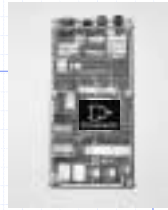


CS150 Project Checkpoint 1



Controller Interface

Dreamkatz **Controller Interface**

1. The N64 Controller

2. Physical interface
3. Communication protocol
4. Design structure and FSM
5. Design implementation
6. Hints & cautions



The N64 controller

- ◆ The *Dreamkatz* Controller has 9 action buttons, a digital directional pad and an analog stick.
- ◆ Fits nicely in the average person's hand
- ◆ The original connector end have been replaced with a *Dreamkatz* compatible plug.
- ◆ Has 3 pins: VCC, GND, DATA



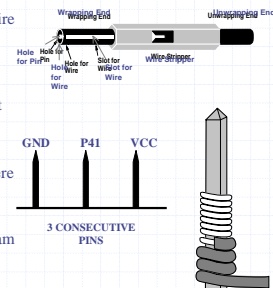
Dreamkatz **Controller Interface**

1. The N64 Controller
2. **Physical interface**
3. Communication protocol
4. Design structure and FSM
5. Design implementation
6. Hints & cautions




Establishing the physical interface: wire wrapping

- ◆ Materials needed: wire wrap, wire wrap tool, and a little practice.
- ◆ Wire wraps available from the IEEE office at 204a Cory. (Next door)
- ◆ Pick three consecutive pins where controller will connect to.
- ◆ Wire wrap according the diagram on the right.




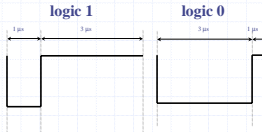
Dreamkatz **Controller Interface**

1. The N64 Controller
2. Physical interface
- 3. Communication protocol**
4. Design structure and FSM
5. Design implementation
6. Hints & cautions



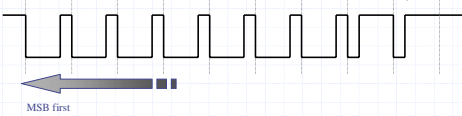
Establishing data communication: **Protocol Specifications Part I**

- ◆ One data line means it handles receiving and transmitting serially. The way you had input data into your LFSRs from a ROM this week is an example of a serial input.
- ◆ Board sends a request and controller responds with its status.

- ◆ Data line when IDLE, is high.
- ◆ Bits are encoded, each 4µs long:


Establishing data communication: **Protocol Specifications Part II**



- ◆ Example, the above sequence is 0x01. This happens to be the request byte.
- ◆ Notice transmission always begins with a falling edge.
- ◆ MSB of the data is sent out first. The last bit is always a stop bit.
- ◆ After the request byte is sent, the controller will respond with a 32 bits of data and also a stop bit.
- ◆ Basically, each bit will correspond to a button.

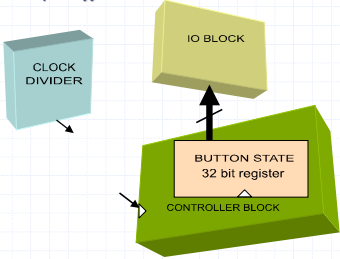
Dreamkatz **Controller Interface**

1. The N64 Controller
2. Physical interface
3. Communication protocol
- 4. Design structure and FSM**
5. Design implementation
6. Hints & cautions

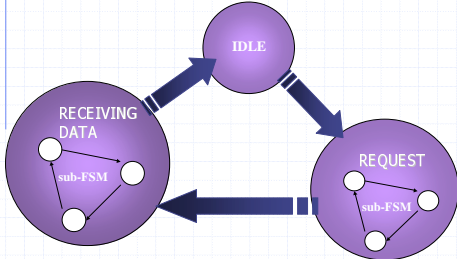


Design Structure

Τηρεε μαιν βλοχκς:



Controller FSM



FSM Timing

- ◆ Top-level FSM should cycle every 1ms as shown above.
- ◆ We are basically polling the controller.
- ◆ Thus, the idea is we have updated info on the controller button status every 1ms.

Transmission Timing

- ◆ Transmitting should be done in the request state. The byte we send is always the request byte, 0x01.
- ◆ How to generate logic 1 and logic 0?
- ◆ 4Mhz clock each clock cycle lasts 250ns.
- ◆ Thus, logic 0: 12 cycles low, 4 cycles high
- ◆ There is no idle time between consecutive bits.
- ◆ Remember: 9th bit must be a stop bit

- ◆ Receiving is much trickier.

Super-sampling

- ◆ Data rate is 250Khz, can we just stick the data wire to a shift register running on a 250Khz clock?
- NO!** Have to worry about phase shifts!
- ◆ Imagine the waveform going into your flipflops are all shifted in time!
- ◆ Communication between devices running on different clocks requires a special mechanism to synchronize receiving, known as SUPERSAMPLING.
- ◆ Watch the incoming wire at 16 times the send rate. (4Mhz vs. 250Khz)

Receive Timing

- ◆ Line when idle is high. The start of a data stream is characterized by a falling edge.
- ◆ We can use that edge for synchronization.
- ◆ Count 5 or 6 4Mhz clock cycles after a falling edge run and then sample the wire.

Quick pause... questions?

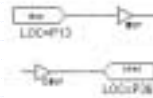
Dreamkatz Controller Interface

1. The N64 Controller
2. Physical interface
3. Communication protocol
4. Design structure and FSM
- 5. Design implementation**
6. Hints & cautions

I/O Block & Clock divider

◆ IO Block

- ◆ Make sure there's always a BUF corresponding to the type of PAD connected.



◆ Clock divider

- ◆ A counter can divide clocks by multiples of 2.
- ◆ Make sure you use a BUFG before sending the clock out.

Controller Block

- ◆ Recall that P41 is wired wrapped to the data line.
- ◆ That IOPAD serves as input and output of bits.



- ◆ In the RECEIVING DATA state, we must not be driving that node.
- ◆ We can use a TRI-STATE buffer to control what's driving that wire.
- ◆ The rest will be your FSM!

Hints & Cautions

- ◆ Don't make ad-hoc state machines!
- ◆ Counters and shift registers.
- ◆ Look at your error messages.
- ◆ Avoid the use of copy and paste in your schematics, ESPECIALLY IPADs and OPADs.
- ◆ Oscilloscope...knows all.(Unless its probes are broken...)

MUAHAHAHA....

