Assignment 1

Due January 30th 2009

Timing, transmission of bits, distributed clocks

The problem

You are to write a simple simulator of a simplistic Ethernet hub of my own design (real–life hubs are much fancier than this one).

The hub has 7 local lines to which individual hosts can be connected and one external to the rest of the world. It has one central processor which can handle only one task at a time. The central processor orders the line drivers to start a transmission, accepts full frames from them and makes all the decisions that are made by the hub. The details will be visited in due time.

Transmission details

The bit rate of this system is 1 Gb/s. Propagation speed is $2 \times 10^8 m/s$. Bits are encoded using your favourite encoding method; if imagination fails you, use Manchester. The two possible values of a bit will be represented by two voltage levels (your choice) that are non–zero. No signal implies silence; a signal much stronger than the legal voltage levels is considered a jam signal and is cause for alarm.

A frame is made of:

- A preamble made of 64 bits forming the sequence $10^{31}11$ (1010...1011).
- A body of the frame made of a variable number of bits with a maximum of 12144 bits followed by a period of silence lasting at least 1 ns.

As signal moves, it attenuates; we will assume attenuation of 0.004 dB/m.

As signal moves it also spreads. This phenomenon is not easy to model without heavy mathematics, so we will use a simplistic approximation described in appendix 1. The approximation uses a parameter α which is totally fictitious. Different values of α should be considered; $\alpha = 0.01$ is the default (you may find this value inconvenient for long links).

Your task

For this assignment, your task is to implement a simulator of a hardware driver (NIC) for one local line. The lines are point–to–point (the NIC is at one end; another identical NIC is at the other end which is located at an unknown but small distance).

The basic components of a card are 2 buffers (input and output) and 3 processors: receiver, transmitter and manager. Each of these processors is driven by its own clock ticking at a rate of 50 GHz. Each clock has a programmable register that allows its own processor to set the rate at which the clock will interrupt the processor[1].

A description of the behaviour of the receiver is given below. The behaviour of the manager is irrelevant for this assignment. You will have to design the transmitter so that its behaviour matches the behaviour of the receiver. Please remember that both ends of the line have identical hardware: one receiver and one transmitter.

A simple simulator needs to be written; its only randomised part, the frame arrivals, can be done in any reasonable way. The simulator must be event–driven (your computer is not fast enough to do a real–time simulation of picosecond rates).

The receiver

The receiver can be in one of 6 states:

Listening every time its clock interrupts, the receiver monitors the signal level present in the local line. When it detects a signal, it changes its state to **Synchronising**.

Synchronising the receiver monitors the incoming signal and sets its clock's register to receive interrupts at the same frequency as the bit arrival rate 9or, more likely, a multiple of it). Then the receiver switches to the **Checking** state.

Checking in this state the receiver continues to monitor the incoming signal checking if it correctly times it. when the end of the preamble is decoded, the receiver initialises the input buffer and the switches to the Accepting state; otherwise, the receiver switches to the **Jamming** state after 64 bit durations.

[1]See Appendix 2

Accepting in this state the receiver accepts incoming bits storing them in the input buffer until silence is detected or a bit is perceived as a jamming signal. If silence is detected, the receiver alerts the manager and moves to the **Waiting** state. If it is a jam, the receiver interrupts the transmitter and switches to the **Jamming** state.

Jamming in this state the receiver sends 96 bits of illegal bit values (3 times the highest legal power). Then it switches to the **Waiting** state.

Waiting in this state the receiver does nothing for 10 ns and then starts listening for silence; when a period of 1 ns of no signal is detected, it signals that fact to the transmitter and switches to the **Listening** state.

When the receiver interrupts the transmitter any transmission in progress is immediately aborted and the transmitter enters a wait state waiting for a signal from the receiver.

Note that the receiver will sense the signal produced by its own transmitter counterpart.

Assignment requirements

Your solution must satisfy these requirements:

1. All your clocks will operate at their individual rates (there will be at least 5 different rates: 2 receivers, 2 transmitters and UTC). Propagation phenomena are expressed in UTC time.

2. You will simulate successfully the transfer of at least 10^5 bits per simulation run.

3. Your transmitter will not be trivialised and will have all the functions corresponding to the behaviour of the receiver.

4. The manager is irrelevant and can be truncated to a supplier of frames to transmit.

5. A demonstration will be necessary to get credit for the assignment.

Appendix 1

Let us assume that signal moves from location 0 to location L in the form of discrete values $v(d)$ where d is the distance from 0. Since the signal moves in time, too, the actual values will be $v(d,t)$ denoting the value of the signal at location d and time t.

As the signal travels at the speed of $2 \times 10^8 m/s$, it will move by 1 cm in 50 ps. Taking 50 ps as the unit of time (and turning time and distance into discrete variables) we can express the value of $v(d+1,t+1)$ as a function of $v(d,t)$ with the formula:

$$v(d+1,t+1) = (1-2\alpha)v(d,t) + \alpha \times (v(d+1,t) + v(d-1,t))$$

Some comments:

- You will need values of v for arguments that are not exact multiples of 50ps (moreover: whose ps?). Pick the nearest value available.
- You will be sampling the incoming signal several times per bit duration (so I hope). The current arrangement (1 cm) allows you to sample no more than 20 times per bit duration. This should be sufficient; if it is not, just rewrite the formula reducing α a bit. On the other hand, if you find that you can manage with substantially fewer than 20 samplings per bit, you may increase the distance from 1 cm to, say, 5 cm (this will speed up you program almost 5 times).

Appendix 2

Each clock is independent of all other clocks: its pulses are not synchronised with the pulses of other clocks and its pulse rate is different than the rate of other clocks (they all think they are 50 GHz clocks, but that is because they all have their own definitions of what constitutes a second).

Each clock has two registers associated with it:

Internal counter this counter is decremented on every pulse (if the clock ticks at 50 GHz, a pulse occurs every 20 ps i.e., 50 times per nanosecond). When the internal counter becomes 0, the clock interrupts the processor.

External register When the clock interrupts the processor, it stores in its internal counter the contents of a register called external register (and every pulse will decrement the counter starting with this new value). Thus, if you want an interrupt from the clock every 500 ps, you put 25 in the external register ($25 \times 20ps = 500ps$).

Nobody can change the rate of pulses nor the moment when a pulse is issued by the clock.