Embedded Systems with ARM Cortex-M3 Microcontrollers in Assembly Language and C

Chapter 21
Digital-to-Analog Converter (DAC)

Dr. Yifeng Zhu
Electrical and Computer Engineering
University of Maine

Spring 2015

Digital-to-analog converter (DAC)

- Converts digital data into a voltage signal by a N-bit DAC

$$DAC_{output} = V_{ref} \times \frac{Digital\ Value}{2^N - 1}$$

- For 12-bit DAC

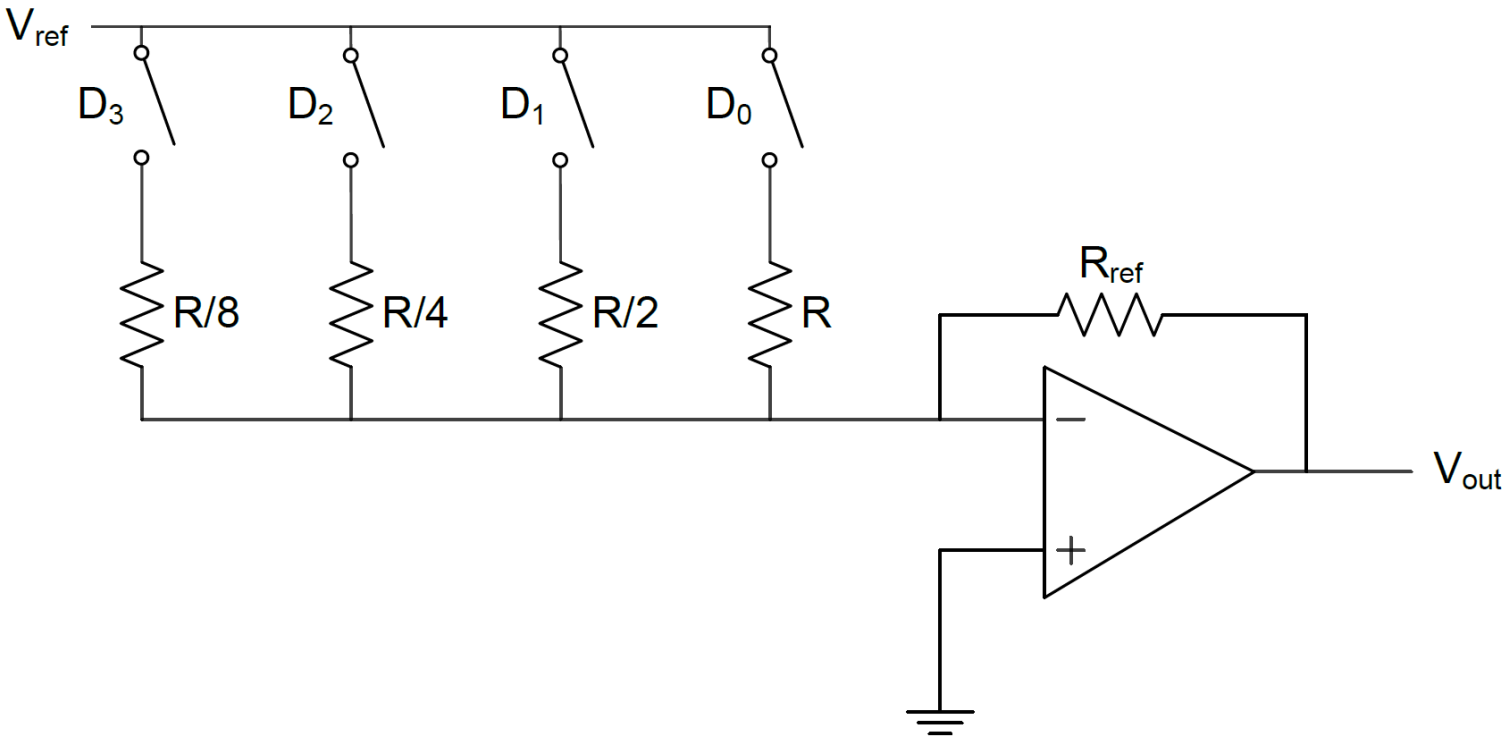
$$DAC_{output} = V_{ref} \times \frac{Digital\ Value}{4095}$$

- Many applications:
  - digital audio
  - waveform generation
- Performance parameters
  - speed
  - resolution
  - power dissipation

DAC Implementations

- Pulse-width modulator (PWM)
- Binary-weighted resistor (We will use this one as an example)
- R-2R ladder (A special case of binary-weighted resistor)

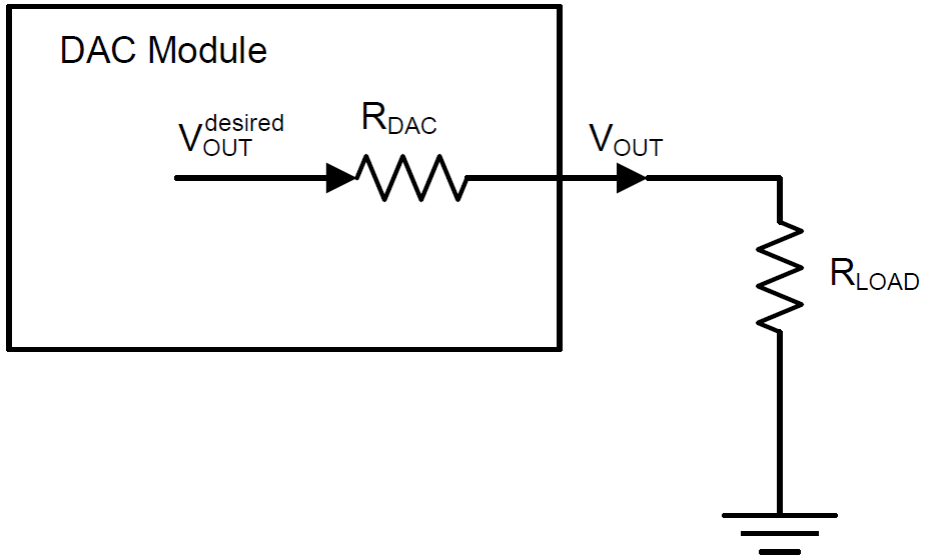
Binary-weighted Resistor DAC



$$V_{out} = V_{ref} \times \frac{R_{ref}}{R} \times (D_3 \times 2^3 + D_2 \times 2^2 + D_1 \times 2 + D_0)$$

Buffered Output

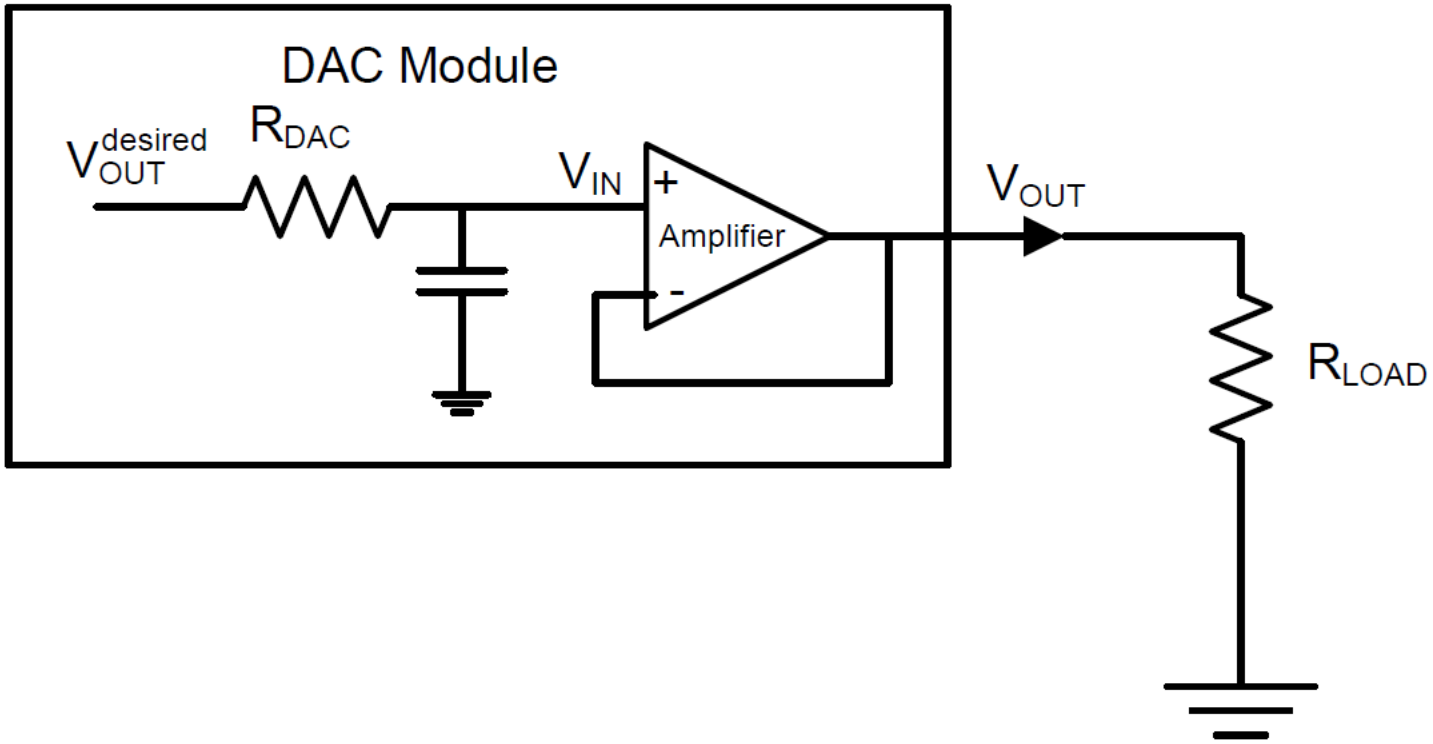
- Load Effect



$$V_{OUT} = \frac{R_{LOAD}}{R_{DAC} + R_{LOAD}} \times V_{OUT}^{desired}$$

Buffered Output

- Use buffer to remove load effect



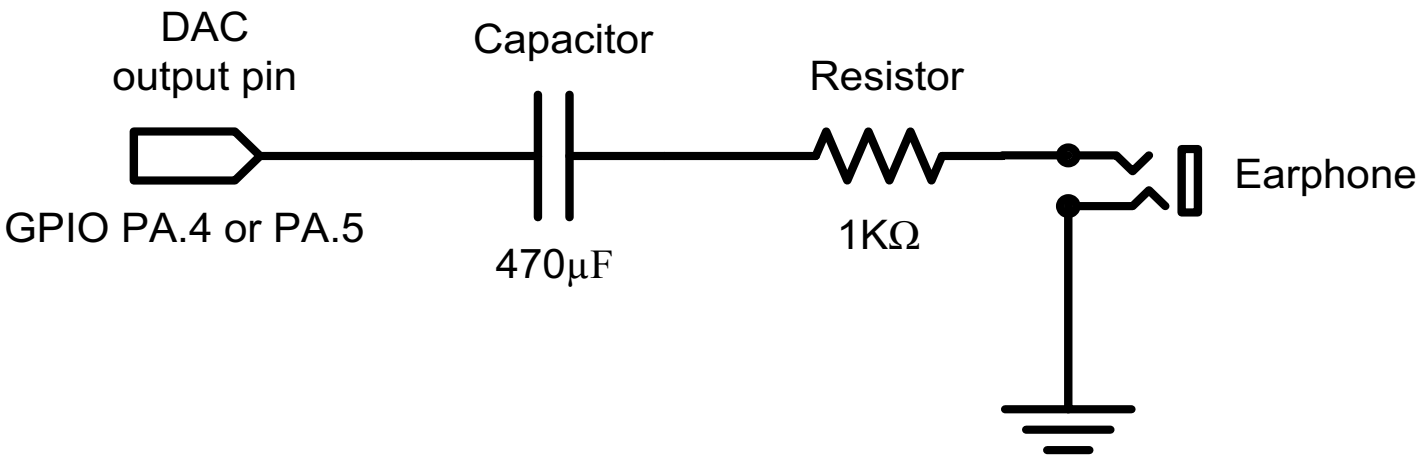
$$V_{OUT} \approx V_{OUT}^{desired}$$

DAC Update

- DAC Output 1: PA.4
  - Connected to external IDD measurement
  - PA.4 is connected to ground via a capacitor
- DAC Output 2: PA.5
  - Connect to 3V via a small resistor
- In the lab, use DAC Output2 (PA.5)
- Have to clear CC1F or UIF flags in status register in Interrupt Service Handler
- ARR: CNT counts from 0 to ARR

$$\frac{f_{HSI}}{(1+PSC)(1+ARR)} = 44.1KHz$$

Digital Music



Digital Music

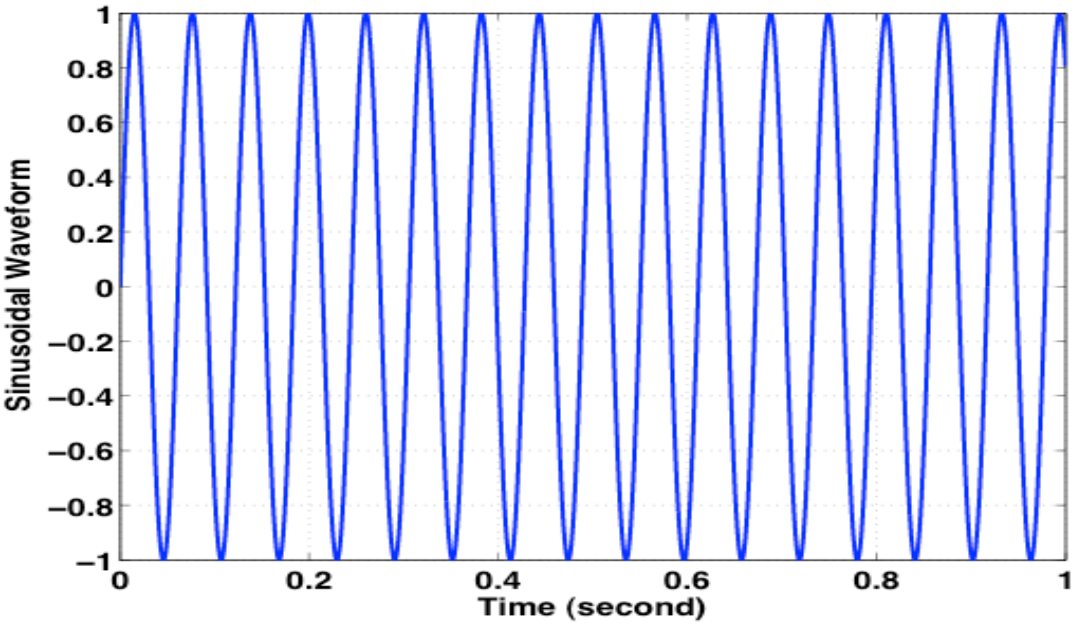
	0	1	2	3	4	5	6	7	8
C	16.352	32.703	65.406	130.813	261.626	523.251	1046.502	2093.005	4186.009
C#	17.324	34.648	69.296	138.591	277.183	554.365	1108.731	2217.461	4434.922
D	18.354	36.708	73.416	146.832	293.665	587.330	1174.659	2349.318	4698.636
D#	19.445	38.891	77.782	155.563	311.127	622.254	1244.508	2489.016	4978.032
E	20.602	41.203	82.407	164.814	329.628	659.255	1318.510	2637.020	5274.041
F	21.827	43.654	87.307	174.614	349.228	698.456	1396.913	2793.826	5587.652
F#	23.125	46.249	92.499	184.997	369.994	739.989	1479.978	2959.955	5919.911
G	24.500	48.999	97.999	195.998	391.995	783.991	1567.982	3135.963	6271.927
G#	25.957	51.913	103.826	207.652	415.305	830.609	1661.219	3322.438	6644.875
A	27.500	55.000	110.000	220.000	**440.000**	880.000	1760.000	3520.000	7040.000
A#	29.135	58.270	116.541	233.082	466.164	932.328	1864.655	3729.310	7458.620
B	30.868	61.735	123.471	246.942	493.883	987.767	1975.533	3951.066	7902.133

For a pitch p, its frequency f

$$f = 440 \times 2^{(p-69)/12}$$

Digital Music

Generate Sine Wave

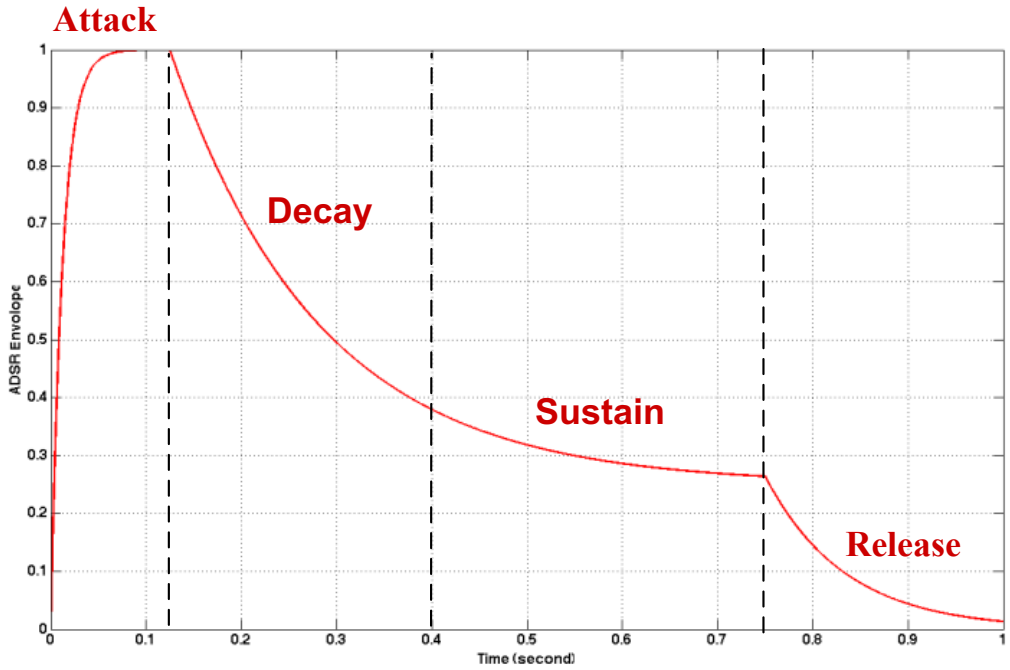


- No FPU available on the processor to compute sine functions
- Software FP to compute sine is slow
- Solution: Table Lookup
  - Compute sine values and store in table as fix-point format
  - Look up the table for result
  - Linear interpolation if necessary

sin(0.4014257)
= 0.4014257 – 0.40142573/3! + 0.40142575/5! – 0.40142577/7!
= 0.4014257 – 0.0107811296737492 + 0.000086864959350 – 0.000000333277256
= 0.390731102008345

floating-point operations

Digital Music:
Attack, Decay, Sustain, Release (ADSR)

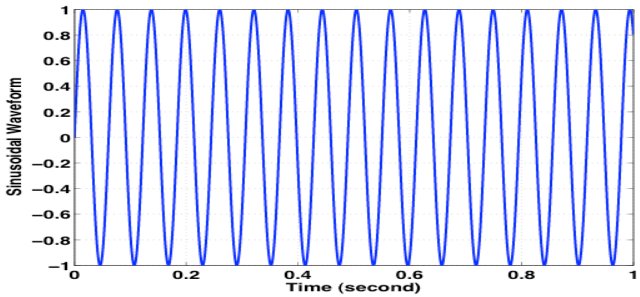


Implemented by a simple digital filter:

$$ADSR(n) = g \times \overrightarrow{ADSR} + (1 - g) \times ADSR(n - 1)$$

where $\overrightarrow{ADSR}$ is the target modulated amplitude value,
g is the gain parameter.

Digital Music
ADSR Amplitude Modulation



+