Real Sound for 8-bit Apple II’s

Michael Mahon
The Apple II speaker
(and cassette output)

- Can only toggle “1-bit output”
- Can’t choose “polarity”
- Doesn’t respond to alternate low-frequency toggles
Simple Sound

- Double-frequency timing loop
  - Squarewaves (beep, etc.)

- Actual frequency timing loop
  - Non-50% duty cycle, timbre (Red Book)

- Infinitely-clipped 1-bit sound
  - Distorted speech and music (Hex Dump Reader, Audex, etc.)
Complex Sound

- Ultrasonic timing loop with variable duty cycle pulses
  - Electric Duet (4 duty cycles = 2-bit precision)
  - Software Automatic Mouth (?)
  - Sampled sound (many duty cycles)
Variable Duty Cycle

Sample value

| 00 | 7 | 86 |
| 01 | 8 | 85 |
| 02 | 9 | 84 |
| 03 | 10 | 83 |
| . | . | . |
| 63 | 70 | 23 |
Software DACs

- Sample rate (11.025kHz)
- Pulse rate (11.025kHz / 22.05kHz)
- Bits of precision (3, 4, 5, 6)
Timing Constraints

- 11kHz = 93 Apple II cycles/sample
  - 1/93cy = 10.973kHz, -52Hz, -0.005%
  - 1/92cy = 11.092kHz, +67Hz, +0.006%
- Apple II timing resolution = 1 cycle
- 4 cycles required to flip speaker
  - 8 cycles per pulse, 4-cycle minimum width
- Sound amplitude = $\Delta$width / period
Software DAC loop example

first try: constant 93 cycles, 6-bit precision

loop:

4 \textbf{[Start pulse]}
3 \textbf{JMP vector1}
\textbf{<0-63 Variable delay>}
4 \textbf{[End 7-70 cycle pulse]}
3 \textbf{JMP vector2}
\textbf{<63-0 Complementary variable delay>}
5 Fetch next sample
2 Check for end (if =0)
10 Increment ptr,Y
6 Shift sample
4 Get vector1
4 Set vector1
2 Transform vector1--> vector2
4 Set vector2
3 \textbf{JMP loop}

117 cycles \hspace{1cm} \textbf{** 24 cycles too long! **}
Pulse frequency of 11kHz is intolerable to many listeners.

[Greg Templeman, 1993]
Logically 32 separate pulse generators
- Each generates two pulses in 92 cycles
- Each fetches next sample and sets vector
- Then it vectors to next generator
- Computation is distributed between pulse edges

DAC522
Constant 92 cycles, 46-cycle pulse period, 5-bit precision

Pulse frequency of 22kHz is inaudible!
## DAC522 Pulse Generators

<table>
<thead>
<tr>
<th>Sample value</th>
<th>46 cycles</th>
<th>46 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>6 6</td>
<td>40 40</td>
</tr>
<tr>
<td>01</td>
<td>7 7</td>
<td>39 39</td>
</tr>
<tr>
<td>02</td>
<td>8 8</td>
<td>38 38</td>
</tr>
<tr>
<td>03</td>
<td>9 9</td>
<td>37 37</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>37 37 37</td>
<td>9 9 9</td>
</tr>
</tbody>
</table>
Other Problems

- Starting and stopping without “pops”
  - Solution: Don’t stop!
    - “Ramp” waves to start and stop at 0 level
    - generate continuous 0-level pulses (except for key clicks)

- Generating long sounds at 11kB/second
  - Solution: Direct Digital Synthesis
    - Resample wavetables on-the-fly
    - Use envelope table for dynamics
RT.SYNTH
Single-voice multi-timbral real-time wavetable synthesizer

- Voice waves are resampled on-the-fly to note frequencies
  - Frequency = Integer.fraction sample index increment
- Can support as many different voices as fit in memory
  - A voice is represented as a set of single-cycle waveshapes selected by a table representing the envelope of the voice.
  - This supports
    - Tonal voices with complex attacks that are resampled
    - Atonal sounds that are played as “waves”
- Wavetable waveforms are stored starting and ending at zero amplitude to minimize pops
- “Resting” or idling sound is the zero-level pulse train
DAC522 Generator Code

```
0800: 8D 30 C0 >6    gen0     sta   spkr       ; <==== start time: 0
0803: EA       >7             nop              ; Kill 2 cycles
0804: 8D 30 C0 >8             sta   spkr       ; <==== stop time: 6
0807: 85 EB    >9             inc   scount     ; Compute envelope
0809: E6 ED    >10            inc   scount     ; Compute envelope
080B: F0 01    >11            beq   *+3        ; If =, branch to iny
080D: A5       >11            dfb   $A5        ; "lda $C8" to skip iny
080E: C8       >11            iny              ; Compute next sample
080F: 18       >12            clc
0810: A5 EC    >13            lda   frac       ; Compute next sample
0812: 65 FE    >14            adc   freq
0814: 85 EC    >15            sta   frac
0816: 8A       >16            txa
0817: 65 FF    >17            adc   freq+1
0819: AA       >18            tax
081A: B1 06    >19            ldd   (env),y    ; Next sample page
081C: 8D 30 C0 >20             sta   spkr       ; <==== start time: 46
081F: EA       >21             nop              ; Kill 2 cycles
0820: 8D 30 C0 >22             sta   spkr       ; <==== stop time: 52
0823: 85 EB    >23             sta   ztrash     ; Kill 3 cycles
0825: 8D 2A 08 >24            sta   :ptr+2
0828: BD 00 00 >25            ldaa  0*0,x      ; Fetch sample.
082B: 8D 3C 08 >26            sta   :sw0+2
082E: C6 FC    >27            dec   dur        ; Decrement duration
0830: F0 02    >28            beq   *+4        ; If eq, branch to dec
0832: EA       >28            nop              ; Else kill 2 cycles and
0833: AD       >28            dfb   $AD        ; "lda xxxx" to skip dec
0834: C6 FD    >28            dec   dur+1      ; of zero-page param.
0836: A5 FD    >29            lda   dur+1
0838: F0 03    >30            beq   :quit      ; Finished.
083A: 4C 00 00 >31            jmp   0*0        ; Switch to gen, T = 89
083C: 4C 40 09 >33            :quit jmp quit
```
CRATE.SYNT

8-voice multi-timbral MIDI playback wavetable synthesizer

- Uses AppleCrate machines as eight digital oscillators
- **MIDI.COMPILER**
  - Merges multi-stream MIDI events
  - Tempo changes complicate timekeeping
  - Schedules 8 digital oscillators in one pass
  - Tries to re-use oscillators with a voice history
  - When >8 oscillators needed, “steals” from oldest note
- **CRATE.SYNT**
  - Uses NadaNet to load the 8 oscillator machines
  - Starts them all in sync (AppleCrate drift is ~1 ms. in 40 sec.)
Questions and discussion...