

Today is the
LAST
LECTURE



Mandatory
WRITING
WORKSHOP
Thu, 10/30
2:30-4:00
32-141

Potpourri

- Serial communication links
- FFTs
- FPGAs @ home
- Project Schedule

Lab #5 due tonight, project abstract next Monday

Serial Communications

- Sending information one bit at a time vs. many bits in parallel
 - Serial: good for long distance (save on cable, pin and connector cost, easy synchronization). Requires “serializer” at sender, “deserializer” at receiver
 - Parallel: issues with clock skew, crosstalk, interconnect density, pin count. Used to dominate for short-distances (eg, between chips).
 - BUT modern preference is for parallel, but independent serial links (eg, PCI-Express) as a hedge against link failures.
- A zillion standards
 - Asynchronous (no explicit clock) vs. Synchronous (CLK line in addition to DATA line).
 - Recent trend to reduce signaling voltages: save power, reduce transition times
 - Control/low-bandwidth Interfaces: SPI, I²C, 1-Wire, PS/2, AC97
 - Networking: RS232, Ethernet, T1, Sonet
 - Computer Peripherals: USB, FireWire, Fiber Channel, Infiniband, SATA, Serial Attached SCSI

RS232 (aka “serial port”)

- Labkit: simple bidirectional data connection with computer.
- Characteristics
 - Large voltages => special interface chips
(1/mark: -12V to -3V, 0/space: 3V to 12V)
 - Separate xmit and rcv wires: full duplex
 - Slow transmission rates (1 bit time = 1 baud); most interfaces support standardized baud rates: 1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K
 - Format
 - Wire is held at 1/mark when idle
 - Start bit (1 bit of “0” at start of transmission)
 - Data bits (LSB first, can be 5 to 8 bits of data)
 - Parity bit (none, even, odd)
 - Stop bits (1, 1.5 or 2 bits of 1/mark at end of symbol)
 - Most common 8-N-1: eight data bits, no parity, one stop bit

RS232 interface

- Transmit: easy, just build FSM to generate desired waveform with correct bit timing
- Receive:
 - Want to sample value in middle of each bit time
 - Oversample, eg, at 16x baud rate
 - Look for 1→0 transition at beginning of start bit
 - Count to 8 to sample start bit, then repeatedly count to 16 to sample other bits
 - Check format (start, data, parity, stop) before accepting data.

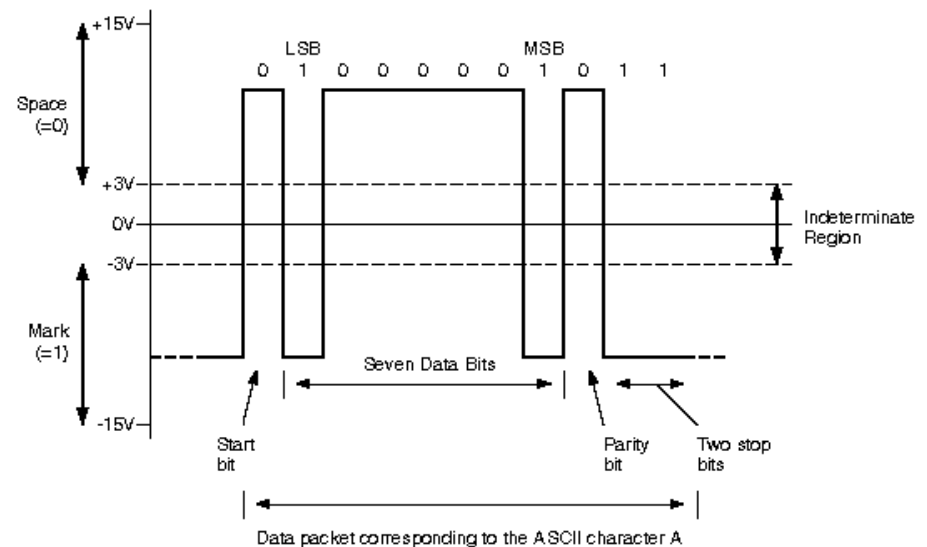
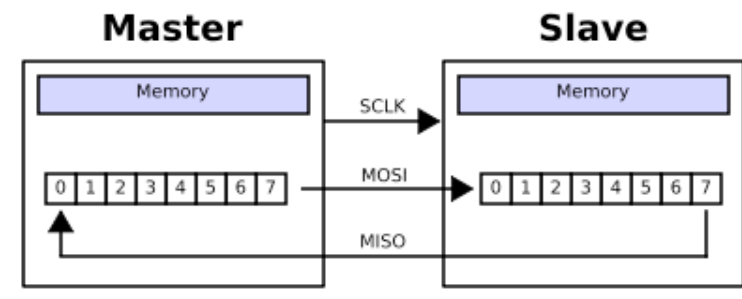
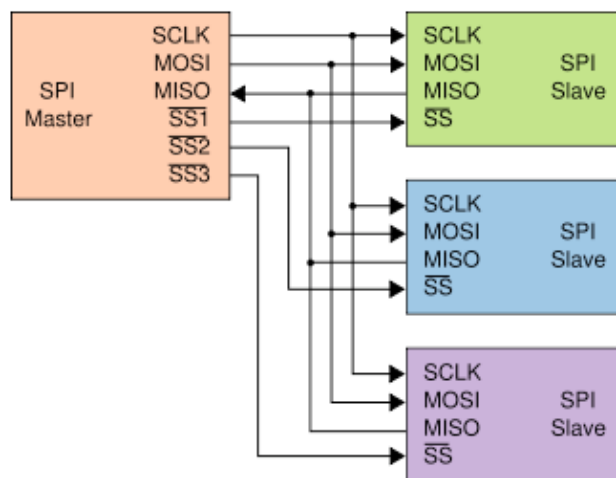


Figure from
<http://www.arcelect.com/rs232.htm>

SPI (Serial Peripheral Interface)

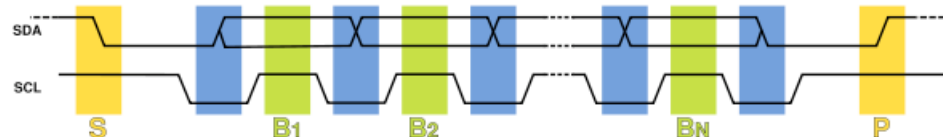
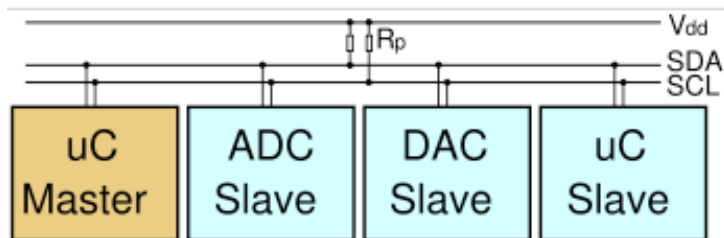
- Simple, 3-wire interface + devices selects
 - SCLK generated by master (1-70MHz). Assert data on one edge, sample data on the other. Default state of SCLK and assignment of edges is often programmable.
 - Master Out Slave In (MOSI) data shifted out of master register into slave register
 - Master In Slave Out (MISO) data shifted out of slave register and into master register
 - Selects (usually active low) determine which device is active. Assertion often triggers an action in the slave, so master waits some predetermined time then shifts data.



Figures from Wikipedia

I²C (Inter-Integrated Circuit)

- 2 open-drain wires (SCL = clock, SDA = data)
- Multiple-master, each transmission addresses a particular device, many devices have many different sub-addresses (internal registers)
- Format (all addresses/data send MSB first):
 - Sender: Start [S] bit (SDA↓ while SCL high)
 - Sender: One or more 8-bit data packets, each followed by 1-bit ACK
 - Data changed when SCL low, sampled at SCL↑
 - Receiver: Active-low ACK generated after each data packet
 - Sender: Stop [P] bit (SDA↑ while SCL high)
- SCL and SDA have pullup resistors, senders only drive low, go high-impedance to let pullups make line high (so multiple drivers okay!)
 - Receiver can hold SCL low to stretch clock timing, sender must wait until SCL goes high before moving to next bit.
 - Multiple senders can contend using SDA for arbitration

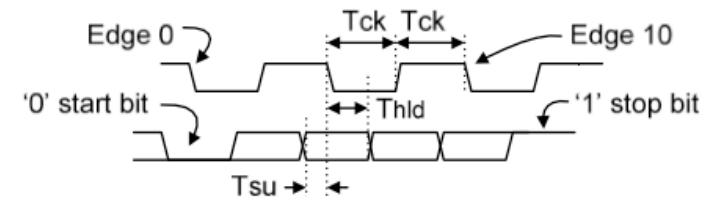


PS/2 Keyboard/Mouse Interface

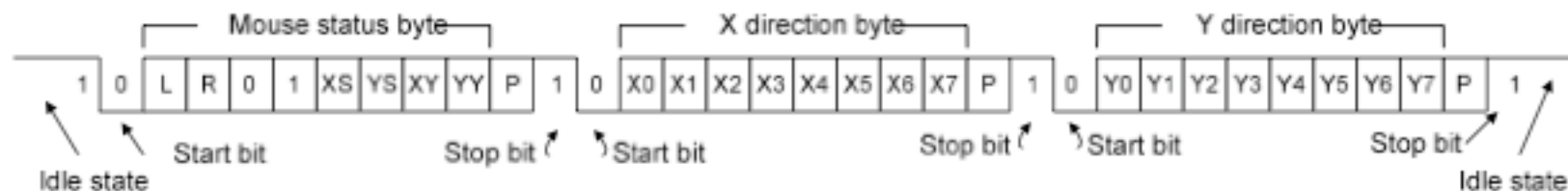
- 2-wire interface (CLK, DATA), bidirectional transmission of serial data at 10-16kHz

- Format

- Device generates CLK, but host can request-to-send by holding CLK low for 100us
- DATA and CLK idle at "1", CLK starts when there's a transmission. DATA changes on CLK↑, sampled on CLK↓
- 11-bit packets: one start bit of "0", 8 data bits (LSB first), odd parity bit, one stop bit of "1".
- Keyboards send scan codes (not ASCII!) for each press, 8'hF0 followed by scan code for each release
- Mice send button status, Δx and Δy of movement since last transmission

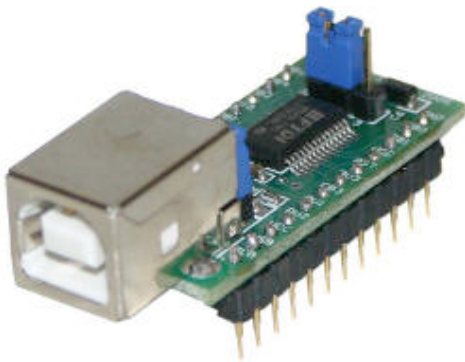


Symbol	Parameter	Min	Max
T_{CK}	Clock time	30us	50us
T_{SU}	Data-to-clock setup time	5us	25us
T_{HLD}	Clock-to-data hold time	5us	25us

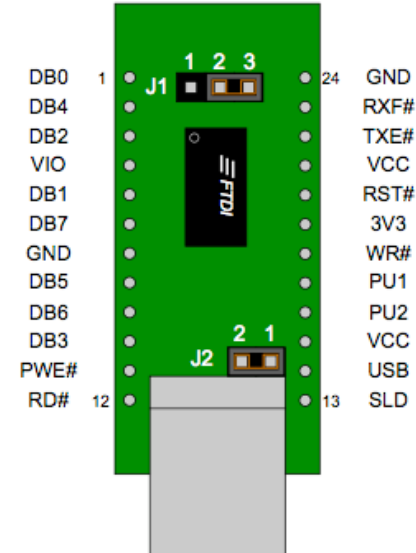


USB (Universal Serial Bus)

- 2-wire (D+,D-) for high-speed, bidirectional polled transmission between master and addressable endpoints in multiple devices. Full speed (12Mbps) and High speed (480Mbps) data rates.
- Multi-level tiered-star topology (127 devices, including hubs)
- FTDI UM245R USB-to-FIFO module for bidirectional data transfer using a handshake protocol, also asynchronous "bit-bang" mode with selectable baud rates.
 - 24-pin DIP module, wire to user pins
 - Drivers for Windows workstations in lab



Figures from ftdi.com



Audio Feature Extraction

- Most features are best recognized in the frequency domain
- Use Discrete Fourier Transform
 - Algorithm used: Fast Fourier Transform (FFT)
 - Input: N data values acquired at sample frequency ω_s
 - Nyquist rate is $\omega_s/2$
 - Output: N complex values representing DFT coefficients in the frequency range $-\omega_s/2$ to $+\omega_s/2$.
 - Each value covers a frequency range of ω_s/N
 - Indices $(0, (N/2)-1)$ are for frequencies $i * (\omega_s/N)$
 - Indices $(N/2, N-1)$ are for frequencies $-\omega_s/2 + (i - N/2) * (\omega_s/N)$
 - If N is even, output is symmetric, so we can calculate magnitude using only positive frequencies. Magnitude $\approx \sqrt{r^2 + i^2} * \text{constant factors}$.
- Example
 - Audio data from AC97 sampled at 8kHz
 - 2048 data points \Rightarrow 2048-point FFT
 - 2048 complex results, each result covers $8k/2048 = 4\text{Hz}$ range

Iterative Sqrt module

```
// takes integer square root iteratively
module sqrt #(parameter NBITS = 8, // max 32
              MBITS = (NBITS+1)/2)
    (input wire clk, start,
     input wire [NBITS-1:0] data,
     output reg [MBITS-1:0] answer,
     output wire done);

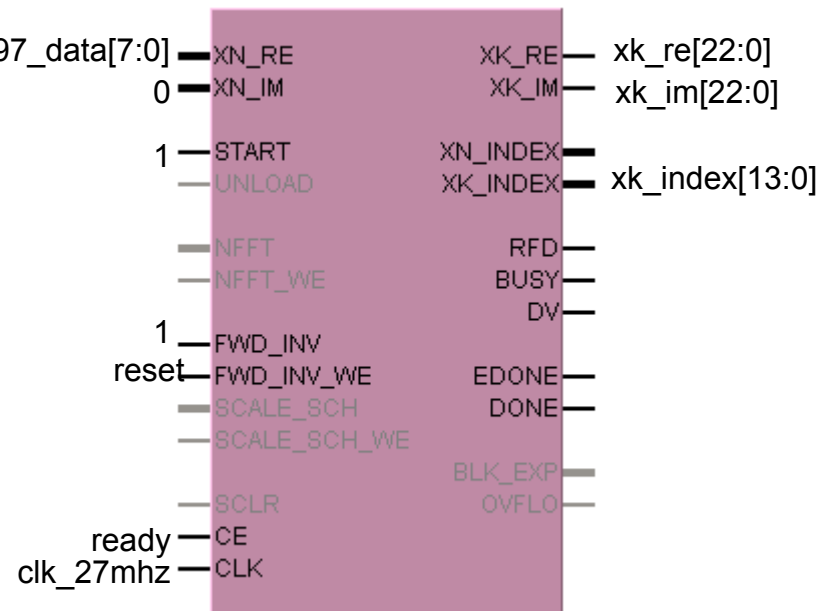
    reg busy;
    reg [4:0] bit;
    // compute answer bit-by-bit, starting at MSB
    wire [MBITS-1:0] trial = answer | (1 << bit);

    always @(posedge clk) begin
        if (busy) begin
            if (bit == 0) busy <= 0;
            else bit <= bit - 1;
            if (trial*trial <= data) answer <= trial;
        end
        else if (start) begin
            busy <= 1;
            answer <= 0;
            bit <= MBITS - 1;
        end
    end

    assign done = ~busy;
endmodule
```

FFT example

- IP wizard will build a N-point FFT module
 - WARNING: they're big!
- In theory, there are two operating modes (select at build time)
 - "pipelined" where you get a complex value out for every sample you send the module - runs continuously
 - "burst" where you load up N samples, wait a while and get your answer while loading the set of samples.
- In practice, we've been having trouble making "pipelined" mode work on the labkit with ISE8.1
- Demo: audio spectrum analyzer
 - Uses "pipelined" mode



FFT of AC97 data

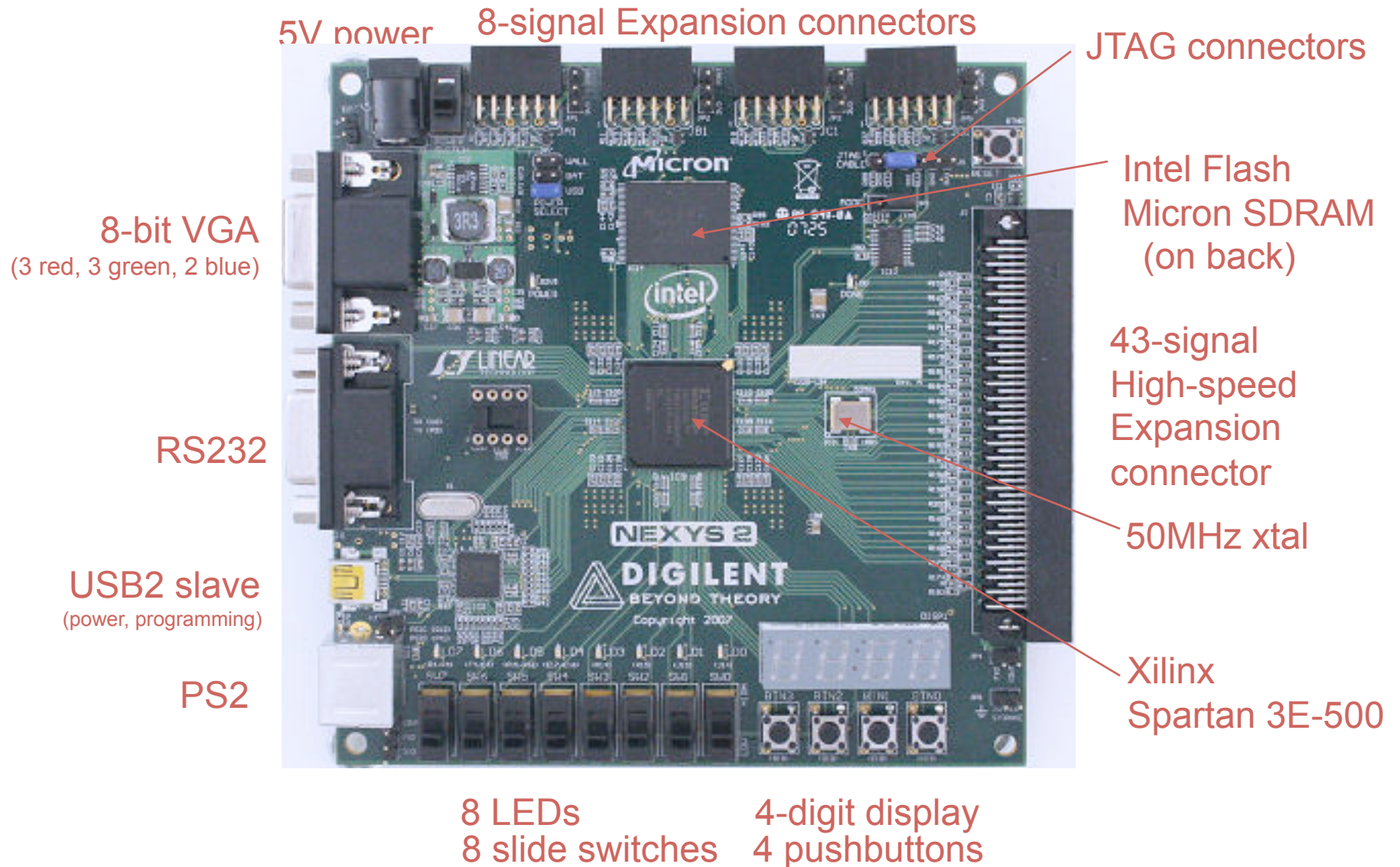
To process AC97 samples:

- use Pipelined mode (input one sample in each cycle, get one sample out each cycle).
 - FFT expects one sample each cycle, so hook READY to CE so that FFT only cycles once per AC97 frame
- use Unscaled mode, do scaling yourself
 - Number of output bits = (input width) + NFFT + 1
 - NFFT is $\log_2(\text{size of FFT})$
- let number of FFT points = P , assume 48kHz sample rate
 - there are P frequency bins
 - positive freqs in bins 0 to $(P/2 - 1)$
 - negative freqs in bins $(P/2)$ to $(P-1)$
 - each bin covers $(48k/P)\text{Hz}$
 - Use XK_INDEX to tell which bin's data you're getting out
 - Typically you want magnitude = $\sqrt{xk_re^2 + xk_im^2}$

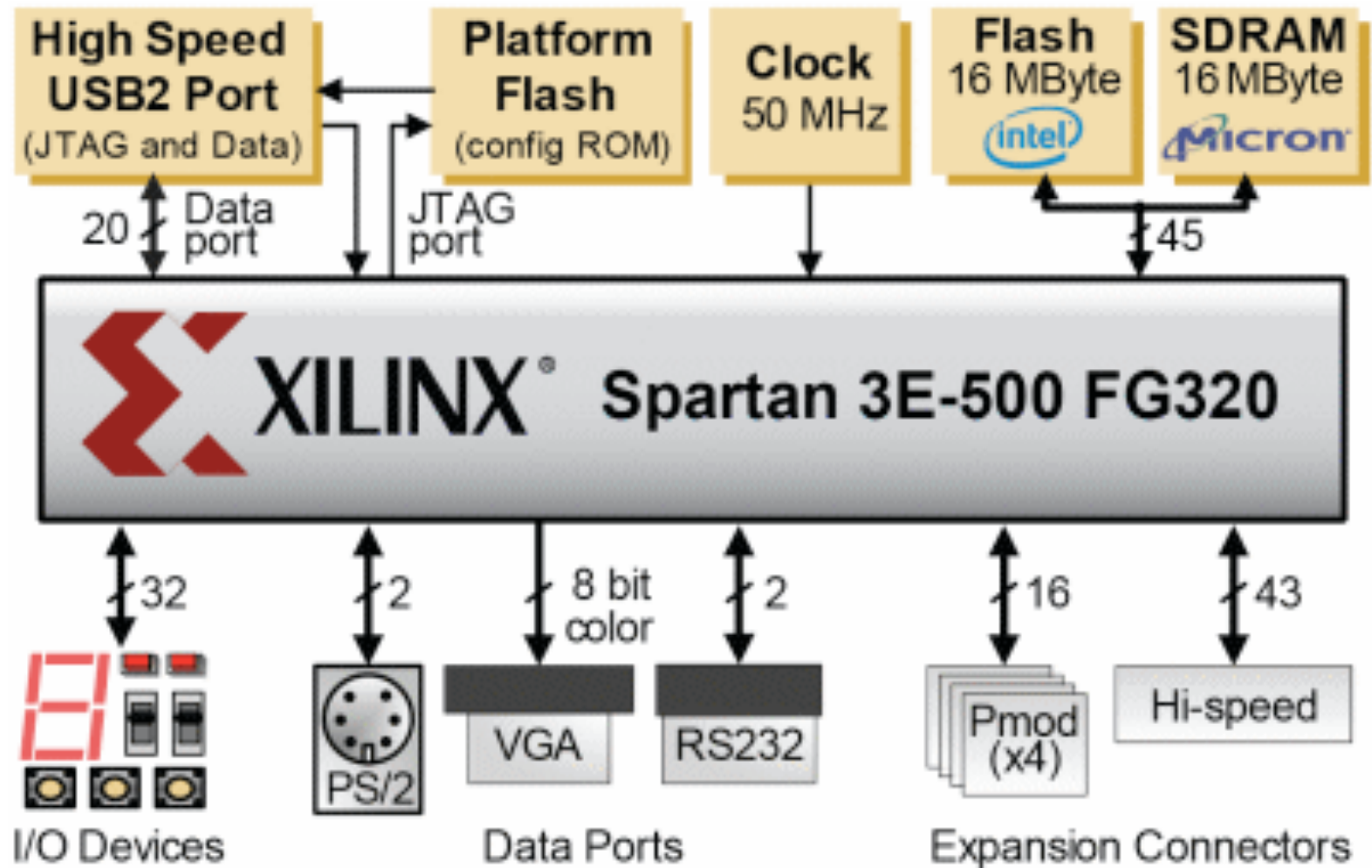
FPGAs @ Home

- 6.111 labkit: the Lexus of FPGA protoboards
 - XC2V6000 (67,586 LUT/FFs, 144 BRAMs)
- Two affordable alternatives (lots more out there)
 - Nexys2 Board (www.digilentinc.com)
 - \$99 = Spartan 3E-500 (9,312 LUT/FFs, 20 BRAMs)
 - \$119 = Spartan 3E-1200 (17,344 LUT/FFs, 28 BRAMs)
 - Switches, buttons, leds, 4-digit seven-segment display
 - 16Mbyte flash, 16Mbyte SDRAM
 - USB2 slave (power, programming, 8-bit host data stream)
 - PS2, serial port, 256-color VGA, 4 expansion connectors
 - XSA-3S1000 @ \$199 (www.xess.com)
 - Spartan XC3S1000 (15,360 LUT/FFs, 24 BRAMs)
 - Switches, buttons, 1-digit display
 - 32Mbyte SDRAM, 2Mbyte Flash
 - PS2, 512-color VGA
 - 80-pin expansion connector (protoboard friendly)

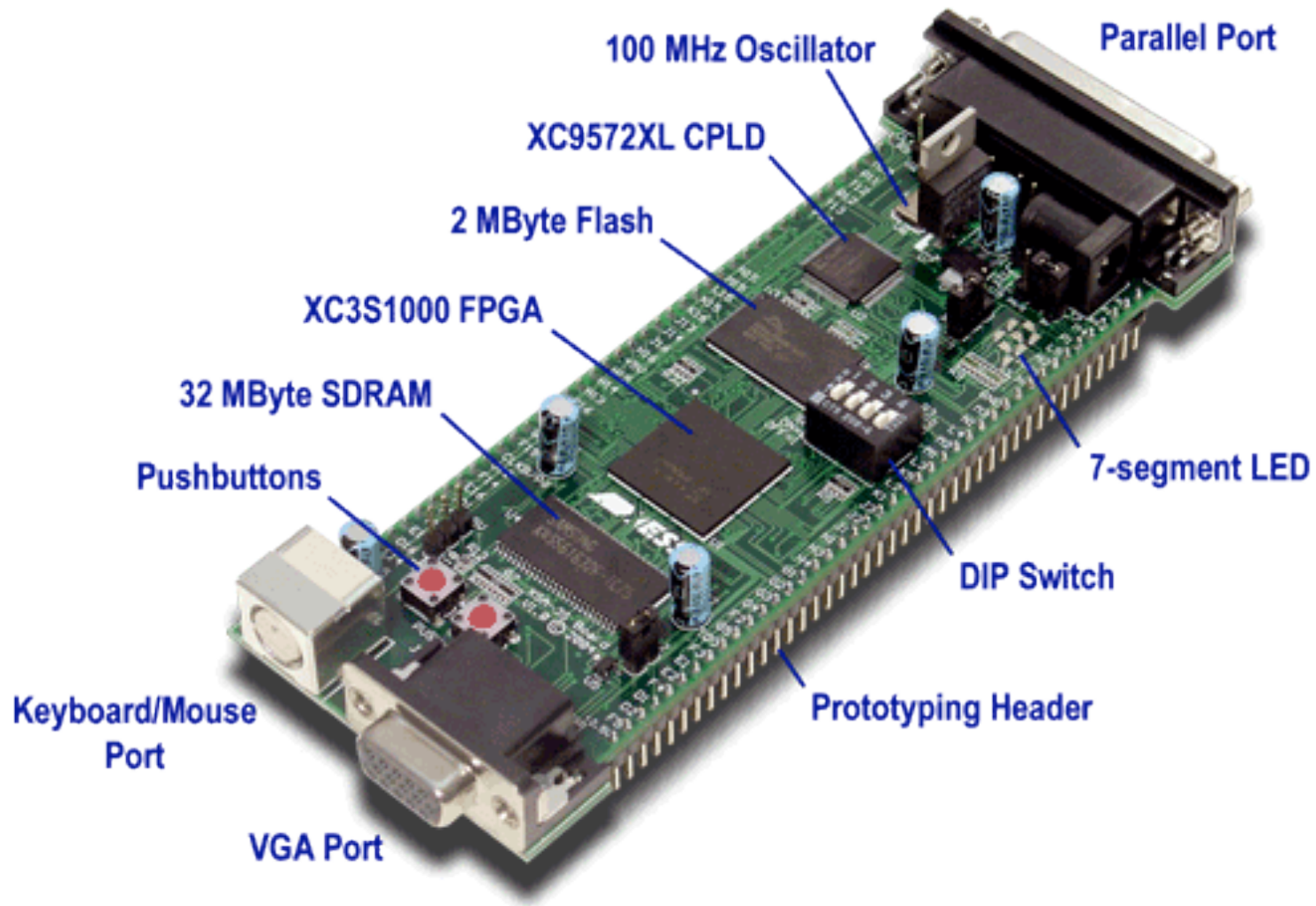
Digilent Nexys2 Board



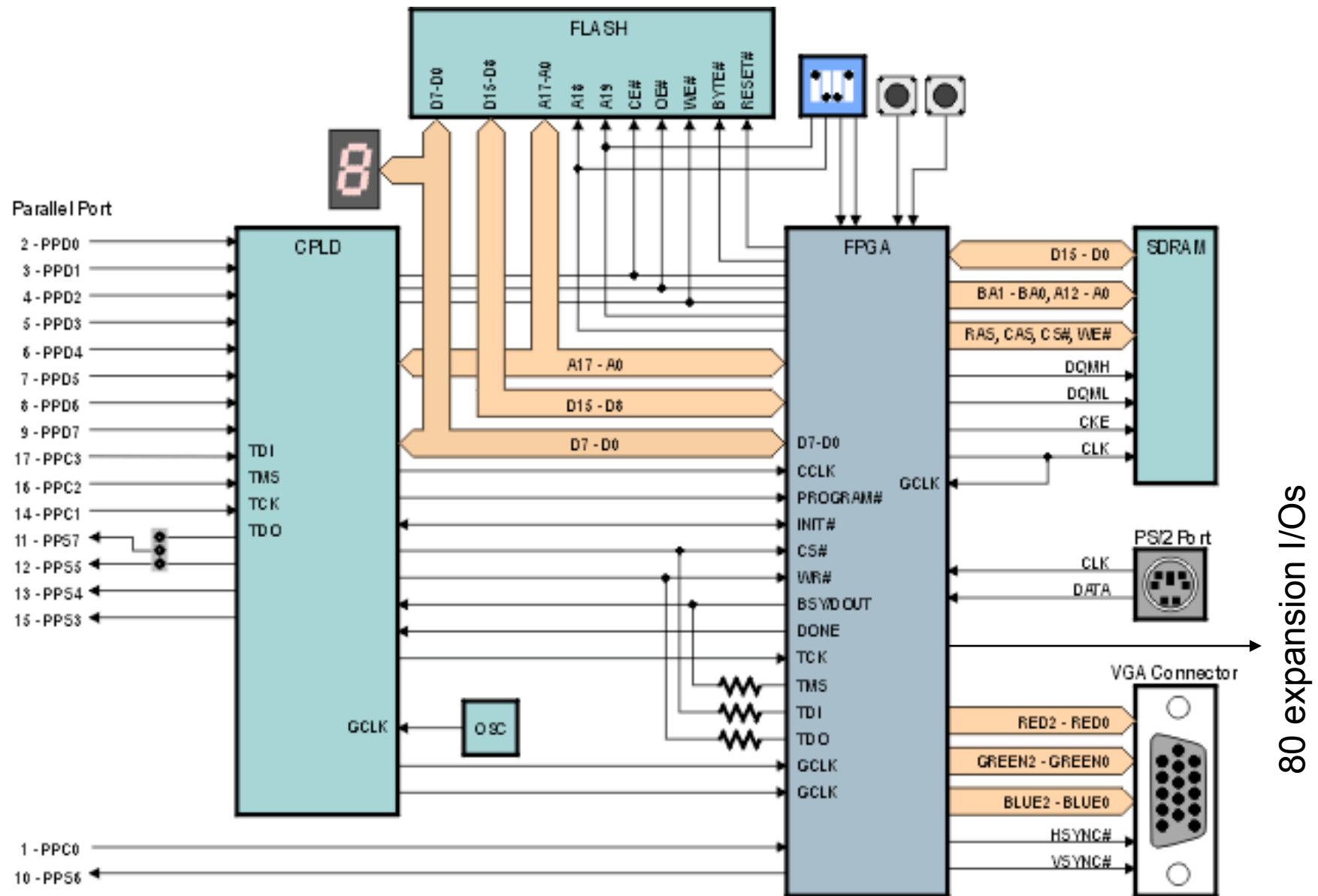
Nexys2 Diagram



XSA-3S1000



XSA-3S1000 Block Diagram



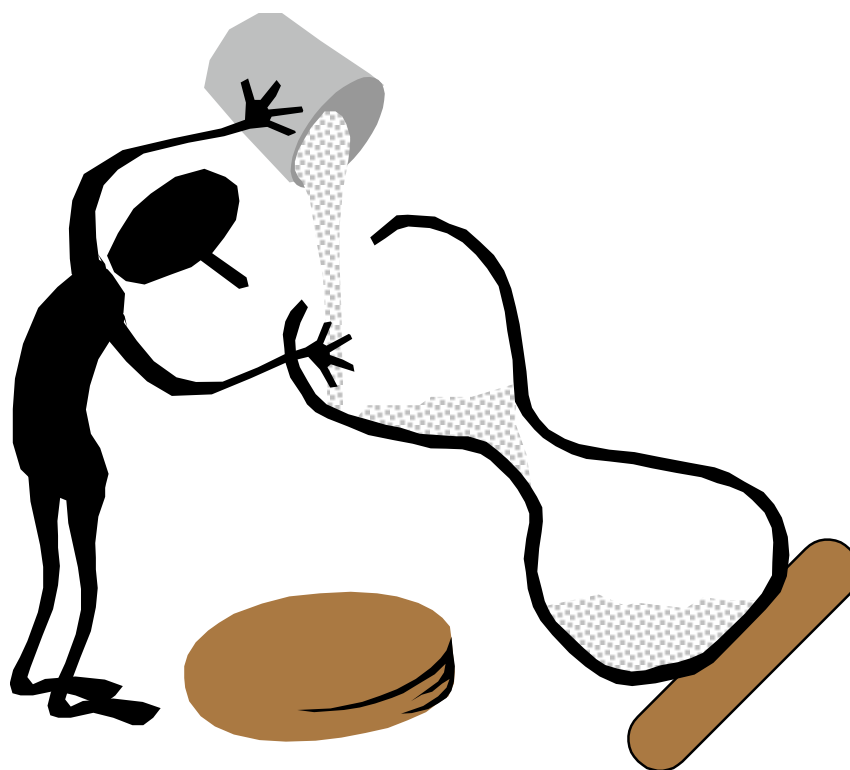
FPGA Software

- Xilinx ISE Web-pack
 - Free!
 - Windows or Linux
 - Supports subset of Xilinx FPGAs (but covers the chips used in the boards listed on the previous slide)
 - No IP Wizard, but
 - You can build memories, logic “by hand” using available components (eg, RAMB16_Sxx) and appropriate defparams or attribute assignments - see Xilinx documentation
 - A **lot** of very good design info in Xilinx App Notes (on-line)
 - Built-in simulator
 - Digilent USB cable for Nexys2 boards under WinXP (use VMWare if running under Linux or MacOS!).

Schedule Reminders

- Thu, 10/23: LAST LECTURE, Lab #5 checkoff
- Mon, 10/27: upload Project Abstract by 5pm
- Thu, 10/30: mandatory writing workshop, 1p, 34-101
- Fri, 10/31: complete proposal meeting with mentor
upload Project Proposal by 5pm
- Fri, 11/07: upload CI-M final version by 5pm
complete block diagram meeting w/ mentor
- W, Th: 20min design presentations
11/12-13 schedule TBA (we'll email you!)
please upload slides to website
- Fri, 11/14: upload Project Checklist by 5pm
- M, Tu, W : project presentations & videotaping
12/8-10 schedule TBA (we'll email you!)
- Wed, 12/10: upload Final Project Report by 5pm
(sorry, no extensions possible!)

See you in lab!



Final project represents 72 hours of credit, so you should average 2-3 hours/day of work on your project assuming you give yourself the occasional day off...