general comments & conclusions.

2. Static Reconfiguration

García and Duato (G&D) describe in their 1993 paper [1], a reconfiguration mechanism that we would later come to know of as a static reconfiguration protocol.

Since this was one of the first transparent reconfiguration protocols, G&D begin by covering definitions and trade-offs for network reconfiguration. In this section, they cover the differences between local and global reconfiguration, pointing out that the later can result in deadlock issues if not handled properly. They also discuss how reconfiguration can be performed without changing the network topology, by instead simply swapping node positions within the network to reduce congestion. Furthermore, they discuss which metrics can be monitored to determine when to perform a reconfiguration operation and how often they should be checked. Specifically, they consider the weight of a communication (# of messages in each direction) and the distance (number of hops) in their cost function. Finally, they consider the tradeoffs between centralized and distributed control of the reconfiguration procedure.

The reconfiguration algorithm proposed by G&D uses local reconfiguration, preserves topology via node-swapping, uses a cost function based upon weight/distance & reconfiguration time, and makes small alterations under centralized control.

When a node decides that the network needs to be reconfigured, it sends a signal to the system controller through a separate control bus. The system controller informs all nodes of the pending reconfiguration, prompting them to stop sending messages. Any messages in transit are buffered at the next intermediate node. Each node sends an acknowledgement to the controller when message delivery has stopped. The requesting node then sends the reconfiguration data to the system controller, which modifies the network topology and broadcasts the new configuration to all nodes. At this point normal communication can resume.

G&D go on to discuss the results of their experiments, claiming to have reduced total message traffic by 35% or more in their transputer-based system. Furthermore they discuss two challenges they encountered while implementing their algorithm: endless cyclic changes and multiple optimal choices.

The former occurs in a hypercube when three nodes are communicating in a triangle-trade pattern; their original algorithm would repeatedly move each node closer to its sender, causing it to move further from its receiver. This cycle would result in a large number of changes with little traffic reduction. G&D resolved this problem by evaluating both inbound and outbound traffic when considering a node swap.

The latter occurs when a node is sending to two different receivers that want to move closer to it. If the node-swap candidates are chosen deterministically, the two receivers can get caught repeatedly swapping locations with each other in an attempt to get closer to the sender, while ignoring an equally suitable node location. They resolved this by using a round-robin strategy among minimal-cost node-changes.

2.1. Evaluation

This paper, written at a time when other reconfigurable networks required the programmer to explicitly decompose his application into phases and optimize them manually, was groundbreaking in that it describes a reconfigurable interconnection network that can *transparently* optimize node locations. The result is they could cut down on traffic congestion without requiring the programmer to hand-tune his/her expected traffic patterns.

On the other hand, this technique has a few drawbacks that limit its use for SDR applications:

- It relies upon a shared bus & centralized controller to coordinate reconfiguration.
  – This limits how well the system can scale. It may work well in the 32-node PARSYS SN 1000 system, but may cause trouble if expected to scale into the hundreds or thousands of nodes.

- All communication must stop during reconfiguration (static reconfiguration). – Depending how long reconfiguration takes and what our latency/jitter bounds are, this may prevent the system from supporting some time-sensitive air-interfaces.
- It doesn't support arbitrary topology changes. – The methods described in this paper only cover node-location swapping. This could still help reduce traffic in fixed interconnection networks, but wouldn't let us take advantage of any programmable interconnect.

3. Dynamic Reconfiguration with Deadlock Avoidance

Pang, Pinkston, and Duato (PP&D) describe a collection of deadlock-free dynamic reconfiguration techniques called "Double Schemes" in their paper for ICCP 2000. [2]

PP&D begin by discussing the existing implementations of reconfigurable networks. They point out that most implementations at the time use a static reallocation mechanism (similar to that introduced in [1]) which drastically reduces system performance during reconfiguration, making them ill-suited for multimedia, real-time, or QoS-sensitive applications. PP&D acknowledge the improvements offered by a new dynamic method called "Partial Progressive Reconfiguration" (PPR), but point out that PPR's design causes unnecessary complexity / reconfiguration latency and restricts which types of networks it can be used on. (Only a specific type of virtual cut-through network is supported.)

PP&D go on to discuss the potential deadlock scenarios that arise due to dynamic reconfiguration. They point out that the deadlocks that can occur are due to an interaction between the old dependency graph before reconfiguration and the new dependencies after. Called "ghost dependencies," these interactions occur when a packet sent out before reconfiguration holds resources (that were only allowed to be held before reconfiguration) encounters a channel held by a packet sent under the new network configuration. These deadlock scenarios can occur whether the reconfiguration occurs in steps (allowing a momentary cycle to appear in the dependency graph), or synchronously (due to ghost dependencies).

PP&D continue by discussing the effects of topology changes on the creation of unroutable packets. According to their paper, a packet becomes unroutable if its head flit is headed for a node which either supplies a set of escape virtual channels that are no longer available, or does not supply any escape channels. The first condition can occur if the set of escape channels got removed or disabled (the actual topology change we are responding to). These packets may become routable again once the reconfiguration completes. The second condition can occur if the new routing function specifically disallows the use of certain channels (to avoid cycles in the new dependency graph). These packets unless discarded will normally remain in the network blocking other traffic indefinitely. On the other hand, if the routing function always provides for a deadlock-free route for packets in this state, then they can become routable.

The "Double Schemes" proposed by PP&D work by incrementally applying a new routing function in a sequence of steps and associated conditions. The premise behind these steps / conditions, is that deadlocks can be avoided by spatially separating resource allocations that could otherwise create dependency cycles on escape paths (as opposed to the temporal separation achieved by static reconfiguration techniques).

The simplest way to achieve this separation, called the "Basic Double Scheme," is to double the number of escape channels, allowing dependencies from one set to the other, but not both at any given time. That is, the network could have two distinct sets of virtual channels, one for the current routing function to escape deadlock, and another for the next/previous routing function. Other than during reconfiguration, only one set of VC's is used for routing packets at a given time – the other remains drained waiting for the next reconfiguration.

The "Fully Adaptive Double Scheme" is an enhancement upon the basic scheme which allows for a third set of virtual channels to be used for adaptive routing. The "Optimized Fully Adaptive Double Scheme" is a further enhancement, that doesn't need a separate third set, but instead uses the second set of virtual channels (that would remain drained under the basic scheme) for adaptive routing. Ghost dependencies are avoided in this scheme by carefully alternating the adaptive and escape channels. For more information, see the detailed steps in section 3 or the proof of deadlock freedom in Appendix II of [2].

3.1. Evaluation

Based upon PP&D's description of prior work, this paper was a significant contribution to the state of the art in reconfigurable of interconnection networks. Unlike previous methods, which at best would only work on a specific network type, and at worst required all network traffic to be paused during reconfiguration, the "Double Schemes" allow for reconfiguring the routing function of any wormhole / cut-through network without halting traffic or causing deadlock, all while also cutting down on un-routable packets. PP&D then take their contribution one step further, and describe a deadlock-free way to get a dual use out of the new virtual channels (via their "Optimized Fully Adaptive Double Scheme").

Assuming PP&D's analysis is correct, and that the time to drain packets from old escape paths is minimal / bounded (for condition 2 of each scheme), PP&D's "Double Schemes" are all viable methods for route reconfiguration in an SDR-based system. As the authors point out though, in faulty / highly dynamic environments, repeated reconfiguration could lead to delays in delivering new traffic, as the reconfiguration scheme waits for old packets from the previous reconfiguration to drain out of the virtual channels to be used.

4. Dynamic Reconfiguration with Deadlock Detection

In their 2001 paper, Fernández, García and Casado (FG&C) analyze the feasibility of a dynamically reconfigurable network-of-workstations without the type of deadlock avoidance suggested by PP&D. [3]

FG&C begin by explaining the tradeoffs in efficiency between deadlock avoidance and deadlock recovery methodologies. Due to the relative efficiency of deadlock avoidance mechanisms, a deadlock recovery mechanism would only be preferred if it can be shown that deadlocks normally occur very infrequently. FG&C's paper focuses upon estimating deadlock frequency and relative number of lost messages.

FG&C go on to describe methods for detecting (Timeout or Inactive Channels Time) and recovering (progressive or regressive) from deadlock. For this study, FG&C selected ICT for deadlock detection due to a lower false positive rate and simpler design. The recovery mechanism chosen was regressive – the message responsible for a detected deadlock is discarded, allowing other packets to make progress. In a separate trial, FG&C opted to misroute these deadlocked packets (as well as those that would otherwise be unroutable) rather than discarding them.

FG&C did two initial experiments in order to determine a sufficient number of virtual channels to keep deadlock down (3) and to determine appropriate parameters for ICT deadlock detection (64 cycles). All experiments were performed on virtual cut-through networks with randomly generated topologies containing 16, 24, 32, & 64 nodes, a packet length of 512 bytes, and uniform traffic. Reconfigurations were triggered by deactivating network nodes one at a time for different network loads.

FG&C found that for light traffic load, deadlock frequency remained low (below 1-2%) for all networks, and that the closer to network saturation, the more frequent deadlocks would become. They paradoxically stated that the ratio of deadlocked / delivered messages does not depend on which node is deactivated, then gave specific evidence to the contrary. Regarding lost messages, they found that the ratio of lost / received messages strongly depended upon which node was deactivated, but not at all on network load.

When simulating with misrouting, deadlock frequency was higher than without it for non-saturated networks, which they attribute to the higher load imposed by message misrouting. Also, with misrouting, the ratio of lost / delivered messages decreased considerably with network size – the larger the size, the greater the enhancement.

4.1. Evaluation

The authors of this paper claim to have demonstrated that deadlock frequency can be kept low enough to justify using a deadlock recovery mechanism for reconfigurable networks. However, since they failed to quantify exactly how low this ratio needed to be in order to be acceptable, it is difficult to interpret their results objectively. Similarly, they didn't quantify what would be an acceptable message loss rate.

Overall this was a rather difficult paper to follow. Grammatical inelegancies were frequent enough to be distracting and in some cases actually made it difficult to decipher

exactly which experiments were being run for this investigation (particularly with regard to what is done with messages responsible for deadlock).

Looking at the results, it appears as if we could expect up to 3% of traffic to result in deadlock during reconfiguration and up to 12% of traffic to be dropped (without misrouting), depending on the network load and # of nodes. With misrouting, lost messages drop to about 4%. There are few air interfaces that can tolerate this high of a bit-error rate or the retransmission delay it would take to recover lost data (assuming A/D samples are being affected). As a result, deadlock recovery during reconfiguration is most likely not a viable option for SDR-based systems.

5. General Comments & Conclusions

Based on the results of these three papers, we can conclude that, depending upon the types of topology changes desired and the particular bandwidth / latency requirements of the air-interfaces being used, either static reconfiguration, or dynamic reconfiguration with deadlock avoidance would allow us to re-route traffic following a topology change in a multi-channel SDR system. The former could be used in smaller systems when simple node-swaps would be sufficient to reduce traffic congestion, and our air interfaces don't have strict latency requirements. The latter could be used in other situations, provided reconfiguration events are infrequent enough to allow for channels to drain between reconfigurations.

It should be noted that, since it was designed to handle unexpected reconfiguration events, the more advanced algorithm (dynamic reconfiguration with avoidance) is reactive in nature. It may be interesting to consider how this algorithm could be better suited to our needs if adapted to be proactive, based upon traffic flows between line-cards, and the drain-status of the backup escape channels.

6. References

[1] García, J.M., Duato, J.: Dynamic reconfiguration of multicomputer networks: Limitations and tradeoffs. In P. Milligan, A. Nuñez, editors, *Euromicro Workshop on Parallel and Distributed Proces.*, IEEE Computer Society Press (1993) 317-323. http://ieeexplore.ieee.org/iel2/908/7904/00336386.pdf

[2] R. Pang, T. Pinkston and J. Duato, "The double scheme: Deadlock-free dynamic reconfiguration of cut-through networks," *International Conference on Parallel Processing* (ICCP 2000), August 2000. http://ieeexplore.ieee.org/iel5/7045/18965/00876160.pdf

[3] L. Fernández, J. García and R. Casado, "On Deadlock Frequency during Dynamic Reconfiguration in Nows," 7th International Euro-Par Conference, August 2001. http://www.ditec.um.es/%7Ejmgarcia/papers/europar254.pdf

[4] Mohammed A. S. Khalid and Jonathan Rose, "A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems," International Symposium on Field Programmable Gate Arrays, 1998. http://portal.acm.org/citation.cfm?id=275119&coll=portal&dl=ACM