# ECE 471 – Embedded Systems
# Lecture 10

Vince Weaver

`http://www.eece.maine.edu/∼vweaver`

`vincent.weaver@maine.edu`

2 October 2014

# Announcements

- Homework #4 due Friday
- Homework #5 posted soon.
- Hand out i2c displays today. Be careful with them!
- Midterm coming up on Tuesday, October 21st. Let me know if that conflicts.

# System Busses

- Older busses often exposed CPU pins directly to connector: Apple II, S-100, ISA
- This was not sustainable, if only because number of CPU pins grew rapidly. Also speed issues.

# Parallel vs Serial Busses

- Originally most busses were Parallel. More bits at a time means higher bandwidth. IDE, Parallel Port, 32-bit PCI, 64-bit PCI
- Problems with parallel: keeping signals in sync. As busses go faster, skew comes into things. Wire length matters. Power issues with driving wide busses.
- Newer busses are serial: SATA, PCIe, USB, Firewire, etc. Also advantage of having fewer wires to route.

- People (especially HPC) still grumble about speed of PCIe

# Embedded Busses

# i2c

- Inter-Integrated Circuit, Invented by Philips (now NXP) in 1982
- Broadcom and others for some reason call it “Two Wire Interface”
- Two-wires (4 if you include Vdd and Ground)
- Since 2006, no licensing fees (though do have to pay to reserve number)

# Why is i2c popular?

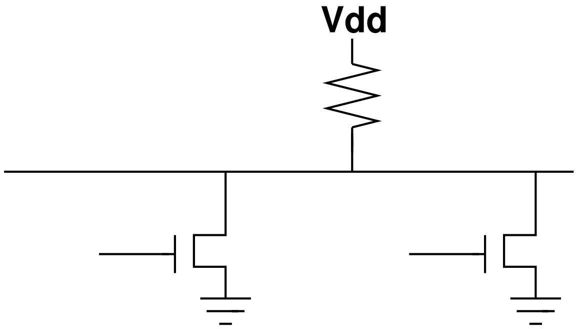
- Stable standard
- Relatively easy to implement
- Not many wires
- Good enough
- Cheap

# Uses of i2c

- SMbus
- DDC (video card / monitor communication)
- Configuring SDRAM
- Temp sensor and fan chips on motherboards
- Wii nunchuck

# Protocol Overview

- Serial Data Line (SDA) and Serial Clock (SCL), Open Drain, Pulled up by resistors
- Open drain means output can be wired together



- 7-bit (or 10-bit) address speeds

- Standard=100kbits/s, slow=10kbits/s, fast=400kbits/s fast plus 1Mbits/s, high 3.4Mbits/s (actual transfers slower due to overhead)
- Length of bus limited to a few meters
- Master (generates clock, init transaction), Slaves (responds)
- Can be multiple masters and slaves
- Master sends start bit, 7-bit address of slave, then read/write bit

- Slave responds with ACK then interacts
- Address and Data set Most-significant Bit first

# Protocol

- Start bit is SDA high-low while SCL high. Stop is SDA low-high while SCL high.
- To transmit bit, set SCL low, set SDA to value, set SCL high, wait 4us, sets low
- After every 8-bits an ACK bit is sent. If 0, more to come. If 1, we are done (or there is an error)
- Clock stretching: slave can hold SCL low until it is done processing

- Arbitration: masters monitor SDA and won't start unless idle. Deterministic arbitration.

  If tries to send a 1 and notices something else is pulling to zero, then a collision and stops. Low addresses automatically win.

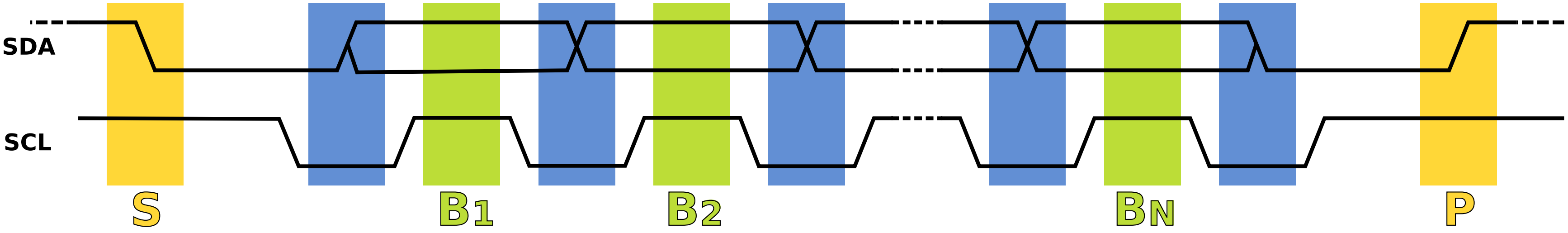


Figure 1: Protocol diagram from Wikipedia

# i2c and Rasp-pi

- 2 busses, only one easily accessible on Model B (other on Camera interface).

- Insmod i2c-bcm2708 and i2c-dev. /etc/modules also remove from blacklist /etc/modprobe.d/raspi-blacklist.conf

- Also want to install i2c-tools if possible apt-get i2c-tools

- i2c port 1 (/dev/i2c-1). Used to be i2c-0 on older machines. Other boards (beaglebone black) likely different.

- Clock stretching buggy on the rasp-pi
- 3.3V
- default speed is 100kHz. You can change this with the baudrate= module parameter.
- i2c-1 on pins SDA=3, SCL=5
- i2c-0 on the camera interface (pad5)

# i2c and Linux

- Like with GPIOs, kernel can drive it, or be exposed to userspace
- i2c-dev module must be installed (and i2c driver)
- Open the device node, /dev/i2c-1
- Use ioctls I2C_SLAVE to set the address of the device we wish to talk to.
- Use standard read or write calls to communicate with

the device

- Close the device when done.
- i2c slave addresses are 7 bits, but when sent the r/w bit is put at end. This can be confusing; some spec sheets will list a slave address as 0x38 (7 bits) but Linux exports this as 0x70 (0x38 shifted left by 1).

# Sample i2c Linux code

For more details on this, see the HW3 handout.

```
unsigned char buffer[17];
int display_fd;

/* open */
display_fd = open("/dev/i2c-1", O_RDWR);
if (display_fd < 0) fprintf(stderr,"Error!\n");

/* set slave address */
result=ioctl(display_fd, I2C_SLAVE, 0x70);
if (result < 0) fprintf(stderr,"Error!\n");

/* writing */
buffer[0]= HT16K33_REGISTER_SYSTEM_SETUP | 0x01;
```

```
if ( (write(display_fd, buffer, 1)) !=1) fprintf(stderr,"Error!\n")

/* closing */
close(display_fd);
```

# i2c on the Pi – detecting

```
i2cdetect -y -r 1
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

# LED Driver Chip

- This is a ht16k33, datasheet available: `http://www.adafruit.com/datasheets/ht16K33v110.pdf`

- Supports up to 16x8 LEDs, as well as keypad input. Can dim display, also blink. Common cathode.

```
  -|>|- common
```

- Works by rapidly scanning all segments fast enough cannot see.

- To set up, write byte commands, high 4 bits command lower 4 bits data.

- To set up full display, write the pointer offset of internal framebuffer, than 16 bytes of on/off data.

- Actual LED hooked up is a BL-Q56D-43UG 4x7 segment Ultra-Green display common cathode.

# Benefit of OS

- Code is portable across all machines with i2c bus
- Can use same code on Gumstix, Rasp-Pi, Beaglebone, etc.
- Will probably need to change the bus number (It's i2c-3 on gumstix).