DESIGNING A ROBUST USB SERIAL INTERFACE ENGINE(SIE)

What is the SIE?

A typical function USB hardware interface is shown in Fig. 1.

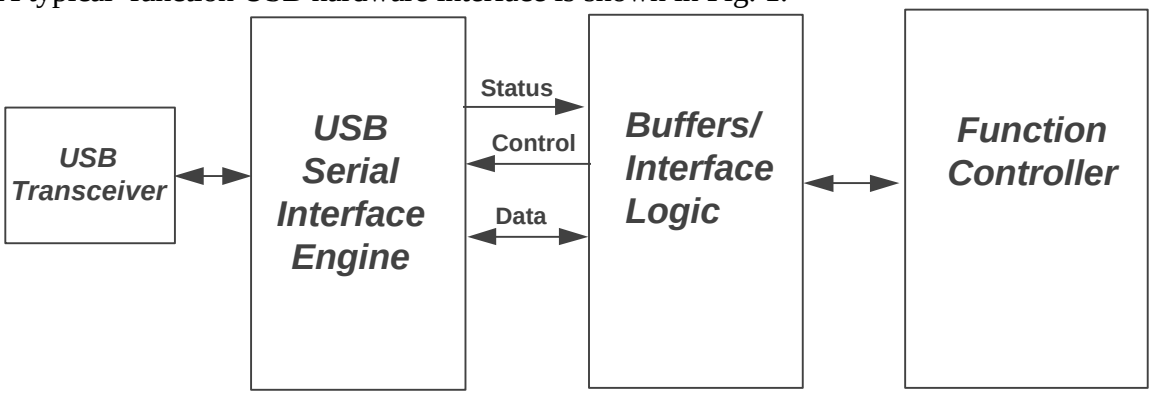

Fig. 1: USB Function hardware interface

The SIE is the frontend of this hardware and handles most of the protocol described in chapter 8 of the USB specification. The SIE typically comprehends signaling upto the transaction level. The functions that it handles could include:

- Packet recognition, transaction sequencing
- SOP, EOP, RESET, RESUME signal detection/generation
- Clock/Data separation
- NRZI Data encoding/decoding and bit-stuffing
- CRC generation and checking (Token and Data)
- Packet ID (PID) generation and checking/decoding
- Serial-Parallel/ Parallel-Serial Conversion

A typical implementation of an SIE with these functions takes about 2500 gates. So the module itself is fairly small; and the functionality is straightforward. In spite of this apparent simplicity, it is possible to end up with a design that doesn't work reliably i.e. a design which is not robust. This paper will point out the reasons for this and describe techniques that will eliminate these problems from a design.

Sources of robustness problems

The primary source of robustness problems is the existence of multiple clock domains in the SIE, some of which are asynchronous to each other. If signaling between these domains doesn't adhere to synchronization rules, intermittent problems can result. These problems are invariably difficult to track down and fix.

Other areas which have the potential for robustness problems include:

- out-of-band signal handling on per-packet basis
- bit stuffing/unstuffing
- special casing for setup, iso etc
- special casing for low speed
- suspend /resume support

The following sections will review each of these areas in turn and discuss techniques to address the problems.

Multiple clock domains

The typical SIE has to deal with four clock zones in three domains:

- USB host 12Mhz clock or receive clock
- internal 4x clock (48Mhz) and transmit clock (divided by 4 version)
- SIE backside clock or interface clock

Each of these domains is discussed in turn in the following sections.

DPLL and receive clock

The 12Mhz clocks in the host/hub and the function are aysnchronous since they are derived from different crystals which can have a tolerance of +/-0.25%. So bitwise synchronization is needed and achieved with the help of a PLL. A typical implementation would use a digital PLL with 4x oversampling to derive the received clock.. At any bit, the derived clock period can be +/-25% of the nominal period as explained below.

A typical DPLL state machine is shown in Fig. 2 and runs on the 48Mhz clock.

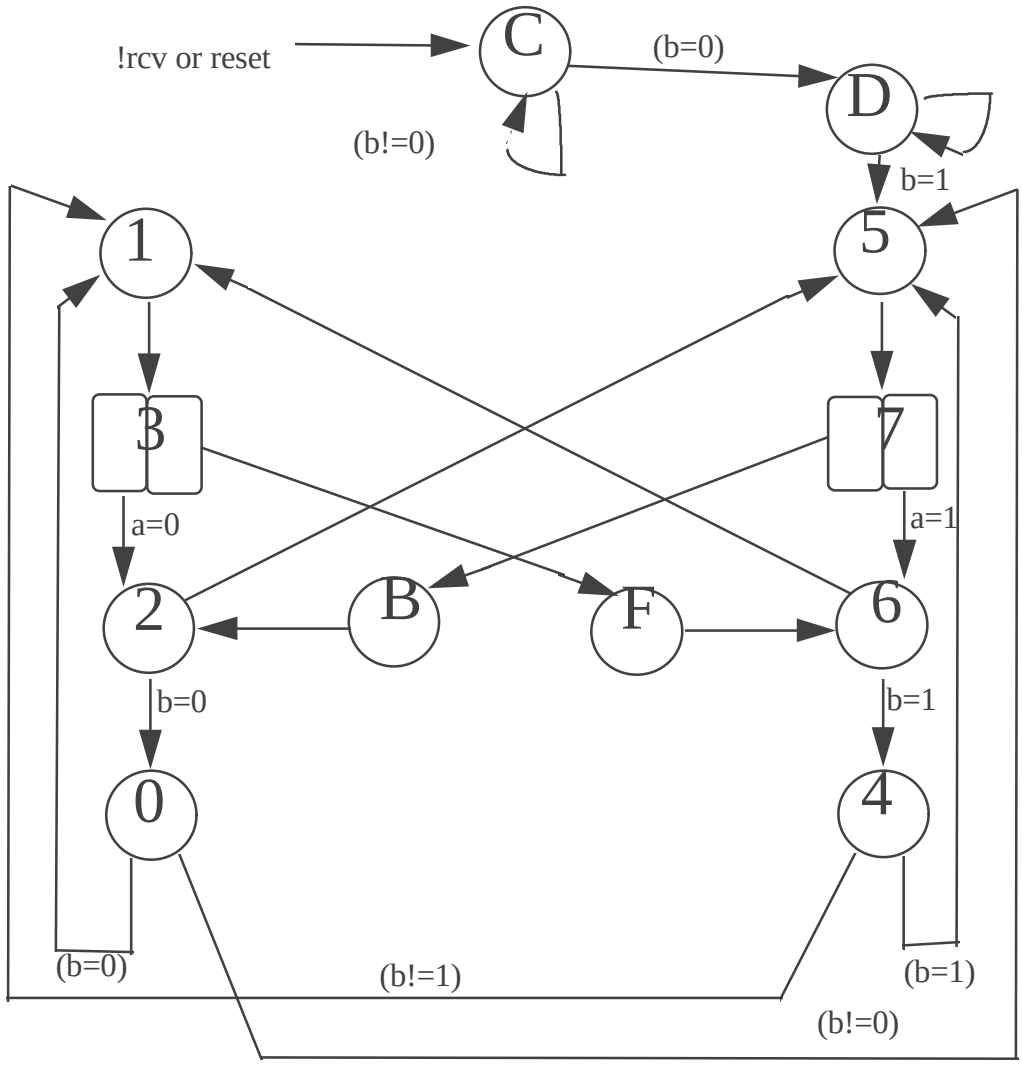

Fig. 2: Example DPLL implementation

‘a’ and ‘b’ are the differential receiver output synchronized by a stage of the 48Mhz (also ‘a’ is synchronized on the rising edge and ‘b’ is synchronized on the falling edge.

In this state machine, states C and D are used to lock to the incoming bitstream using the initial transitions (edges); after lock is achieved the DPLL circulates in either the right half (when bit stream is a ‘1’) or in the left half (when bitstream is a ‘0’). The nominal loop is through the 4 states in a vertical line (5 7 6 4 or 1 3 2 0). In this nominal loop, the inner two states generate the clock high period and the outer states generate the clock low period for the receive 12Mhz clock. These states are chosen to ensure adequate setup and hold time for clocking the extracted data. Transitions in the bit stream result in switching between the loops; if there is no change in bit width (perfectly synchronous clocks), the transitions will be from states 0 and 4. If the bit width is shorter (host is

running at a higher frequency), the transition will be from states 2 and 6. So the low period of the derived clock is shortened by one 4x period for the short bit. A long bit width will cause detours from states 3 and 7 through states F and B before going to states 6 and 2 respectively; this will add an extra 4x high period to the derived clock.

With the state encoding shown, bit 2 of the state (encoded as bits 3..0) represents the data; a glitch-free receive clock can be obtained by running the derived clock through a flip-flop clocked by the 4x clock. Further qualification and use of the received bitstream (e.g. sync detect) should be based on this extracted clock on the extracted data.

A corollary of the way this DPLL works is that if there are not enough edges (at least 2) received, lock cannot be achieved (i.e. DPLL will not progress past the D state) and no clock can be derived. This can happen for example when resume signaling is received by the awake function. It could also arise if the differential receiver generates an edge when it sees the transition from idle to single-ended 0 signaling during reset.

Race Conditions in the transmit domain

The clock zones in the second domain are synchronous; however race conditions could occur in signaling between the 1x and 4x sub domains because the 1x clock is derived from the 4x clock. This may be a bigger problem in some target technologies than in others. The problem is exacerbated by the need to switch the hardware between transmitter and receiver clocks.

Since the USB is half duplex several of the modules in the SIE can be shared between transmit and receive e.g. the crc logic . Since every USB transaction includes receive and transmit phases, the state machines carry state between the phases. So there is a need for a means to reliably multiplex between receive clock and transmit clock. A state machine is shown in Figure 3. It switches btween the receive clock (dclk from the dpll) in state 0 and the transmit clock (clk12s) in state 3. The output clock is 0'in the other states. This mechanism provides a glitch free clock which has the low period extended during the transaction phase transitions. The rcv signal indicates that the SIE is in receive mode. When generating this signal it is important to ensure that the reflection of a transmitted EOP does not get seen as the start of a new packet and cause the mux to switch over to receive (i.e. to the dpll clock which wont be generated because there are not enough edges for the dpll to lock on).

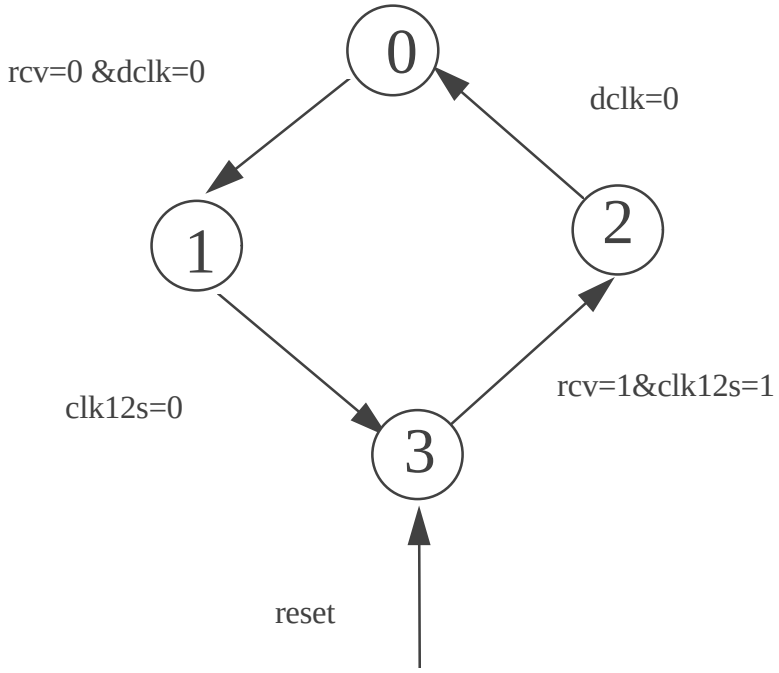

Fig 3: Switching between transmit and receive clocks

All synchronous designs will have some skew between the different branches of the clock tree (shown as skew1 in Figure 4). The addition of logic to generate the clocks and multiplex the clocks in the SIE creates additional skew between the nominally synchronous clock domains. The timing paths of signals which go between these domains should be analyzed closely to avoid race conditions and consequent anomalous behavior. The End Of Packet (EOP) is one such signal which is derived in the 4x domain (as described later) and used by the state machines in the 1x domain.

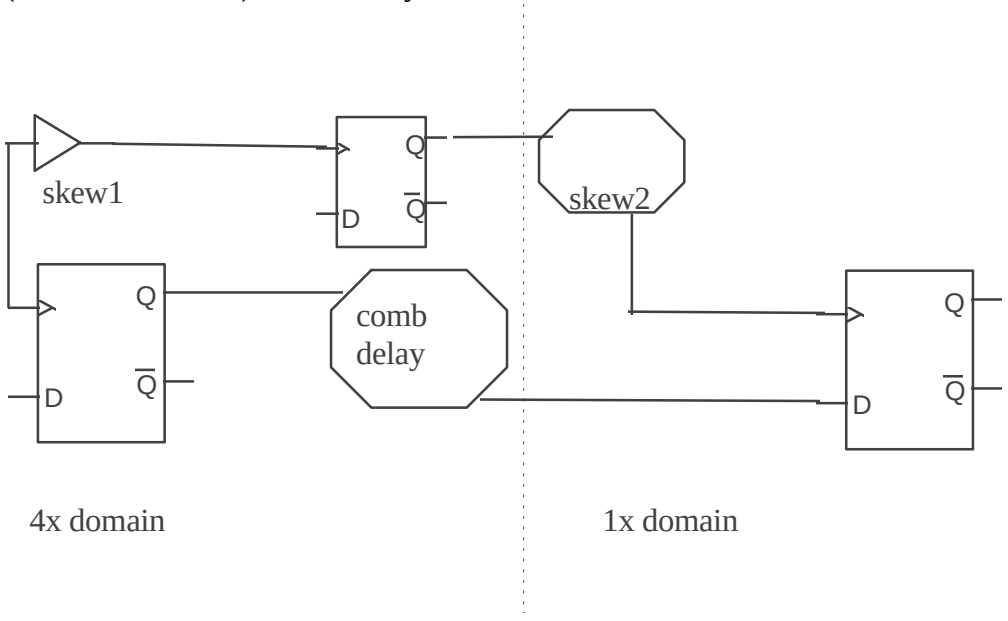

Fig. 4: Inter-domain signaling between nominally synchronous domains

The backside clock

The SIE backside could be running on a different clock than the SIE; since the data transfer interface is byte wide, standard four phase closed loop signaling with handshake signal synchronization can be used for the data transfers. Fig. 5 shows the transfer of received data bytes from the SIE to the backside using READY signal from the SIE and the RD signal handshake from the backside.

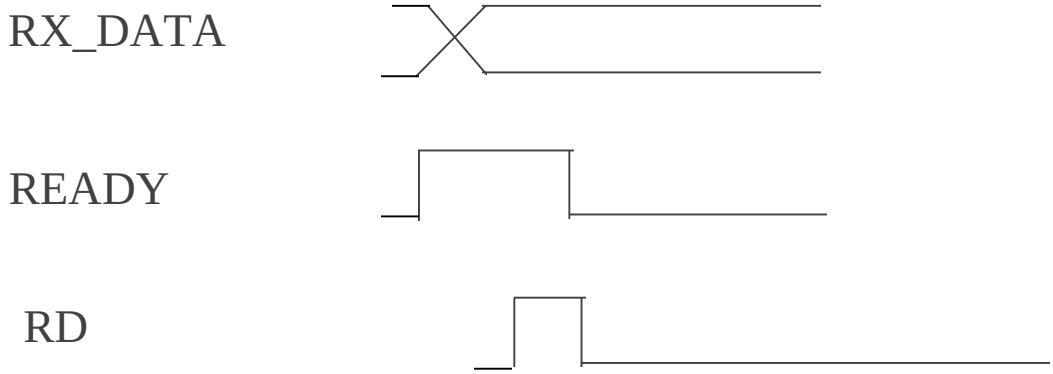

Fig 5: Rx_Data transfer

Fig. 6 below shows the transfer of transmit data from the backside to the SIE using READY from the SIE and WR handshake from the backside.

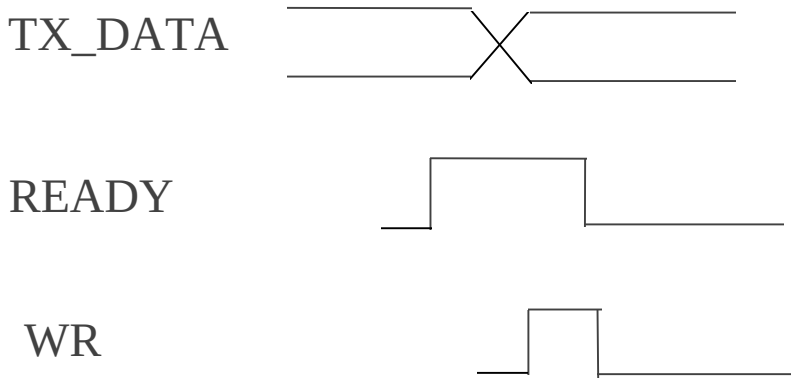

Fig 6: Tx_Data transfer

Other control signals from the backside to the SIE must be synchronized as well; this is typically done with the fastest clock which is the 4x clock in the SIE. The resulting signals will be in the 4x domain and since they are used in the 1x domain, they should be analyzed for race conditions as described in the previous section.

Packet delimiters and out of band signaling

Precise detection of packet delimiters is crucial for robust SIE operation. Each packet has a start delimiter (or sync) and end delimiter (or EOP). The nominal sync field consists of an NRZI KJKJKJKK pattern. Even though this is an in band (made up of differential signals) pattern, the initial bit may be distorted due to hub turn on behavior ; as noted previously, the DPLL may need some edges to achieve lock as well. An easy way to account for this is to use only the latter portion of the sync field to detect sync. This works because only the end of sync is needed to delimit the start of information in a packet.

The sync detection can be done prior to data extraction in the DPLL. The detected end of sync will then need to be synchronized with the data stream extracted by the DPLL. It is easier and recommended to look for the end of sync in the data stream extracted by the DPLL. The pattern being searched for should comprehend that the data output from the DPLL during interpacket gap and unlocked states are not to be used when detecting the sync pattern.

The EOP detection is more difficult because an out of band (single-ended 0 or se0) value needs to be detected and timed followed by an in band signal. The se0 should be assumed to be asynchronous to the inband signaling. This asynchronism could be due to a combination of hub behavior, single ended receiver behavior and differential receiver/DPLL behavior . The signaling passes through several stages of receivers and transmitters between the host and the function. This produces skew between the inband signaling and the out of band signaling. The in band <-> out of band transitions could also result in differential receiver output edges which can affect clock extracted by the DPLL. The USB spec requires full speed USB agents to accept se0 widths greater than or equal to 82 ns and to reject widths of less than 40 ns. There are similar requirements for low speed devices. The se0 of the EOP also needs to be terminated with a J transition.

One way to meet all these requirements for EOP detection is to use a state machine running in the 4x domain as shown in Fig. 7.

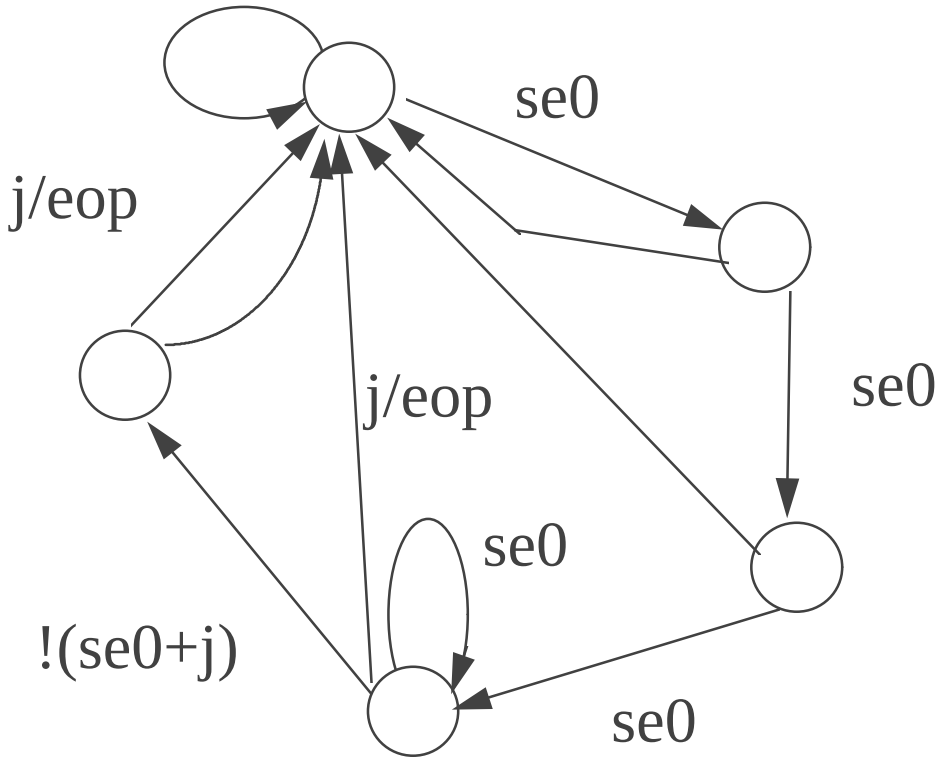

Fig. 7: eop detection state machine

This EOP is detected in the 4x domain; however it is used by the state machines in the 1x domain. This interdomain signaling should be analyzed carefully for race conditions as described in a previous section. Note that the EOP and the J detection are asynchronous; this is accounted for in the dead state on the extreme left. There is a similar asynchronous zone between the inband signaling at the end of the packet and the out of band se0 of the EOP which follows it. This may manifest itself as an extra bit at the end of the packet and so is termed dribble; this must be accounted for in the state machines when delimiting the end of packet. These state machines should also take into account the fact that the possible dribble may follow a bit stuff at the end of a packet.

Other design subtleties

Bit stuffing and unstuffing can be implemented by putting the state machines and datapath on hold while stuffing or stripping the extra bit. Bit unstuffing near the EOP needs to be handled carefully as explained above.

Although most transactions are three phase, ISO transactions are only two phase and the state machines need to comprehend this. Similarly SETUP transactions are identical to OUT transactions except that they cannot be NAKed or STALLed. The data buffering and the state machines need to take this into account. Data toggle sequencing logic at a bidirectional endpoint should take into account the specific requirements for the starting toggle sequence of each stage of a control transfer.

Low speed signaling is identical to full speed signaling except for the inversion of polarity. But low speed devices need to comprehend that while most data entities are defined in terms of number of bits, the se0 width for reset is not. Low speed devices should also be able to handle keep-alive signals (bare EOPs) correctly.

SIE designers should also recognize that the DPLL may not be able to generate a clock during resume and reset signaling even though it is nominally in receive. This is because the DPLL needs at least two good edges before it can lock and start generating the clock as explained earlier.

Devices which implement remote wakeup should also have a means to time the period of driving resume in the absence of the regular device clock. They should comprehend that host wakeup events can occur at any time relative to the remote wakeup event. The device also needs to maintain state information like address, interface setting etc. when suspended.

Detection of USB reset should reset the SIE as well as the rest of the device; this will reset the counter in the SIE which is timing the se0. This could result in a train of reset pulses output from the SIE in response to a single long USB reset pulse from the host. The device should work correctly in this case.

Conclusions

Asynchronous clock domains inherent in any interconnect technology (including USB) makes the SIE design more challenging than may appear from an initial reading of the spec. The use of a DPLL and a proven clock switching mechanism with careful analysis of timing paths can help achieve a robust design here. The need to switch between in band and out of band signaling to delimit the end of packet requires the need for dead states and dribble bits to be handled in the state machines. Other subtleties in the spec should also be kept in mind when designing an SIE. The protocol compliance checklist can be very helpful in identifying areas which lack robustness. SIE cores are available from various vendors; these can be of help to device designers in obtaining a robust design. Bus functional models and other tools should provide a means to stress the robustness areas in the SIE design.