Overview and Comparison of SPI, I2C, and UART

UAV LiDAR Mapping

By Joseph Bessette-Denwood, ECE '19

Introduction

Chip to chip protocols are the methods of communication that different computer chips use to talk to each other. The priorities of these protocols typically fall into one of three categories: cost, speed, and reliability. Reliability refers to how low of a probability of failure the system has. Speed is typically measured in two ways: baud rate and bits per second. Baud rate is symbols per second and bits per second (bps) refers to the effective output in number of bits transmitted per second. This is generally measured in kilobits (kbps) or megabits (mbps) per second for the protocols that are being discussed here, which refers to 8000 (220) bps or 1000000 (230) bps respectively. Cost can refer to a few different things, such as: the literal cost of additional development to implement it, or the power cost of the system. This paper will be focusing on the power costs, as the power costs have become more important as battery technology has stagnated, resulting in power savings becoming more important.

I2C

I2C (Typically said I-Two-C or I-squared-C) stands for Inter-Integrated Circuit. It is a Client-Server protocol, with two lines, a serial data line (SDL), and a serial clock line (SCL)[7]. There is no maximum size to a message. This protocol allows for multiple clients and multiple servers, the number depending on the addressing scheme. Additionally, a node may switch roles between messages. There are four potential modes of operation on SDL: client transmit, client receive, server transmit, and server receive8. A client will start in client transmit mode, and will start by sending a START signal. It will then send the address field that the message is targeted to. After that, a single bit will signal either read 1 or write 0 from the client to the server. This is followed up by the data, and finally a STOP signal will end the message. Additionally, for every eight data bits sent, the sender will wait for a single return bit. If this is a 0, the sender will continue, if the line is a 1 it will stop, for either the receiver doesn't exist or it is in error[9].

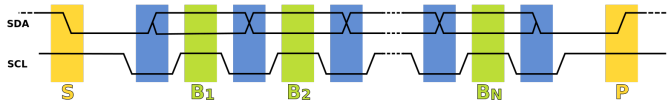

Figure 1. Image of I2C protocol timing diagram

The START and STOP signals are based on the clock signal. I2C uses the clock line (SDL) to control the line, while using the data line (SDA) to send data. The devices use a logic zero on SCL to tell the system that SDA has been changed. When SCL holds logic one, SDA should stay constant. This is only violated for the STOP and START signals. During the START signal, the SCL line is at one, and SDA transfers from one to zero. For the STOP signal, the SCL line is at one, and SDA transfers from zero to one[9].

The wires are pull-up, so multiple nodes can write to the line at the same time, and if any node is driving the line to zero, the line will go to zero. When this is done on SDA, it is called arbitration, and when done on SCL, it is called clock stretching. Clock stretching allows the server to control the speed of data transfer, allowing it to slow down the data coming in if it needs to[9]. This is possible because the lines are active low, meaning that in order for SCL to return high during data transfer, no line can be driving it low. Therefore, if a server is driving it high, the line will stay high, and the client will wait to send the next bit[7]. Arbitration is the behavior of a node attempting to drive a one and seeing the line as a zero, causing it to assume that there is another node transmitting and therefore stop transmitting.

UART

UART stands for universal asynchronous receiver-

transmitter, and is also commonly known as serial. UART has three modes[11]. The first is simplex, which only allows one device to communicate to the other. The second is full duplex, which uses two lines, allowing both nodes to be sending and receiving at the same time. The final is semi-duplex, which uses a single line, with the two nodes switching being sender or receiver. The line has an idle of high, with a low bit signaling a start of a message. Depending on the code set that is being used, there are five to ten bits following this, followed by a high bit that signals the message has ended[11].

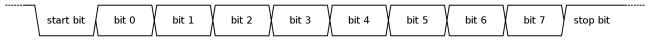

Figure 2. UART Protocol Packet

As this protocol is asynchronous, the transmitter has a relatively simple task. The programmer decides a bitrate, and the transmitter simply has to transmit signals that are long enough for that bitrate11. The receiver waits for a low signal at least half as long as the bit rate, and uses that to synchronize its clock. It then feeds the bits at every expected location based on the clock into a shift register, before using or presenting the data once the full set of bits has been sent11. The protocol is extremely simple, but this limits the maximum speed and limits what the system can do.

SPI

SPI stands for Serial Peripheral Interface. It is a single client, multiple server protocol, which uses four lines for communication. The four lines are: SCLK (Serial clock), MOSI (Client Output Server Input), MISO (Client Input Server Output), and SS (Server select)[6]. The clock is controlled by the client, and is used as the control for when data is sent. The client selects a server using the server select lines, writing it to zero. Then, during every clock cycle, the client uses the MOSI line to transmit one bit to the server, and the server uses the MISO line to transmit one bit to the client[7]. When the client is done with the communication, it will stop using the clock line, and deselect the server.

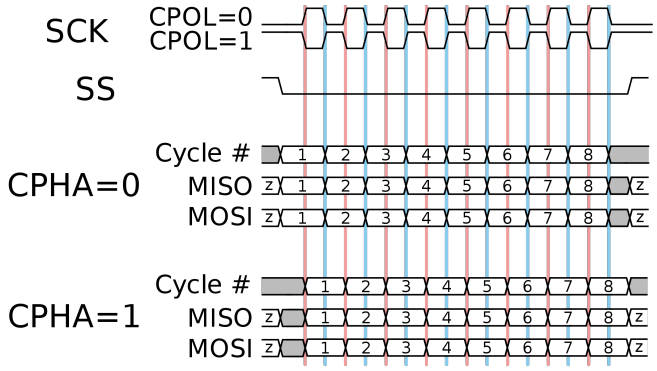

Figure 3. SPI Protocol Packet

Conclusion

While this tech note describes three protocols, there are a huge number of protocols that exist, each with their benefits and disadvantages, each with its benefits and downsides. I2C allows for numerous nodes on its system, and for those nodes to change from client to Server very easily, during operation. However, it has limited data transfer, with a maximum of 3.4 Mbps in recent revisions, which can be an issue in situations where the requirement is higher1. UART is a system that depends on the maximum accuracy of the clocks involved, as well as the maximum sampling rate of the receiver. Therefore, it has no maximum data rate, with systems upwards of 35 Mbps existing. However, UART is also the most limited system as far as number of servers and clients, as per line it allows only one transmitter and one receiver, and relying on the clocks onboard to be accurate enough to detect the signals. Finally, SPI also has not specified upper limit, as it is based on the maximum clock speed that can be sent through the SCLK line1. It also allows for multiple servers, but cannot elegantly handle an arbitrary number of servers, as it either has to increase the number of control lines linearly, or increase the delay on getting the data it needs linearly.

Choosing the correct protocol depends on what your needs are. In some cases, you may want something simple, something that won't produce secondary issues, and don't need a high bandwidth, you would choose UART. In others, you may need a network of chips to send small messages to each other, but again, don't need a high bandwidth, only sending commands instead of large packets of data, so you would choose I2C. If you were sending something larger on a network, and only using a single client, such as an image to a CPU, you would use SPI. In our case, we used I2C to connect our LIDAR to our

CPU, as the LIDAR doesn't send large data packets, but does need to act as both client and server.

References

1. Mikhaylov, Konstantin, and Jouni Tervonen. "Evaluation of power efficiency for digital serial interfaces of microcontrollers." New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. IEEE, 2012.
2. Myers, P. (2007). Interfacing Using Serial Protocoles: Using SPI and I2C. In Proc. ESP 2005 (pp. 1-9).
    a. Direct comparison of SPI & I2C - brief
3. Hanabusa, R. (2007). Comparing JTAG, SPI, and I2C. Spansion's application note, 1-7.
    a. JTAG is typically used for debugging or testing systems – not typically used for communications.
4. Thomas, G. (2008). Introduction to the modbus protocol. The extension—a technical supplement to control network, 9.
    a. Modbus is the protocol the LeddarVu uses
5. Thomas, G. (2008). Introduction to Modbus Serial and Modbus TCP. Ext. A Tech. Suppl. to Control Netw, 9(5), 4-7.
    a. Comparison between Modbus over wires and over the internet – only interested in wires.
6. https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all
7. Oudjida, A. K., et al. "FPGA Implementation of I 2 C & SPI Protocols: a Comparative Study."
8. Leal-del Río, Tatiana, Gustavo Juarez-Gracia, and L. Noé Oliva-Moreno. "Implementation of the communication protocols SPI and I2C using a FPGA by the HDL-Verilog language." Research in Computing Science (2014): 31-41.
9. Leens, Frédéric. "An introduction to I 2 C and SPI protocols." IEEE Instrumentation & Measurement Magazine 12.1 (2009): 8-13.
10. Bibin, M. C., and B. S. Premananda. "Implementation of UART with BIST Technique in FPGA." International Journal of Inventive Engineering and Sciences (IJIES), ISSN (2013): 2319-9598.

Figures

1.https://en.m.wikipedia.org/wiki/File:I2C_data_transfer.svg
2.https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter#/media/File:UART_timing_diagram.svg
3.https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#/media/File:SPI_timing_diagram2.svg