Training-Sequence-Indifferent Decision-Feedback Equalizer

Yan Huang and Zhipeng Li

1. Overview

On the receiver side of communication channel, decision feedback equalizer (DFE) has been widely used to cancel inter-symbol interference (ISI). In most designs, the DFE training algorithm assumes that the training-sequences are known or purely random. However, the generation mechanism of training sequence is often biased; in particular, the standard SATA training algorithm generates more zeros than ones. The training sequence heavily influences the steady state equalizer peformance. Training on an unexpected, or otherwise biased patterns, couldcause unnecessary error. In the industry, it is most cost effective to design a generic DFE for various applications. However, because of the traditional standards set for these different applications, the nature of the training sequence may be different.

In this project, we will program the FPGA to act as a DFE with mean squared error (MSE) cost and added features to account for biased training-sequences. We want to design a blind DFE that can hold simple compare-and-store data that will be able to adjust for things like: an unbalances number of 0's to 1's and other non-random behavior in the training sequence, and adjust for these such that our DFE will be able to minimize bit error. Given some input image, distorted though an ISI channel, we display the received data , with DFE or bypassing DFE, on a screen through VGA to demonstrate the effectiveness of the designed DFE.

2. Background and Motivation

The DFE has long been a staple of receiver technology and have gone through many different incarnations. Yan Huang became interested in receiver technology while interning this summer at Marvell Semiconductors. As an intern in the testing team for the Central Analog Group, I tested various chips which implemented different receiver architectures.

During data transmission, the signal is susceptible to intersymbol interference (ISI) resulting from linear and nonlinear fiber effects. One method of dealing with this is to use a linear feed-forward equalizer (FFE) combined with a decision-feedback equalizer (DFE). This process can be most clearly views using an eye diagram. We can see that without equalization, the "eye" is almost closed. The DFE has the function of expanding the eye. By calculating the effect of a symbol on subsequent symbols, the DFE can widen the eye, so the data stream can be more easily quantized into zeros and ones. The DFE can do this because intersymbol interference tends to cause subsequent symbols to skew toward earlier symbols, e.g. the voltage of a 0 after a 1 is higher than a 0 after a 0. The DFE identifies how much deviation there is and deducts that from the signal, restoring its level.

In this project I'd like to improve upon the traditional approach in several ways. Firstly, in my experience as a tester, I've noticed that the same DFE can pass the bar on one transmission protocol and fail on another. This is partly due to the strictness of the protocol, partly to do with the training sequence A blind DFE assumes the training pattern is random. If the training pattern is not random, the DFE can train incorrectly, and cause tap errors. This error is exacerbated when there are more taps. This is the main problem we'd like to address.

Training patterns fall into several types. Firstly, the best is "purely random" sequences like Pseudorandom bit sequences (PRBS). Blind DFEs are designed to train on such sequences. However there are other training sequences, that are not unbiased. The SATA training sequence is generated in such a way that 1's and 0's are not produced with the same probability, there is a 51-49 split, rather than 50-50. There are other sequences that overcompensate and create the same number of 0's to 1's, so that after a 0, there is a much higher chance of getting a 1 than another 0. The construct of a

traditional blind DFE depends on there being a balanced number of 0's to 1's, and a balance number of transitions to non-transitions. These effects are not noticeable with two taps, but become troublesome with five or six taps. Because later DFE taps train according to earlier DFE taps, a mis-placed tap value can cause -undesired effects down the line.

As new applications always demand higher transmission speeds and data bandwidth , ISI will become a more serious problem, making additional taps necessary for detailed compensation. It could be argued that once could specifically build different DFEs for different channels ad different protocols (i.e. different traing sequences). However, that is an unviable industrial solution. Thus, we'd like design a generic DFE that can adapt to various channels and protocols.

3. Technical Approach

Channel

We will use two different approaches to emulate channel distortion. The first approach is to simulate the ISI by filtering. The second approach is to send the data out through a physical channel, specifically a 4-feet ribbon cable, and receive the data back at the other end.

Input Interface

If we distort data through a physical channel, we need a high-speed analog-to-digital input interface. The intended throughput of this decision-feedback equalizer is 10Mbps. The input signal to the DFE is digital signals distorted by an ISI channel. Because we are doing the equalization digitally, each distorted input sample from the ISI channel must be quantized. As the audio ADC onboard at tens of kilohertz is too slow for the input interface, we propose to use a sigma-delta ADC. Running the FPGA input interface at 80MHz, we could oversample the 10MHz input signal by 8X; with a 1-bit DAC and second order of noise shaping, we could achieve 5.4 effective bits of quantization. The FPGA processing core receives 5-bit input samples at 10MHz from the sigma-delta ADC interface, equalizes the samples, and outputs 1-bit signal at 10MHz.

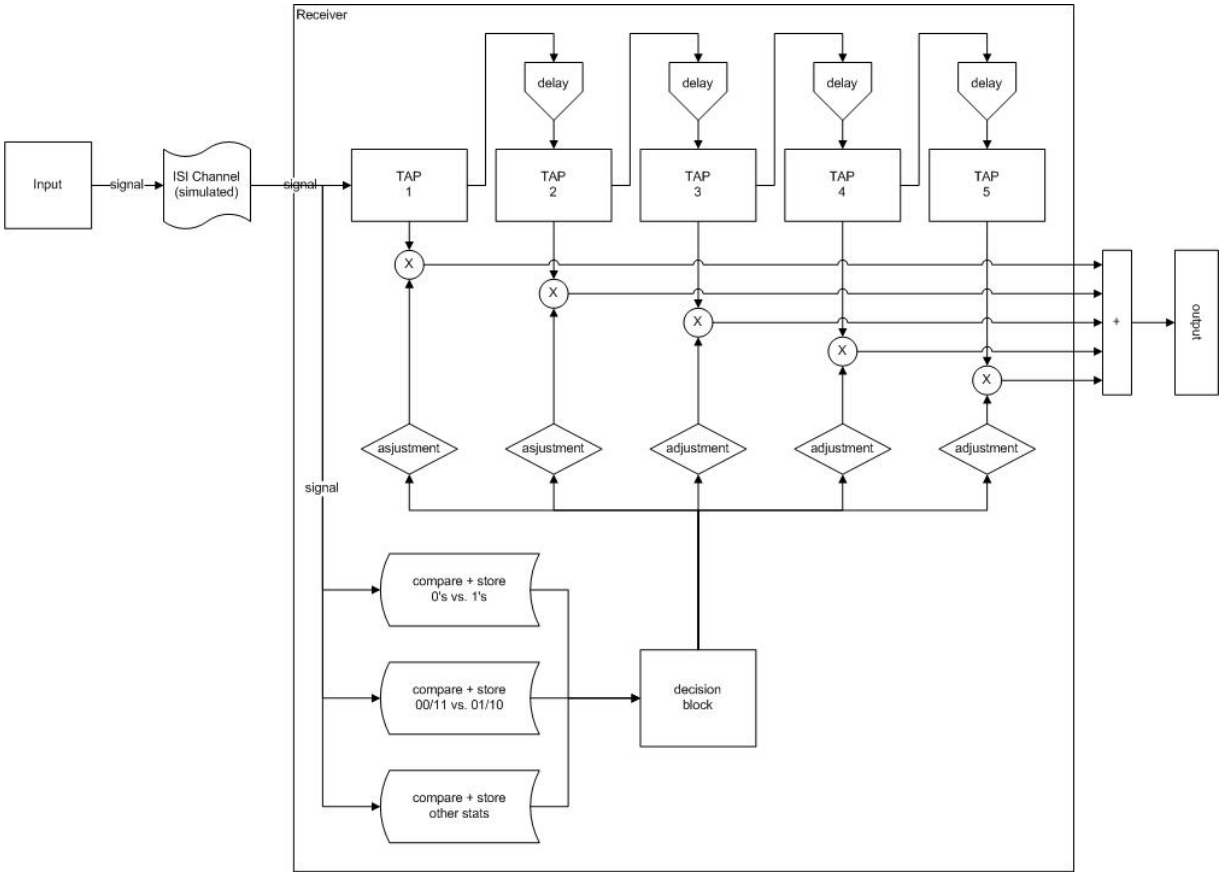
Table 1: Effective quantization bits determined by oversampling factor and order of noise shaping [1]

Quantizer order p	Oversampling factor M				
	4	8	16	32	64
0	1.0	1.5	2.0	2.5	3.0
1	2.2	3.7	5.1	6.6	8.1
2	2.9	5.4	7.9	10.4	12.9
3	3.5	7.0	10.5	14.0	17.5
4	4.1	8.5	13.0	17.5	22.0
5	4.6	10.0	15.5	21.0	26.5

[1] A. Oppenheim, R. Schafer. Discrete-time Signal Processing. Prentice-Hall 2009

Processing

The FPGA will be used to design the DFE as pictured below:



As the signal comes in, the DFE will calculate each tap in turn. Since traditionally a DFE is partly analog and partly digital, our implementation will differ from tradition, since we are using all digital components.

The DAC input is stored into a register. Tap 1, has the special functionality of also finding the "center point," the 50% point where half the signals evaluate as 1 and half evaluate as 0. This point is stored in a special register than will continue adjusting as the training continues. Then we start to calculate tap 1. This process is identical for all other taps, except they have a delay time of 1 from the previous tap. If the ISI is causing the signal high, we push it down, if the ISI is causing the signal low, we give it a boost.

At the same time, we are also accumulating data on the training sequence. We will be using this data to determine adjustments to the taps, so when we are through with training, we can multiple tap value and adjustment and sum to the signal. The output signal is a composite of the adjustments made to each data point at each tap.

Output

The output interface sends the data to the video DAC. The FPGA processing core sends 1-bit signal at 10MHz to the output interface. The core produces data at 10Mbps. This is enough for a 300 pixel by 200 pixel video refreshing at 60 frames per second and 3 bit color per pixel.

300 x 200 pixel x 3 bit /pixel x 60/sec = 10.8 Mbit/sec

We will display 6 different frames of 300x200 video on the same screen.

1. Video based on data equalized by our proposed DFE;

2. Quantization histogram of data equalized by our proposed DFE;
3. Video based on data equalized by a standard 2-tap DFE;
4. Quantization histogram of data equalized by a standard 2-tap DFE;
5. Video based on unequalized data;
6. Quantization histogram of unequalized data.

4. <u>Work Plan</u>

We are planning on first designing the tap module, then connecting it up for a simple two tap classic DFE. We will then expand to five taps, and add additional training compensating circuitry. If time permits we will add an module in the equalizer to compensate for non-liner fiber effect.

Table 1: Division of tasks

Yan	Zhipeng
Tap Compensator Decision block	Sigma-Delta ADC at input interface Video module