Ultra Low Power Bus v1 to v2 Changes Document

Now with 1/4 the clocking!

(But still in need of a better name...)

Contents

1 Introduction

This document is an exploration of an idea, it assumes you are familiar with ULPB v1.0.

2 Changes

Loosely ordered as a series of changes from v1.0.

2.1 Double Data Rate

We dismissed this when we concluded that latching on both the positive and negative edges was not feasible in a general sense. What we neglected, however, was so long as we could design a system where each flop latched only on the positive edge **or** the negative edge (never both), then the option is back on the table (since what is latching on negative clock edges but an inverter and a clock that is 180° out of phase?).

The insight is that there is no need for a DRIVE1 or DRIVE2 *state*. The only transition that occurs on DRIVE* edges is a new bit being placed onto the bus. Logic and decisions are all consequences of the LATCH* states. If you control and clock the MUX that drives `DOUT` with the 0° clock, but feed the data entry to the MUX with a FIFO clocked by a 180° clock, the effect is the value on the data line changing on the negative edge of clock.

This change ignores details of arbitration, termination, acknowledgment, and reset for now, it is an optimization meant for during data transmission.

2.2 Removing the double-latching

The motivation for a separate LATCH1 and LATCH2 was to (in a purely digital manner) allow for arbitrary reset events (interruption / preemption) and to very cleanly distinguish Reset, Stop, and Acknowledge sequences from the data stream. But there is another way…

In this version, both the clock and the data are connected in a buffered loop (yes this does cause some issues for injection, TBD). When any node wishes to interrupt a transmission—where interrupt is defined as both ending a transmission (stop bit) or forcing an abort—it stops forwarding the `CLK` signal. The control layer will detect that its `CLK` input has not gone low at the next `CLK` output. In response, the control layer will start toggling the *data* lines. Each member node will have a small counter that is clocked from *data* and asynchronously reset by `CLK`. During normal transmission, the counter should never be able to count higher

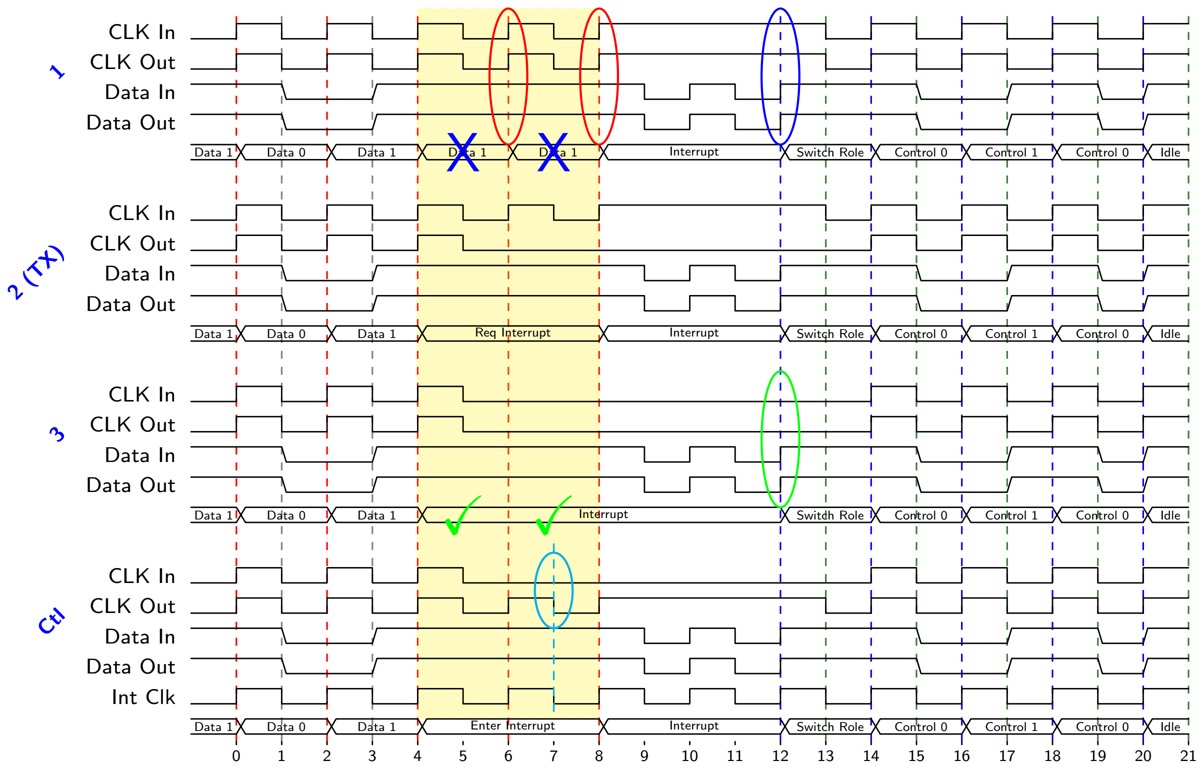

Figure 1: Detail of the Interrupt entry procedure. In this example, Node 2 is the TX node, thus when it elects to enter Interrupt it is already forwarding its data lines. Node 2 decides to enter Interrupt at time 4. At time 6 Node 2 suppresses the clock edge, requesting Interrupt. The control layer detects this at time 7[1]. At this point the control node begins toggling the *data* lines to act as a clock. After two data pulses, all nodes will enter Interrupt. A few control bits are sent, after which the bus returns to Idle.

than 1 as the clock is constantly resetting it. If the counter reaches 2 then the clock and data loops *switch roles*.

Figure 1 demonstrates an entry into interrupt. Note the careful subtelty depending where a node is physically located. At time 6 Node 1 will record an extra data bit that Node 3 will not record. This is resolved at time 10 when Interrupt is entered. When entering Interrupt, a node must sample its `CLK` line. If `CLK` is high, the node should discard the most recently recorded bit. The control node must wait until the third data edge (time 12) to begin forwarding the `CLK` line to ensure a stable value is sampled for the possible bit-deletion. The control node should rely on the incoming `CLK` to latch data bits, otherwise it would also have to discard a bit recorded at time 6.

Arbitration of interrupt entry acts as other arbitration has before. Considering Figure 1, had Node 3 decided at the same time as Node 2 (time 4) that it wished to interrupt it would never see another `CLK` edge before the data pulses arrived, causing the node to lose arbitration.

[1]This detection must occur on the falling edge since the control layer requires an egde to sample on and it is sampling its `CLK_IN` pin detecting when it is not high. A simple detector can sample this every edge asserting a signal which is sampled by the controller's FSM at time 8, transitioning the controller into interrupt mode.