USB 1.1 Hub Implementation

Rudy Sorensen

April 10, 2025

1 Abstract

2 Design

2.1 Clock and Data Recovery

2.1.1 Design Discussion

As per the specification of USB 1.1, a hub must support both full-speed and low-speed transmissions. These correspond to data rates of 12 Mbps and 1.5 Mbps respectively. Since USB is an asynchronous protocol, the clock must be recovered through the data. The general approach for this is to use a PLL because the SYNC portion of a packet usually guarantees that the PLL will be able to lock onto the frequency prior to the actual data transmission.

Alternatively, oversampling coupled with a moving average calculation can be used as well. Typically this is done by oversampling the data at a rate of 8 or 16 times the original frequency to minimize sampling errors caused by jitter or long switching times; however I will be oversampling at a rate of only 4 times the original frequency due to hardware limitations. This means that the hub clock will be running at 48 MHz. Assuming that the USB Host and devices I interact with adhere to the specification, this design decision is safe. This is proven by the following information:

- Data rise and fall times are restricted between 4 ns and 20 ns.
- Host clock tolerance is $\pm$ 0.005% $\Rightarrow f_{min} = 11.994$ MHz and $f_{max} = 12.006$ MHz
- So, $p_{min} =$ 83.292 ns and $p_{max} =$ 83.375 ns
- So in the worst case, being the longest rise/fall time and shortest period. We would only have 63.292 ns of stable data
- In the worst case of sampling, we would have sampled right before the stable period. This is time 0. Then our first accurate sample would be at time 20.833, the second at 41.666, and the third at 62.499. Since $62.499 < 63.292$, we have at least 3 accurate samples even in the worst possible case if we assume that a hold time violation does not occur
- If the hub clock frequency were to drop down to 47.4 MHz, then the third sample would be missed. Leaving us with only two accurate samples. So it must be verified that the hub's clock frequency will not drop to this value

With that said, we can now run off the assumption that we will always receive at least 3 accurate samples per bit period when sampling at 48 MHz; in other cases we may have 4 accurate samples. Using a moving average calculation, this means that we will always be able to accurately determine the desired value.

Also, due to the host clock variation and packet lengths (up to 1023 bytes for a DATA packet), we must be resynchronizing our clock on edges whenever possible. Thankfully, the USB specification provides ample edges to resynchronize on due to the way in which data is encoded. In the worst case, we will only ever go 6 bit periods without an edge because of mandatory bit-stuffing.

To summarize:

- The hub clock will run at 48 MHz to achieve a 4x oversampling rate
- This results in 3 to 4 accurate samples per bit period
- Data will be determined through a moving average calculation
- Due to clock variation, the sampling counter must be resynchronized on edges; edges occur at least every 6 bit periods

2.1.2 Verification

Assumptions

- The serial_in signal is asynchronous. This is modeled by creating a different clock with the same period as the hub clock, but without any phase restrictions
- It is possible for the same value to be sampled 3, 4, or 5 times due to setup and hold time violations, this will be modeled by restricting the serial signal to always be stable for 3, 4, or 5 hub clock cycles
  - This is done by randomizing the first sampled value if the previous value and current value differ, i.e. the value will always be stable for 3 cycles but may be stable for 4. Since the next bit period can model this same error, it is possible for the value to extend into the next bit period, making it be stable for 5 clock cycles. Cover properties are used to ensure that all of these situations occur

Assertions

- Anytime valid is asserted, the sampled value should be equal to the last value used to drive serial_in