

Bypass Paths

Adam Megacz

August 9, 2007

Abstract

This memo introduces *bypass* paths and explains the motivation for including them and exposing them to the programmer/compiler. The memo then describes the assembler syntax for using such paths and the semantics and obligations which accompany them.

1 Motivation

The Fleet specification requires that there be a fifo between the switch fabric and each inlet port. The size of this fifo must be exposed to the programmer (or the compiler), because it must be taken into account in the calculations necessary to prevent switch fabric cloggage.

It is expected that most Fleet implementations will use the same size fifo for most inlet ports.

However, in writing programs for Fleet, we have found that often a ship's output will be looped back to one of its inputs. When this is the case, the path from that output to the input is often on the critical path for some computation. Furthermore, in such situations it is often possible to establish that at most one datum will ever be on that path at a time.

Given these assumptions, it would be quite beneficial for the programmer to be able to send a datum from the output to the input, and have it *bypass* the fifo at the inlet. Doing so increases the risk of deadlock, so the programmer (or compiler) must be especially careful when using such a mechanism.

2 Syntax

We introduce an additional keyword, `bypass` into the Fleet assembler syntax, to indicate that such bypass paths are desired. This keyword appears *before* the name of the inlet which is the destination of the `sendto`. For example,

```
alu3.out2: take, sendto bypass alu3.in2;
```

This command indicates that it is safe for Fleet to use a path with only a single stage of fifo storage between `alu3.out2` and `alu3.in2` for this command, if such a path exists.

Note that the `bypass` keyword is merely advisory. If no such bypass path exists, the assembler is free to substitute a path which has fifo storage.