Homework 1, Fall 2021

Due: in a week

Directions: No late homeworks. You can talk it over but you must write up and do all the computation by yourself. This one is a lot of work but it will really teach you the main ideas of the Physical layer. Don't worry if you can't solve them all, but put down your ideas. Will grade one of Problems 1, 2, and 3. The last 2 are for exam practice only but we will provide solutions. Try them to prepare for the midterm!

1. **Fourier Analysis**, 25 points: In this problem, we will learn (not for the exam, do not fear, I know that many of you are not EEs) how to calculate Fourier coefficients and how much bandwidth you need for a good approximation.

-***A***. *The square wave:* Consider the square wave that we claimed in class was approximated by the Fourier series sin(2 π *t)* + sin(6 π *t)/3* + sin(10 π *t)/10* + … Why do the scaling factors go down linearly for increasing frequencies? To see this lookup how to find Fourier coefficients (scaling factors) by integration. To find the Fourier analysis you want the coefficient of the n-th harmonic, in other words the scaling factor for the sine wave sin(2 π *n x).* To find the coefficient for an arbitrary function f(x) simply integrate f(x) sin(2 π *n x)* from negative infinity to positive infinity. All you have to remember (or lookup!) is how to integrate a sine function. Write down the proof (even if you copy it from the net). But in one line and your own words, why does the integral result in a scaling factor is inversely proportional to n?

-***B***. *The approximation:* Go to https://www.desmos.com/calculator (you may have to login using Google) and visualize the series of approximations of the series sin(2 π *t)* + sin(6 π *t)/3* + sin(10 π *t)/10* + . . . First, go to the spanner icon (graph settings) on the upper right corner. Set the x-limits from 0 to 3 and the y-limits from -1.5 to +1.5. Also, choose radians. Now do the following. Start with sin(2 π *t),* then sin(2 π *t)* + sin(6 π *t)/3,* then do in(2 π *t)* + sin(6 π *t)/3* + sin(10 π *t)/10* Do one more term that you should be able to guess. For each of these 4 approximations to the square wave, answer the following questions:

- **B.1** Turn in the graph of the function (you should be able to save, worst case do a screenshot) in your report
- **B.2** How much bandwidth in Hertz do you need?
- **B.3** If you sample the signal anywhere in the middle third ((clock recovery is not perfect), what is the largest percentage error in the amplitude you find (it should reduce with better approximations (hover over points to see values)
- **B.4** How long does it take to ramp up? That is for each approximation, how long does it take to reach its peak (for the first basic sine approximation, it

should be 0.25. Hover over the first peak to see the x-coordinate

B.5 Also, answer the following question, why do we get a bad approximation if we add the even harmonics (e.g., 2Hz, you can use Desmos to try), also what happens if you do not scale the higher harmonics down (try using 1 instead of 1/3 for the third harmonic). Have some fun playing with various "paintbrushes" for approximating functions!

2. Nyquist Limit: This will take some work but at the end you should have a good idea about how bits are sampled, and what inter-symbol interference means. Figure 1 shows the response of a wire to two input bits, a slow bit of width 2 usec and a fast bit of width 1 usec. The response to ***both*** slow and fast bits (this shows that the wire cannot respond faster than once every 2 usec, its Nyquist limit) is shown at the bottom of Figure 1. Note that I have chosen to make the response a triangular approximation to the ***sinc*** function we studied in class. It rises to 4 Volts in 1 usec, falls to 0 after 2 usec. It then falls to -2 Volts at 2.5 usec, climbs to 0V at 3 usec, rises to 2 V again at 3 usec, and finally falls to zero at 4 usec. Unlike the real sinc function, this idealized output goes to 0 after 4 usec.

Assume that a 1 is encoded as a 4 Volt signal, and a 0 is encoded as 0 Volts. Assume that the output to a 0 Volt input of any bit width is also 0 Volts for all time. If at any sampling instant, the receiver measures a voltage of 2 Volts or more, it assumes it has received a 1; else it assumes it has received a 0. The sender is going to send just 3 bits 101

- **A**, 7 points First assume the sender sends bits at the slow rate of once every 2 usec. Use graph paper and color pens (or a program) to draw the 3 bits: the first bit as red, the second in blue, and the third in black (using these colors will help the grader). Assume the sender sends its 3 bits at times 0, 2 usec, and 4 usec. The sampling instants the receiver uses are 1 usec, 3 usec and 5 usec. At any sampling instant, the receiver measures the output voltage as the sum of the voltage values of the red, blue, and black waves. Write down the measured outputs. What bits does the receiver output?
- **B,** 7 points Repeat the same process you did in Part B but this time using the "fast bit" of width 1 usec (you are now signalling at the fabled Nyquist limit). The sender now sends its bits at times 0, 1 usec, and 2 usec and the receiver samples at 1 usec, 2, and 3 usec. Write down the measured outputs. What bits does the receiver output?
- **C**, 11 points Repeat the same process you did in Parts A and B except with a "super- sonic" bit whose width is 0.5 usec. The output response to the supersonic input bit is exactly the same as for the fast and slow bits. The sender now sends its bits at times 0, 0.5 usec, and 1 usec and the receiver samples at 1 usec, 1.5 usec, and 2 usec. Clearly, the sender is being cheeky and you should see intersymbol interference (at which sampling instants?). What bits does the receiver output?

3. Clock Recovery, 25 points: The first problem taught you about how outputs can be computed using Fourier series, the second problem taught you output distortions can cause inter-symbol interference if you send too fast. However, we assumed in Problem 2 that the sender and receiver clocks were perfectly synchronized. In reality, they are not and we need clock synchronization. In Problem 3 you will simulate the effect of clock recovery on some bits sent using 4-5 encoding using a clock synchronization algorithm I give you below.

   Assume the preamble has been received and the receiver is basically in sync except for possible clock drift. Thus the receiver is sampling according to its current clock (see figure 2) and should be expecting transitions only at what it thinks are bit boundaries (see dotted lines in figure). However, because of clock drift the actual transitions may be a little off (see the solid line in Figure 2).

   Remember that in 4-5 coding you are guaranteed to get ***at least*** one transition in every 5 consecutive bits; however, you may get ***up to*** 5 transitions. Pseudocode for the receiver clock recovery algorithm is as follows.

   Receiver Code

   Data

Structures:

```
T: real constant; (* nomimal time to send a bit, input to
program *) P: real; (* predicted next time at which a
transition might occur*)  A: real; (* actual real time at
which a transition occurs *)
lag: real; (* difference between predicted and expected *)
```

After preamble is detected:

```
Initialize real time clock to start at
0; lag = 0;
P = 0;
```

```
StartTimer (T/2);
Wait (TimerExpiry);

Do until end of
   frame P = P + T +
   lag;
   Output (SampleSignal); (* output sampled value when timer expires *)
   StartTimer (T + lag);
   Wait (Timer Expiry);
   In parallel with Wait look for a Transition if
     any If Transition is detected at actual time
     A
        lag = A - P; (* difference between real and predicted *)
end
```

You are going to run this code assuming a nomimal bit time of 1 usec (e.g., T = 1 usec) and a sender who is sending 6% slower than the receiver. Thus the sender sends his first bit from 0 to 1.06, the second bit from 1.06 to 2.12, the third from 2.12 to 3.18 . Without doing any clock recovery or lag adjustment the receiver would sample at what it thinks is the middle of a bit and so at 0.5, 1.5 etc. We would like to see what happens on the ten bit sequence 0101101010 with and without clock recovery. Assume a 0 is encoded as 0 volts and a 1 as 1 volt.

- Use graph paper to draw a waveform of the 10 bits sent by the sender (5 points)
- If the receiver does not run the clock recovery code, how far off is the sampling by the 10-th bit? (5 points)
- On the picture of the waveform you drew, draw the sampling instants and values of lag assuming the receiver uses the pseudocode above for clock recovery and there is no noise. (30 points)
- Now suppose there is a sharp noise spike of 1V at time 0.3 usec. How would it affect your sampling times. (5 points)
- Finally, suppose there is a sharp noise spike of 1V at time 2.5 usec. How would it affect your sampling times? (5 points)

5 Extra credit points if you can suggest a precise and sensible modification the receiver pseu- docode to deal with noise.

4. Eye Patterns: Figure 3 shows the response of two wires to the same input signal. The input signal is shown as a pattern of rectangular pulses. Assume that the pattern keeps repeating. The interval between the vertical dotted lines represents one bit time. The two outputs $S1$ and $S2$ are shown using dashed bold lines. Notice that $S1$ closely follows the input and all rise and fall times are symmetrical. However, $S2$ represents the output of a more "sluggish" wire (i.e., a wire with less

bandwidth) and so takes longer to track the input signal. Notice that the rise and fall times in $S2$ are also data dependent – it takes longer to rise to a 1 after two zeroes than it does after a single zero.

Using graph paper draw the signals $S1$ and $S2$ (you don't have to measure the signal values from Figure 2; use any values that gives you roughly the same shape as the figures in Figure
2. Then draw the eye pattern for these two signals. From your figures, deduce what would

happen to the eye pattern if the input signal was run through a wire with even less bandwidth (than the wire which produced *S*2) such that intersymbol interference starts occuring.

5. Modal Dispersion in Fibre: Figure 4 shows a multimode fibre. Assume that every 1 is encoded as a pulse of light that splits into four signals, one of which goes the direct route and the other two go by long routes (bouncing all the way). Suppose the time taken for the longest route is 150 nsec, the second route is 120, the third is 110 nsec, and the time taken for the shortest route is 100 nsec. What is the maximum bit rate we can use on the fibre without causing intersymbol interference? What happens if we double the length of the fibre link?

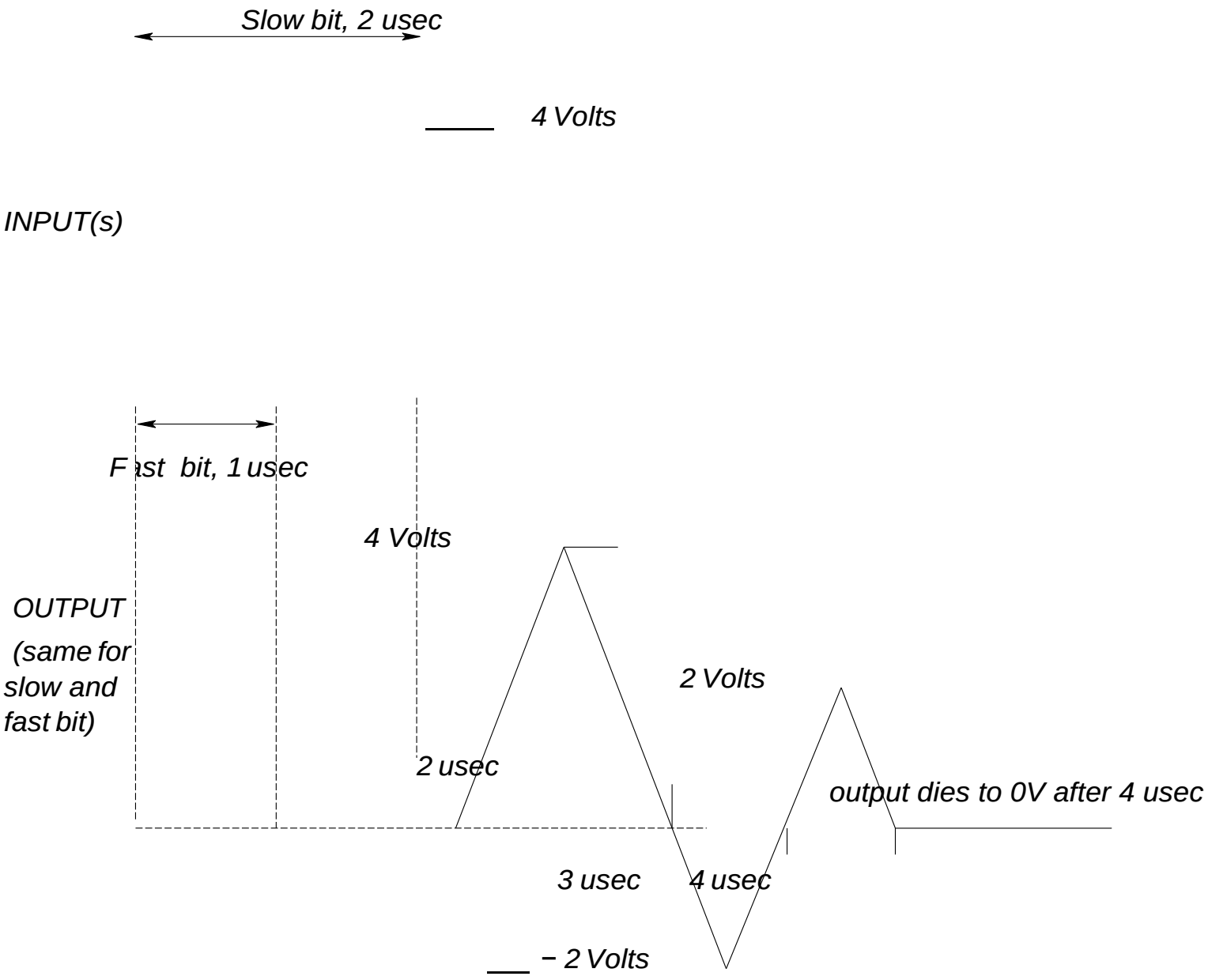

Figure 1:

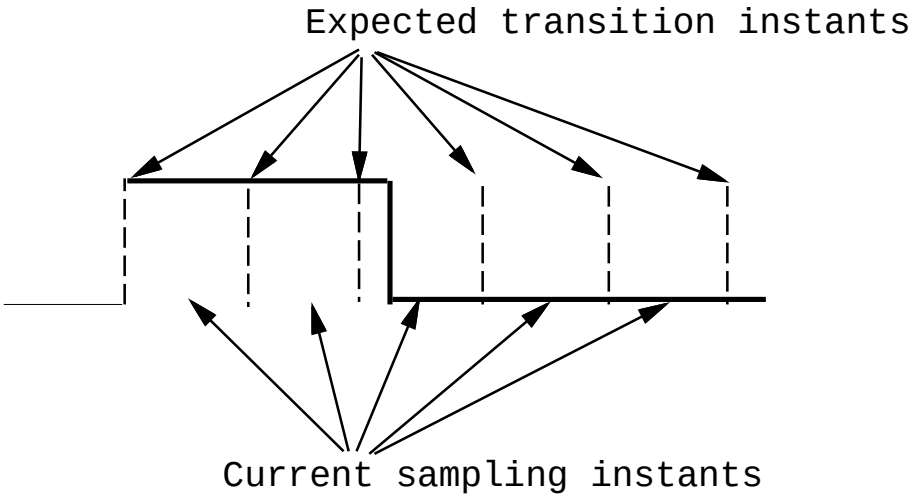

Figure 2:

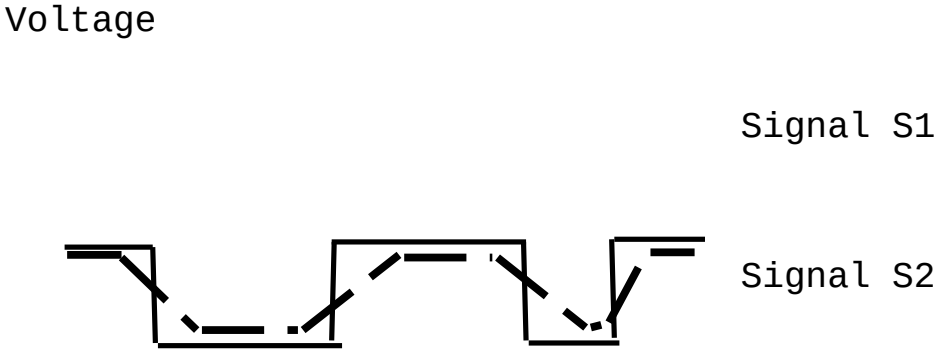

Figure 3

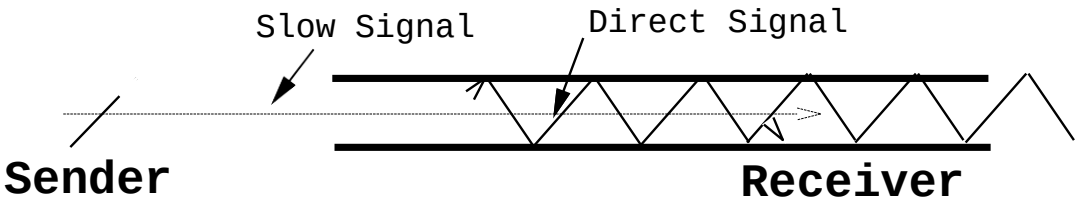

Figure 4: