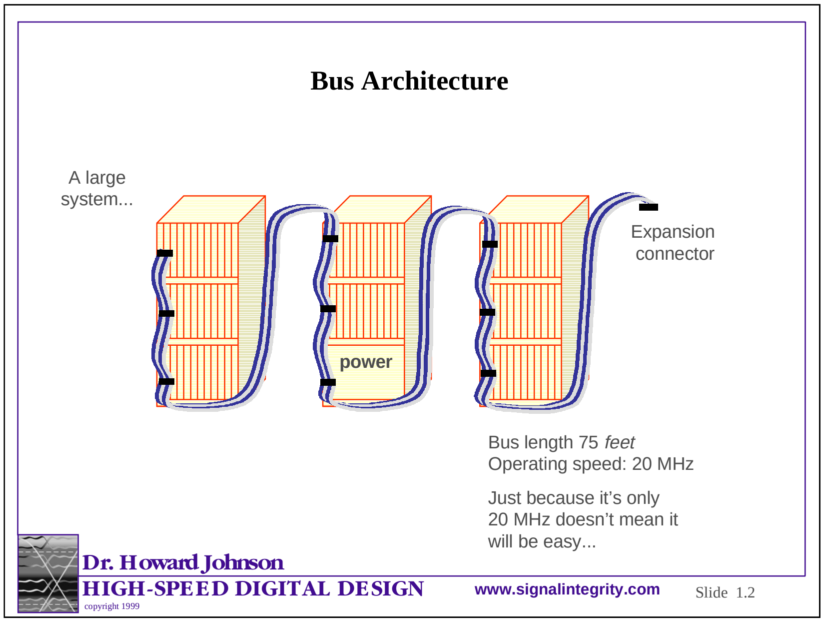
Bus Architecture & Timing

DesignCon99
Santa Clara, CA
February 2, 1999

Dr. Howard Johnson
HIGH-SPEED DIGITAL DESIGN

www.signalintegrity.com
Slide 1.1



I remember the first time I got involved in a bus retro-fit project. That was about 20 years ago, at a company called ROLM.

ROLM is a manufacturer of telephone equipment for medium and large businesses.

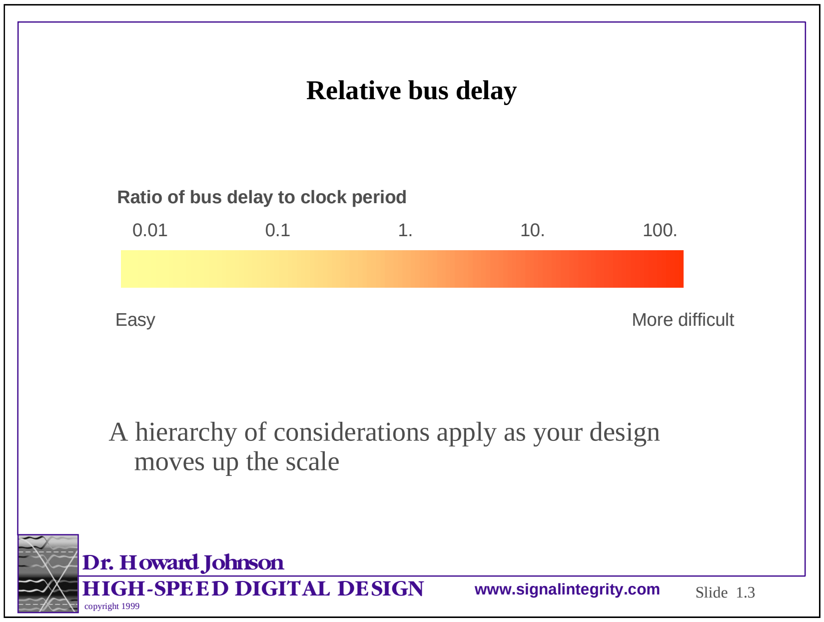
At the time, the main backplane bus in the ROLM system ran at 4.5 MHz. This speed limited the number of customers ROLM could serve from one product. If we could speed up the bus, the product could serve more people.

My assignment was to figure out how to quadruple the bus capacity, boosting the operating speed up to about 20 MHz.

Looking back on it, an operating speed of 20 MHz doesn't sound that difficult, but what you need to know to appreciate this problem is the length of the bus. The bus was 75 feet long.

It snaked up and down through rack after rack of equipment, tying together literally hundreds of PCB's.

I came to recognize on this project that the difficulty of a bus design is related not just to the operating speed, but to the relation between speed and bus length.



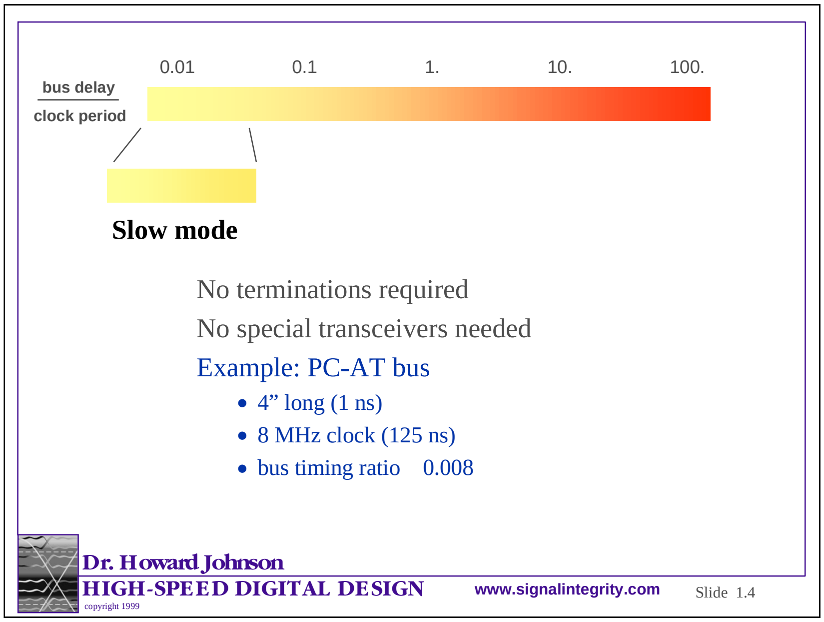
The crucial ratio is the bus delay divided by the clock period.

I've plotted that ratio above, indicating a typical range of operation for most bus designs, ranging from a low of 0.01 to a high of about 100.

Bus designs which have a low bus timing ratio are relatively easy to design, at least in terms of their bus timing. Buses with a big bus timing ratio are more difficult.

As you work your way up the scale, a hierarchy of techniques is employed to produce designs of greater and greater sophistication, which can reach to the uppermost area of the chart.

Let's look at a few examples.

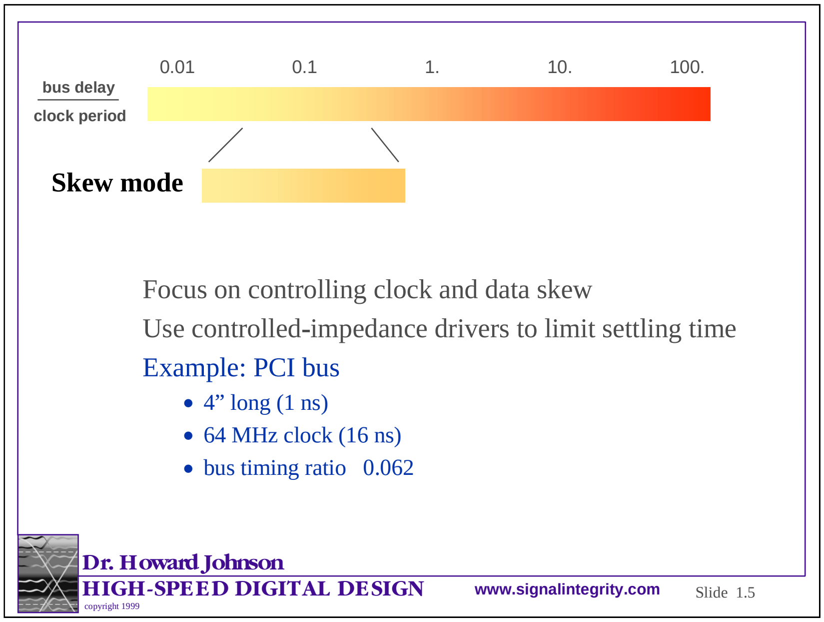


The PC-AT bus, also called the ISA (industry standard architecture) bus, is a fairly simple animal.

It rates a length/speed ratio of less than one percent.

As a consequence, the timing constraints on this bus are rather relaxed. It’s easy to get it to work, and most cards inter-operate (at the physical level, anyway).

You can build this bus from just about any old CMOS ASIC cell.



The next step up in performance occurs when the bus delay significant enough compared to the clock period that we start thinking about optimizing the clock skew.

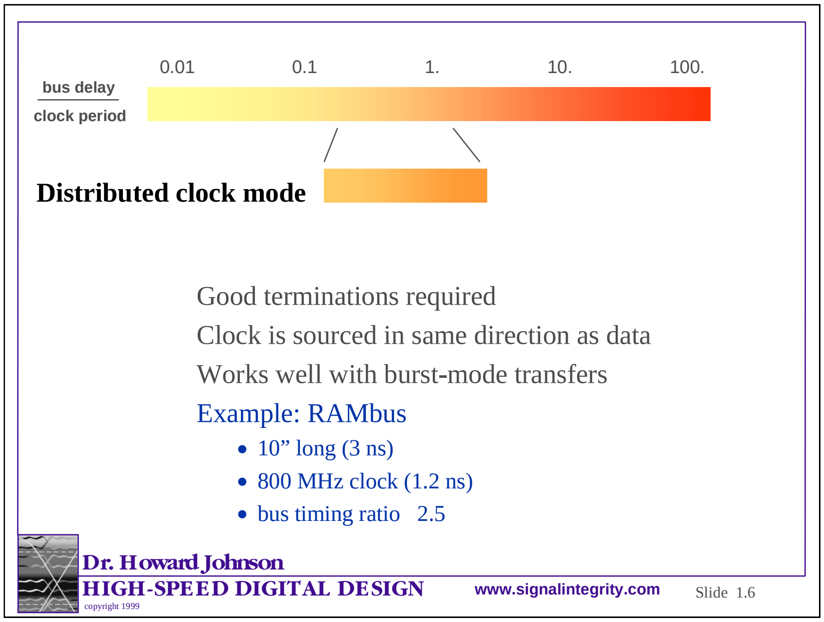
When you write out the equations for bus timing, the clock skew between cards has as much to do with the cycle time as any other delay component. If we squeeze the clock skew, we can usually improve timing.

And there’s a big incentive to work on the clock skew (as opposed to trying to improve the performance of any other element in the system). That’s the fact that there exists only one clock, compared to the many, many other data and control signals.

So, a lot of designers spend time optimizing the clock skew. This is probably a fairly effective thing to do.

There’s a lot of help available for this, too, in the form of low-skew clock generators and buffers.

Just working on clock skew gets you up into the territory of a 30 percent bus timing ratio.



A bus timing ratio near unity requires some very careful thought about the timing budget.

These systems often give up on the idea that everyone along the bus should be operating in the same cycle at the same time.

Instead, they go with some form of a distributed clock, either

(1) two clocks, one for each direction

(2) A separate clock sourced from each transmitter

The idea is to arrange to the clocking scheme so that each receiver gets a clock right in the middle of its received data bit *as that data bit passes along in front of it*.

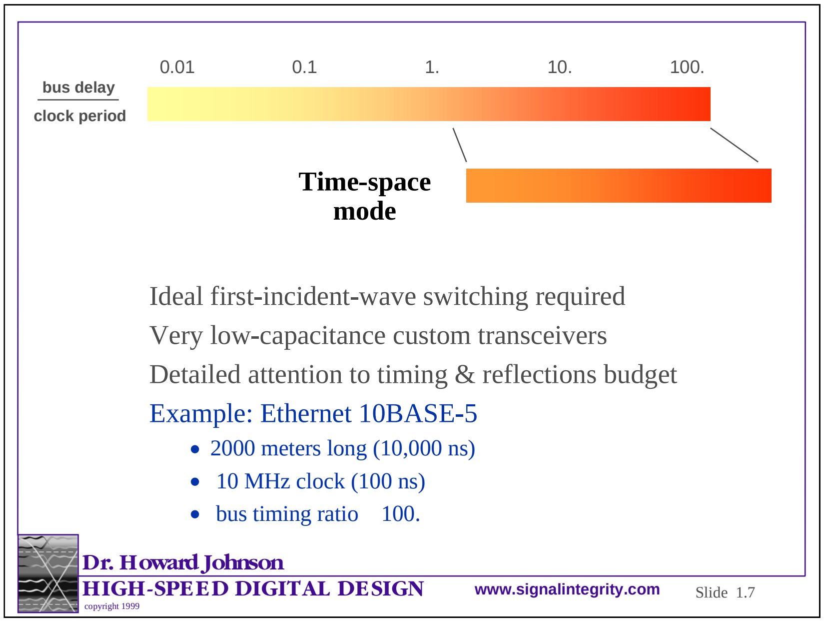
Obviously, all the receivers won’t be getting data at precisely the same time. Therefore, they all benefit from tiny adjustments in their timing.

A good example of this architecture is RAMBUS. They use the two-directional clock idea.

Note that there is often some overhead associated with switching clocks, so this is a scheme that works best with burst-mode transfers, where you set up the clocks, let the system fly for a while, and then reset the clocks again.

This system is in use at speeds of up to 800 MHz. As we go forward into the future, however, at even higher speeds, the bus timing ratio will continue to grow. What happens in the extreme?

At the top end of the scale is the true distributed system architecture.



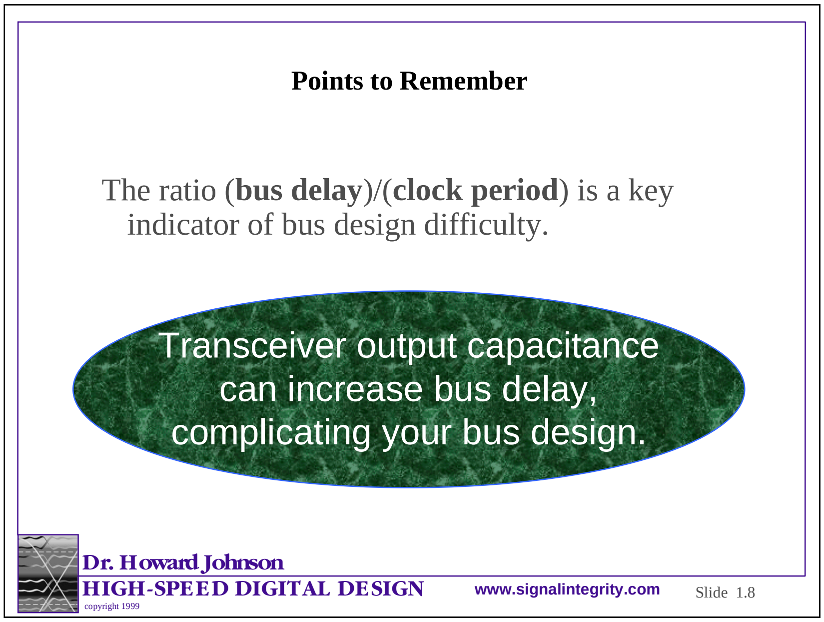
Probably among the premier examples of this type of system is the original Ethernet 10-Mbs coax-based system. This system bears the moniker 10BASE-5.

The original Ethernet operates at a relatively pedestrian rate of 10 MHz, which doesn't sound like much until you realize that with repeaters it can span up to 2-km.

At that maximum radius there can be more than 100 bits in storage, distributed along the cable, at any given time.

The 10BASE-5 Ethernet system is a serial data communications system which encodes its clock as part of the data stream. The clock is extracted by PLL circuits within each receiver. There is an elaborate distributed-control algorithm for deciding who gets to talk when.

Ethernet serves as an extreme example of a large bus timing ratio. If we study this example, it may provide some clues as to how we are the printed-circuit board bus structures of the future.



In summary, I’d like to just make one final point.

This point, if you haven’t heard it before, or haven’t recognized its importance, will make this whole talk worthwhile.

The output capacitance of each transceiver on a bus loads it down.

It also *slows* it down.

A bus with capacitive loading always goes more slowly than one without, **sometimes by a factor of two or more**.

This effect magnifies your bus timing ratio. It makes the bus more difficult to build than you may have first thought.

Don’t get caught short by this effect. Take the transceiver capacitance into account when you do your timing calculations.