EECS 373 – lecture 4

Buses, serial communication and digital design

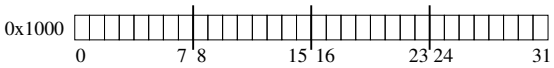
Announcements

- I saw very few people in for questions on prelab 3.
  - Feel free to use my office hours for help as needed.
  - Don't put off the prelabs. You certainly don't have time to do them in the lab
- Lab 3
  - Lab 3 is a 2 week lab. You will likely need to use open hours a LOT to get this done.
- Ron is holding extra lab hours this week from 4:30-7:30
- We will be arranging to insure you can get into the lab during normal hours. See the website...

Today is variety day

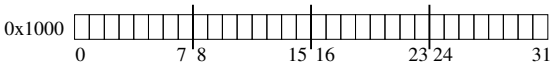
- Bus stuff (almost finished *last* Tuesday)
  - Reads and writes as they appear on the bus
  - Unaligned accesses
- Digital Design
  - Finish example from Thursday
  - State machines in digital design
  - Other examples

Transfer Alignment



MPC823 external bus supports natural address alignment
- Byte access: Any address alignment
- Half-word access: Address bit 31 equal to 0
- Word access: Address bits 31 and 30 equal to 0

Dealing with Smaller Accesses: Reads



Assume that the word value 0x12345678 is stored at 0x1000 and that r4 contains 0x1000. What happens on the following transfers?

D[0:7] D[8:15] D[16:23] D[24:31]

- lbz r3 , 0 (r4)
- lbz r3 , 1 (r4)
- lbz r3 , 2 (r4)
- lbz r3 , 3 (r4)
- lhz r3 , 0 (r4)
- lhz r3 , 2 (r4)

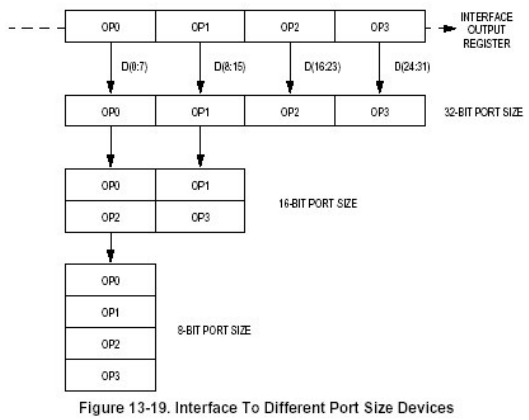


Figure 13-19. Interface To Different Port Size Devices

Table 13-2. Data Bus Requirements For Read Cycles

TRANSFER SIZE	TSIZE [0:1]		INTERNAL ADDRESS		32-BIT PORT SIZE				16-BIT PORT SIZE		8-BIT PORT SIZE
			A30	A31	D0–D7	D8–D15	D16–D23	D24–D31	D0–D7	D8–D15	D0–D7
Byte	0	1	0	0	OP0	—	—	—	OP0	—	OP0
	0	1	0	1	—	OP1	—	—	—	OP1	OP1
	0	1	1	0	—	—	OP2	—	OP2	—	OP2
	0	1	1	1	—	—	—	OP3	—	OP3	OP3
Half-Word	1	0	0	0	OP0	OP1	—	—	OP0	OP1	OP0
	1	0	1	0	—	—	OP2	OP3	OP2	OP3	OP2
Word	0	0	0	0	OP0	OP1	OP2	OP3	OP0	OP1	OP0

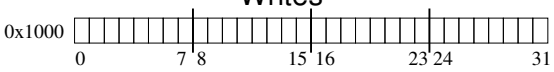
NOTE: — Denotes that a byte is not required during that read cycle.

Table 13-3. Data Bus Contents for Write Cycles

TRANSFER SIZE	TSIZE [0:1]		INTERNAL ADDRESS		EXTERNAL DATA BUS PATTERN			
			A30	A31	D0–D7	D8–D15	D16–D23	D24–D31
Byte	0	1	0	0	OP0	—	—	—
	0	1	0	1	OP1	OP1	—	—
	0	1	1	0	OP2	—	OP2	—
	0	1	1	1	OP3	OP3	—	OP3
Half-Word	1	0	0	0	OP0	OP1	—	—
	1	0	1	0	OP2	OP3	OP2	OP3
Word	0	0	0	0	OP0	OP1	OP2	OP3

NOTE: — Denotes that a byte is not required during that read cycle.

Dealing with Smaller Accesses: Writes



How about the following transfers?

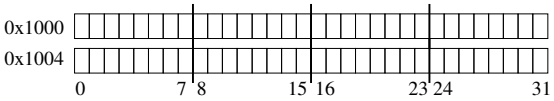
D[0:7] D[8:15] D[16:23] D[24:31]

- stb r2 , 0 (r4)
- stb r2 , 1 (r4)
- stb r2 , 2 (r4)
- stb r2 , 3 (r4)
- sth r2 , 0 (r4)
- sth r2 , 1 (r4)

Dealing with Smaller Accesses

- On a write access, which two factors determine which bits in a 32-bit word are updated?
- On most wide buses, the master drives *byte enable* lines instead of less significant address bits
  - Moto 68000 (16 bits): LDS', UDS' (no address LSB)
  - 32-bit buses: Replace low 2 address bits with 4 byte enables
- MPC823 does not:
  - Full byte address provided
  - Size (byte, halfword, word) encoded on two control lines TSIZ[0-1]

Unaligned Accesses



Consider two adjacent 32-bit memory locations and assume that r4 = 0x1000. What happens when the CPU executes the following instructions?

- lwz r3 , 2 (r4)
- lwz r3 , 1 (r4)

Problems with Unaligned Accesses

What are some problems with unaligned accesses?

Now back to digital design.

Last time we discussed:

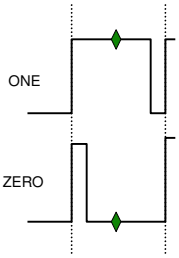
- Serial communication
  - Problems with serial communication
    - Want shared clock
    - Use different techniques instead.
      - One was "change if real data stays same"
      - Used XOR to decode this one
    - PWM signal
      - Use "duty cycle" to indicate state
      - Low bandwidth (why?)
    - Did a bit on counters

Correction

USB stuff

- 0 is encoded as a transition, 1 as no transition
- Bit stuffing
  - If no change for 6 cycles (6 1s), insert a change.

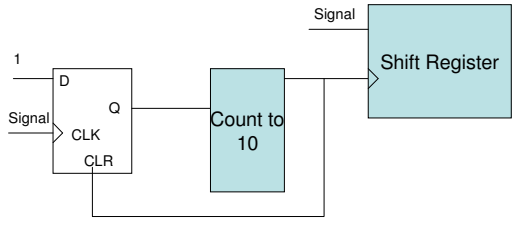
Digital PWM signal



- This is a simple technique for sending low bandwidth data.
  - Start on the rising edge
  - Count till about ½ the cycle
  - Measure the value.
  - Do it again.
- Useful
  - Sender and reciever can have clocks way out of sink. (almost factor of 2)
  - Could even have different clocks with some way of agreeing on clock period
    - Say sender starts with a pulse and receiver times it with his clock.

Let's design it.

- Assume signal period is 20 local clocks.
- Assume either 30/70 or 70/30 duty cycle +/- 10%
- Assume signal might have some noise
- Mr. State machine.



Say you used this solution and found you were often getting more data than you had sent. What might you suspect was wrong?

Design again

- Rather than that *ad hoc* method, let's design a state machine which does this.
  - We will not go to a circuit solution
    - Could do it, but takes too long
  - Use tools to do this for you
    - ABLE, FSM tool in Xilinx, etc.

Now let's design an "analog" version

- Say we want to measure duty cycle…
  - Design that.
  - Pure FSM may not be the best way to go…
- Time from rising edge to rising edge is about 100 to 1000 cycles
  - Min/Max is 10/90 90/10
  - Some noise possible.

And one more design problem?