Black Duck Down

ECE532 Final Demo

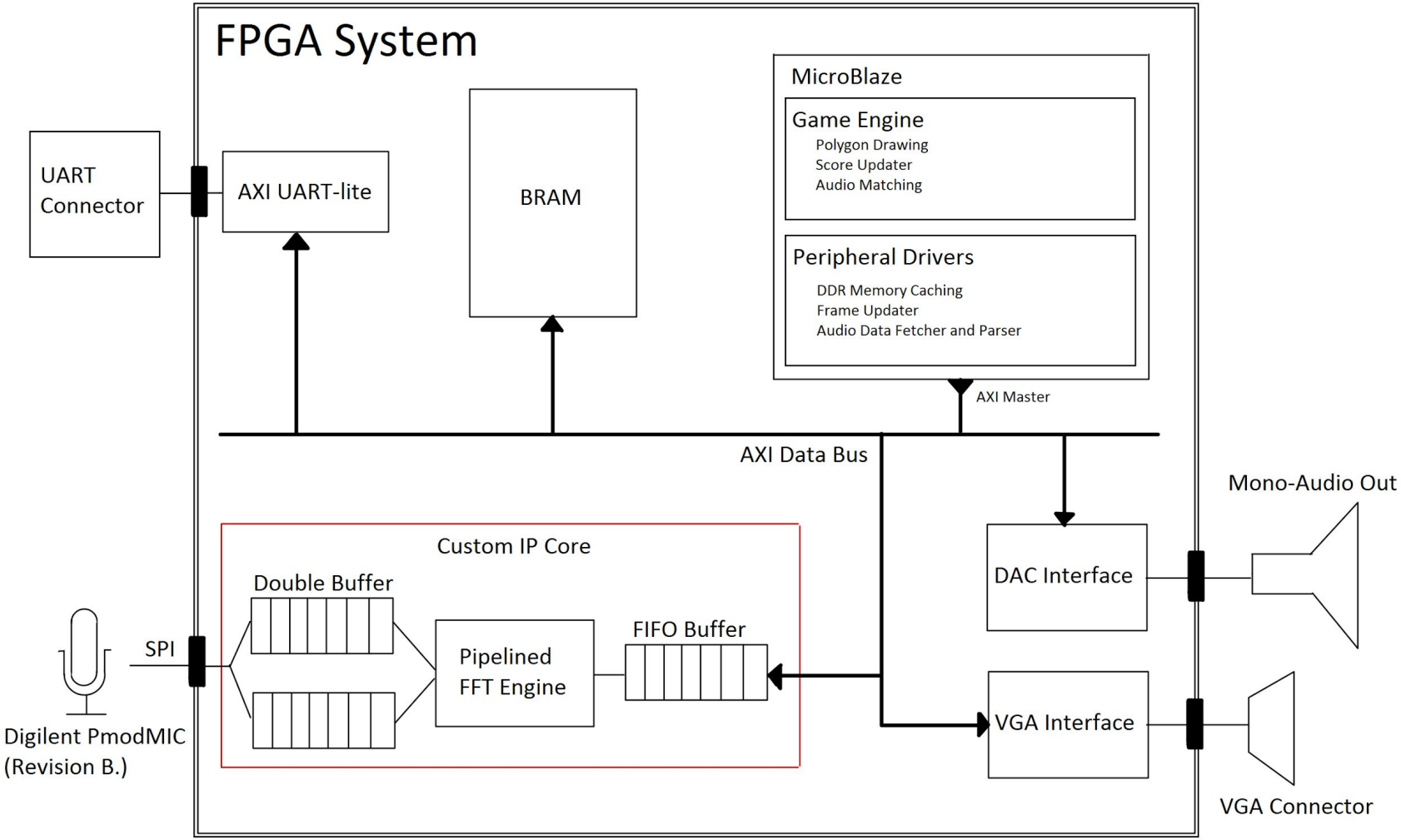
Wen Bo Li
Jianwei Sun
Wenyi Yin

Intro-duck-tion

- Black Duck Down is a voice-training game.
- Player sings specific notes to down the duck.
  - Trains note-hitting ability for singers.
- During gameplay, ducks appear at random on the screen.
- Each duck corresponds to a unique note in the range C2 - C5
-
- Players who are able to find pitches faster score higher!

Why Black Duck Down?

- FFTs are popularly implemented in FPGAs due to speed.
- Practice how to build an FFT.
- At high level: A fun, useful, but feasible project.
- But something different from guitar hero/rock band.
    - Hence the randomly-appearing notes.


FPGA System
MicroBlaze
Game Engine
Polygon Drawing
Score Updater
Audio Matching
Peripheral Drivers
DDR Memory Caching
Frame Updater
Audio Data Fetcher and Parser
AXI Master
UART Connector
AXI UART-lite
BRAM
AXI Data Bus
Mono-Audio Out
Custom IP Core
Double Buffer
FIFO Buffer
Pipelined FFT Engine
DAC Interface
SPI
VGA Interface
Digilent PmodMIC (Revision B.)
VGA Connector

Design Process

1. Together, decided on interfaces for all components
2. Then split up FFT, VGA, and mic IPs across members
3. Wrote software in parallel
4. Integrated all components

We wrote all code from scratch, aside from auto-generated AXI peripheral verilog.

Major Problems/Changes

FFT:

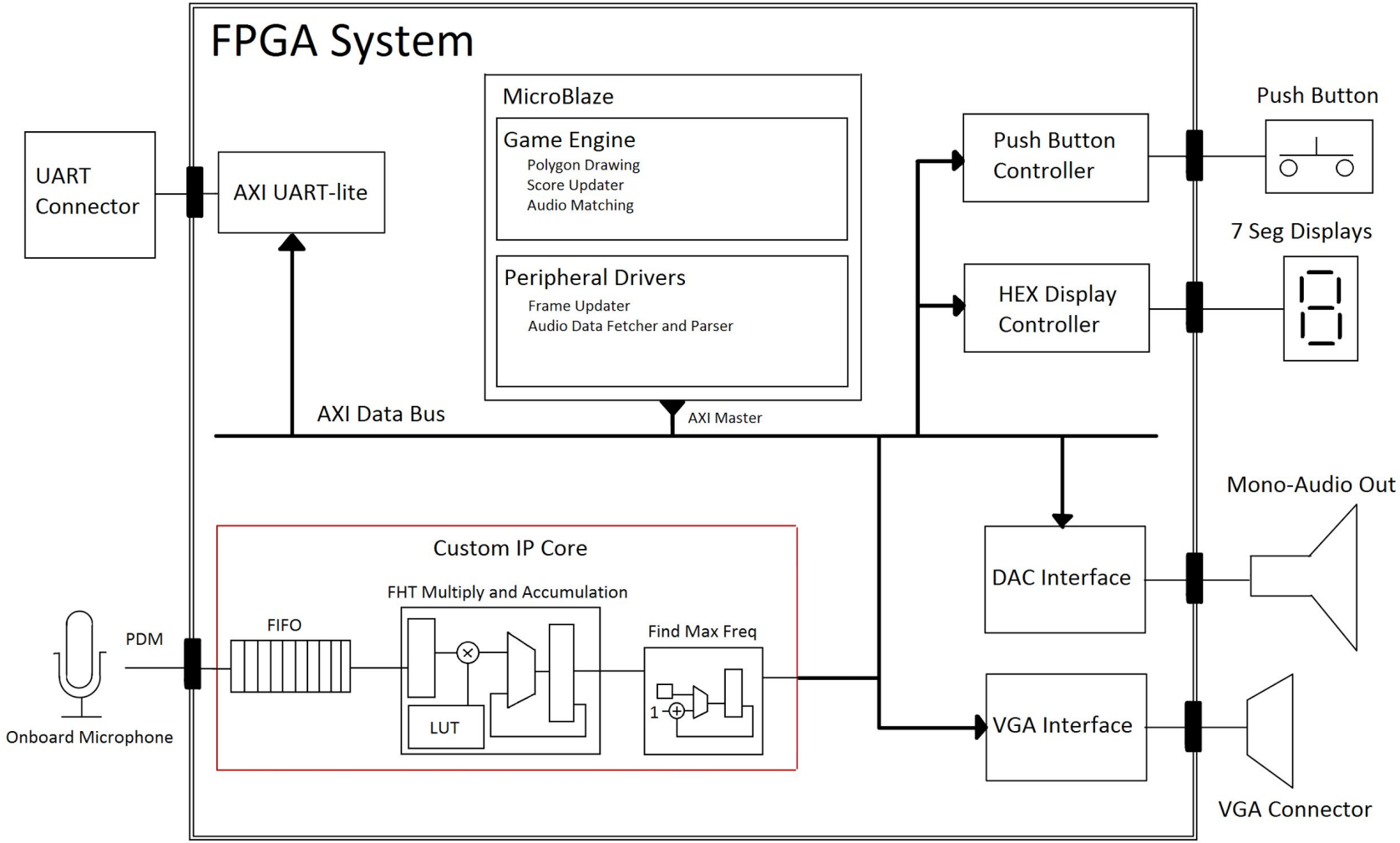
- Hard to implement complex arithmetic, built FHT instead
- Trouble meeting timing requirements. Pipelined operations, reduced resolution

VGA:

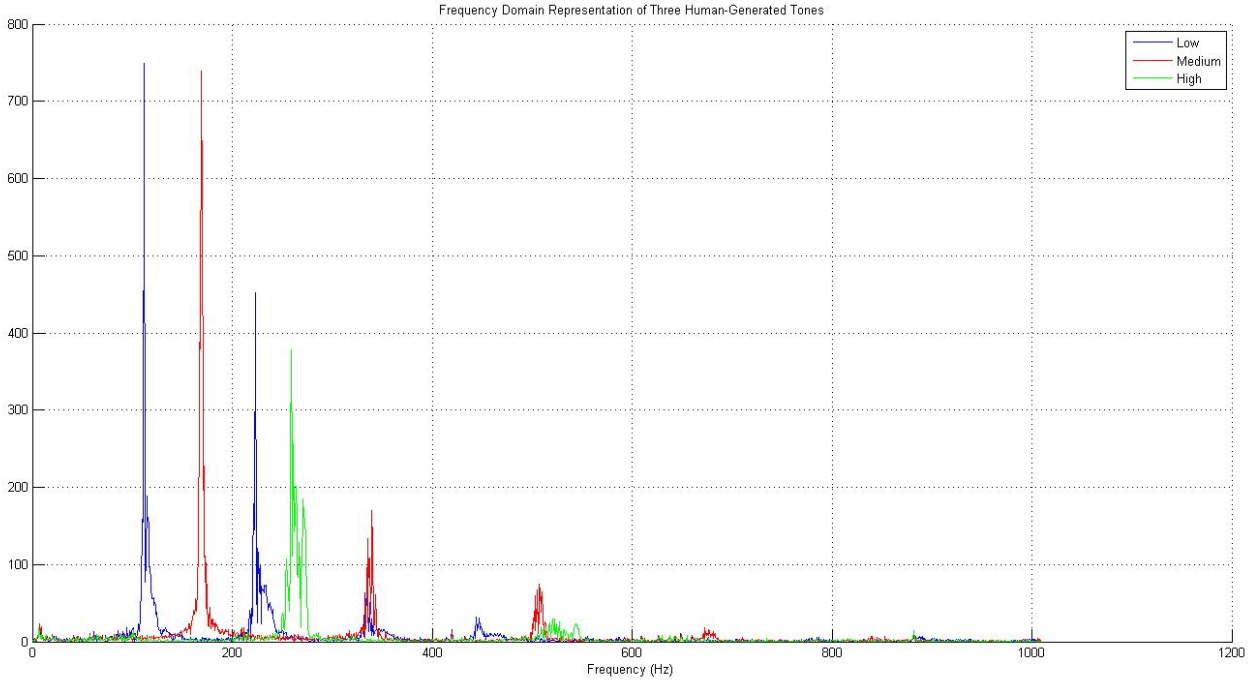
- Simplified graphics due to time tradeoff - animated ducks to motionless ducks
- Specific locations were assigned to specific notes, rather than a dynamic scheme

Software

- Had to average frequency info from FHT due to noisy data
- Had various issues with memory sizes and SDK errors


FPGA System
MicroBlaze
Game Engine
Polygon Drawing
Score Updater
Audio Matching
Peripheral Drivers
Frame Updater
Audio Data Fetcher and Parser
UART Connector
AXI UART-lite
AXI Data Bus
AXI Master
Push Button
Push Button Controller
7 Seg Displays
HEX Display Controller
Mono-Audio Out
DAC Interface
VGA Interface
VGA Connector
Custom IP Core
FHT Multiply and Accumulation
FIFO
LUT
Find Max Freq
1
PDM
Onboard Microphone

Fast Hartley Transform (FHT)



Fast Hartley Transform (FHT) - Continued

- Integration of PDM data to convert to time series amplitude audio data
- Audio data stored into a queue. Start of FHT takes a snapshot of the queue
- Uses FHT instead of FFT due to arithmetic with real numbers, and no requirement for phase information
- Avoids the use of floating point operations by scaling all numbers by a large integer, making use of primitive DSP48 Blocks
- Look-Up-Table of constants for computing the Hartley transform, initialized with readmemh() command
- 8 - stage pipelined, with FSM, implementation ensures fast operation
- Usage of RAMB18/36 primitives ensures low resource utilization

Display Logic (VGA)

AXI slave peripheral: outputs are colours, HSYNC, VSYNC

- Basic VGA module generates sync signals for 1280x1024 display at a 60Hz refresh rate (100 MHz pixel clock)
- Use sync signals to keep track of what position you are at on the screen
- Screen is segregated into 256 80x64 pixel blocks (16 in each direction), each block has a corresponding register in the peripheral
- Store one 80x64 array of 12-bit colours per type of picture (duck, score digit, etc)
- Depending on position on screen and values in registers, select main output colour from one type of picture by indexing into the correct array

Software

- Averages last 32 frequency readings from the FHT to determine current frequency.
  - Prevent detection of frequency jittering.
- Kills and randomly spawns ducks in O(1) time using smart data structure.
  - 2 arrays: 1) state array for all ducks, 2) dead duck array for only dead ducks.
- Frequency to note mapping.
  - As note increases linearly, frequency grows exponentially.
- Other minor features.
  - Score recording and display.
  - Hex display of note and score.
  - Speeds up duck spawning over time.
  - Push-button start and re-start of game.

What we learned

- Detailed documentation and communication is important.
  - Easy to have misunderstandings/disagreements on game design details.
- Debugging software on FPGAs is fraught with danger.
  - Memory constraints.
  - Memory constraints also => function constraints.
  - Google and trial & error is what got us through.
- Examples
  - i = 0 was not registering until stack was increased.
  - rand() and printf() do not work - need lighter functions.

Demo

Thank you