MIDI Synthesizer

Kyle, Peter, and Eric

Current Work Accomplished

Kyle: Started work on envelope generator.

Eric: Working on getting Audio Chip interfaced.
Peter: Wrote Simple NCO, Working with Eric on Audio Chip

Feature List

(BASIC FUNCTIONALITY)

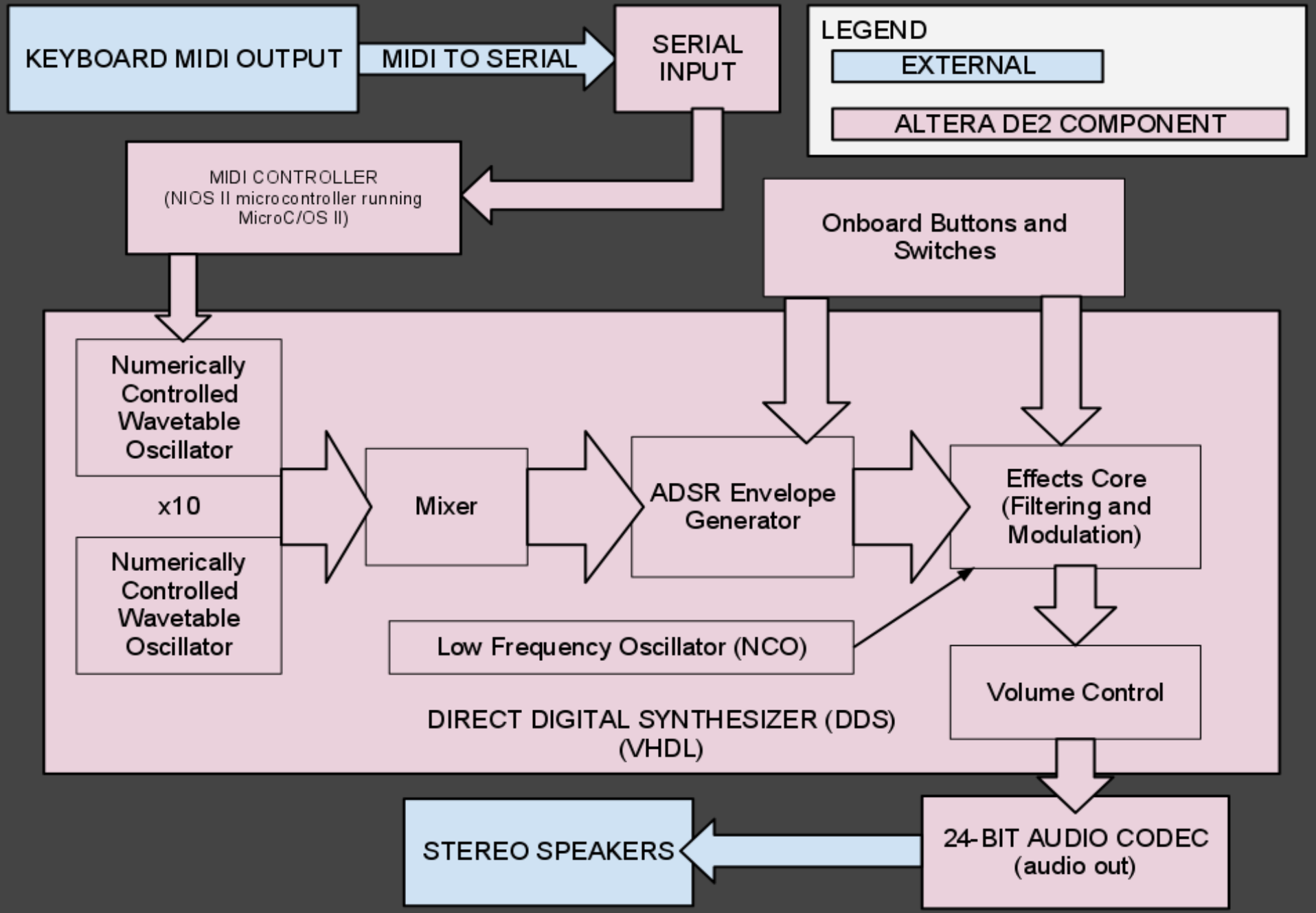
1. Play accurate notes when keyboard keys are pressed, via the MIDI protocol.
2. Ability to add effects. (Wah, Tremelo, Vibrato)
3. Pre-recorded performances (MIDI Files)
4. Up to 10 keys played at once.
5. 48kHz 16bit Audio

(POSSIBLE ADDITIONS)

1. Arpeggiator.
2. Display notes played on screen.
3. Multiple sound sets.
4. External Custom MIDI Instrument.

Motivation

- Digital synthesis is an interesting area.
- Straightforward core project with lots of room for expansion.
- Fun demonstration.
- Something a bit different.
- Interesting DSP applications.



Basic Functionality Component Diagram

Challenges

The challenges currently ahead.

1. Create the VHDL components for basic functionality
2. Connecting the components together
3. Create MIDI controller in MicroC/OS II on NIOS II
4. Connect the external components, keyboard, speakers

Components

Hardware Based:

- Physical MIDI Interface to UART

FPGA Based:

- Numerically Controlled Oscillator
- Envelope Generators
- Mixer
- Effects Core
- Volume Control

Code Example

```
entity nco is
  port (
    clk : in std_logic; -- System Clock
    rst : in std_logic; -- Reset
    nco_inc : in std_logic_vector(31 downto 0); --Freq Increment
    wave_out : out std_logic_vector(11 downto 0) --Output Waveform
  );
end nco;

architecture arch of nco is
  component lut
    port (
      clk : in std_logic; --Clock
      addr : in std_logic_vector(11 downto 0); --Address in LUT
      waveform : out std_logic_vector(11 downto 0) --Waveform Val
    );
  end component;

  signal accumulator : std_logic_vector(31 downto 0);
  signal lut_address : std_logic_vector(11 downto 0);
```

Code Example 2

```
begin  -- arch
  lut_address <= accumulator(31 downto 20); --12 bits=4096 samples

  inc: process(clk, rst)
  begin
    if rst = '0' then
      accumulator <= x"00000000";
    elsif rising_edge(clk) then
      accumulator = unsigned(accumulator) + unsigned(nco_inc);
    end if;
  end process;

  lut: lut port map(
    clk => clk,
    addr => lut_address,
    waveform => wave_out
  );
end arch;
```

Test Plan

We will be testing each component as we complete them. Then do various integration tests as the system is connected together.

- NIOS II: To test setup, connection and operation we will compile and run a simple program. Will also be used used for other tests.
- RAM: We will test connection and functionality by running a standard memory tester. Similar to Lab 1.
- Flash: To test connection,storage and loading we will store and load a simple program on the flash. Power cycle the board and have the program start and run from flash.
- MIDI Controller: We will write testing harnesses to simulate input and verify the expected output is produced.

Test Plan

- MIDI Input: We will test proper connection and signal reading by connecting the keyboard, pressing a key and having it displayed on the LCD.
- Sub-components: For the various sub components that will make up the system (ADSR Envelope generator, oscillators, etc) we will run basic functional testing.

Application Notes

- None so far.
- When we get audio working we'll release that...

MIDI Synthesizer

Questions?