

14.30 Consequences For Programmers

As Figure 14.16 shows, the set of mapped addresses do not need to be identical in both address spaces. The goal is to make a bridge *transparent* so the processor does not know about the auxiliary bus. From a programmer's point of view, however, the software may need to accommodate the mapping. For example, when a device is installed in an auxiliary bus, the device is configured with an address, A. If the computer's owner enters A as the device address, software must understand the mapping and use the corresponding address on the main bus. Similarly, if the interaction between the device and the processor involves addresses, the processor must use addresses that the a device on the auxiliary bus understands†.

14.31 Switching Fabrics

Although a bus is fundamental to most computer systems, a bus has a disadvantage: bus hardware can only perform one transfer at a time. That is, although multiple hardware units can attach to a given bus, at most one pair of attached units can communicate at any time. The basic paradigm always consists of three steps: wait for exclusive use of the bus, perform a transfer, and release the bus so another transfer can occur.

Some buses extend the paradigm by permitting multiple attached units to transfer N bytes of data each time they obtain the bus. For situations where bus architectures are insufficient, architects have invented alternative technologies that permit multiple transfers to occur simultaneously. Known as *switching fabrics*, the technologies use a variety of forms. Some fabrics are designed to handle a few attached units, and other fabrics are designed to handle hundreds or thousands. Similarly, some fabrics restrict transfers so only a few attached units can initiate transfers at the same time, and other fabrics permit many simultaneous transfers. One of the reasons for the variety arises from economics: higher performance (i.e., more simultaneous exchanges) can cost much more, and the higher cost may not be justified.

Perhaps the easiest switching fabric to understand consists of a *crossbar switch*. We can imagine a crossbar to be a matrix with N inputs and M outputs. The crossbar contains $N \times M$ electronic switches that each connect an input to an output. At any time, the crossbar can turn on switches to connect pairs of inputs and outputs as Figure 14.17 illustrates.

†Chapter 15 explains why a processor and a device exchange address information.