Real-Time Systems

Week 4

Interfaces

Outline

- General concepts of communication
  - Number of participants
  - Direction of communication
  - Communication Model
- Communication buses
  - Parallel
  - UART
    - RS232/485/...
  - I2C
  - SPI

Number of Participants

- **Point-to-point**
  - The simplest of communication protocols that involves just two devices exchanging data
  - A common example is RS-232
- **Multi-drop**
  - A more complicated structure that consists of a single master and a number of slaves, thereby facilitating more complicated data transfers and control architectures.
  - Some protocols, such as I2C, can have multiple masters.

Direction of Communication

- Simplex
  - This refers to data communication that is unidirectional.
  - A typical implementation of a simplex data link could be a sensor broadcasting data to an FPGA or a microcontroller.
- Duplex
  - The ability of a protocol to support communication in one direction at a time is referred to as “half duplex.” If the protocol can support communication in both directions simultaneously it is referred to as “full duplex.”

Communication Model

- **Layer One: Physical Layer.**
  - Describes the physical interconnection.
- **Layer Two: Data Link Layer.**
  - Describes the means for actually transferring data over the physical layer
- **Layer Three: Network Layer.**
  - Describes the means to transfer data between different networks
- **Layer Four: Transport Layer.**
  - Provides transfer of data between end users.
- **Layer Five: Session Layer.**
  - Controls the connections between end users.
- **Layer Six: Presentation Layer.**
  - Transforms data presentation between higher-level and low-level layers.
- **Layer Seven: Application Layer.**
  - Interfaces to the final application where the data transferred to is used or data is gathered for transfer.

Communication Busses

- Parallel
  - Perhaps the simplest method of transferring data between an FPGA and an on-board peripheral.
  - supports half-duplex communications between the master and the slave.
  - Depending upon the width of data to be transferred, coupled with the addressable range
- UART (Universal Asynchronous Receiver Transmitter):
  - This comprises a number of standards defined by the Electronic Industry Association (EIA), the most popular being the RS-232, RS-422, and RS-485 interfaces.
  - These standards are often used for inter-module communication as opposed to between the FPGA and peripherals on the same board,
  - These standards defined are a mixture of point-to-point and multi-drop buses.

Communication Busses

- I2C (Inter-Integrated Circuit)
  - This is a multi-master, two-wire serial bus that was developed by Phillips in the early 1980s with a similar purpose as SPI.
  - Due to the two-wire nature of this interface, communications are only possible in half-duplex mode.
- SPI (Serial Peripheral Interface)
  - This is a full-duplex, serial, four-wire interface that was originally developed by Motorola, but which has developed into a de facto standard.
  - This standard is commonly used for intra-module communication (that is, transferring data between peripherals and the FPGA within the same system module).
  - Often used for memory devices, analog-to-digital converters (ADCs), CODECs, and MultiMediaCard (MMC) and Secure Digital (SD) memory cards, the system architecture of this interface consists of a single master device and multiple slave devices.

UART - General

- A UART may be used when:
  - High speed is not required
  - An inexpensive communication link between **two** devices is required
- UART communication is very cheap
  - Single wire for each direction (plus ground wire)
    - Asynchronous because no clock signal is transmitted
  - Relatively simple hardware
- PC devices such as mice and modems used to often use UARTs for communication to the PC

UART - General

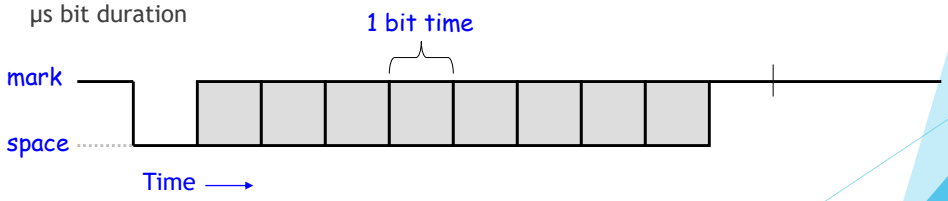
- Becoming much less common
- Largely been replaced by faster, more sophisticated interfaces
  - PCs: USB (peripherals), Ethernet (networking)
  - Chip to chip: I^2C, SPI
- Still used today when simple low speed communication is needed

UART Functions

- Transmitter
  - Convert from parallel to serial
  - Add start and stop delineators (bits)
  - Add parity bit
- Receiver
  - Convert from serial to parallel
  - Remove start and stop delineators (bits)
  - Check and remove parity bit

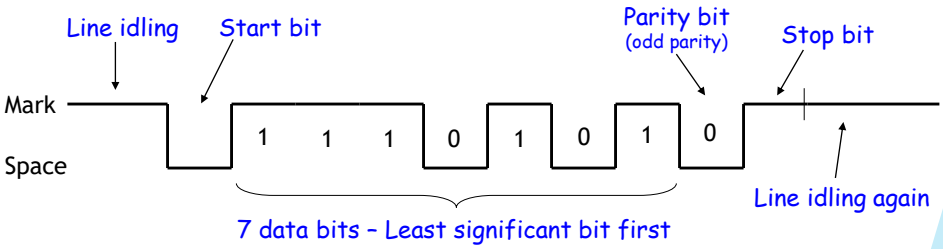
UART - Timing

- Below is a timing diagram for the transmission of a single byte
- Uses a single wire for transmission
- Each block represents a bit that can be a **mark** (logic '1) or **space** (logic '0')
- Each bit has a fixed time duration determined by the transmission rate
- Example: a 1200 bps (bits per second) UART will have a 1/1200 s or about 833.3 µs bit duration



UART Transmission Example

- Send the ASCII letter 'W' (1010111)



UART - Options

- UARTs usually have programmable options:
  - **Data:** 7 or 8 bits
  - **Parity:** even, odd, none, mark, space
  - **Stop bits:** 1, 1.5, 2
  - **Baud rate:** 300, 1200, 2400, 4800, 9600, 19.2k, 38.4k, 57.6k, 115.2k…
- Baud Rate
  - The "symbol rate" of the transmission system
  - For a UART, same as the number of bits per second (bps)
  - Each bit is 1/(rate) seconds wide
- Example:
  - 9600 baud → 9600 Hz
  - 9600 bits per second (bps)
  - Each bit is 1/(9600 Hz) ≈ 104.17 µs long

UART - Throughput

- Data Throughput Example
  - Assume 19200 baud, 8 data bits, no parity, 1 stop bit
    - 19200 baud → 19.2 kbps
    - 1 start bit + 8 data bits + 1 stop bit → 10 bits
  - It takes 10 bits to send 8 bits (1 byte) of data
  - 19.2 kbps • 8/10 = **15.36 kbps**
- How many KB (kilobytes) per second is this?
  - 1 byte = 8 bits
  - 1 KB = 1,024 bytes
  - So, 1 KB = 1,024 bytes • 8 bits/byte = 8,192 bits
  - Finally, 15,360 bps • 1 KB / 8,192 bits = **1.875 KB/s**

UART – RS232

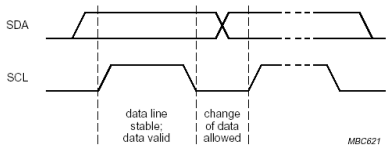
- RS232 is the most common UART standard
  - Used by PC serial ports
- RS232 does NOT use positive logic
  - Logic 1 is any signal from -25V to -3V
  - Logic 0 is any signal from +3V to 25V
  - The range -3V to +3V is a transition region that is not assigned to a logic level
- On an oscilloscope, an RS232 waveform looks inverted from the actual data values transmitted

I2C - General

- Two-wired bus
- Originally to interact within small num. of devs (radio/TV tuning, …)
- Speed options
  - 100 kbps (standard mode)
  - 400 kbps (fast mode)
  - 3.4 Mbps (high-speed mode)
- Data transfers
  - serial, 8-bit oriented, bi-directional
- Master/slave relationships with multi-master option (arbitration)
  - master can operate as transmitter or receiver
- Addressing: 7bit or 10bit unique addresses
- Device count limit: max. capacitance 400 pF

I2C - General

- Two-wired bus
  - serial data line (SDA)
  - serial clock line (SCL)
- voltage levels
  - HIGH 1
  - LOW 0
  - not fixed, depends on associated level of voltage
- bit transfer
  - SCL = 1 ⇒ SDA = valid data
  - one clock pulse per data bit
  - stable data during high clocks
  - data change during low clocks



I2C - Masters and Slaves

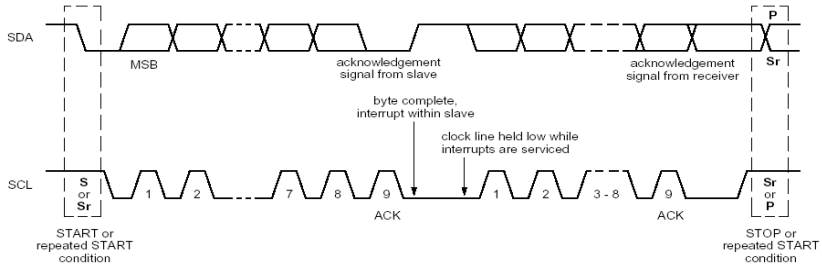
- Master device
  - controls the SCL
  - starts and stops data transfer
  - controls addressing of other devices
- Slave device
  - device addressed by master
- Transmitter/Receiver
  - master or slave
  - master-transmitter sends data to slave-recevier
  - master-receiver requires data from slave-transmitter

I2C - Data Transfer

- Data bits are transferred after start condition
- Transmission is byte oriented
  - Byte = 8 bits + one acknowledge bit
- Most significant bit (MSB) first
- Slave address is also datum
  - first byte transferred
  - during the first byte transfer:
    - master is transmitter
    - addressed slave is receiver
  - next bytes: depends on the last bit in address byte

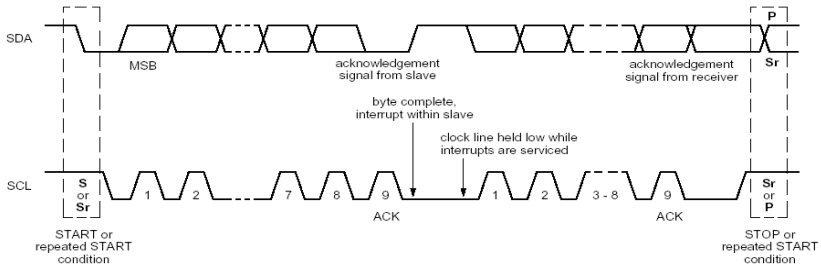
I2C - Timing

- master sets SCL = 0 and generates pulse for each data bit
- 8 pulses for data bits are followed by one pulse for ack. bit
- after ack.
  - master tries to generate next byte's first pulse
  - slave can hold SCL low → master switches to wait state



I2C - Timing

- data bits are generated by transmitter as SCL pulses
- 9-th pulse:
  - transmitter releases SDA
  - receiver must hold SDA low in order to ack. received data
  - slave must release SDA after ack. bit (allows master to end frame)



I2C - Multiple Masters

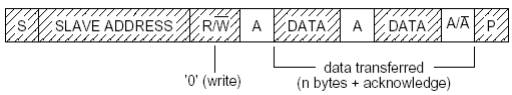
- more bus controllers can be connected
- several masters can start frame at once
- synchronization needed on SCL
- arbitration needed on SDA
- using wired-AND connection to SCL/SDA

I2C - Addressing

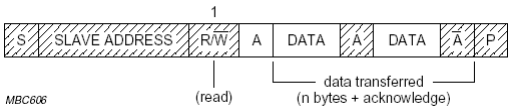
- The first byte transmitted by master:
  - 7 bits: address
  - 1 bit: direction (R/W)
    - 0 … master writes data (W), becomes transmitter
    - 1 … master reads data (R), becomes receiver
- Data transfer terminated by stop condition
- Master may generate repeated start and address another device
- Each device listens to address
  - address matches its own → device switches state according to R/W bit
- Address = fixed part + programmable part
  - fixed part assigned by I^2C committee

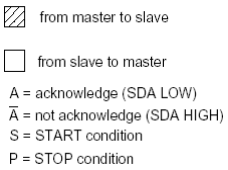
I2C - Frame Format

master-transmitter



master-receiver (since second byte)





I2C - Special Addresses

- General call 0000 000 | 0
- Start byte 0000 000 | 1
- CBUS address 0000 001 | *
  - used for cooperation of I^2C and CBUS
- High-speed mode master code 0000 1** | *
- 10-bit slave addressing 1111 0** | *

SPI - General

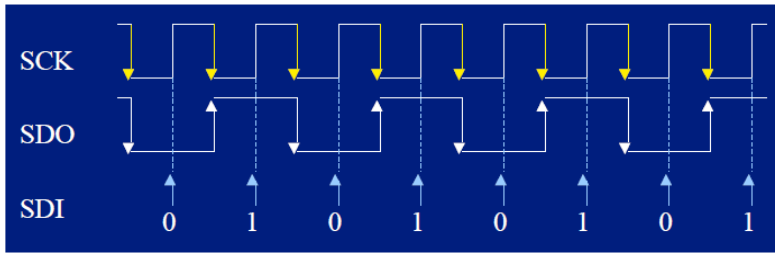
- SPI is a Synchronous protocol
  - The data is clocked along with a clock signal (SCK)
- The clock signal controls when data is changed and when it should be read
- Since SPI is synchronous, the clock rate can vary, unlike RS-232 style communications

SPI - General

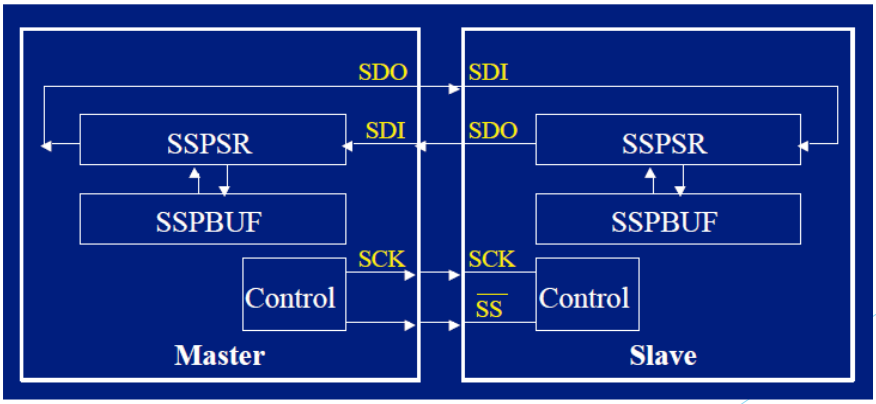
- SPI is a Master-Slave protocol
  - The Master device controls the clock (SCK)
  - No data is transferred unless a clock signal is present
  - All slaves are controlled by the master clock
  - The slave devices may not manipulate the clock
- SPI is a Data Exchange protocol
  - As data is being clocked out, new data is clocked in
  - Data is exchanged
    - no device can just be a transmitter only or receiver only
  - the master controls the exchange by manipulating the clock line (SCK)
  - Often a signal controls when a device is accessed
  - CS or SS signal is known as “Chip Select” or “Slave Select” and is frequently an active-low signal.

SPI – Data Transmission

- Data is only output during the rising or falling edge of SCK
- Data is latched during the opposite edge of SCK
- The opposite edge is used to ensure data is valid at the time of reading



SPI Data Transfer



References

- BYU
  - Slides about UART from Ecen 224
- Tomáš Matoušek
  - Inter Integrated Circuits bus by Philips Semiconductors
- Microchip
  - Overview and Use of the PICmicro Serial Peripheral Interface