Project 1: 8b/10b Encoding with a State Machine

Due on October 5th, 2020 at 23:59

Introduction

You may recall that data can be represented as a sequence of Boolean values called "bits", which commonly take values of '0' or '1'. For data communication, a sender transmits these bits to a destination using some agreed upon method. The simplest way for the sender to send a message to the destination is to simply send it bit. However, it is often useful for the sender to apply some reversible transformation to the data before sending it that the receiver will simply undo at the destination. One such transformation is the 8b/10b encoding, which will be the focus of this project.

On a communication channel, we would like to send any possible combination of '0's and '1's. However, it is often undesirable to send a message with many more '0's than '1's or vice versa. One case where this is true is in Ethernet communication over twisted wire pairs. For example, to send a '0' over the wires, the sender can put a voltage of perhaps $-2.5\,\mathrm{V}$ between the wires and for a '1' we can put a voltage $2.5\,\mathrm{V}$. If a roughly equal amount of '0's and '1's are sent across the channel, the average voltage will be about $0\,\mathrm{V}$, i.e., the data signal is "DC free". If instead there is a large disparity between the number of '0's and '1's the average voltage may be very far from $0\,\mathrm{V}$, i.e., there is a non-zero DC component. Since Ethernet over twisted pairs uses coupling transformers at each end of the link, the DC component will be blocked (we learn this later in the course), which leads to a distortion of the data signal and eventual detection errors.

To solve this problem, the sender may apply a procedure to the data message that is called "8b/10b encoding". In 8b/10b encoding, blocks of eight original data bits are mapped to blocks of 10 bits which are transmitted. The disparity of the block of 10 bits, which is the difference between the numbers of '1's and '0's in the block, is either 0 (five '1's and five '0's), +2 (six '1's and four '0's) or −2 (four '1's and six '0's). The "running disparity" is the accumulation of block disparities, and the 8b/10b encoding ensures that the running disparity at the end of every block is either +1 or −1. For this, the mapping is organized in a way that the every combination of eight bits maps to either one or two combinations of ten bits. When an eight bit input maps uniquely to one ten bit output it is guaranteed that this ten bit output will have an equal number of '0's and '1's (i.e., zero block disparity). Otherwise, the eight bit input can map to either an output with two more '1's than '0's (block disparity +2) or two more '0's than '1's (block disparity −2). Therefore if the running disparity of '1's over '0's is ± 1 at any point, the sender will always be able to choose an encoding of their message that keeps the disparity at ± 1 (i.e. the encoding has an equal number of '0's and '1's) or flips the disparity to ∓ 1. Thus, at at point, the running disparity of '1's over '0's or '0's over '1's will be one.

The 8b/10b encoding scheme is used in many common applications including USB 3.0, Gigabit Ethernet and Fibre Channel. The USB 3.0 protocol runs at an astonishing $4\,\mathrm{Gbps}$. This means that the above encoding must be performed 500 million times every second. At this speed, it is impossible to use a regular CPU to perform the encoding, instead specialized hardware must be used. However, it is difficult for digital hardware to use logic as sophisticated as that employed by CPUs. One piece of logic that is available (and commonly used) in digital hardware is the state machine (see CPEN 211/312 for more).

Task 1 (5 marks): Detecting Running Disparity in a Data Channel

In this first task, we explore the detection of running disparities. Let $\mathbb{B} = \{0, 1\}$.

Design a state machine that takes as input a discrete-time signal, $x \in [\mathbb{N} \to \mathbb{B}]$ and outputs another discrete-time signal $y \in [\mathbb{N} \to \mathbb{B}]$ such that $y(n) = 1$ if and only if at any point in the past the signal x had a running disparity ± 3 or more. In other words,

$$y(n) = 1 \iff \text{there exists a } k \text{ that is less than or equal to } n \text{ where } \left| \sum_{i=0}^{k} \sigma(x(i)) \right| \geq 3$$

where $\sigma \in [\mathbb{B} \to \mathbb{Z}]$ such that $\sigma(0) = -1$ and $\sigma(1) = 1$.

As the solution of this task, write

1. your set of states S,
2. your initial state $s_0 \in S$,
3. and your update function $f \in [\mathbb{B} \times S \to \mathbb{B} \times S]$

on the answer sheet.

Task 2 (5 marks): Analyzing the 8b/10b Encoder

Next, we focus on the 8b/10b encoder system. Read the following description of the 8b/10b encoder.

The 8b/10b encoder is a system that transforms a sequence of eight bit inputs into a sequence of ten bit outputs. The encoder keeps track of the previous running disparity between '0's and '1's in the output sequence to determine the disparity of the next output. By design, this disparity may only be -1 (to denote there has been one more '0' than '1' in the previous outputs) or $+1$ (to denote there has been one more '1' than '0' in the previous outputs). Although there is no disparity at the beginning of the sequence, the initial disparity is set to -1.

In the first stage, the eight bit input is split into a three bit part and a five bit part. The three most significant bits of the eight bit input are fed into a 3b/4b encoder, while the other five bits are fed into a 5b/6b encoder.

In the second stage, the three bit part and the five bit part are fed into a block that produces a two bit output. The first bit of this output is a '1' only if the six bit output corresponding to the five bit input has disparity (i.e. more '0's that '1's). Similarly, the second bit is a '1' only if the four bit output corresponding to the three bit input has disparity. This output is fed into both the 3b/4b encoder and the 5b/6b encoder to ensure the encoders cooperate to produce an output that satisfies the design requirement that the next running disparity is -1 or $+1$.

The third stage of the encoding process is the 5b/6b encoder. In the 5b/6b encoder, a lookup table is used find the six bit output that corresponds to the five bit input. This output will either have three '0's and three '1's or four '0's and two '1's. In the former case, no action is needed. If the latter is true and the running disparity is -1, then the resulting disparity would be -3, which violates the design. Therefore, in this case, each bit in the output is flipped, either from a '0' to a '1' or a '1' to a '0'. This new output has two '0's and four '1's and therefore the new disparity is $+1$. Finally, the block must use the input from the second block to predict whether the 3b/4b will also flip its output as this will effect its new state.

The fourth stage of the encoding process is the 3b/4b encoder. The 3b/4b encoder operates similarly to the 5b/6b encoder, using a lookup table to generate a four bit output from a three bit input and flipping the bits of the output if needed to maintain a proper running disparity. In addition, the 3b/4b encoder must use the input from the second stage to prevent the case where both the 3b/4b encoder and the 5b/6b encoder choose to flip the disparity from -1 to $+1$ (or vice versa) by outputting more ones than zeros (this would result in a new disparity of $+3$ violating the design).

Finally, in the fifth stage a ten bit output is created by concatenating the output of the 3b/4b encoder with the output of the 5b/6b encoder. The output of the 3b/4b should precede the output of the 5b/6b encoder.

Draw a block diagram that describes the overall system. Your diagram should include a block for each of the five stages above, but does not need to show any of the logic within. Label each block in your system as memoryless or non-memoryless and time variant or time invariant.

Submit your written response for these written questions via PDF on Canvas.

Task 3 (10 marks): Implementing the 8b/10b Encoder in Hardware

In the final task you will use Python to implement a software model of the 8b/10b encoder. Since strings are made up of ASCII characters with a length of eight bits, you will then use the encoder to encode inputted lines of text.

This portion of the project is available via Github Classroom at https://classroom.github.com/a/8Ewft6m8. Use Git to clone the project directory from the Github repository created by Github Classroom. You can then navigate into the cloned folder using Terminal or Git Bash and run `python main.py`. At first you should see no output from the command, it is working as long as there are no warnings or errors. Next, try typing a message into the terminal (stick to ASCII characters here, i.e., just English letters, numbers and punctuation), the program will attempt to encode each character of the message into a ten bit output. You should notice that the program outputs only zeros, that is where you come in. By implementing the functions in the `implementation.py` file you will be able to make a functioning 8b/10b encoder.

There are four Python files within the repository, `luts.py`, `main.py`, `implementation.py` and `helpers.py`. `helpers.py` contains helpful functions including the ones that poll the lookup table reference above in the 3b/4b and 5b/6b encoders. `main.py` shows how we will exercise the functions you write in this section, **do not change this file**.

To complete the assignment, implement the five functions in `implementation.py` and use Git commands to push your changes to Github. Make sure you pass all the provided test cases for full marks. We may also run your code on additional text cases. If you do not pass all the test cases at first, you can make additional changes to your code and continue to push these changes to Github. **Only your last commit pushed to Github before the deadline will be marked.**