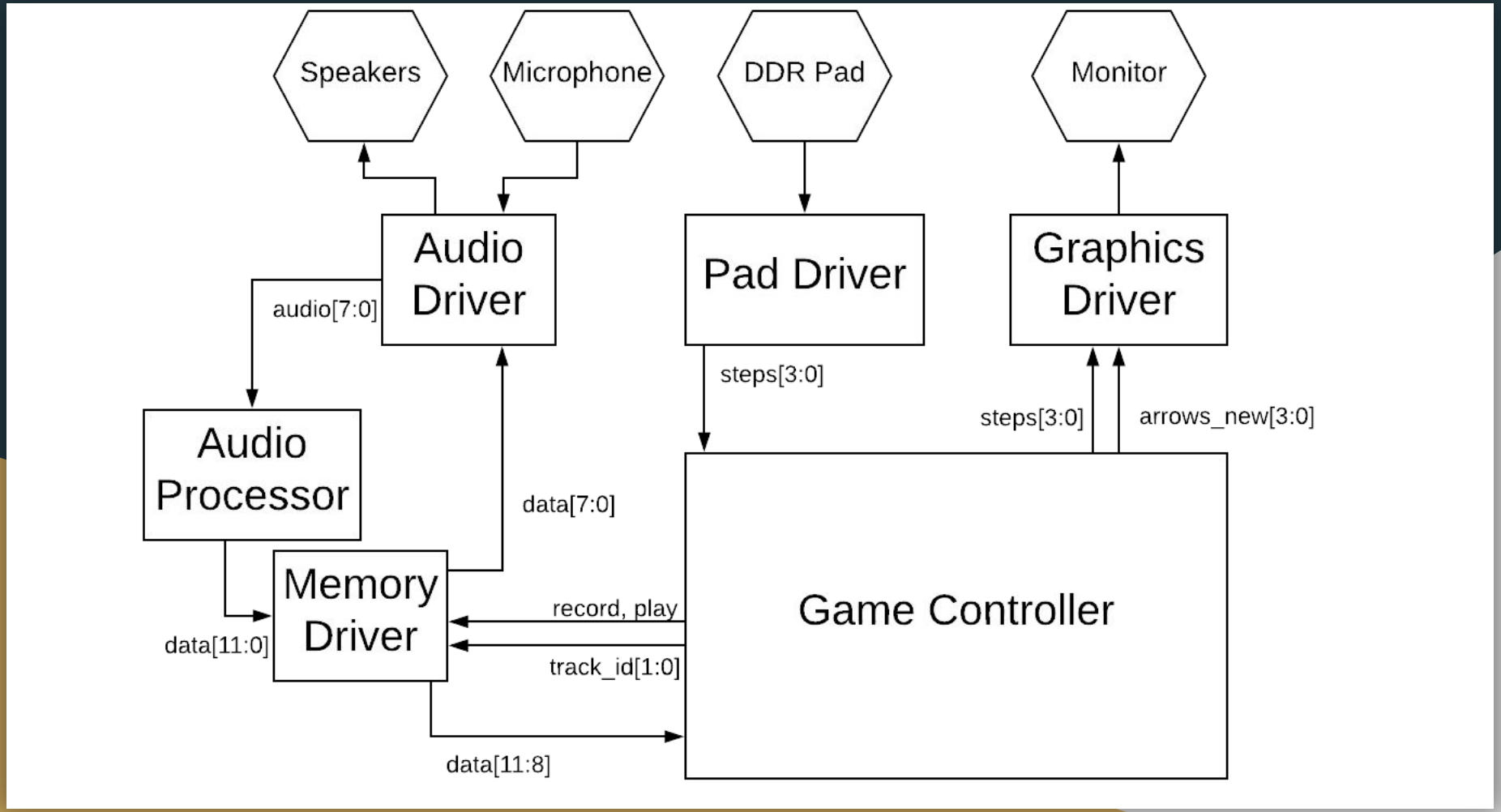
F.R.E.D.D.Y

FPGA Required Electronic Digital Dance Yackadoodle

Jamie Bloxham & Matthew Tung

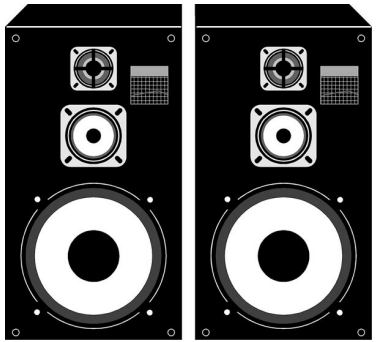
TL;DR : FPGA Powered DDR (Dance Dance Revolution)

- Arcade Game where players hit steps for points
- Match specific steps displayed on screen to music
- All about rhythm, reaction, and speed

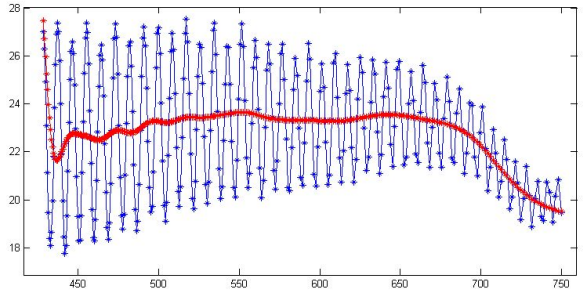

Speakers
Microphone
DDR Pad
Monitor
Audio Driver
Pad Driver
Graphics Driver
audio[7:0]
steps[3:0]
steps[3:0]
arrows_new[3:0]
Audio Processor
data[7:0]
Memory Driver
record, play
Game Controller
data[11:0]
track_id[1:0]
data[11:8]

Audio Driver

- Receives and outputs audio data
- As in Lab 5, 8-bit audio bit depth @ 6 kHz
  - **Strongly** subject to change
- Input: standard microphone, play music with a third-party device
  - Stretch goal: rework data flow to support input from e.g. a USB thumb drive
- Output: standard speakers

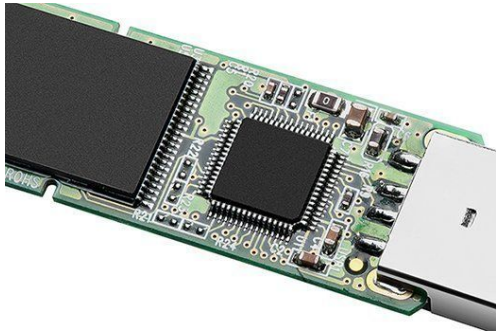


Audio Processor



- Input: audio stream
- Output: audio stream augmented with additional 4 bits for arrow data
- Step one: extract features from audio stream
  - Primary goal: find low frequency energy peaks, by passing signal through LPF
  - Stretch goal: FFT analyses, e.g. tempogram
- Step two: convert features into arrows
  - Primary goal: at low frequency energy peaks, insert a random arrow
  - Stretch goal: feature-informed arrow selection, difficulty settings

Memory Driver



- Flash memory interface
- Data is concatenation of audio stream, and 4-bit arrow stream
  - i.e. stores 12 bits per timestep, for 8-bit audio bit depth
- As in audio lab 5, keeps track of current position in song
- 2 States (Recording and Playback)
  - In recording, writes audio-arrow-stream from Audio Processor to memory
  - In playback, reads audio-arrow-stream from memory, sends audio stream to Audio Driver, sends arrow stream to Game Controller
- Multiple “slots” for tracks, track slot is specified by Game Controller

Game Controller

- Main Controller of the Game
- Tells Memory Driver when to record or…PLAY!
- Regularly takes 4 bit step data from Memory Driver
- Sends commands and data to Graphics Driver to display
  - Position of falling arrows/steps
  - Score (Stretch Goal : Combos and Streak Multipliers)
  - Scoring Region (Where you should match the arrows to as they fall)
- Checks from Pad Driver if steps have been matched based on pixel pos.
  - If so, increments score value according and the falling arrows disappear
  - If not, falling arrows go away to bottom of screen
  - Stretch Goal: have different scores based on accuracy (OK, Good, Perfect, etc.)

Graphics Driver

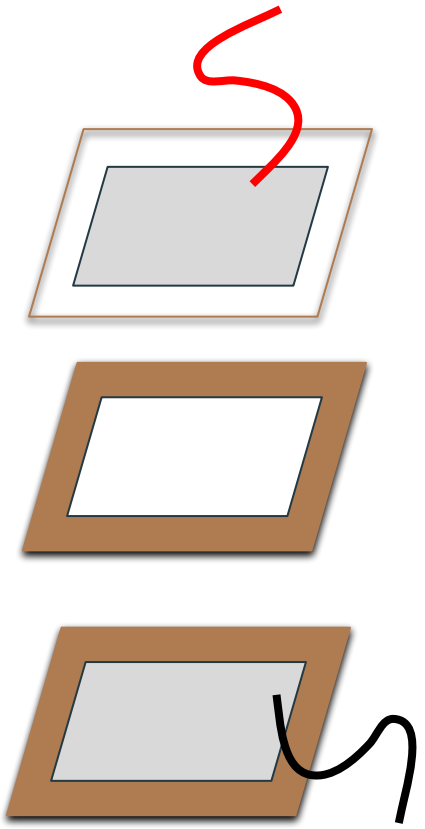
- Displays all data sent from Graphic Drivers
  - Falling arrows/step to match
  - Outline of arrows for where to match
  - Score
- Uses COE image data to display on monitor
  - Via VGA

Pad Driver

- Takes signals sent from DDR Step Pad
- Debounce and send to Game Controller for gameplay

DDR Step Pad

- 4 Switches
- Each switch is an arrow/step
- Pressure Plate design
  - When metal plates connect, circuit is completed
    - Pull-up Resistor



Challenges & Stretch Goals

- Synchronization between many different modules
- “Good” arrow pattern generation
- Allowances for imperfectly timed player input

- Audio input via USB
- FFT audio processing, tempograms, etc
- Configurable difficulty
- Scoring enhancements (Bad/Good/Perfect, streaks, etc)
- UI

Timeline

- 11/5 - 11/11 : Got all desired images to display and move
- 11/12 - 11/18 : Get Flash Memory and Base Game Functionality
- 11/19 - 11/25 : Implement Audio Processor and Complete Game Controller
- 11/26 - 12/2 : Complete Audio Processor/Game Functionality and Integrate Game Pad
- 12/3 - 12/9 : Debug and work on Stretch Goals

Questions?