Modular Synthesizer
6.111 DESIGN PRESENTATION

MIKE MILLER
TEJA VISHWANADHA
ANDREW MUTH

12 NOVEMBER 2008
What is a synthesizer?
- Electronic instrument that generates output tones from user-controlled inputs
- Modular Synth: break audio processing into discrete units for arbitrary routing

Why a synthesizer?
- Highly modular design
- Easy to implement in DSP → 48kHz refresh frequency
- Interesting routing/UI needs
- Fun to demo!
System Block Diagram

- **3 Blocks:**
  - Audio DSP Units
  - Control System
  - User Interface

- **User Interface System**
  - Inputs (digital, analog)

- **Control Module**
  - AC97 input
  - AC97 output

- **Audio DSP Modules**

- **RS-232 (115200 8N1)**
Control System

- **2 Parts:**
  - **Control Module**
    - Serial parsing
    - DSP control
    - DSP setup
    - System memory
  - **Routing Network**
    - “Ring” topography
    - 2 Loops
      - Audio Data
      - Control Data
Control Module

- **Control Module**
  - Transmit & Receive instructions from PC or User Interface unit
    - Serial interface
    - 115200 8N1
  - Control Logic
    - Drives synthesizer DSPs with RUN, SET, BYPASS lines
  - System State Memory Buffer
    - \((n \times m)\) registers @ 8 bits wide
  - Ring Network I/O
    - For DSP state setting/updates

\(n\) – number of DSP modules implemented
\(m\) – number of control registers needed by DSP module \(n\)
The Problem:
- Inter-module signal routing should support arbitrary configuration
- Should be on-the-fly configurable

Previous Solution:
- Analog signal routing between jacks on synth board:
  an audio “patch panel”
The “Digital Patch Panel”

- **Our Solution:**
  The “Digital Patch Panel”

- **Parallel data streams**
  - 16-bit PCM audio
  - 8-bit control values
  - Bin(n) address lines

- **Like an SPI bus:**
  - Data rotates “through” each module

- **Triggered on certain signals:**
  - Audio: network rotates on every `<AC97 ready>` signal
  - Control: rotates on every command update via serial port

\[ n \] – number of DSP modules implemented
\[ m \] – max number of control registers needed by DSP modules
Audio Module Design

- **Standard DSP layout:**
  - State Registers
  - Control Logic
  - Ring Network I/O
    - Audio (16+ bits)
    - Control (8+ bits)
  - Audio DSP

- **Standard I/O interface**
  - Ring Networks
  - Control Signals
Audio Modules

- **11 (13) Digital Audio modules**
  - **Basic DSP modules:**
    - Oscillator
    - Filter
    - Sequencer
    - Slew Limiter
    - Amplifier
    - Envelope
    - Noise
    - Ring Modulator
    - Delay
    - Reverb
    - N-to-1 mixer
  - **“Reach” DSP modules**
    - Vocoder
    - Pitch Corrector
  - **Peripheral Audio modules**
    - AC97 - Input
    - AC97 - Output
User-Interface Component

- **2 Issues:**
  - Module settings
  - Audio path routing

- **Previous Solution:**
  - Knobs, Sliders, Switches
  - Point-to-Point wiring

- **Our Solution:**
  - Buttons & Switches
  - 2-axis analog joystick (SPI)
  - Resistive touchscreen (serial)
Display Module

- **Output Display:**
  - Custom LCD system – Sharp 4.3” Widescreen (PSP)
  - 480 x 272 pixels @ 8bpp, RGB
    - 391,680 image bytes/color/frame @ 8bpp
  - 24+4 driver pins
    - 8 bits x R,G,B data bus
    - HS, VS, VCLK, DISP
  - 2.5V interface logic levels
  - LED backlight driver onboard

- **Fallback Plan A:**
  - VGA LCD drive (a la Lab 5)

- **Fallback Plan B:**
  - PC Software/Human-Readable commands via serial terminal
Input Modules

- **Main Input - Touchscreen:**
  - Custom touchscreen for 4.3” Widescreen LCD
  - Resistive 4-wire interface
  - Touchscreen Controller: AD7843 12-bit touchscreen ADC
  - TTL serial interface

- **Aux Inputs – Joystick:**
  - 2-axis (XY) joystick
  - 10-bit ADC with SPI interface

- **Aux Inputs – Buttons:**
  - 6 buttons, momentary

- **Aux Inputs – Switches:**
  - 8 switches, SPST
U/I “Wrapper” System

- Based on NEXSYS2 board
  - Spartan 3E-1200k Xilinx FPGA
  - 16 Mb SRAM
  - 128 Mb Flash
  - RS-232 UART

- Serial comm link between U/I unit and Synth
  - Computer replacement during testing

- Offloads display control, refresh, user input from master FPGA
# Milestones & Timeline

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tues</th>
<th>Weds</th>
<th>Thurs</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Design Presentation</td>
<td></td>
<td>Oscillator – basic functionality</td>
<td>AC97 input, output – basic functionality</td>
</tr>
<tr>
<td>Audio Ring Network – basic functionality</td>
<td></td>
<td></td>
<td>Design Meeting Control Logic – basic functions</td>
<td></td>
<td>Basic Audio DSPs - implemented</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Design Meeting</td>
<td></td>
<td></td>
<td>Thanksgiving, Mike &amp; Muth travelling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cont’d</td>
<td>Design Meeting</td>
<td></td>
<td></td>
<td>New Features END DATE</td>
<td></td>
<td>Design Meeting</td>
</tr>
<tr>
<td>Design Meeting</td>
<td></td>
<td></td>
<td>Checkoffs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Overview** - [CTRL MODULES] - [AUDIO MODULES] - [I/O MODULES] - [MILESTONES]