Splitting and arbiting a bidirectional serial bus

Jeff Anderson - March 18, 2016

Introduction to Bidirectional Buses

Bidirectional buses (e.g., I²C, SMBus, and LIN) have become ubiquitous in today’s electronics due in part to their simplicity. Using only two wires – data and clock – multiple devices can communicate with each other. According to the I²C bus specification, up to 128 devices can share the same data and clock lines; this being enabled by the use of external pullup resistors, and open-drain drivers on each device. If none of the devices are transmitting a 0, the bus is naturally pulled to 1 by the pullup resistor. However, any of the devices on the bus can pull it to 0.

Bus masters are allowed to drive the bus at any time, and slaves respond to the bus master’s query within a certain time period after the query has been received. In multi-master scenarios, individual devices acting as bus masters are required to perform their own bus arbitration. The bus master that wishes to assert control over the bus is required to test it by pulling it to 0; this informs the other masters that the bus is being used.

Why split a bidirectional bus?

The I²C bus specification[1] contains a reference circuit which allows it to be split into input and output pairs. This configuration is needed for several reasons. First, the split-bus can be used to optically isolate the bus master and slaves in cases of safety, noise, and grounding issues (Figure 1). Additionally, performance gains can be realized by amplifying the split-bus (Figure 2) or replacing the amplifier with a circuit that performs media conversion. This can increase the operating distance of the bus and increase performance by reducing capacitance. As capacitance goes down, the signal time constants, t, become dominated by the pullup resistors.
For those designing bidirectional bus controllers, the bus splitting technique can be used for debug. Typical debug of bidirectional protocols is made difficult because a misbehaving controller may pull the bus to 0 while another controller is asserting control of the bus. This situation makes identification of the transmitting device on the bus impossible without knowledge of the internal states of each device’s controller. However, monitoring the /gateB1 and /gateA2 lines from Figure 2
enables the identification of both the transmitting devices and any simultaneous assertions of the
bus using only standard lab equipment and debug techniques.

Lastly, the bus splitting technique can be used to connect an I²C supported device to another device
that does not have an I²C controller. In this case, the split bus can be connected to the GPIOs of the
other device (Figure 3).

There are several published circuits which enable splitting of a bidirectional bus. Unfortunately, the
reference circuits for split-bus applications either require specially designed circuits for a particular
application (as seen in published articles), or require external control logic (as seen in the I²C standard)
that uses transmission gates to allow the sender and receiver to communicate without creating a feedback path that causes
latchup. The latchup condition is evident in Figure 2, where IOA pulling the
bus to 0 forces IOB to pull to 0 though /gateB1, which then forces IOA to get pulled to 0 indefinitely
through /gateA2.

The bilateral arbiter presented in this Design Idea can split a bidirectional bus into transmit and
receive pairs, and is constructed in a generic way that allows it to be used for any split-bus
application. Additionally, it requires no external control logic - the bus is controlled solely by the
state of the data bus:
Bilateral Arbitration

The arbiter shown in Figure 5, comprised of cross-coupled enabling circuits from Figure 4, works because bidirectional buses, by definition, only support half-duplex communications. In a steady state, the DATA bus is pulled high by pullup resistors R1 and R2, which forces OUT1 and OUT2 to 0. This keeps both NMOS FETs in the cutoff state. When IC1 pulls DATA low, OUT1 goes to 1, which makes Q2 pull down the DATA bus of IC2. Simultaneously, OUT1 is fed to an input of NOR gate U2, breaking the feedback loop from OUT2 back to Q1. This breaking of the feedback loop removes the latchup condition, making any other control logic unnecessary, since the first circuit to assert its data line wins the race and blocks the other circuit via the NOR gate.

Figure 4 Bilateral arbiter enabling circuitry

Figure 5 Bilateral arbitration of a bidirectional bus
Figure 6 Bilateral arbitration enabling bus amplification

Due to the generic nature of the circuit, bilateral arbitration can be applied to any split-bus application. Figure 6 shows bilateral arbitration applied to amplification. This can easily be extended to media conversion and bus isolation, by replacing the amplifier component with either a media conversion circuit or an optoisolator, respectively. For bus controller debug purposes, the lines in between the amplifiers can be monitored to help identify malfunctioning bus controllers. Figure 7 shows bilateral arbitration being applied to an I²C-to-GPIO connection.

Figure 7 I²C to GPIO connection

Conclusions

There are several reasons to split a bidirectional bus into transmit/receive pairs. From increased performance to enabling debug to isolation, splitting of the bus is a technique that many designers will find useful at one time or another.

Bilateral arbitration is a bus arbitration technique that enables most bus splitting applications. It relies on the behavior of bidirectional bus communication to eliminate the need for external control.
circuitry, and is generic enough to be used for many applications without adding unneeded complexity.

References

3. Thoren, M., “Method provides fast, glitch-free isolation of I2C and SMBus signals”, EDN, June 2004