Enhancements to Initialization Procedure

Sept ’05 Presentation to Ethernet over Backplane working group
Joe Abler

Requirements for Initialization

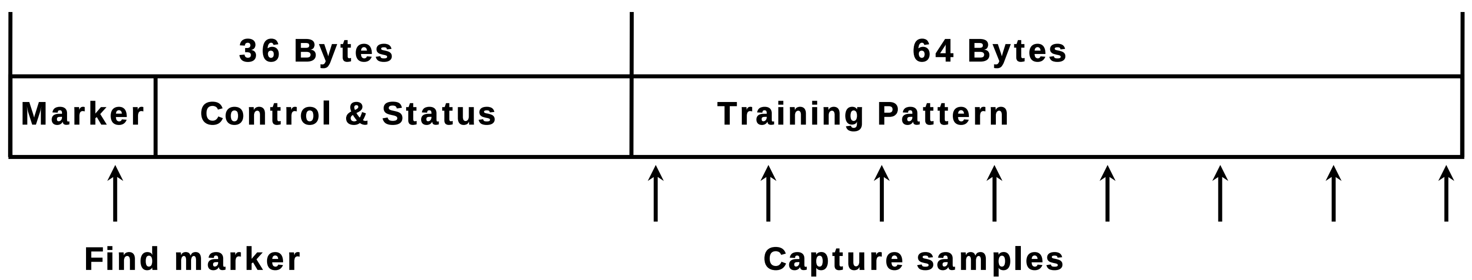
- **Procedure needs to be efficient**
  - Minimize initialization time
  - Minimize complexity, but allow for flexible implementations
    - Hardware, software, or mix
  - Minimize cost (silicon area)
- **Basic requirements of training pattern**
  - Data content should be random
    - Initialize FFE to average or nominal conditions, essentially centering it's capability
    - Swings (pattern extremes) are handled adaptively by DFE
  - Data content should cycle through all possible FFE combinations
    - Spec allows for up to 7 FFE taps
- **Basic procedure**
  - Training pattern is transmitted
  - Receiver monitors incoming data, collects statistics to evaluate coefficient updates
    - Statistics are integrated over a large number of samples, upwards of 1000
  - Receiver sends update coefficient update commands to transmitter
  - Cycle completes until convergence

Problems with current training pattern definition

- **Ratio of control overhead to training pattern is very high**
  - Becomes very inefficient to cycle through and gather sufficient statistics
- **Training pattern has insufficient random content**
  - Definition will actually lend normal implementations to get repeated samples of only a handful of pattern types
  - Impulse patterns will force FFE well off-center, not where desired
- **Training pattern definition drives unnecessary complexity/area**
  - Most reasonable implementation would be to store pattern in a 64 byte memory or register array
    - Could implement a state machine, area usage would still be high to generate the pattern
  - This may be acceptable for traditional Ethernet port interfaces, but this is Ethernet over Backplane….
    - Consider the total cost in a switch chip with upwards of 100 links!

Design implications of current training pattern definition

- **Sample capture will be relatively slow compared to baud rate**
  - Consider a hardware implementation running at 160MHz
    - Currently defined training pattern length would only result in 8 samples per pattern
- **Training pattern is found by identifying the marker position**
  - This implies subsequent iterations of the training cycle will have the same sample capture points
- **Capturing 1000 samples would require 125 iterations of the training cycle**
  - However the statistics will contain info from only 8 distinct patterns!!
  - 10us elapsed time, but 36% of it spent in overhead
- **Software solution would not be realistic with such a definition**
  - Samples taken only have a 64% likelihood of landing in training pattern period

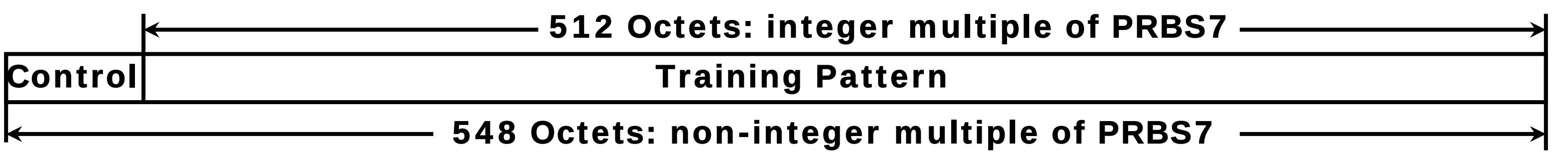


Proposed changes to training frame structure

- **Change training pattern to a PRBS7 generated stream (entire pattern field)**
  - Provides randomization
  - Provides a pattern which will cycle through all FFE pattern combinations
  - Simplifies implementation
    - Most serdes already incorporate this pattern generator
- ~~**Do not reseed the PRBS generator during each iteration**~~
  - ~~Provides randomization across iterations, eliminating the common sample point problem~~
- **Lengthen the training pattern period**
  - Suggest length of 512 octets
  - Increases efficiency of training relative to overhead
  - Simplifies sampling approach – receiver can “train through” overhead section
    - No need to search for markers, start, and stop
    - Enables software approaches
    - Small number of samples taken in overhead region are simply integrated into the total
  - Capturing 1000 samples would take 6.4us
    - vs. 10us for current approach

Implications of continuous PRBS7 training pattern

- **Receiver PRBS checker may have difficulty synchronizing to pattern**
  - This is not a problem per se, checker synchronization is not necessary to evaluate coefficient update commands
- **However, to allow the option of a receiver synchronizing to PRBS, the recommended approach is to reseed the generator each iteration**
  - Reseeding the generator allows receiver to lock to marker and synchronize to pattern each iteration
    - Statistics sampler should include logic to vary start of statistic gathering in order to obtain random samples
    - This is more reasonable given the increased pattern period
  - An algorithm not requiring the receiver to synchronize to the pattern can “train through” the control period
    - Since Control + Training period is a non-integer multiple of the PRBS7 period, the PRBS pattern will “slide across” a fixed period sample and provide random samples



Implications of proposed training format changes

- **Handshake process for coefficient updates potentially takes longer?**
  - Increased time is not necessarily a problem, and could be eliminated
    - Capturing 1000 samples would still require about 15 iterations with a hardware implementation, with 15 corresponding overhead fields
    - Turnaround time of updated request to update status is still a tolerable percentage (about 12% of total)
  - Additional round trip time for handshake is about 1us
    - More than offset by the 3.6us gained in capturing samples
  - Updates are now likely to occur in a single cycle
    - Additional time allows for processing within a single cycle
    - Therefore it's not an absolute increase in handshake time
- **This could be further optimized by training through the handshake**
  - It's not critical that the receiver know when the Tx has been updated to begin acquiring new samples
    - Integrating a few "old" samples in with the new samples is not a problem
    - An implementation could use a simple timer delay after update request if it wanted to eliminate or minimize the number of "old" samples

Recommended Changes:

- **Increase training pattern length to 512 Octets**
- **Change training pattern to PRBS7**
  - 1 seed at start of each iteration