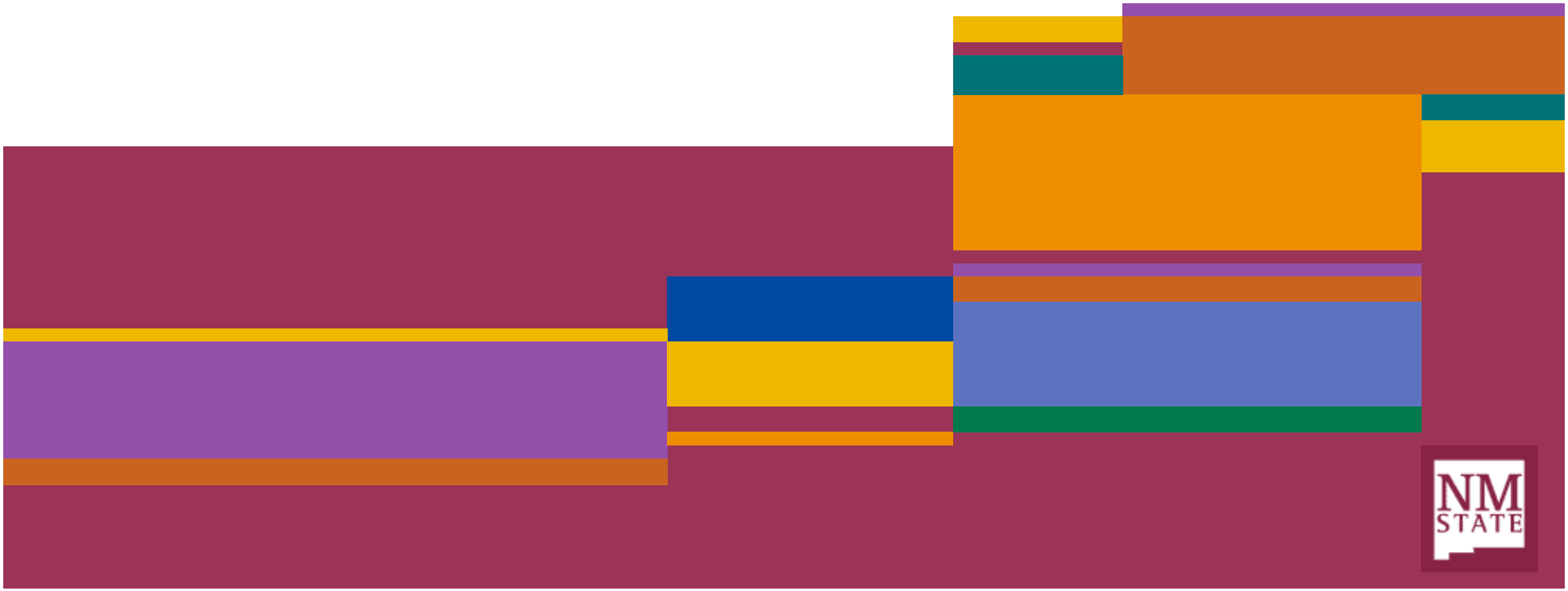
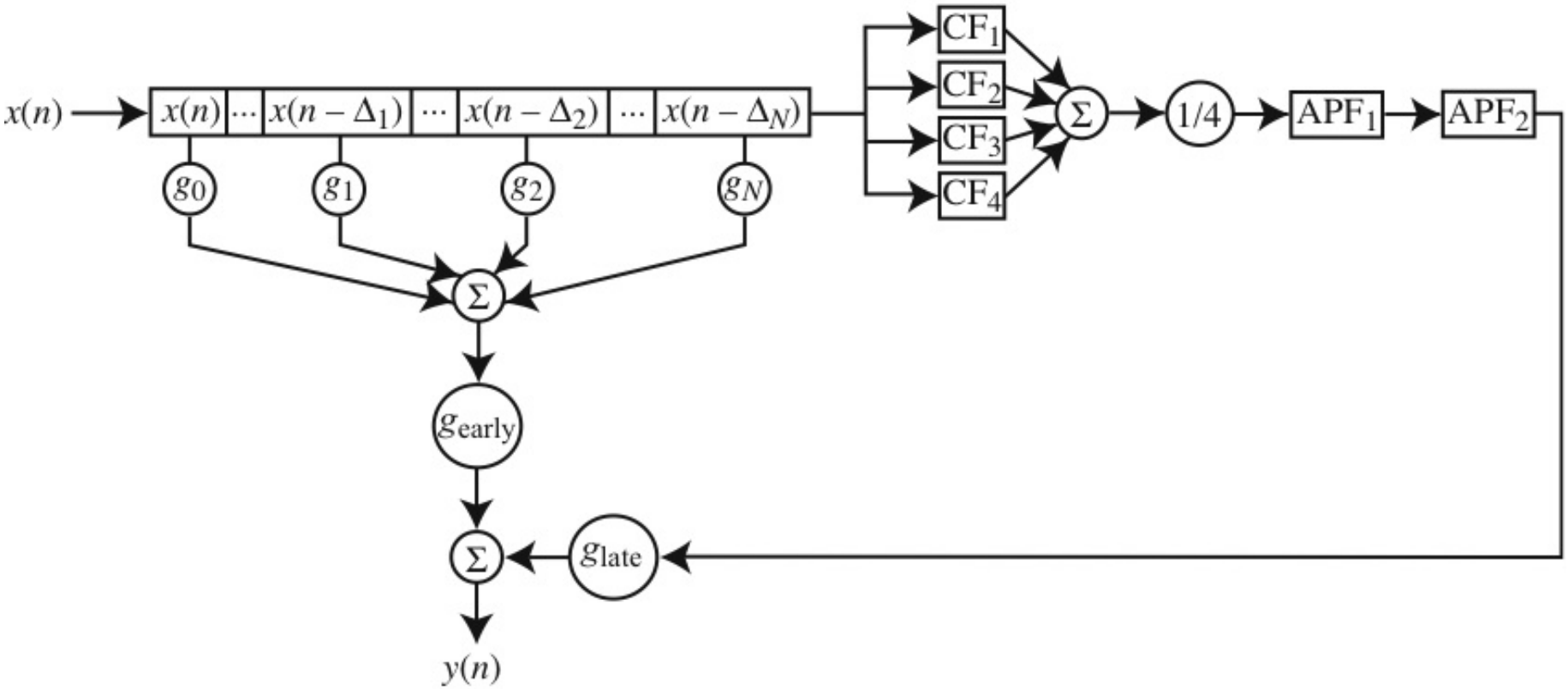
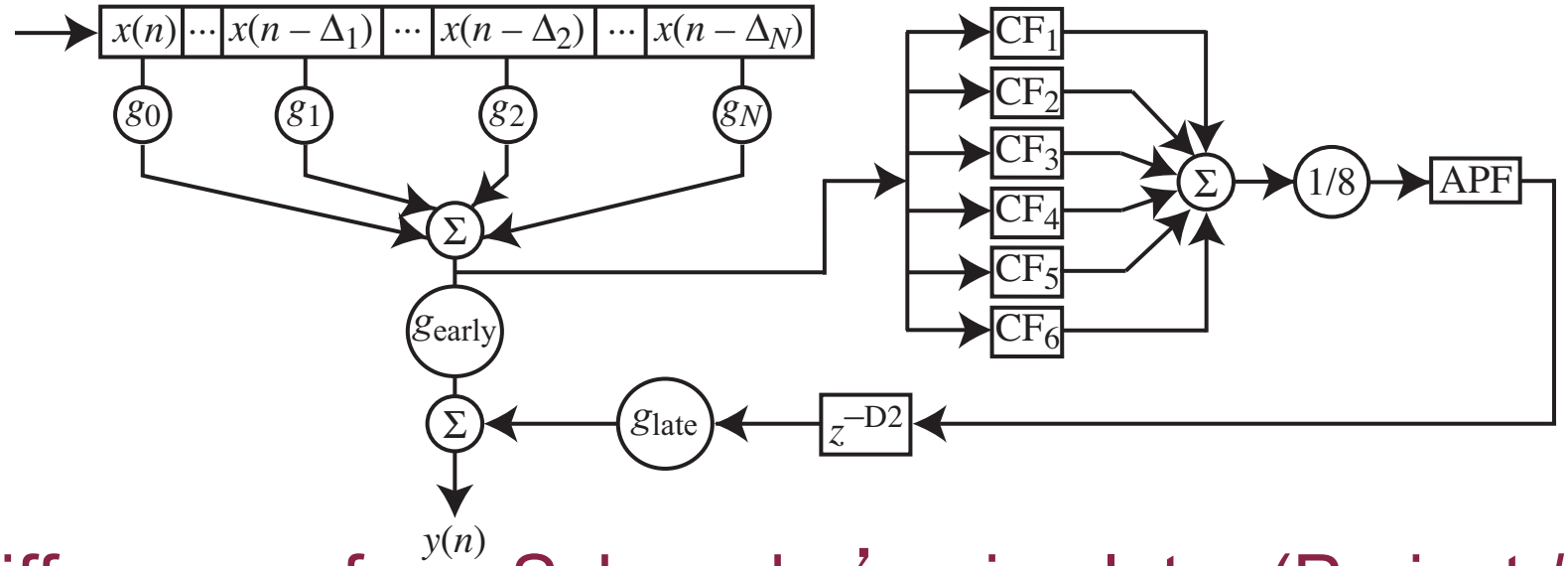
Project #3 TI-Based Sound Field Simulator (Moorer's)



Schroeder's Sound Field Simulator



Moorer's Sound Field Simulator



- Differences from Schroeder's simulator (Project #1)
  - TDL has 19 taps (data from ideal simulation of Boston Symphony Hall)
  - TDL output feeds reverberator (not oldest sample in TDL queue)
  - Reverberator has parallel bank of 6 CFs followed by APF
  - CFs have LPF in feedback loop (models high-freq absorption by air)
  - Reverb output is delayed so that first reverb pulse coincides with last TDL pulse

Tapped Delay Line

- Tapped delay line for Moorer's simulator operates in same way as Project #1 except for more taps

Table 1: Tapped delay line parameters for Moorer's sound field simulator.

Tap	Delay (ms)	Gain	Tap	Delay (ms)	Gain
0	0.0	1.0	10	58.7	0.193
1	4.3	0.841	11	59.5	0.217
2	21.5	0.504	12	61.2	0.181
3	22.5	0.491	13	70.7	0.180
4	26.8	0.379	14	70.8	0.181
5	27.0	0.380	15	72.6	0.176
6	29.8	0.346	16	74.1	0.142
7	45.8	0.289	17	75.3	0.167
8	48.5	0.272	18	79.7	0.134
9	57.2	0.192			

tap.c

- tap() (p. 178 Orphanidis) taps the ith element of a circular delay line
  - D is order of the array, i.e. length of queue is D + 1
  - *w is a pointer to base address of array
  - *p is a pointer to the current sample [doesn't change in tap()]
  - i is the delay element to tap, $0 \leq i \leq D$
- This routine returns the tapped value by extending i words beyond p, modulo (D+1)

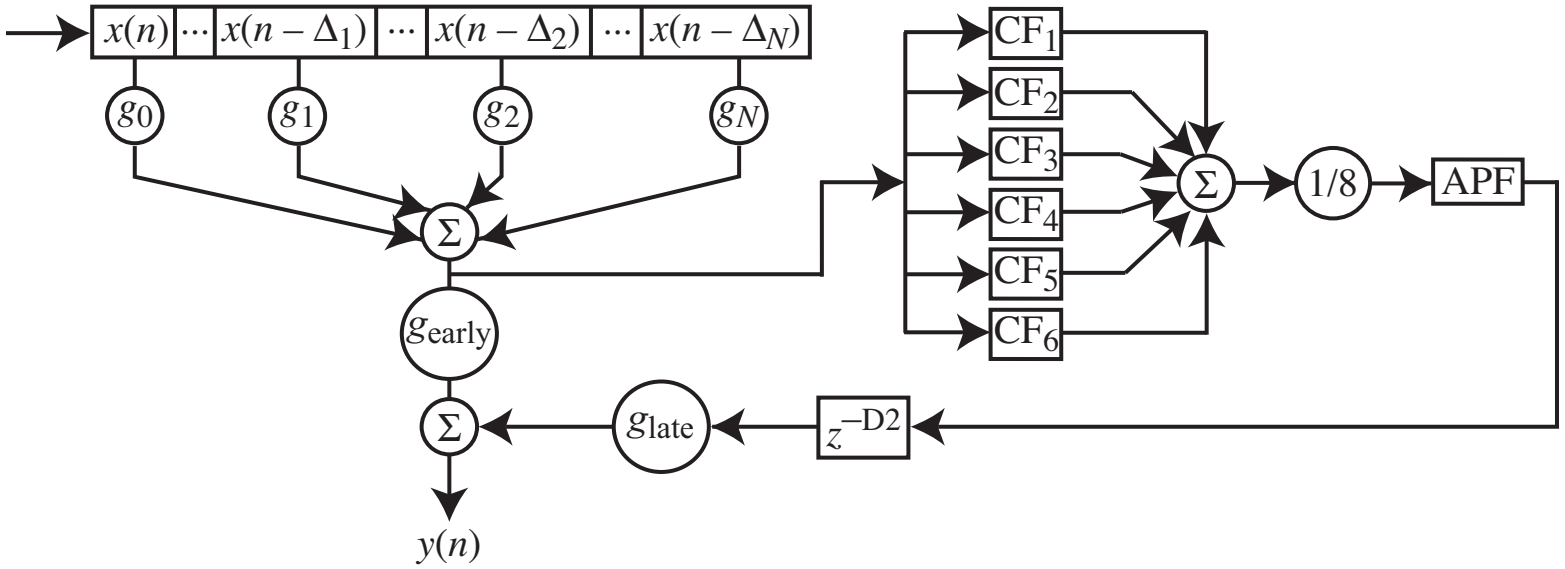
tap.c

```
short tap(short M, short *w, short *p, short i)
{
  return w[(p - w + i) % (M + 1)];
}
```

tdl.c

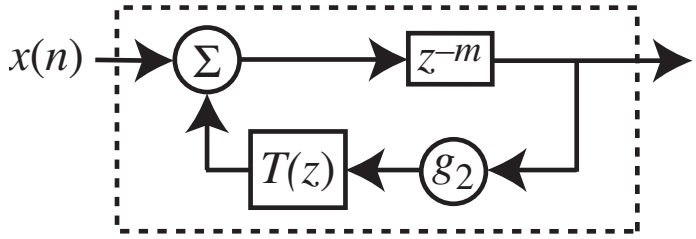
```
short tdl(short M, short *w, short **p, short N, short *gain,
   short *delta, short x)
{
  short i, sample;
  int y=0;

  **p = x;/* read input sample */

  /* compute filter output, y */
  for(i=0; i<N; i++) {    /* loop over the taps */
    sample = tap(M, w, *p, delta[i]);/* get sample from tdl */
    y += gain[i]*sample;  /* mac gain and sample  */
  }
  /* update filter states, i.e. wk(n+1) = w{k-1}(n) */
  cdelay(M, w, p); /* p -> wm */
  return cround(y)>>15; /* round output and back to 16 bits */
}
```

Moorer's Sound Field Simulator

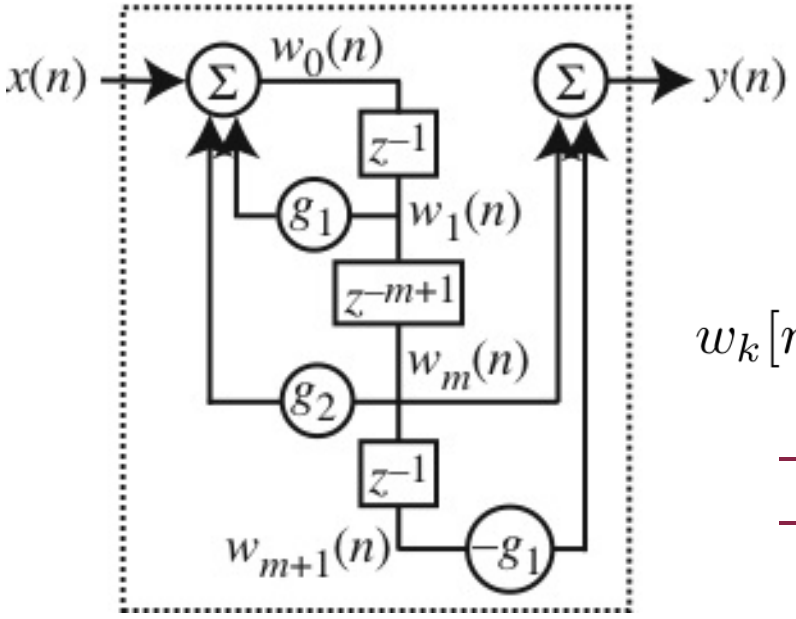


- Differences from Schroeder's simulator (Project #1)
  - Reverberator has parallel bank of 6 CFs followed by APF
  - CFs have LPF in feedback loop (simulates absorption of high frequencies by the air)

Comb Filter with LPF in Feedback Loop



$$T(z) = \frac{1}{1 - g_1 z^{-1}}.$$

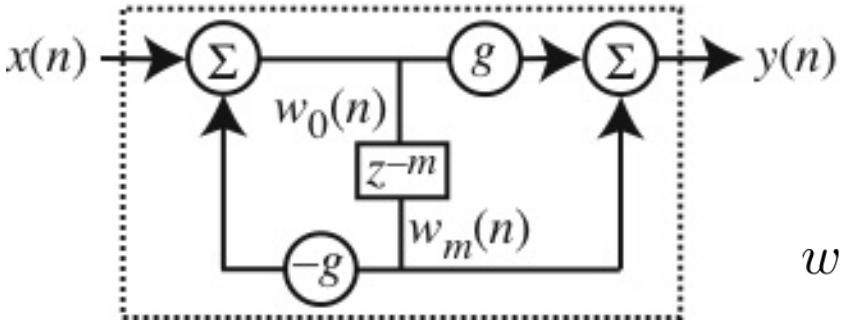


$$\begin{aligned} w_0[n] &= x[n] + g_1 w_1[n] + g_2 w_m[n] \\ y[n] &= w_m[n] - g_1 w_{m+1}[n] \\ w_k[n+1] &= w_{k-1}[n], \quad k = m+1, m, ..., 1 \end{aligned}$$

- g1 LPF gain, g2 CF gain
- filter state queue length of m+1 is needed

Allpass Filter

- APF for Moorer's simulator operates in same way as Project #1

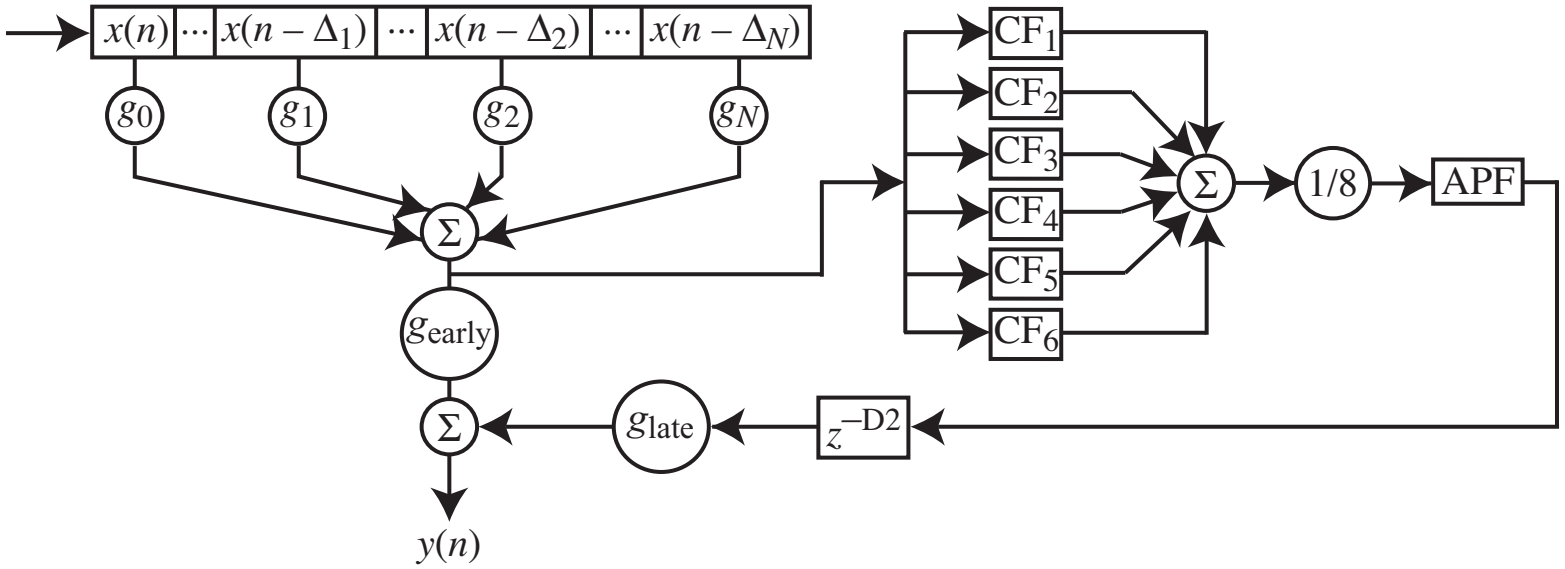


$$
\begin{aligned}
w_0[n] &= x[n] - g w_m[n] \\
y[n] &= g w_0[n] + w_m[n] \\
w_k[n+1] &= w_{k-1}[n], \quad k = m, m-1, ..., 1.
\end{aligned}
$$

Implementation of CF and APF

- We could use ccan to implement the CF and APF, however, since the CF and APF state equations only require using a few of the filter states, ie. most of the a_k and b_k are 0, much computation would be wasted
- Alternately, one could call tap() within a cf() or apf() code in order to “tap ” the state queue for only the states required in computing newest state and output
- State update is accomplished with cdelay()

Moorer's Sound Field Simulator



- Differences from Schroeder's simulator (Project #1)
  - Reverb output is **delayed** so that first reverb pulse is 1ms after last TDL pulse
  - Last TDL pulse @ 79.7 ms, first reverb pulse @ 50 ms (CF1's 50ms delay but no APF delay) so we delay reverb by 79.7–50+1=30.7 ms

Implementation of Delay by D2

- Create a circular queue and feed it with APF output
- Be sure to grab oldest sample in D2 queue before blowing it away--this sample is the reverb output

```
/* Delay reverberator output to align with TDL */
reverbOutput = *D2delay_OldestStatePtr; /* get oldest sample */
*D2delay_OldestStatePtr = APF_Output; /* replace oldest sample */
cdelay(D2, D2delay, &D2delay_OldestStatePtr);
```

user_data.h, initialize_program.c

- In Project #1, a project1.dat file was supplied, for Project #3, we supply user_data.h file and also initialize_program.c