6.111 Final Project: Digital Debussy- A Hardware Music Composition Tool

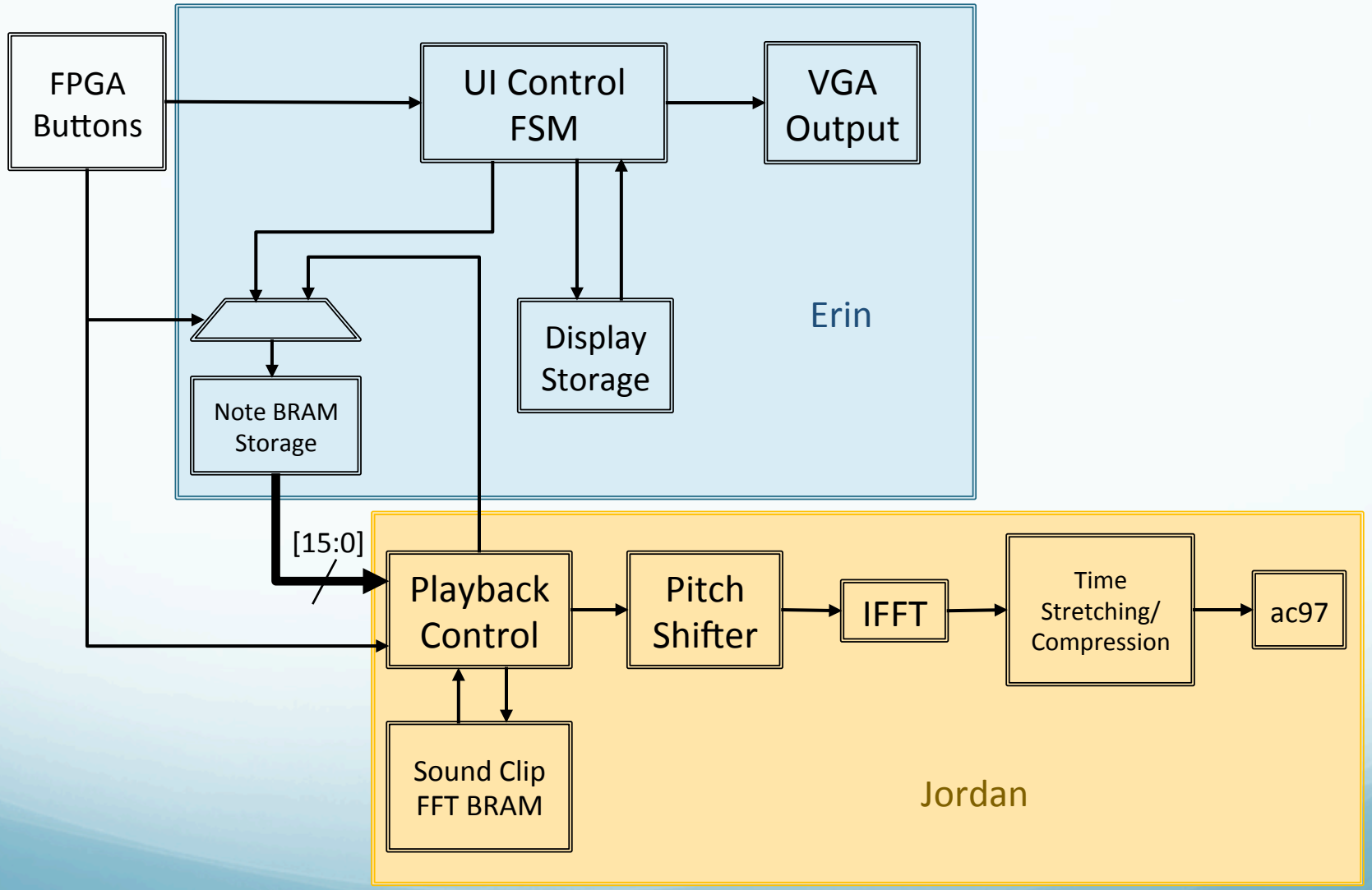
Jordan Addison and Erin Ibarra
November 6, 2014

Purpose

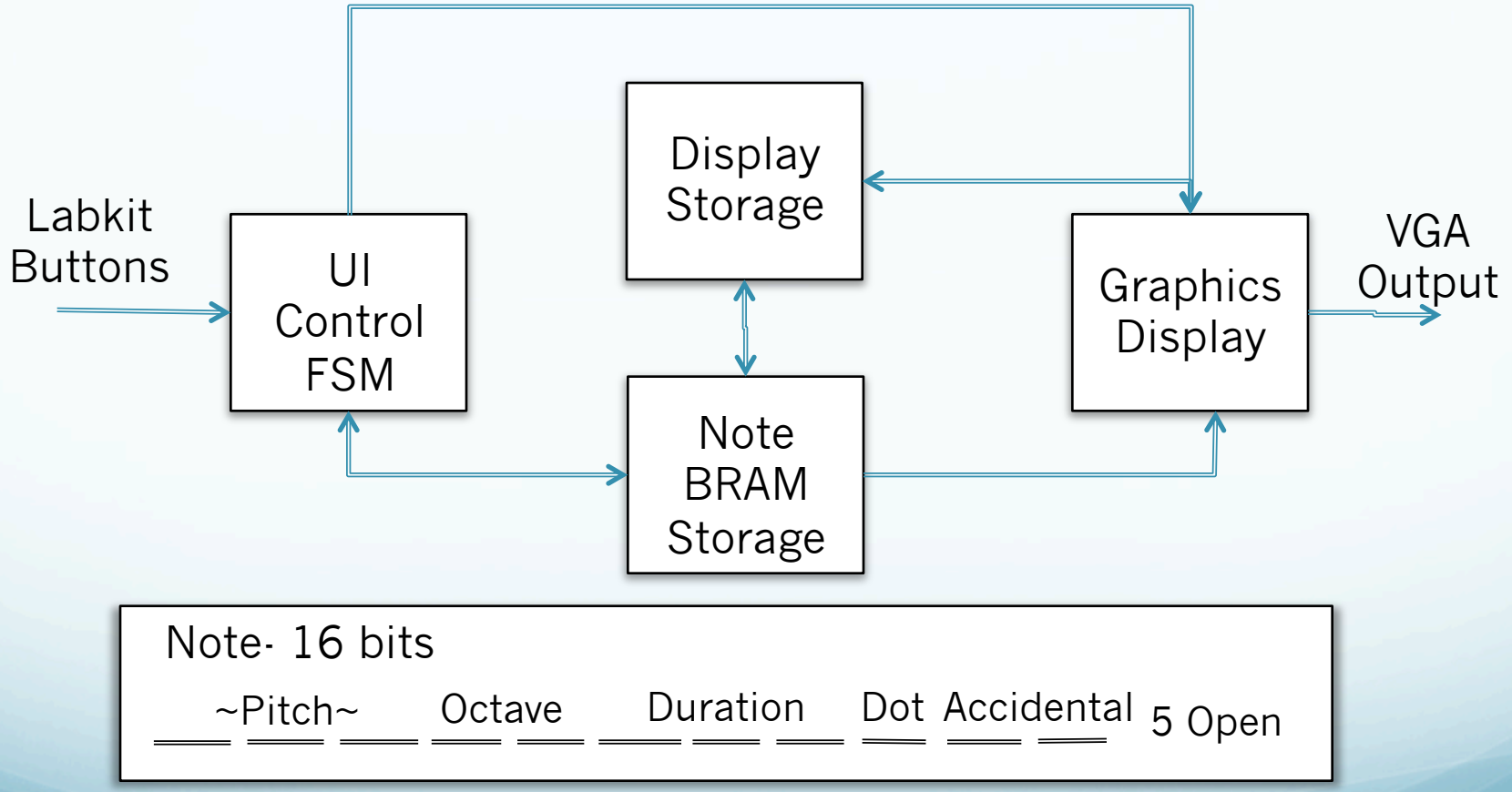
- Professional music composition software is expensive
  - $150-$600, typically between $300-$600
  - Amateurs, beginners do not need the power and functionality of professional software
- Very little information on internal details of music composition software

Outline of Minimum Features

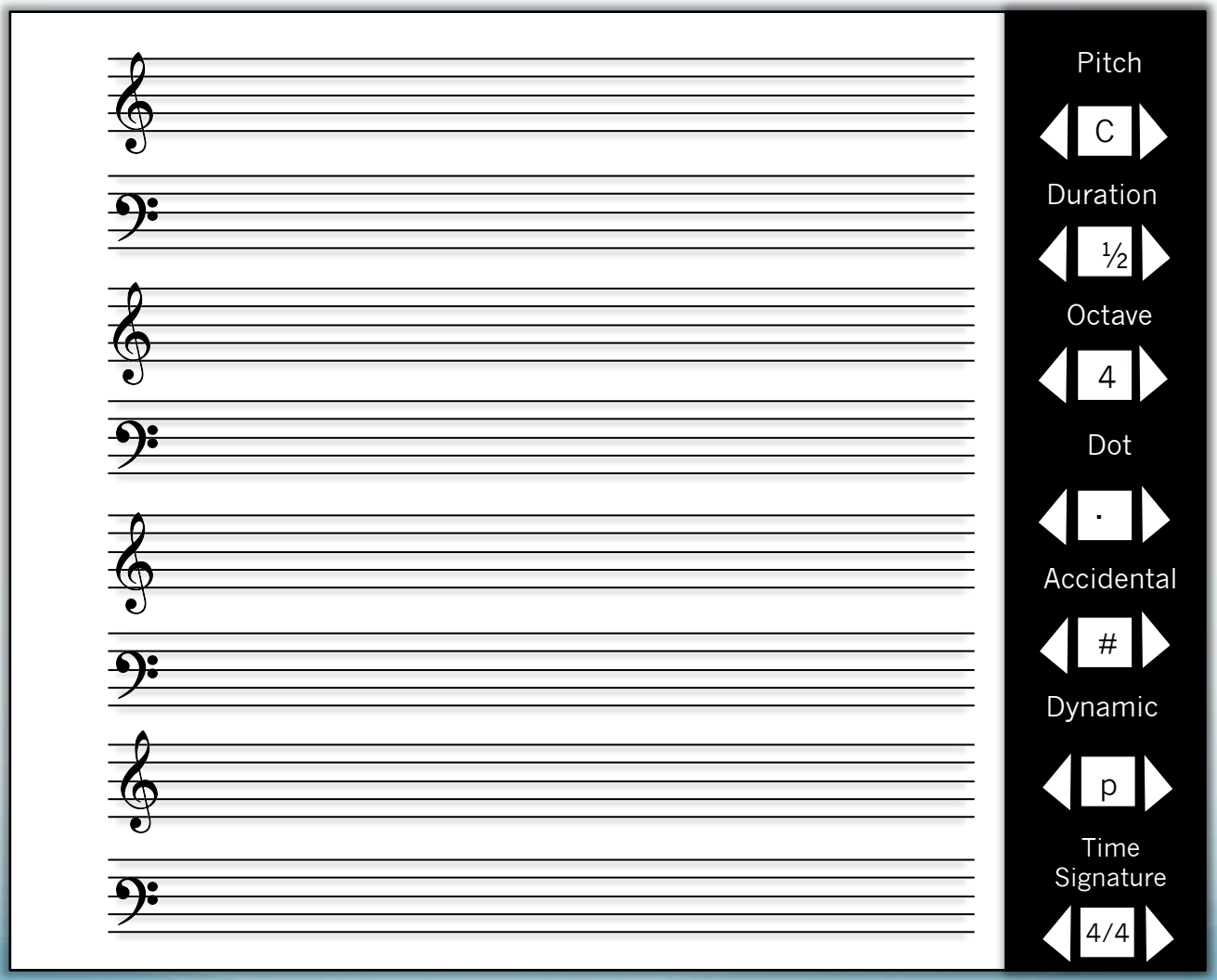
- Single note melodies with up to 64 individual notes and playback
- Bass and treble clefs
- Two time signatures : 3/4, 4/4
- Transcribe pitches from C3-C5
  - One octave below Middle C to one octave above Middle C
- 1 Key Signature (C major) but all flats and sharps available as accidentals
- Note duration from whole to sixteenth notes
- Dynamic volume from piano to forte


FPGA Buttons
UI Control FSM
VGA Output
Erin
Display Storage
Note BRAM Storage
[15:0]
Playback Control
Pitch Shifter
IFFT
Time Stretching/ Compression
ac97
Sound Clip FFT BRAM
Jordan

User Interface Block Diagram



Graphical User Interface



User Interface: Transcription Process

- User Interactions
  - Music transcribed from left to right using labkit buttons
  - Select parameters on toolbar
  - Switch between “Staff” and “Toolbar” modes using “Enter” button
  - Place a note based on chosen parameters by pressing “1” button in staff mode
- Display
  - Entered notes are stored in note memory
  - Addresses of notes and duration values for corresponding notes are stored in measure counter
    - Measures are divided accordingly on display
  - Pitches are displayed as sprites and mapped to appropriate location on staff

UI Testing

- Using generated note files
- Test benches with simulated button pushes
- Visual inspection

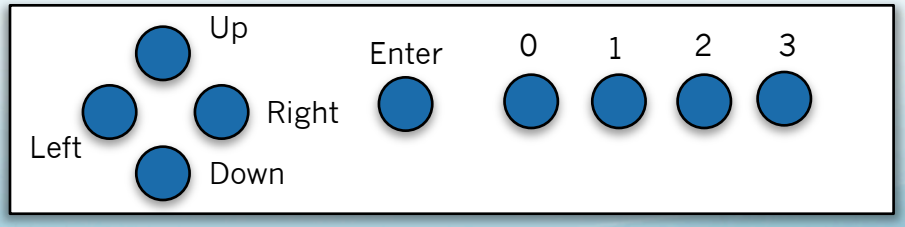
Button Functionality

Toolbar

- **Up/Down:** cycle through attributes (pitch, duration, etc.)
- **Left/Right:** cycle through values of a specific attribute (i.e. pitch: A,B,C, etc.)
- **Enter:** switch to staff mode
- **3:** playback (press and hold)

Staff

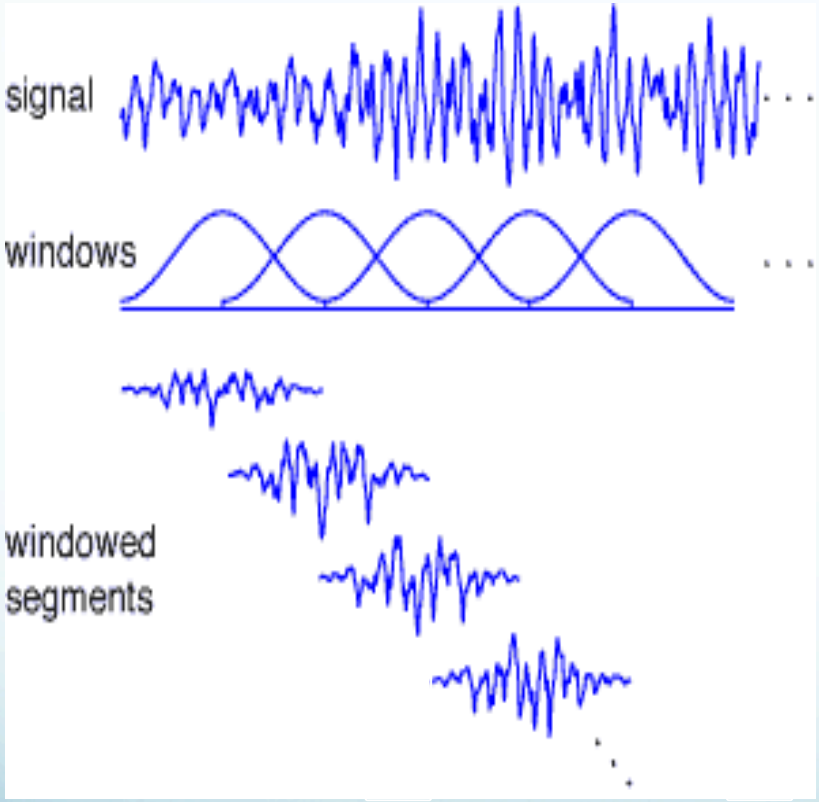
- **Left/Right:** scroll through existing notes on the staff
- **0:** erase the highlighted note (turns into a rest)
- **Enter:** switches to toolbar mode
- **3:** playback (press and hold)
- **1:** place a note



Pitch and Speed

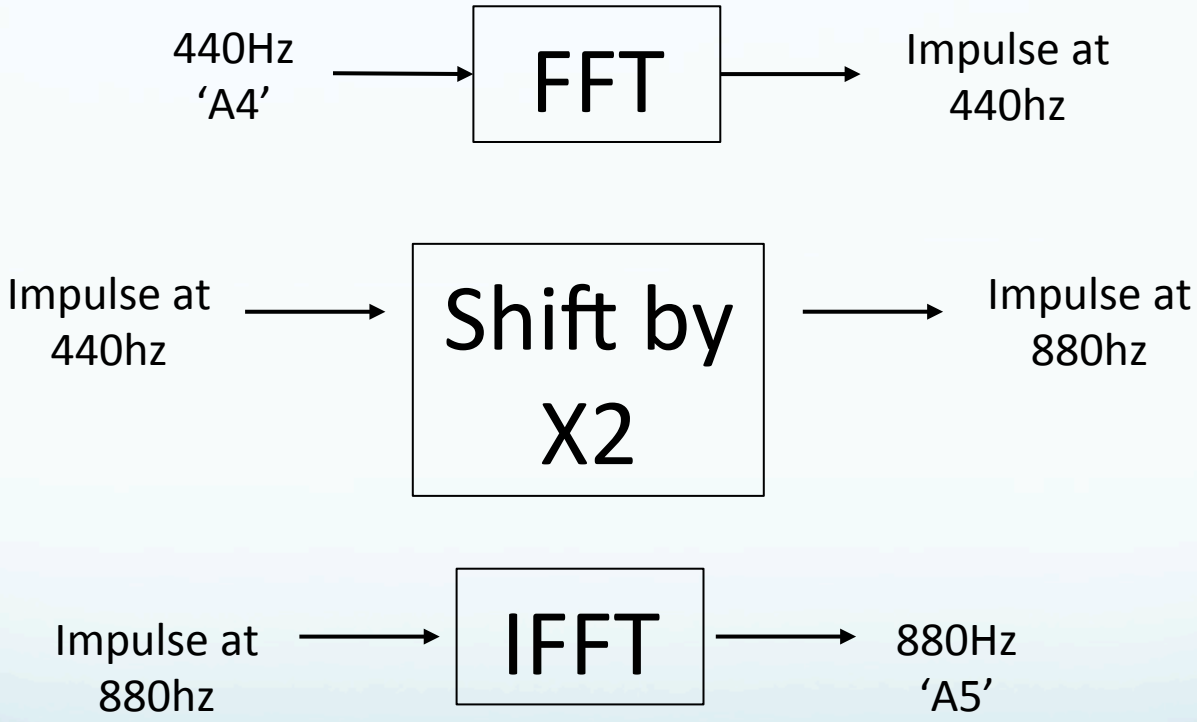
- Seem to be inherently linked
- Higher Speed → Higher Pitch,
  Slower Speed → Lower Pitch
- Standard Solution:
  - Window audio sample
  - Analyze pitch in frequency domain
  - Analyze tempo in time domain
- “Phase Vocoder”

Controlling Speed

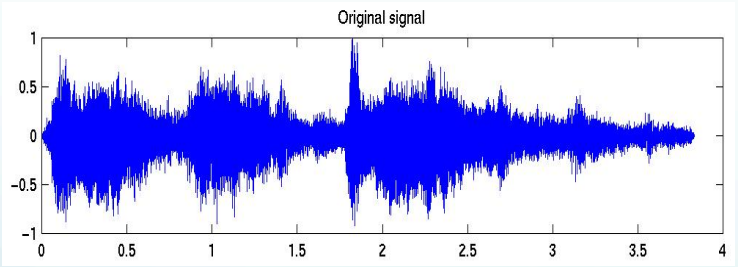


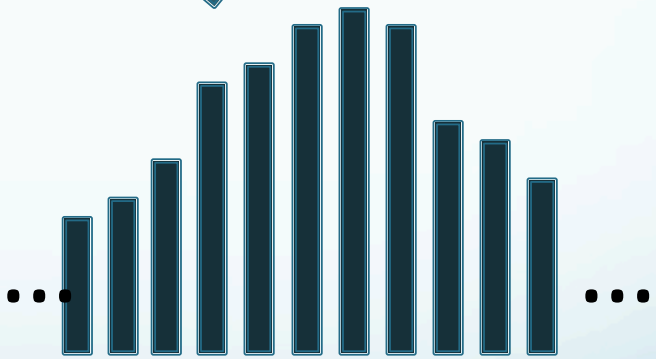
- To change playback speed, change amount of overlap between windowed segments
- No effect on pitch
- Can have other effects on the sound

Pitch Shifting: Concept



Controlling Pitch: Issues

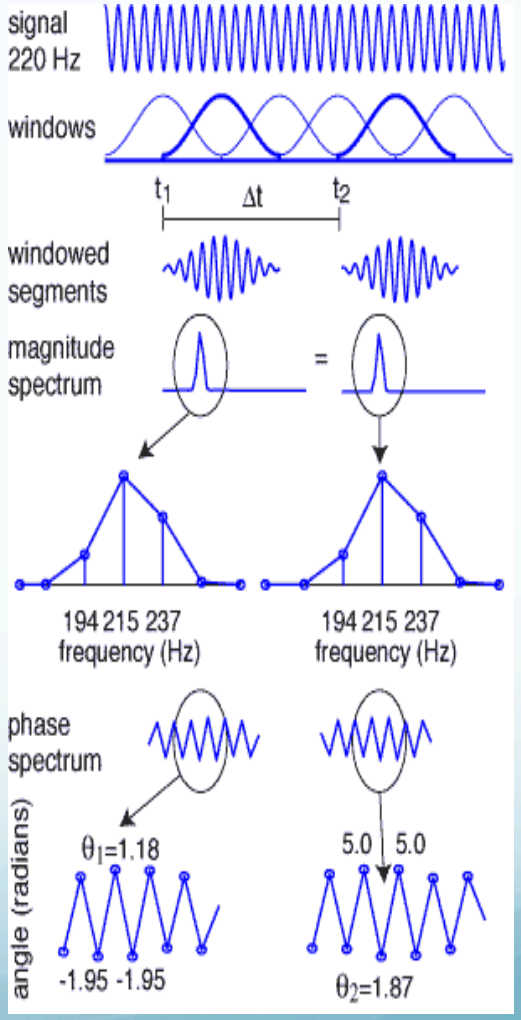




Frequency Bins

Controlling Pitch: Solutions

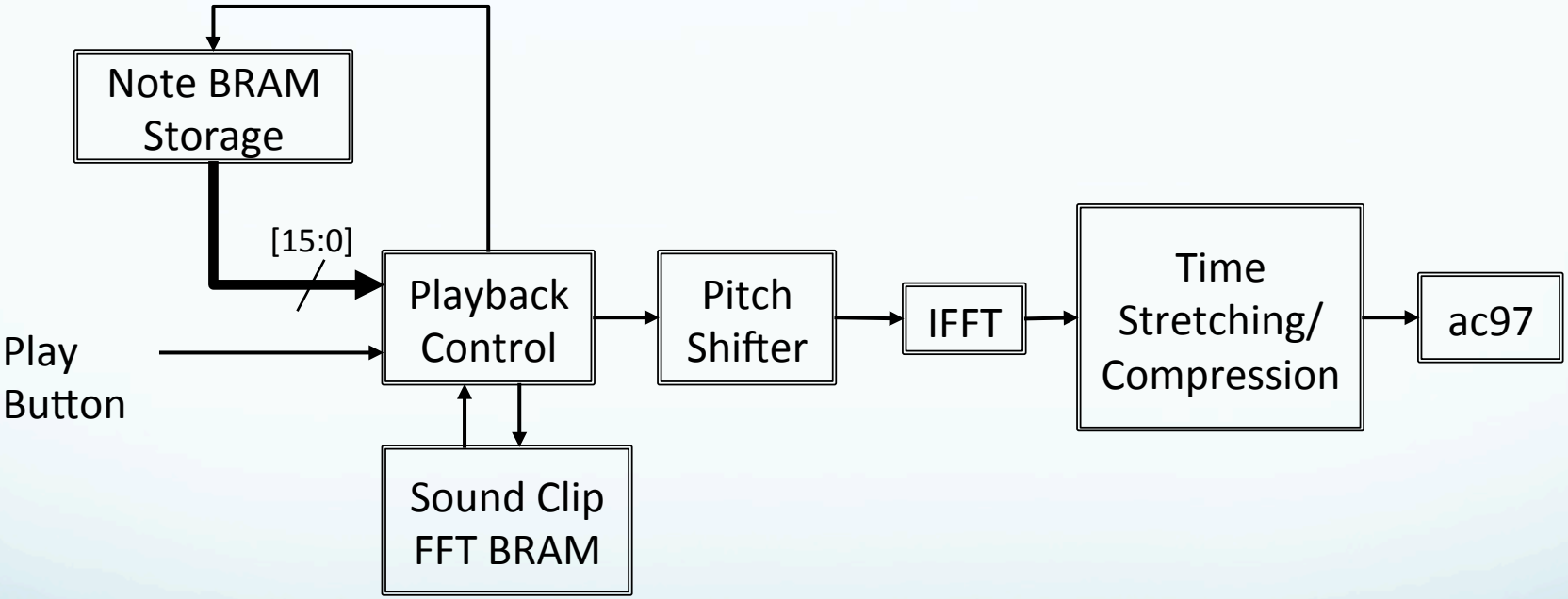
- Audio signals are “short-time stationary”
- Adjacent signals are NEARLY identical
- But differ enough to use the phase difference to extract a very precise estimate of the “true frequency” of a bin
- Without phase information, looks like a 215Hz signal (5Hz error)
- With phase info, estimated frequency is 220.0219! (Δt = 3ms)



Implementation Scheme

- Compute FFTs of sound clips ahead of time in MATLAB
- Store FFT data in FPGA BRAM
- Implement pitch shifting/time dilation on FPGA
- Other considerations:
  - Proper note shaping (Attack, sustain, release), will likely use simple shaping filters if needed
  - Pitch shifting only has a 2 octave range
  - Time scaling is quite limited with this implementation

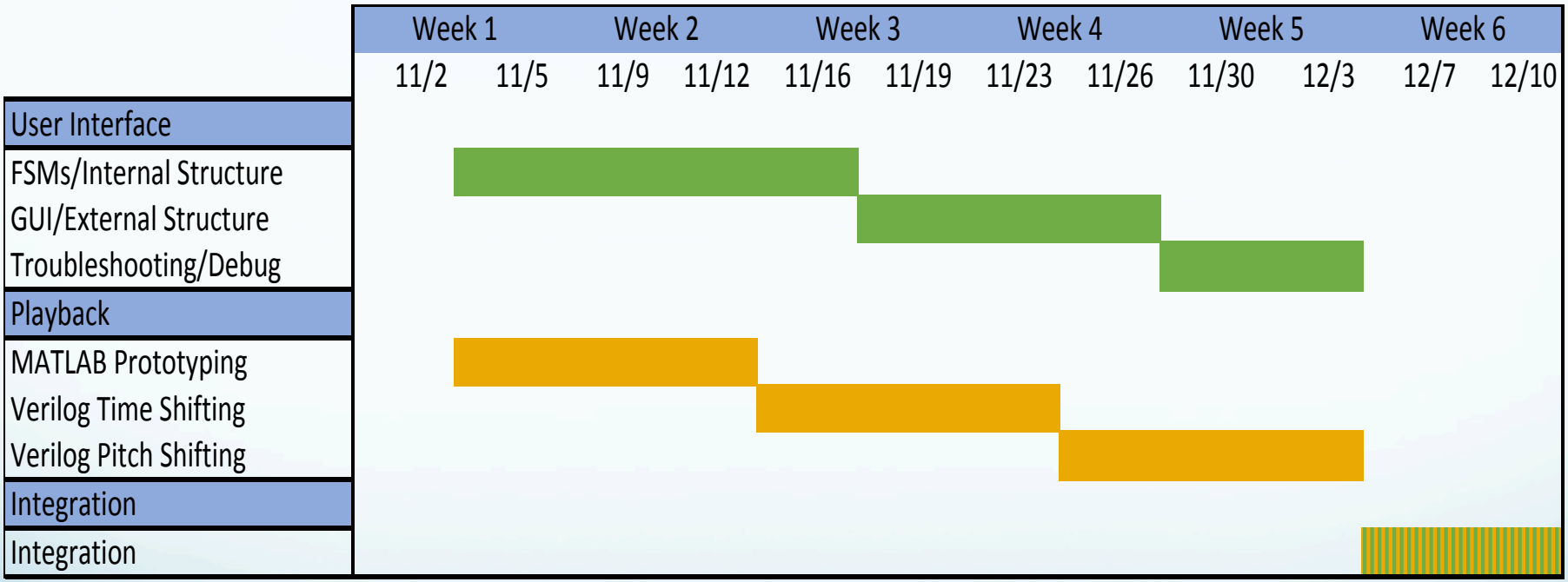
Playback Architecture Overview



Playback Testing

- MATLAB generated test files to put in BRAM
- Use these for unit testing
  - Time Shift
  - Pitch Shift
  - Volume Control

Timeline


Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
11/2
11/5
11/9
11/12
11/16
11/19
11/23
11/26
11/30
12/3
12/7
12/10
User Interface
FSMs/Internal Structure
GUI/External Structure
Troubleshooting/Debug
Playback
MATLAB Prototyping
Verilog Time Shifting
Verilog Pitch Shifting
Integration
Integration