EECS 150 Spring 2004

Lab Lecture 8

Checkpoint2 Part1

3/12/2004

Greg Gibeling

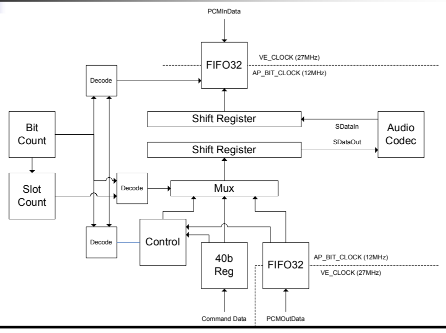
Today

- Checkpoint2
  - Functional Description
  - Block Diagram
- AC97 Audio
- Announcements
- Reset/Timing
- Shift Register
- Simulation Model

Checkpoint2 (1)

- Run audio through the board
  - Interface with AC97 Codec (LM4549A)
  - Initialize analog loopback
  - Volume controls (Part2)
  - Full digital loopback (Part2)
- Basically its just a little amplifier
  - But its fun to have a working checkpoint

Checkpoint2 (2)



Checkpoint2 (3)

- First Week
  - Send some AC97 commands
  - Get analog loopback running
- Second Week (after break)
  - Usable volume controls
  - Loop digital PCM data through the FPGA

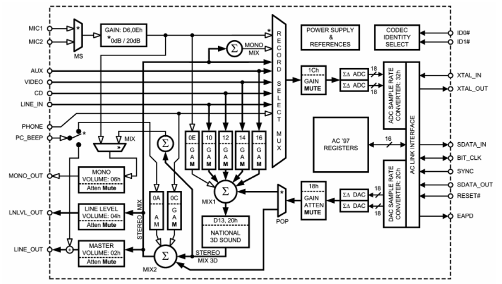
Checkpoint2 (4)

- Microphone input
- Line/Master Output
- Nothing fancy
  - Should be able to hear yourself
  - Just write to a few command registers
- Design for the future
  - Think ahead to the second half!

Checkpoint2 (5)

- To Do List (Not RTL or Bubble&Arc)
  - Ensure AP_RESET_ is held long enough
  - Continuously generate Sync signal
  - Wait for codec to be ready
  - Initialize registers
  - …
  - Send Volume Commands
  - Loop digital audio

The Codec Analog Path



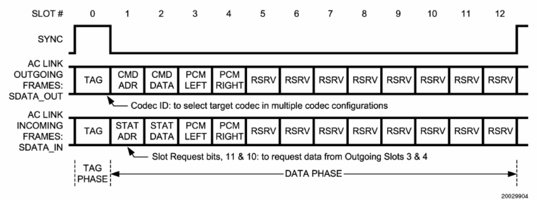
AC97 Digital Audio (1)

- Serial Bit Stream
- You must generate a sync signal
- Two Parts to AC97
  - Command Data (Read/Write Registers)
  - PCM (Wave) audio data
- The Chip
  - LM4549A Audio Codec
  - READ THE DATASHEET (Its short)

AC97 Digital Audio (2)

- Bit serial transmission
  - AP_SDATA_OUT is from FPGA to Codec
  - AP_SDATA_IN is from Codec to FPGA
- Data transmitted in 256b frames
  - 13 slots per frame
  - Slot0 is 16b
  - Slots1-12 are 20b
  - Sync signal high (roughly) during Slot0

AC97 Digital Audio (3)

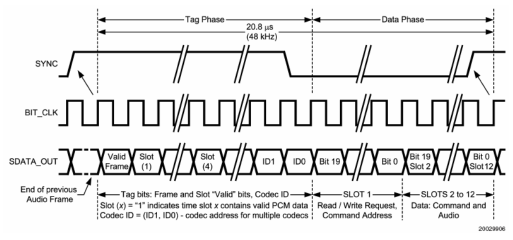


SLOT #
0
1
2
3
4
5
6
7
8
9
10
11
12
SYNC
AC LINK OUTGOING FRAMES: SDATA_OUT
TAG
CMD ADR
CMD DATA
PCM LEFT
PCM RIGHT
RSRV
Codec ID: to select target codec in multiple codec configurations
AC LINK INCOMING FRAMES: SDATA_IN
TAG
STAT ADR
STAT DATA
PCM LEFT
PCM RIGHT
RSRV
Slot Request bits, 11 & 10: to request data from Outgoing Slots 3 & 4
TAG PHASE
DATA PHASE
20029904

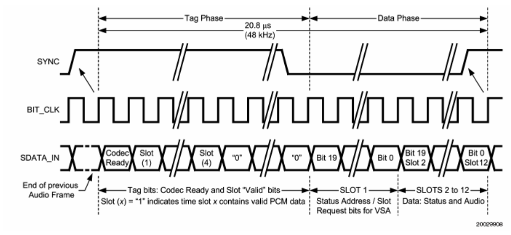
AC97 Digital Audio (4)

- Slot0 - Tag
  - Bit[15] - 1 is frame is valid
  - Bits[14:11] - 1 if corresponding slot is valid
  - Bits[10:2] - Not Used
  - Bits[1:0] - ID, set to 2’b00
- Slot1 - Register Address
- Slot2 - Register Data (for writes)
- Slots3&4 - PCM Audio Data

AC97 Digital Audio (5)


Tag Phase
Data Phase
20.8 µs (48 kHz)
SYNC
BIT_CLK
SDATA_OUT
Valid Frame
Slot (1)
Slot (4)
ID1
ID0
Bit 19
Bit 0
Bit 19 Slot 2
Bit 0 Slot 12
End of previous Audio Frame
Tag bits: Frame and Slot "Valid" bits, Codec ID
Slot (x) = "1" indicates time slot x contains valid PCM data
Codec ID = (ID1, ID0) - codec address for multiple codecs
SLOT 1
Read / Write Request, Command Address
SLOTS 2 to 12
Data: Command and Audio
20029906

AC97 Digital Audio (6)



Announcements (1)

- Lab Policy Reminder
  - Blue bins are for recycling
  - Gray bins are for trash
  - Pick up after yourselves
  - No eating at the stations
  - Not even snacking
- Enforced by account suspension

Announcements (2)

- Bathrooms
  - Don't use the wrong bathroom
  - Even while the custodial staff is cleaning
  - Use another floor or building
  - This applies to both men and women
- Enforced by **ARREST BY UCPD**

Announcements (3)

- Check the website every 24hrs
  - Really read the news
  - We post important information there
- Come to lectures
  - They're for YOU, not our amusement
  - If the lab is unclear WATCH THE WEBCAST
  - Many people are asking questions we answered in lab lecture

Announcements (4)

- Securing the website
  - You'll be able to get all the information
  - …including the IEEE Verilog Standard
  - From anywhere!
- Logging In
  - Username: cs150-xxx
  - Password: SID (not your windows password)

AC97 Reset (1)

- Watch the Reset!
  - AP_RESET_ is active LOW
  - It will disable AP_BIT_CLOCK
  - You must hold reset for 1us
  - How can you deal with this?
- Use two clocks
  - Generate AP_RESET_ using the other clock

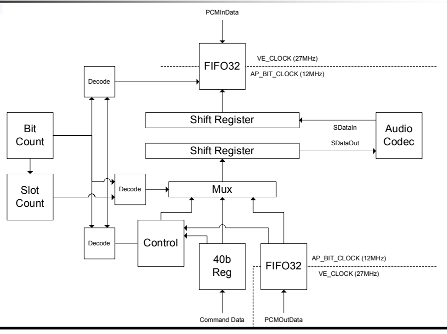
AC97 Reset (2)

- Some Register Need Async Reset
  - always @ (posedge BitClock or posedge BitReset)
- You should generate a "Long" reset
  - Goes active before AP_RESET_
  - Doesn't go inactive until much later
  - Avoids async logic (we don't like it)
- How do you do this?
  - Delays!

AC97 Sync

- First high SAMPLE marks start of frame
  - Not the first time you drive it high
  - What's the difference?
- Should be high for 16 cycles
  - All during Slot0
- This is CRITICAL
  - Without proper sync the codec won't work

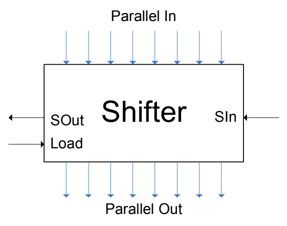
Checkpoint2 (6)



Shift Registers (1)

- Shift Registers!
  - Very simple module
  - Very powerful, can be used for in and out
  - Shift towards high bit
- I/O
  - SIn, SOut – Serial in and out
  - POn, Pout – Parallel in and out
  - Load – Load PIn into the register

Shift Registers (2)



Shift Registers (4)

```
input           SIn, Clock;
output[3:0]  POut;
output          SOut;
reg     [3:0]  POut;
assign          SOut =          POut[3];
always @ (posedge BitClock) begin
        POut <=                 {POut[2:0], SIn};
end
```

Shift Registers (5)

- For part 1 you only need 1 SR
  - Takes 40b and shifts them out
- The only trick is to load the SR!
  - Basically one load per slot/pair of slots
  - You just prepare the right data with a mux
  - Load the shift register at the right time
  - And watch it go…

Volume Controls (1)

- Its backward
  - High numbers mean quiet
  - Low numbers mean loud
- Your controls should have:
  - Up (Bit 1)
  - Down (Bit 0)
  - Mute (Bit 2)
- See FPGA_TOP…

Volume Controls (2)

- Inputs: Up/Down/Mute/Updated
- Outputs: Setting/Changed
  - Up/Down/Mute causes a change
  - Setting changes
  - Changed goes high
  - Changed goes low on Updated
- Why?
  - What if changes happen back to back?

Testing (1)

- We built you a tester!
  - AC97Codec.V
  - This is for simulation ONLY
  - It error checks the AC97 data (somewhat)
- Limitations
  - The tester will not catch everything
  - It may in fact have bugs
  - It's a lot better than nothing

Tester (2)

- What does it do?
  - Sends PCMInput.txt into your design
  - Writes the output to PCMOutput.txt
  - Displays control register writes
  - Checks for many timing mistakes
  - Watches for common bugs
  - GENERATES THE CLOCK

Tester (3)

- PCMInput.txt
  - 10 hex digits per line
  - PCMLeft is the left 5 digits
  - PCMRight is the right 5 digits
- PCMOutput.txt
  - Identical format
  - If you do loopback the files will match

And now…

- Read the datasheet
  - Pages 2, 15 and 16-23
- Don't be afraid to test things
  - We put a lot into that tester
  - It might be buggy though
- Remember: This checkpoint is NEW
  - That means we're nearly as clueless as you
  - Watch the website for updates