University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150 J. Wawrzynek
Spring 2000 E. Caspi

Project Checkpoints #5, 6, 7

This handout covers the remaining checkpoints for the MIDI project. It defines the project schedule for the next few weeks, from now until final check-off and report time.
By now, you have a complete project board and circuit designs for all input/output operations. What remains to be designed is the sound synthesis engine, the heart of the project.

1 Schedule

The following is a week-by-week schedule for the next few weeks of class:

- 3/27/00 – spring break
- 4/3/00 – Project checkpoint #5: monotone notes
- 4/10/00 – Project checkpoint #6: notes of arbitrary frequency
- 4/17/00 – Project checkpoint #7: velocity sensitivity
- 4/23/00 – Spare week (clean-up design, extra credit)
- 5/1/00 – Final project check-off

2 Checkpoints

The remainder of the project is divided into three checkpoints. These checkpoints establish the minimum progress which you should be making each week. You are, of course, free to work ahead. In addition, at the end of the semester, there will be a final check-off and project demonstration in lab.

- **Checkpoint #5: Monotone Notes.**

  Demonstrate a simplified synthesizer that always plays notes at the same frequency and same amplitude. The circuit should play the attack, looped sustain, and release for the instrument's first template (at address `0x200`) using a unit step-size. It is not necessary to read the template entry from the master directory to achieve this functionality. It is, however, necessary to read the template header (at `0x200`) to retrieve end-pointers for the attack, sustain, and release regions. Attack and release should be played in response to MIDI note-on/note-off events, or alternatively, in response to pressing push-buttons on the Xilinx board.

- **Checkpoint #6: Notes of Arbitrary Frequency.**

  Extend the synthesizer to play arbitrary notes at their proper frequency (but same amplitude). Attack, looped sustain, and release samples must now be read from the correct template with the correct fractional step-size. Interpolation is not required, nor is velocity sensitivity. Attack should be played in response to MIDI note-on, and release should be played in response to the active note's note-off. If a note-on arrives while an old note is playing, the old note should be abandoned immediately, and the new note should begin its attack.

  The proposed method of abandoning an old note to begin an overriding one typically creates noise (clicks) due to a discontinuity in the output waveform. For extra credit, you may wish to implement a cleaner-sounding method of switching notes. The old note should probably be "faded out" (scaled down in amplitude) over some short period of time, say 10ms, before or while the new note begins. Monophonic (single-note) mode in commercial synthesizers typically implements a *legato/glissando* transition between notes, fading or even pitch-bending from the old note to the sustain part of the new note (*i.e.* without playing the attack of the new note). You may choose your own semantics for a click-free note transition algorithm, but the algorithm must be approved by the professor before you begin.

- **Checkpoint #7: Velocity Sensitivity.**
  Extend the synthesizer to play each note at an amplitude corresponding to its velocity. The mapping from velocity code (`0x00` – `0xFF`) to amplitude scaling factor (0--1) must come from a lookup table. A flexible mapping is needed to compensate for the fact that different MIDI keyboards send different velocities for a key press of a given pressure (*i.e.* they have different *velocity curves*). The mapping should also compensate for the non-linearity between key pressure, velocity, and amplitude. You may use the table values provided in lecture or experiment with your own (linear, power, etc.).

3 EPROM Configurations

We will provide a number of EPROM bit-files containing instrument samples. These bit-files will reside in: `U:\WVLIB\CS150\INSTRMNT\`, as well as: `C:\CS150\INSTRMNT\` on the EPROM programming workstations in 204B Cory. The files are binary dumps and must be loaded using the EPROM programmer's "binary" mode (File→Format→Binary). The samples include:

- `agtr.bin` – Acoustic guitar
- `loudv.bin` – Loud violin
- `marimba.bin` – Marimba
- `sines.bin` – Sine wave
- `longsines.bin` – Long-playing sine wave

You are invited to create new sample files, following the format described in the project spec.

Name: _____ Name: _____ Lab: _____

4 Check-offs

- Checkpoint #5: monotone notes. TA: _____ date: _____
- Checkpoint #6: notes of arbitrary frequency. TA: _____ date: _____
- Checkpoint #7: velocity sensitivity TA: _____ date: _____