Digital-to-Analog and Analog-to-Digital Converters

M. B. Patil
mbpatil@ee.iitb.ac.in
www.ee.iitb.ac.in/~sequel

Department of Electrical Engineering
Indian Institute of Technology Bombay

Introduction

Introduction

* Real signals (e.g., a voltage measured with a thermocouple or a speech signal recorded with a microphone) are analog quantities, varying continuously with time.

Introduction

* Real signals (e.g., a voltage measured with a thermocouple or a speech signal recorded with a microphone) are analog quantities, varying continuously with time.
* Digital format offers several advantages: digital signal processing, storage, use of computers, robust transmission, etc.

Introduction

* Real signals (e.g., a voltage measured with a thermocouple or a speech signal recorded with a microphone) are analog quantities, varying continuously with time.
* Digital format offers several advantages: digital signal processing, storage, use of computers, robust transmission, etc.
* An ADC (Analog-to-Digital Converter) is used to convert an analog signal to the digital format.

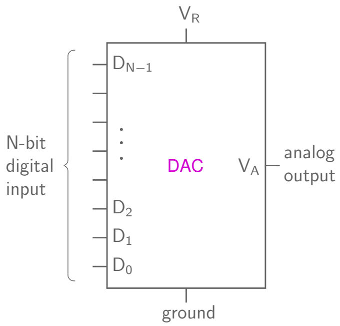
Introduction

* Real signals (e.g., a voltage measured with a thermocouple or a speech signal recorded with a microphone) are analog quantities, varying continuously with time.
* Digital format offers several advantages: digital signal processing, storage, use of computers, robust transmission, etc.
* An ADC (Analog-to-Digital Converter) is used to convert an analog signal to the digital format.
* The reverse conversion (from digital to analog) is also required. For example, music stored in a DVD in digital format must be converted to an analog voltage for playing out on a speaker.

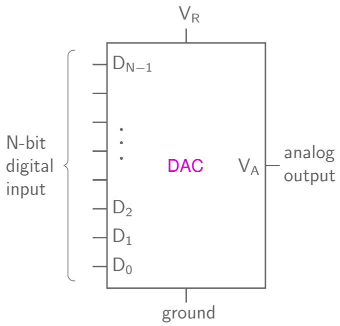
Introduction

* Real signals (e.g., a voltage measured with a thermocouple or a speech signal recorded with a microphone) are analog quantities, varying continuously with time.
* Digital format offers several advantages: digital signal processing, storage, use of computers, robust transmission, etc.
* An ADC (Analog-to-Digital Converter) is used to convert an analog signal to the digital format.
* The reverse conversion (from digital to analog) is also required. For example, music stored in a DVD in digital format must be converted to an analog voltage for playing out on a speaker.
* A DAC (Digital-to-Analog Converter) is used to convert a digital signal to the analog format.

DAC

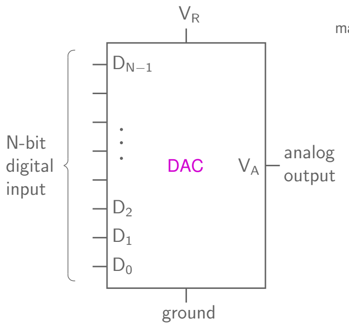

V_R
D_{N-1}
N-bit digital input
DAC
V_A
analog output
D_2
D_1
D_0
ground

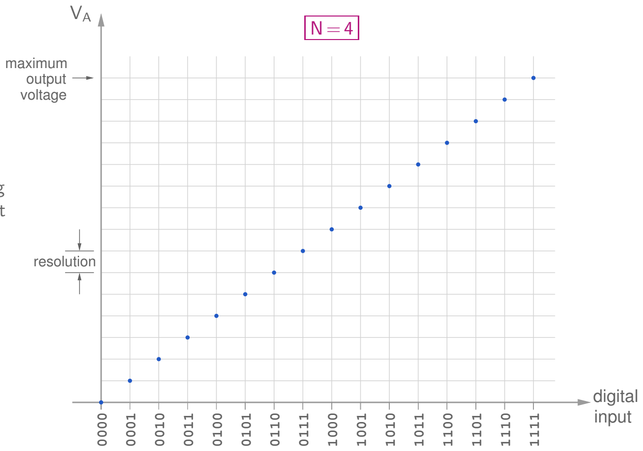
DAC



∗ For a 4-bit DAC, with input $S_3 S_2 S_1 S_0$, the output voltage is
$V_A = K \left[(S_3 \times 2^3) + (S_2 \times 2^2) + (S_1 \times 2^1) + (S_0 \times 2^0) \right]$.
In general, $V_A = K \sum_0^{N-1} S_k 2^k$.

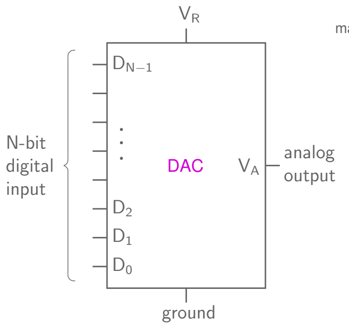
DAC

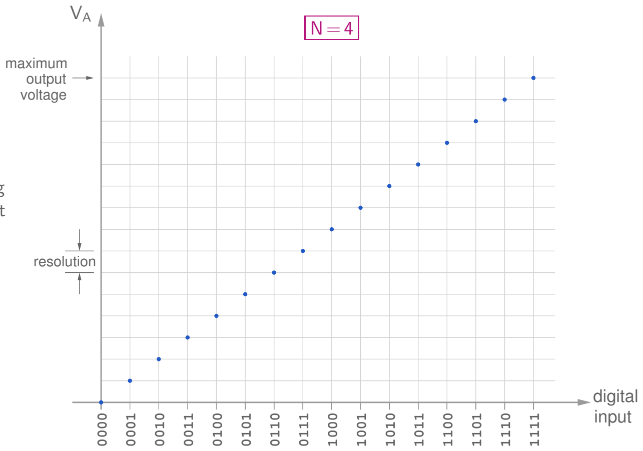




∗ For a 4-bit DAC, with input $S_3 S_2 S_1 S_0$, the output voltage is
$V_A = K\left[(S_3 \times 2^3) + (S_2 \times 2^2) + (S_1 \times 2^1) + (S_0 \times 2^0)\right]$.
In general, $V_A = K \sum_0^{N-1} S_k 2^k$.

DAC





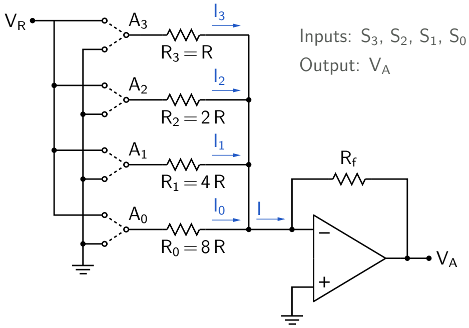
∗ For a 4-bit DAC, with input $S_3 S_2 S_1 S_0$, the output voltage is

$V_A = K\left[(S_3 \times 2^3) + (S_2 \times 2^2) + (S_1 \times 2^1) + (S_0 \times 2^0)\right]$.

In general, $V_A = K \sum_0^{N-1} S_k 2^k$.

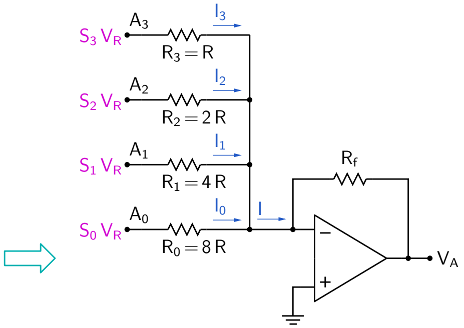
∗ K is proportional to the reference voltage V_R. Its value depends on how the DAC is implemented.

DAC using binary-weighted resistors

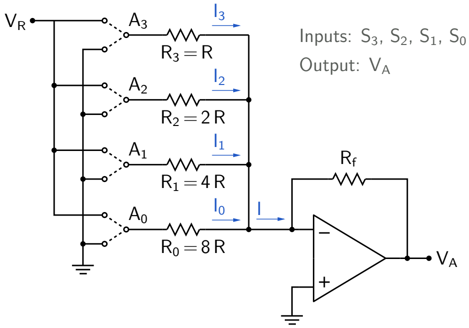


Inputs: S_3, S_2, S_1, S_0

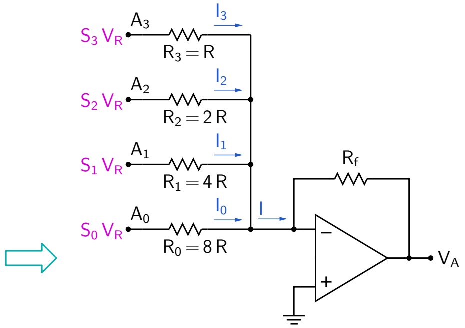
Output: V_A



DAC using binary-weighted resistors

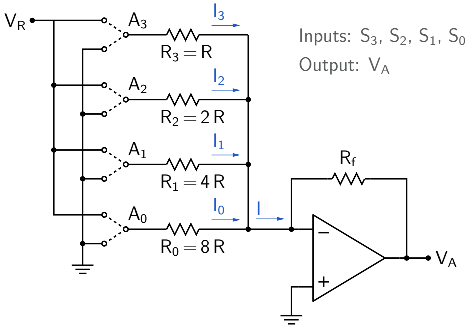


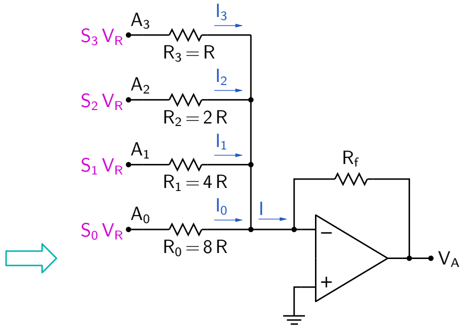
Inputs: S_3, S_2, S_1, S_0
Output: V_A



* If the input bit S_k is 1, A_k gets connected to V_R; else, it gets connected to ground.

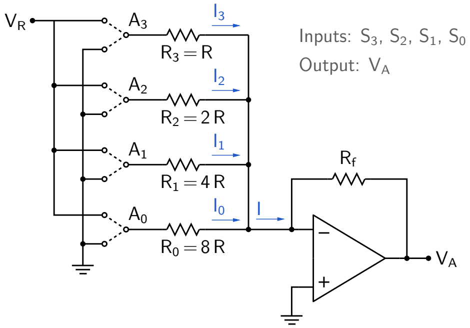
DAC using binary-weighted resistors

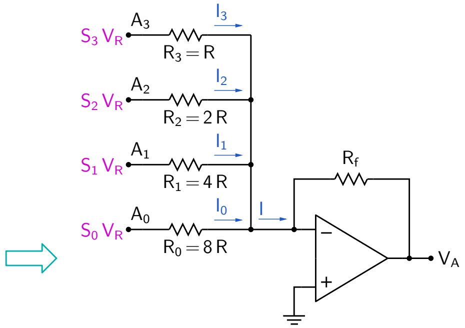




* If the input bit S_k is 1, A_k gets connected to V_R; else, it gets connected to ground. $\rightarrow V(A_k) = S_k \times V_R$.

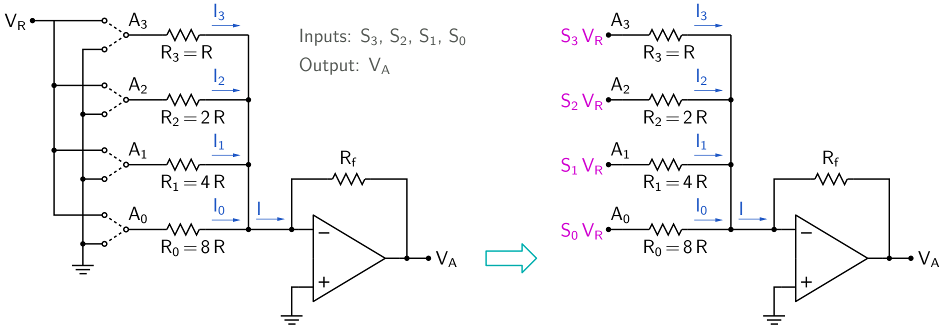
DAC using binary-weighted resistors





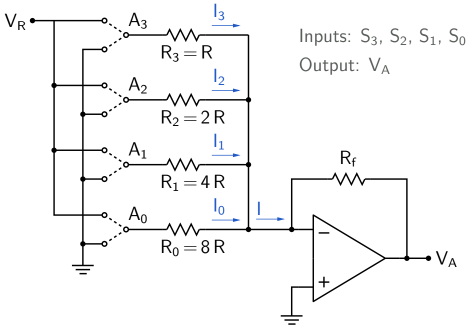
* If the input bit S_k is 1, A_k gets connected to V_R; else, it gets connected to ground. $\rightarrow V(A_k) = S_k \times V_R$.
* Since the inverting terminal of the op-amp is at virtual ground, $I_k = \dfrac{V(A_k) - 0}{R_k} = \dfrac{S_k\, V_R}{R_k}$.

DAC using binary-weighted resistors



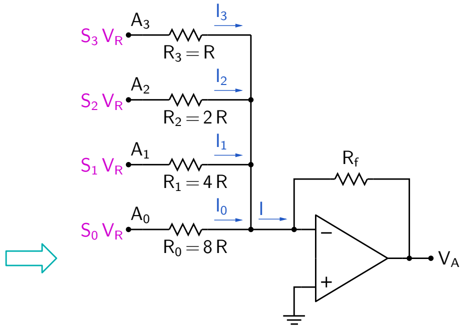
* If the input bit S_k is 1, A_k gets connected to V_R; else, it gets connected to ground. $\rightarrow V(A_k) = S_k \times V_R$.
* Since the inverting terminal of the op-amp is at virtual ground, $I_k = \dfrac{V(A_k) - 0}{R_k} = \dfrac{S_k\,V_R}{R_k}$.
* $I = \dfrac{S_0\,V_R}{8\,R} + \dfrac{S_1\,V_R}{4\,R} + \dfrac{S_2\,V_R}{2\,R} + \dfrac{S_3\,V_R}{R} = \dfrac{V_R}{2^{N-1}R} \sum_0^{N-1} S_k \times 2^k \; (N=4)$.

DAC using binary-weighted resistors



Inputs: S_3, S_2, S_1, S_0

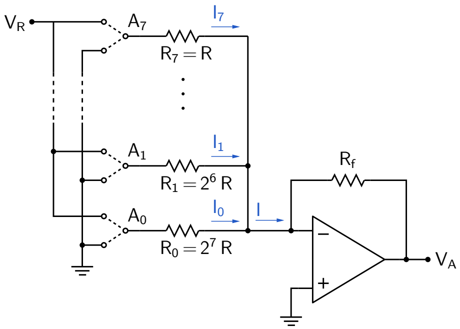
Output: V_A



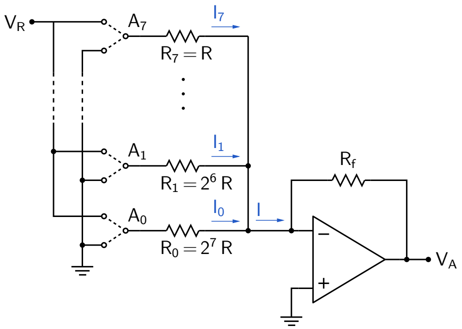
* If the input bit S_k is 1, A_k gets connected to V_R; else, it gets connected to ground. $\rightarrow V(A_k) = S_k \times V_R$.
* Since the inverting terminal of the op-amp is at virtual ground, $I_k = \dfrac{V(A_k) - 0}{R_k} = \dfrac{S_k\,V_R}{R_k}$.
* $I = \dfrac{S_0\,V_R}{8\,R} + \dfrac{S_1\,V_R}{4\,R} + \dfrac{S_2\,V_R}{2\,R} + \dfrac{S_3\,V_R}{R} = \dfrac{V_R}{2^{N-1}R} \displaystyle\sum_0^{N-1} S_k \times 2^k$ $(N=4)$.
* The output voltage is $V_o = -R_f\,I = -V_R\,\dfrac{R_f}{2^{N-1}R} \displaystyle\sum_0^{N-1} S_k \times 2^k$.

M. B. Patil, IIT Bombay

DAC using binary-weighted resistors: Example

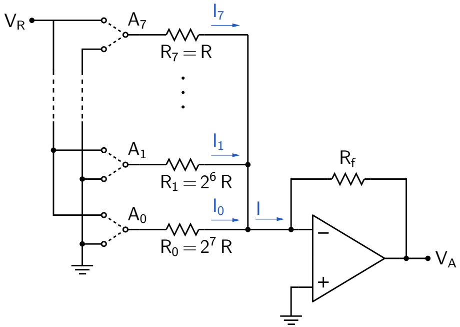


DAC using binary-weighted resistors: Example



* Consider an 8-bit DAC with $V_R = 5\,\text{V}$. What is the smallest value of R which will limit the current drawn from the supply (V_R) to 10 mA?

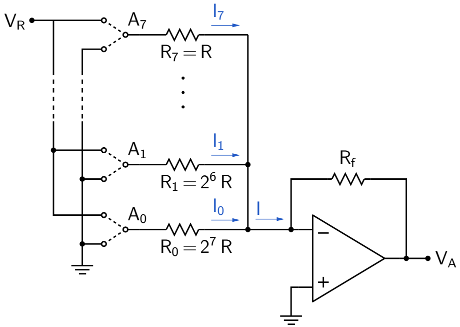
DAC using binary-weighted resistors: Example



* Consider an 8-bit DAC with $V_R = 5\,\text{V}$. What is the smallest value of R which will limit the current drawn from the supply (V_R) to 10 mA?

  Maximum current is drawn from V_R when the input is 1111 1111.

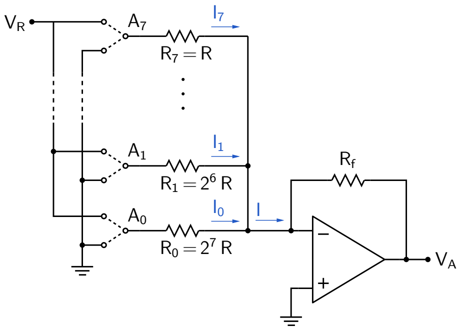
DAC using binary-weighted resistors: Example



* Consider an 8-bit DAC with $V_R = 5\,\text{V}$. What is the smallest value of R which will limit the current drawn from the supply (V_R) to 10 mA?

  Maximum current is drawn from V_R when the input is 1111 1111.
  $\rightarrow$ All nodes A_0 to A_7 get connected to V_R.

DAC using binary-weighted resistors: Example



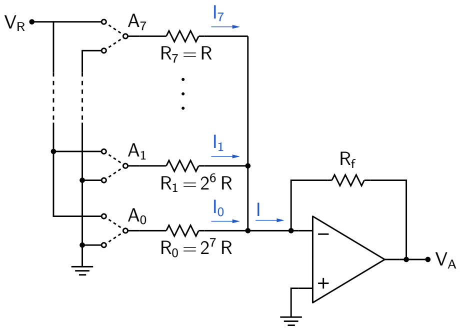
* Consider an 8-bit DAC with $V_R = 5\,\text{V}$. What is the smallest value of R which will limit the current drawn from the supply (V_R) to 10 mA?

Maximum current is drawn from V_R when the input is 1111 1111.
$\rightarrow$ All nodes A_0 to A_7 get connected to V_R.

$$\rightarrow 10\,\text{mA} = \frac{V_R}{R} + \frac{V_R}{2R} + \cdots + \frac{V_R}{2^7 R} = \frac{1}{2^7}\,\frac{V_R}{R}\,\left(2^0 + 2^1 + \cdots + 2^7\right)$$

$$= \frac{1}{2^7}\,\frac{V_R}{R}\,\left(2^8 - 1\right) = \frac{255}{128}\,\frac{V_R}{R}$$

DAC using binary-weighted resistors: Example



* Consider an 8-bit DAC with $V_R = 5\,\text{V}$. What is the smallest value of R which will limit the current drawn from the supply (V_R) to 10 mA?

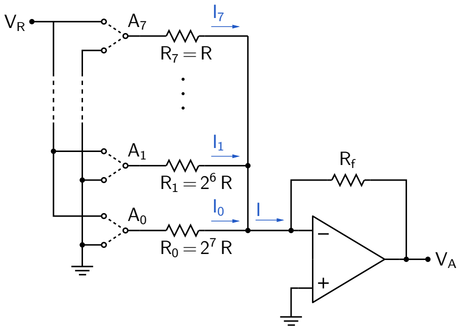
Maximum current is drawn from V_R when the input is 1111 1111.
→ All nodes A_0 to A_7 get connected to V_R.

$$\rightarrow 10\,\text{mA} = \frac{V_R}{R} + \frac{V_R}{2R} + \cdots + \frac{V_R}{2^7 R} = \frac{1}{2^7}\frac{V_R}{R}\left(2^0 + 2^1 + \cdots + 2^7\right)$$

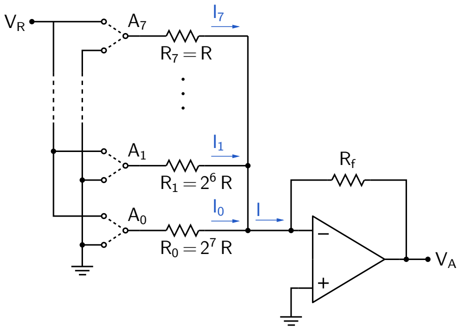
$$= \frac{1}{2^7}\frac{V_R}{R}\left(2^8 - 1\right) = \frac{255}{128}\frac{V_R}{R} \rightarrow R_{\text{min}} = \frac{5\,\text{V}}{10\,\text{mA}} \times \frac{255}{128} = 996\,\Omega\,.$$

(Ref.: K. Gopalan, *Introduction to Digital Microelectronic Circuits*, Tata McGraw-Hill, New Delhi, 1998)

DAC using binary-weighted resistors: Example (from Gopalan)

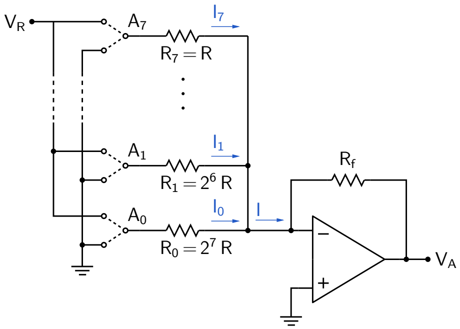


DAC using binary-weighted resistors: Example (from Gopalan)



* If $R_f = R$, what is the resolution (i.e., ΔV_A corresponding to the input LSB changing from 0 to 1 with other input bits constant)?

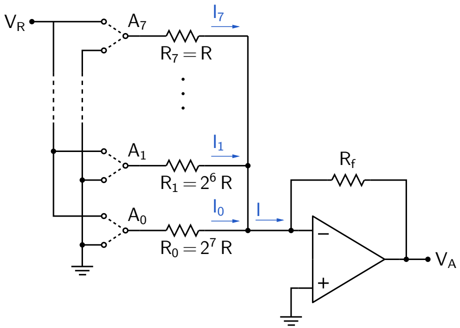
DAC using binary-weighted resistors: Example (from Gopalan)



* If $R_f = R$, what is the resolution (i.e., ΔV_A corresponding to the input LSB changing from 0 to 1 with other input bits constant)?

$$V_A = -V_R\,\frac{R_f}{2^{N-1}R}\,\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right]$$

DAC using binary-weighted resistors: Example (from Gopalan)

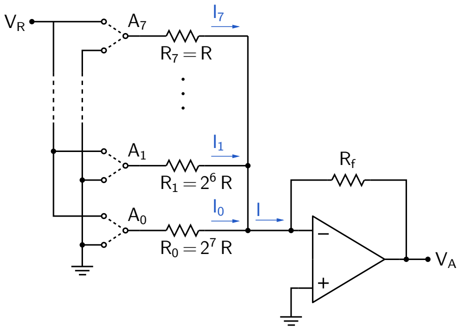


* If $R_f = R$, what is the resolution (i.e., ΔV_A corresponding to the input LSB changing from 0 to 1 with other input bits constant)?

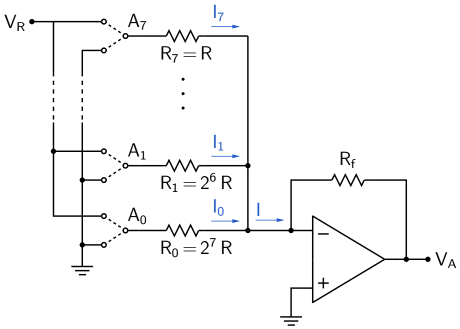
$$V_A = -V_R\,\frac{R_f}{2^{N-1}R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right]$$

$$\rightarrow \Delta V_A = \frac{V_R}{2^{N-1}}\,\frac{R_f}{R} = \frac{5\,\text{V}}{2^{8-1}} \times 1 = \frac{5}{128} = 0.0391\,\text{V}.$$

DAC using binary-weighted resistors: Example (from Gopalan)

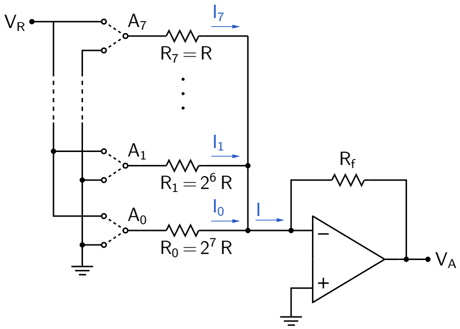


DAC using binary-weighted resistors: Example (from Gopalan)



* What is the maximum output voltage (in magnitude)?

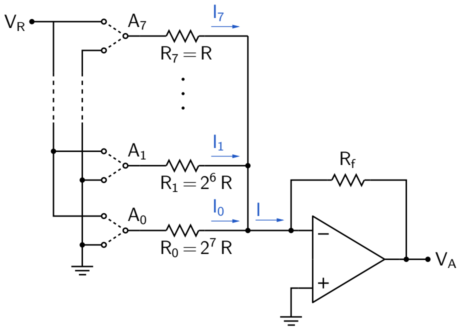
DAC using binary-weighted resistors: Example (from Gopalan)



- What is the maximum output voltage (in magnitude)?

$$V_A = -\frac{V_R}{2^{N-1}}\,\frac{R_f}{R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right].$$

DAC using binary-weighted resistors: Example (from Gopalan)

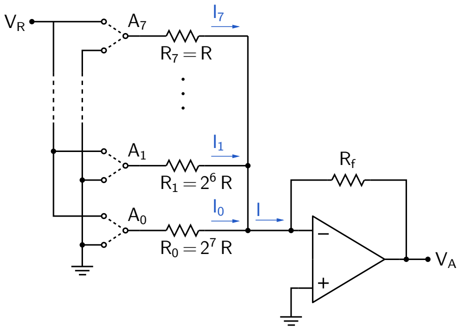


* What is the maximum output voltage (in magnitude)?

$$V_A = -\frac{V_R}{2^{N-1}}\,\frac{R_f}{R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right].$$

Maximum V_A (in magnitude) is obtained when the input is 1111 1111.

DAC using binary-weighted resistors: Example (from Gopalan)



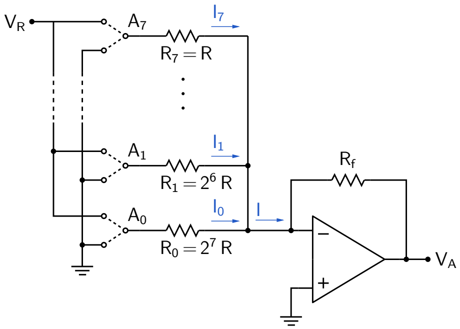
* What is the maximum output voltage (in magnitude)?

$$V_A = -\frac{V_R}{2^{N-1}}\,\frac{R_f}{R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right].$$

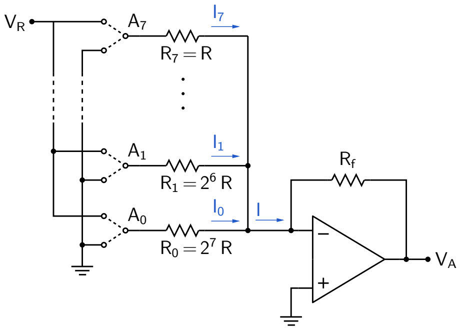
Maximum V_A (in magnitude) is obtained when the input is 1111 1111.

$$|V_A|^{\text{max}} = \frac{5}{128} \times 1 \times \left[2^0 + 2^1 + \cdots + 2^7\right] = \frac{5}{128} \times \left(2^8 - 1\right) = 5 \times \frac{255}{128} = 9.961\,\text{V}.$$

DAC using binary-weighted resistors: Example (from Gopalan)

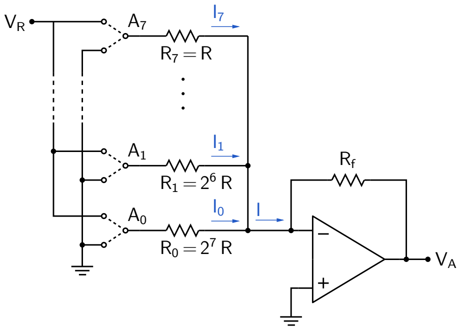


DAC using binary-weighted resistors: Example (from Gopalan)



∗ Find the output voltage corresponding to the input 1010 1101.

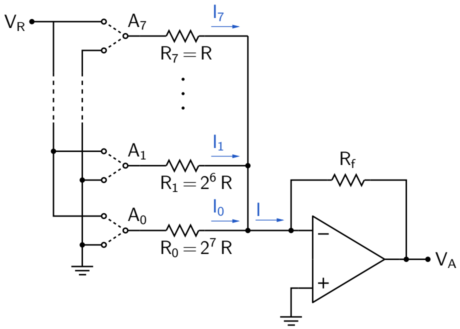
DAC using binary-weighted resistors: Example (from Gopalan)



* Find the output voltage corresponding to the input 1010 1101.

$$V_A = -\frac{V_R}{2^{N-1}}\,\frac{R_f}{R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right].$$

DAC using binary-weighted resistors: Example (from Gopalan)

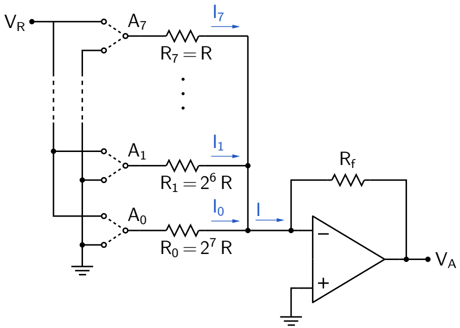


* Find the output voltage corresponding to the input 1010 1101.

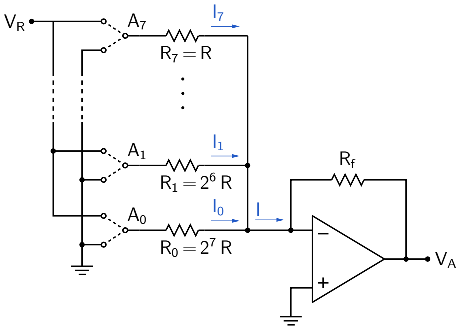
$$V_A = -\frac{V_R}{2^{N-1}}\,\frac{R_f}{R}\left[S_7 2^7 + \cdots + S_1 2^1 + S_0 2^0\right].$$

$$= -\frac{5}{128} \times 1 \times \left[2^7 + 2^5 + 2^3 + 2^2 + 2^0\right] = -5 \times \frac{173}{128} = -6.758\,\text{V}\,.$$

DAC using binary-weighted resistors: Example (from Gopalan)

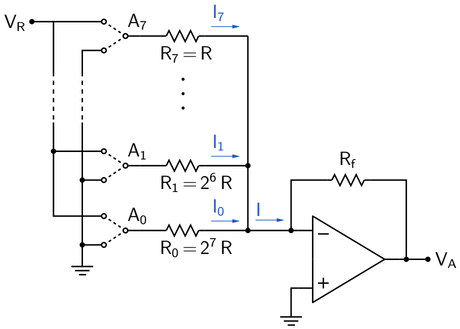


DAC using binary-weighted resistors: Example (from Gopalan)



∗ If the resistors are specified to have a tolerance of 1 %, what is the range of $|V_A|$ corresponding to input 1111 1111?

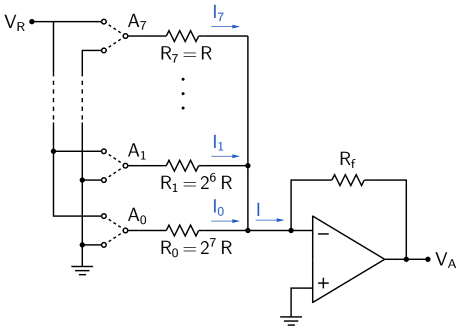
DAC using binary-weighted resistors: Example (from Gopalan)



* If the resistors are specified to have a tolerance of $1\,\%$, what is the range of $|V_A|$ corresponding to input 1111 1111?

$|V_A|$ is maximum when (a) currents I_0, I_1, etc. assume their maximum values, with $R_k = R_k^0 \times (1 - 0.01)$ and (b) R_f is maximum, $R_f = R_f^0 \times (1 + 0.01)$.
(The superscript '0' denotes nominal value.)

DAC using binary-weighted resistors: Example (from Gopalan)

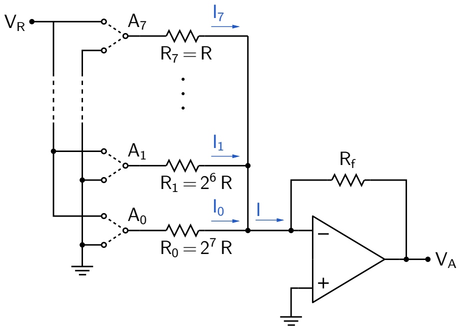


* If the resistors are specified to have a tolerance of 1 %, what is the range of $|V_A|$ corresponding to input 1111 1111?

$|V_A|$ is maximum when (a) currents I_0, I_1, etc. assume their maximum values, with $R_k = R_k^0 \times (1 - 0.01)$ and (b) R_f is maximum, $R_f = R_f^0 \times (1 + 0.01)$.
(The superscript '0' denotes nominal value.)

$$\rightarrow |V_A|^{\text{max}}_{11111111} = V_R \times \frac{255}{128} \times \left.\frac{R_f}{R}\right|^{\text{max}} = 5 \times \frac{255}{128} \times \frac{1.01}{0.99} = 10.162\,\text{V}.$$

DAC using binary-weighted resistors: Example (from Gopalan)



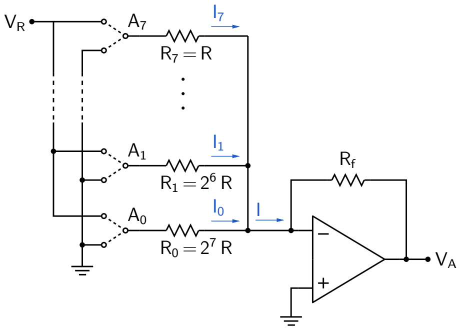
* If the resistors are specified to have a tolerance of 1 %, what is the range of $|V_A|$ corresponding to input 1111 1111?

  $|V_A|$ is maximum when (a) currents I_0, I_1, etc. assume their maximum values, with $R_k = R_k^0 \times (1 - 0.01)$ and (b) R_f is maximum, $R_f = R_f^0 \times (1 + 0.01)$.
  (The superscript '0' denotes nominal value.)

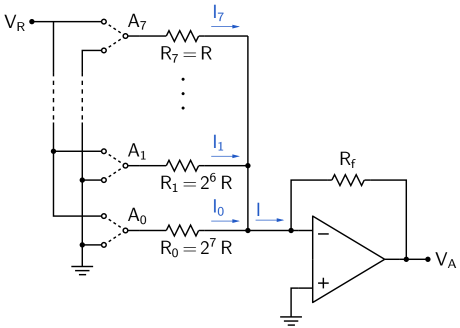
  $$\rightarrow |V_A|^{\text{max}}_{11111111} = V_R \times \frac{255}{128} \times \left.\frac{R_f}{R}\right|^{\text{max}} = 5 \times \frac{255}{128} \times \frac{1.01}{0.99} = 10.162\,\text{V}.$$

  $$\text{Similarly, } |V_A|^{\text{min}}_{11111111} = 5 \times \frac{255}{128} \times \frac{0.99}{1.01} = 9.764\,\text{V}.$$

DAC using binary-weighted resistors: Example (from Gopalan)

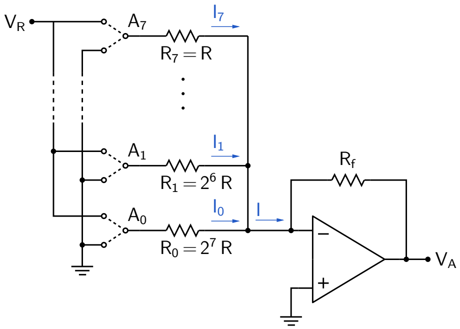


DAC using binary-weighted resistors: Example (from Gopalan)



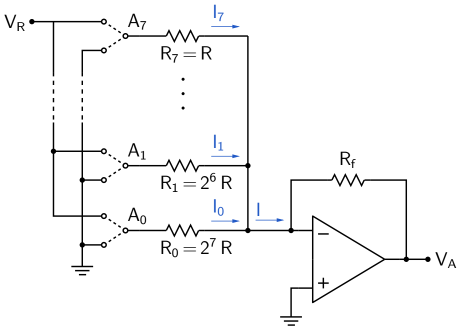
* ΔV_A for input 1111 1111 $= 10.162 - 9.764 \approx 0.4\,\text{V}$ which is larger than the resolution ($0.039\,\text{V}$) of the DAC. This situation is not acceptable.

DAC using binary-weighted resistors: Example (from Gopalan)



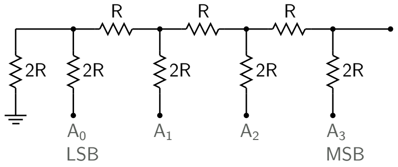
* ΔV_A for input 1111 1111 $= 10.162 - 9.764 \approx 0.4\,\text{V}$ which is larger than the resolution ($0.039\,\text{V}$) of the DAC. This situation is not acceptable.
* The output voltage variation can be reduced by using resistors with a smaller tolerance. However, it is difficult to fabricate an IC with widely varying resistance values (from R to $2^{N-1}R$) and each with a small enough tolerance.

DAC using binary-weighted resistors: Example (from Gopalan)



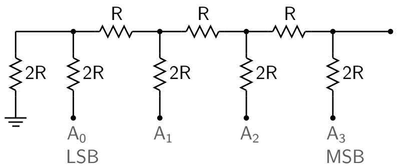
* ΔV_A for input 1111 1111 $= 10.162 - 9.764 \approx 0.4\,\text{V}$ which is larger than the resolution ($0.039\,\text{V}$) of the DAC. This situation is not acceptable.
* The output voltage variation can be reduced by using resistors with a smaller tolerance. However, it is difficult to fabricate an IC with widely varying resistance values (from R to $2^{N-1}R$) and each with a small enough tolerance.
$\rightarrow$ use $R-2R$ ladder network instead.

R-2R ladder network



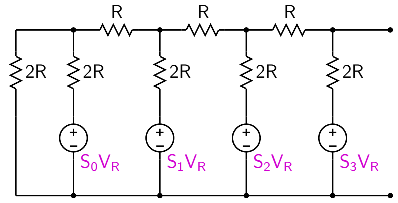
Node A_k is connected to V_R if input bit S_k is 1; else, it is connected to ground.

R-2R ladder network

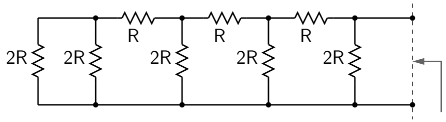


Node A_k is connected to V_R if input bit S_k is 1; else, it is connected to ground.

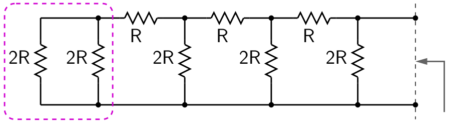
The original network is equivalent to



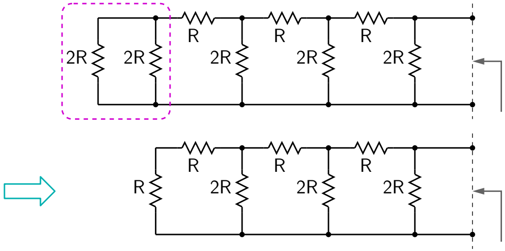
R-2R ladder network: Thevenin resistance



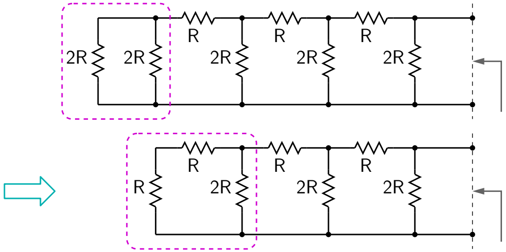
R-2R ladder network: Thevenin resistance



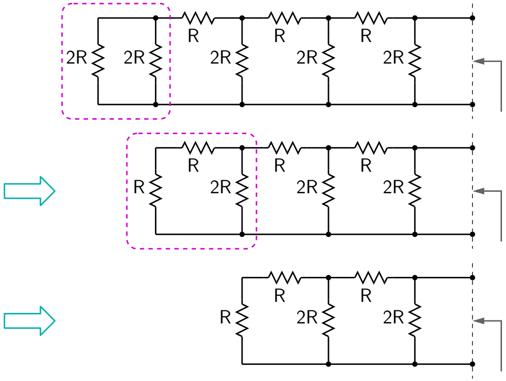
R-2R ladder network: Thevenin resistance



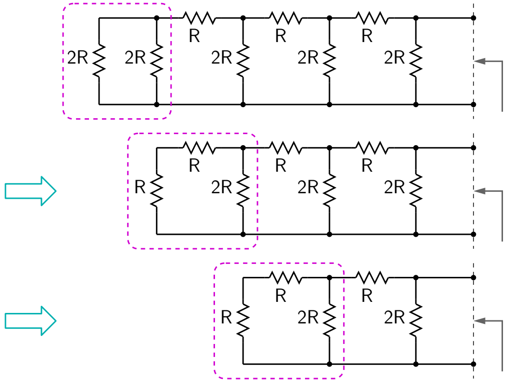
R-2R ladder network: Thevenin resistance



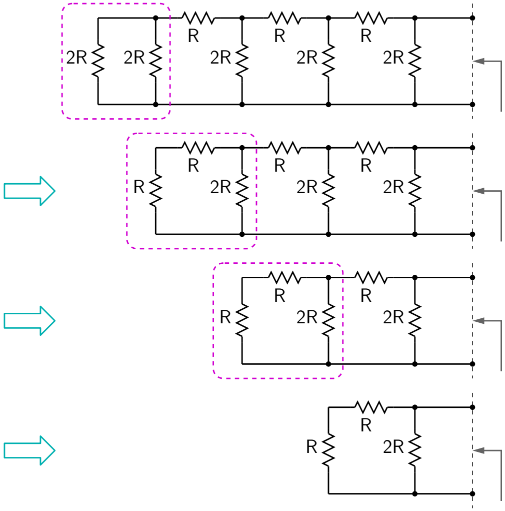
R-2R ladder network: Thevenin resistance



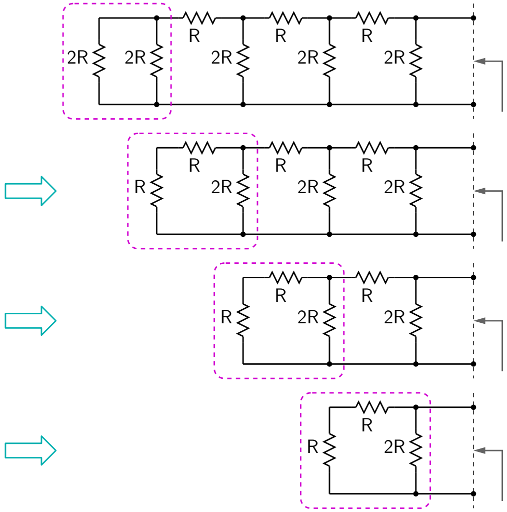
R-2R ladder network: Thevenin resistance



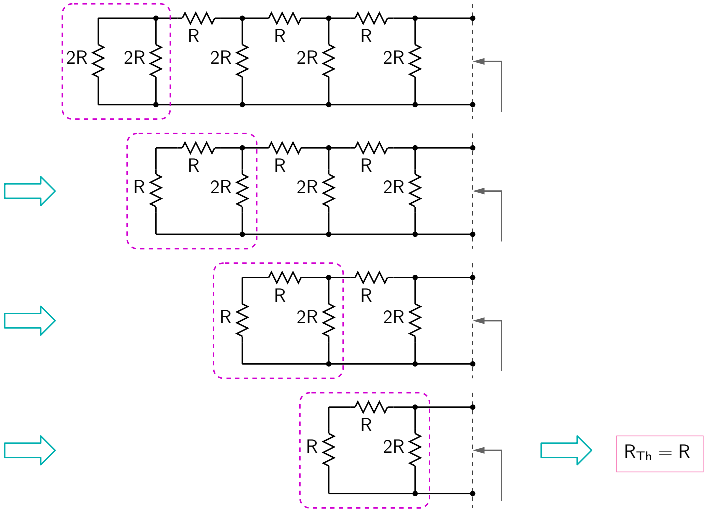
R-2R ladder network: Thevenin resistance



R-2R ladder network: Thevenin resistance

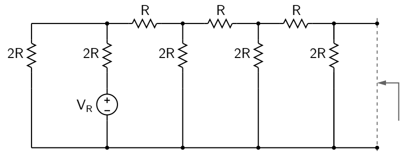


R-2R ladder network: Thevenin resistance

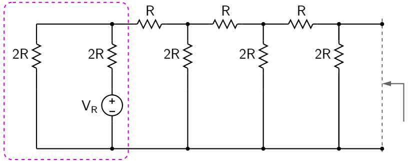


$R_{Th} = R$

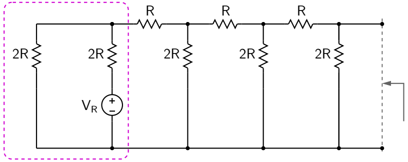
R-2R ladder network: V_{Th} for $S_0 = 1$

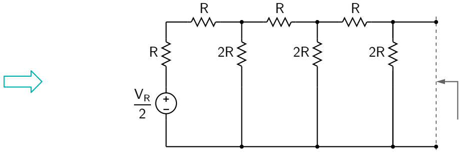


R-2R ladder network: V_{Th} for $S_0 = 1$

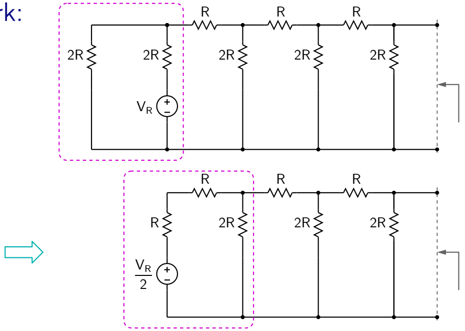


R-2R ladder network: V_{Th} for $S_0 = 1$

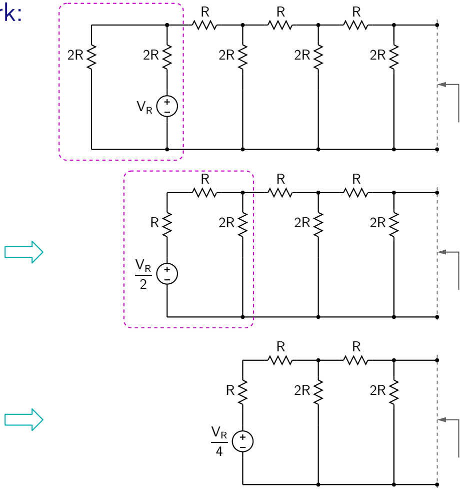




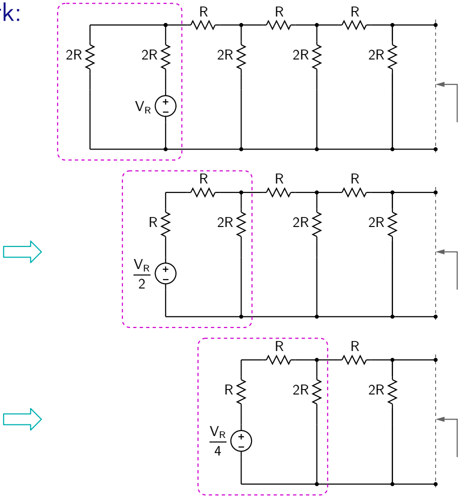
R-2R ladder network: V_{Th} for $S_0 = 1$



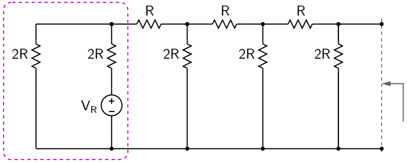
R-2R ladder network: V_{Th} for $S_0 = 1$

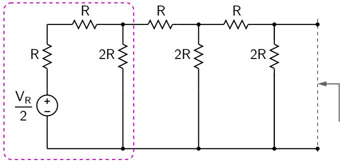


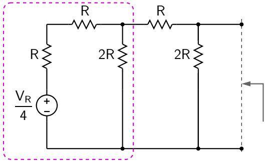
R-2R ladder network:
V_{Th} for $S_0 = 1$

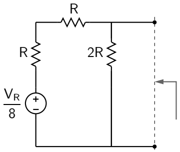


R-2R ladder network: V_{Th} for $S_0 = 1$

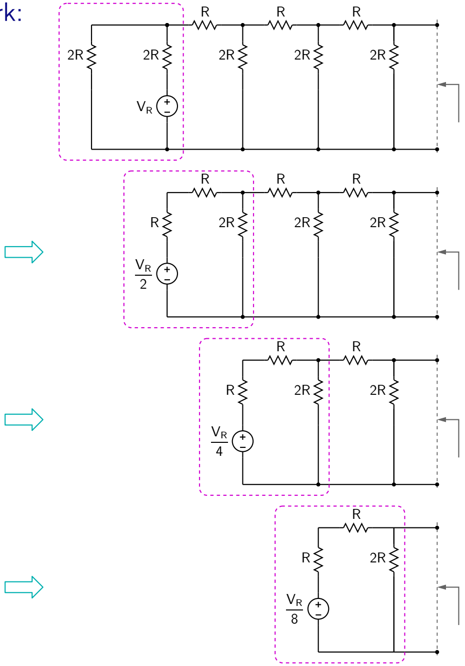




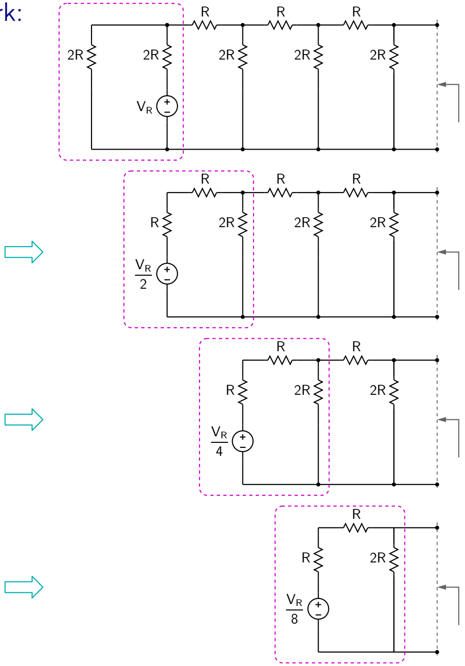




R-2R ladder network: V_{Th} for $S_0 = 1$



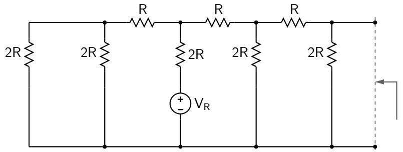
R-2R ladder network: V_{Th} for $S_0 = 1$



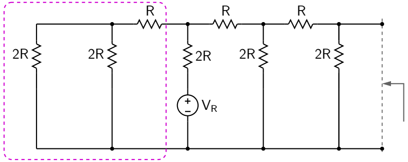
$$\Rightarrow \quad \boxed{V_{Th} = \frac{V_R}{16}}$$

M. B. Patil, IIT Bombay

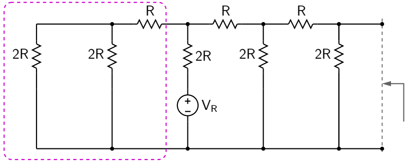
R-2R ladder network:
V_{Th} for $S_1 = 1$

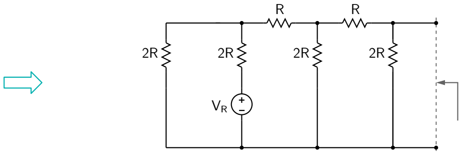


R-2R ladder network: V_{Th} for $S_1 = 1$

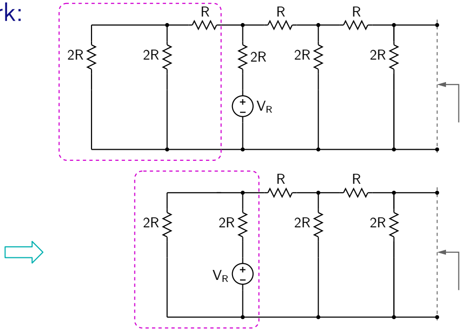


R-2R ladder network: V_{Th} for $S_1 = 1$

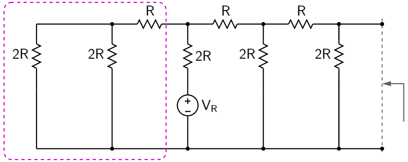


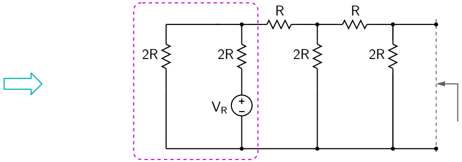


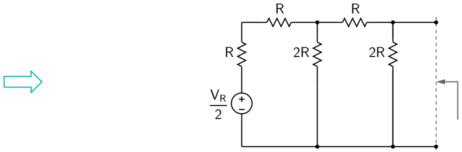
R-2R ladder network: V_{Th} for $S_1 = 1$



R-2R ladder network: V_{Th} for $S_1 = 1$

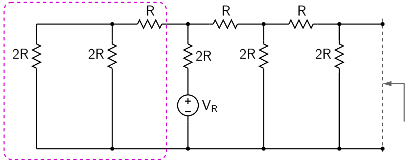


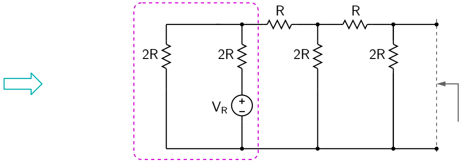


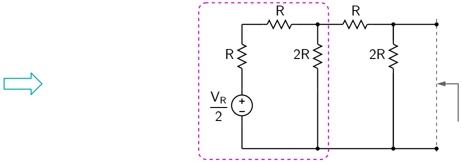


R-2R ladder network:

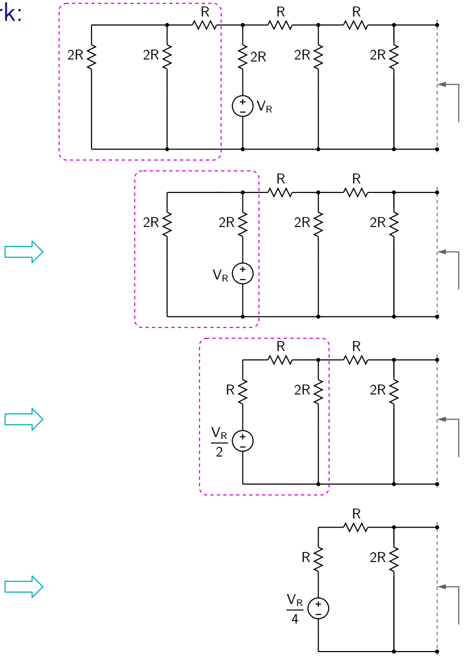
V_{Th} for $S_1 = 1$



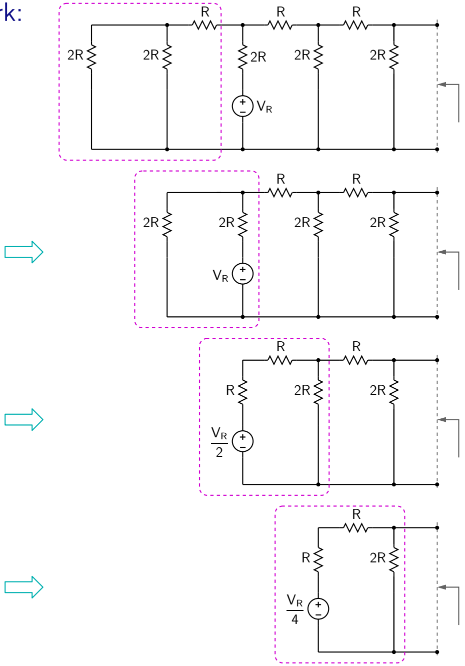




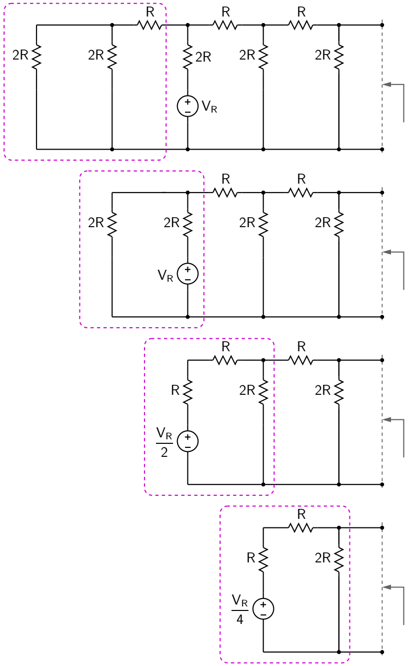
R-2R ladder network:
V_{Th} for $S_1 = 1$



R-2R ladder network: V_{Th} for $S_1 = 1$

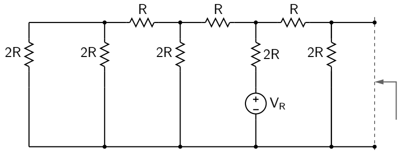


R-2R ladder network: V_{Th} for $S_1 = 1$

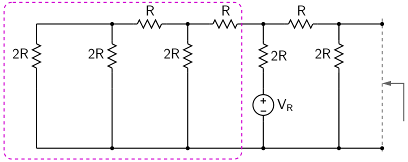


$$\boxed{V_{Th} = \frac{V_R}{8}}$$

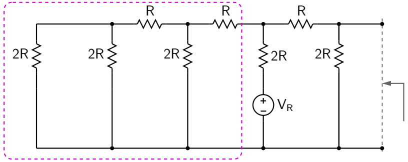
R-2R ladder network: V_{Th} for $S_2 = 1$

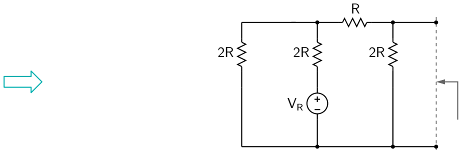


R-2R ladder network:
V_{Th} for $S_2 = 1$

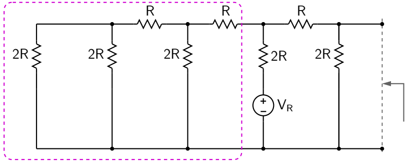


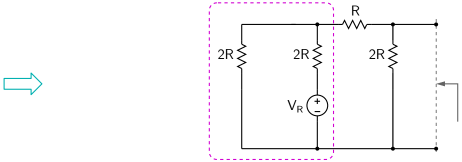
R-2R ladder network: V_{Th} for $S_2 = 1$



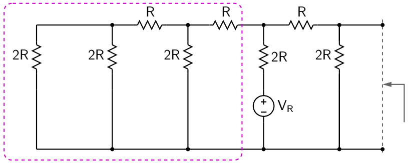


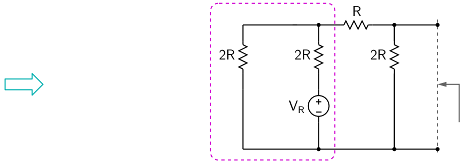
R-2R ladder network:
V_{Th} for $S_2 = 1$

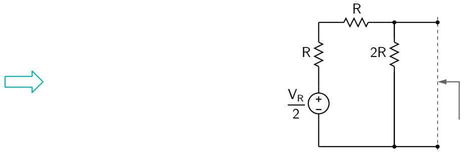




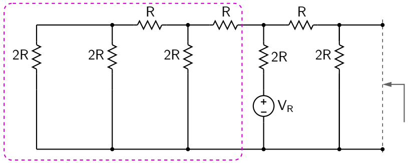
R-2R ladder network: V_{Th} for $S_2 = 1$

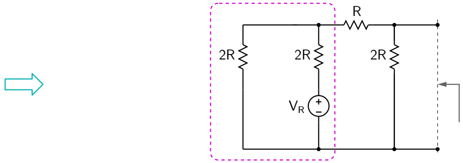


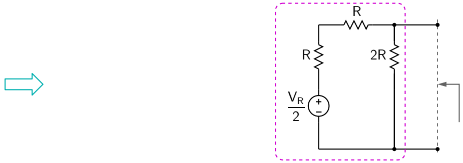




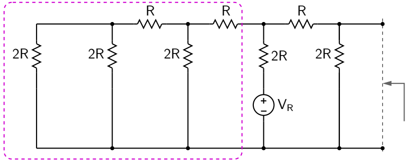
R-2R ladder network: V_{Th} for $S_2 = 1$

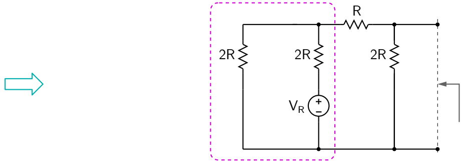


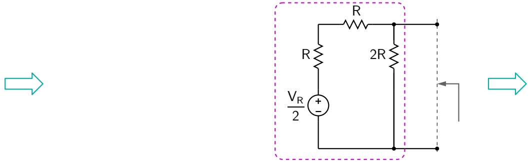




R-2R ladder network: V_{Th} for $S_2 = 1$

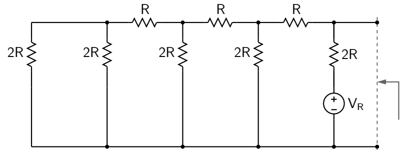




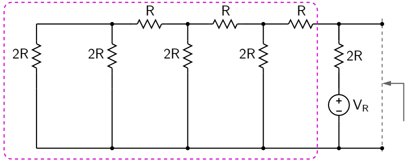


$$\boxed{V_{Th} = \frac{V_R}{4}}$$

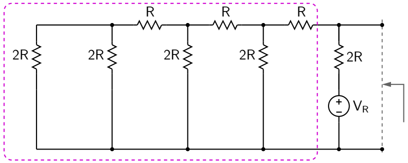
R-2R ladder network: V_{Th} for $S_3 = 1$

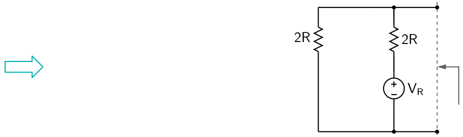


R-2R ladder network:
V_{Th} for $S_3 = 1$

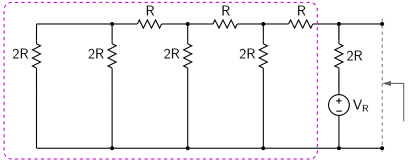


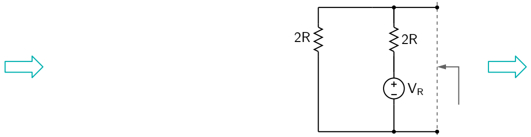
R-2R ladder network: V_{Th} for $S_3 = 1$





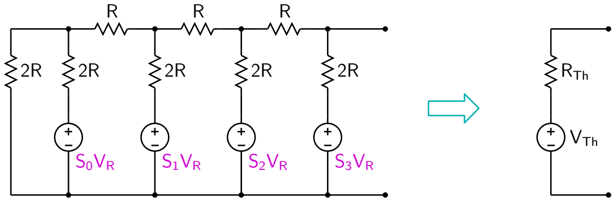
R-2R ladder network: V_{Th} for $S_3 = 1$



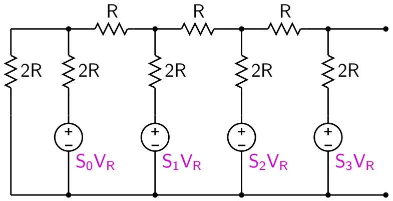


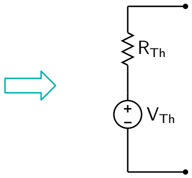
$$\boxed{V_{Th} = \frac{V_R}{2}}$$

R-2R ladder network: R_{Th} and V_{Th}



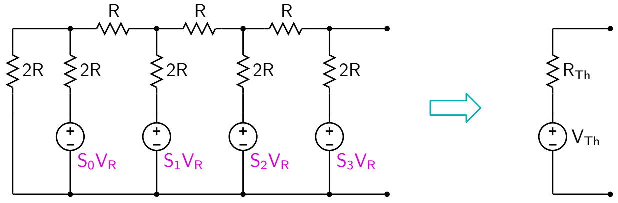
R-2R ladder network: R_{Th} and V_{Th}





* $R_{Th} = R$.

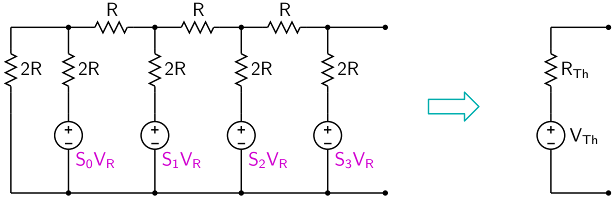
R-2R ladder network: R_{Th} and V_{Th}



* $R_{Th} = R$.
* $V_{Th} = V_{Th}^{(S0)} + V_{Th}^{(S1)} + V_{Th}^{(S2)} + V_{Th}^{(S3)}$

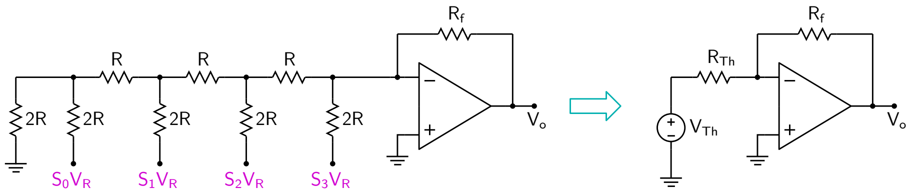
$$= \frac{V_R}{16}\left[S_0\,2^0 + S_1\,2^1 + S_2\,2^2 + S_3\,2^3\right] .$$

R-2R ladder network: R_{Th} and V_{Th}

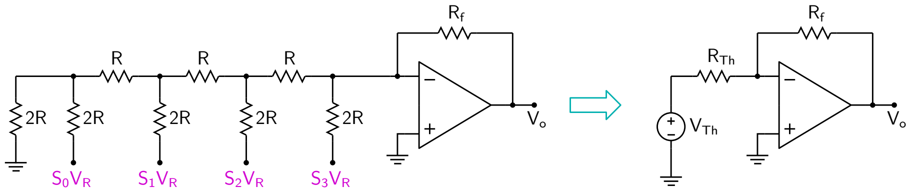


* $R_{Th} = R\,.$
* $V_{Th} = V_{Th}^{(S0)} + V_{Th}^{(S1)} + V_{Th}^{(S2)} + V_{Th}^{(S3)}$
  $$= \frac{V_R}{16}\left[S_0\,2^0 + S_1\,2^1 + S_2\,2^2 + S_3\,2^3\right]\,.$$
* We can use the R-$2R$ ladder network and an op-amp to make up a DAC $\rightarrow$ next slide.

DAC with R-2R ladder

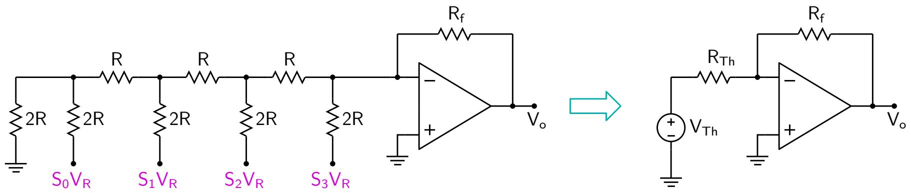


DAC with R-2R ladder



* $V_o = -\dfrac{R_f}{R_{Th}}\,V_{Th} = -\dfrac{R_f}{R_{Th}}\,\dfrac{V_R}{16}\,\left[S_0\,2^0 + S_1\,2^1 + S_2\,2^2 + S_3\,2^3\right]\,.$

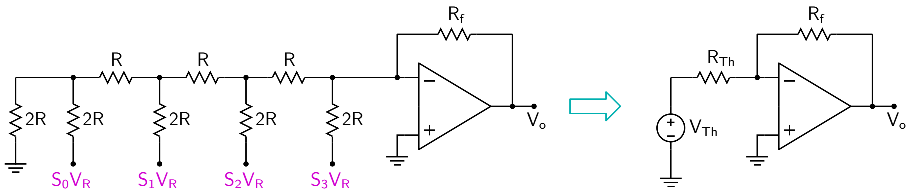
DAC with R-2R ladder



* $V_o = -\dfrac{R_f}{R_{Th}}\,V_{Th} = -\dfrac{R_f}{R_{Th}}\,\dfrac{V_R}{16}\,\left[S_0\,2^0 + S_1\,2^1 + S_2\,2^2 + S_3\,2^3\right]$.

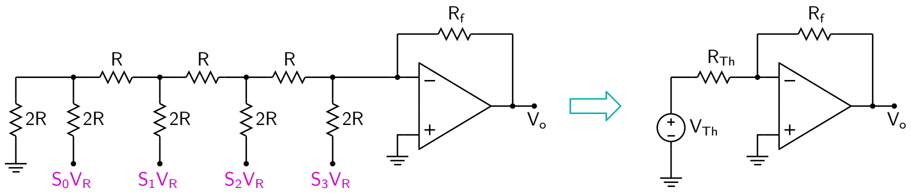
* For an N-bit DAC, $V_o = -\dfrac{R_f}{R_{Th}}\,V_{Th} = -\dfrac{R_f}{R_{Th}}\,\dfrac{V_R}{2^N}\,\sum_0^{N-1} S_k 2^k$.

DAC with R-2R ladder



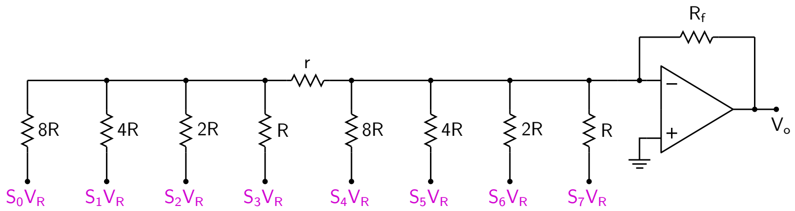
- $V_o = -\dfrac{R_f}{R_{Th}}\,V_{Th} = -\dfrac{R_f}{R_{Th}}\,\dfrac{V_R}{16}\,\left[S_0\,2^0 + S_1\,2^1 + S_2\,2^2 + S_3\,2^3\right]$.
- For an N-bit DAC, $V_o = -\dfrac{R_f}{R_{Th}}\,V_{Th} = -\dfrac{R_f}{R_{Th}}\,\dfrac{V_R}{2^N}\,\sum_0^{N-1} S_k 2^k$.
- 6- to 20-bit DACs based on the R-2R ladder network are commercially available in monolithic form (single chip).

DAC with R-2R ladder



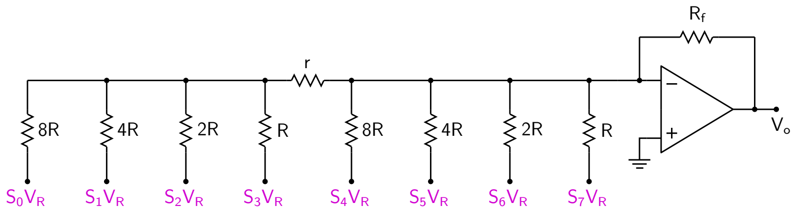
* $V_o = -\dfrac{R_f}{R_{Th}} V_{Th} = -\dfrac{R_f}{R_{Th}} \dfrac{V_R}{16} \left[S_0\, 2^0 + S_1\, 2^1 + S_2\, 2^2 + S_3\, 2^3 \right]$.
* For an N-bit DAC, $V_o = -\dfrac{R_f}{R_{Th}} V_{Th} = -\dfrac{R_f}{R_{Th}} \dfrac{V_R}{2^N} \sum_0^{N-1} S_k 2^k$.
* 6- to 20-bit DACs based on the R-2R ladder network are commercially available in monolithic form (single chip).
* Bipolar, CMOS, or BiCMOS technology is used for these DACs.

DAC: home work



Combination of weighted–resistor and R–2R ladder networks

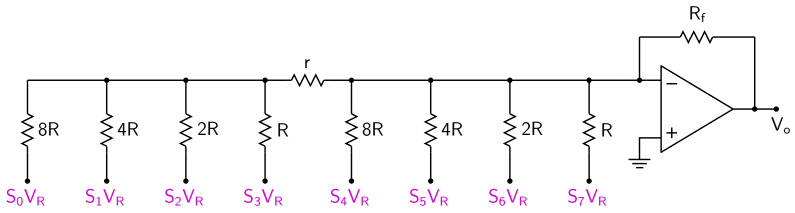
DAC: home work



Combination of weighted–resistor and R–2R ladder networks

* Find the value of r for the circuit to work as a regular (i.e., binary to analog) DAC.

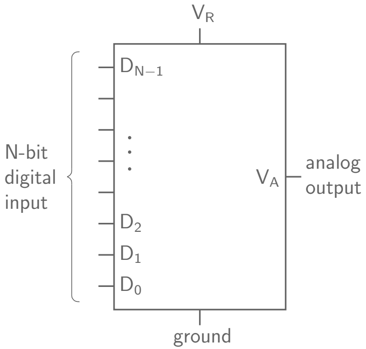
DAC: home work

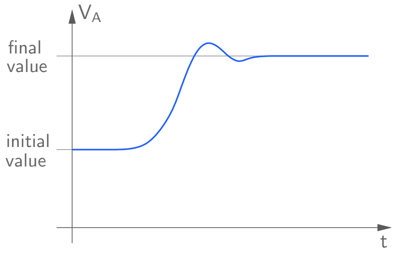


Combination of weighted–resistor and R–2R ladder networks

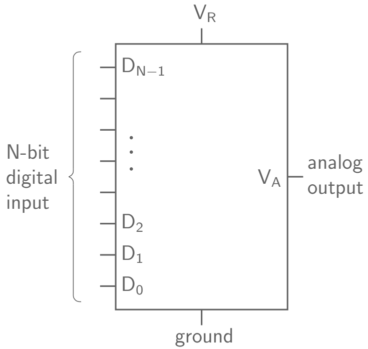
* Find the value of r for the circuit to work as a regular (i.e., binary to analog) DAC.
* Find the value of r for the circuit to work as a BCD to analog DAC.

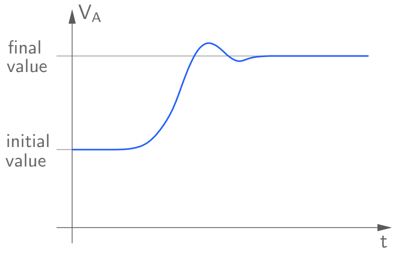
DAC: settling time





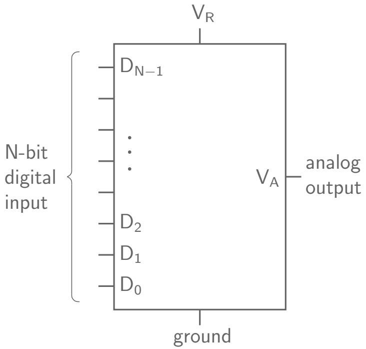
DAC: settling time

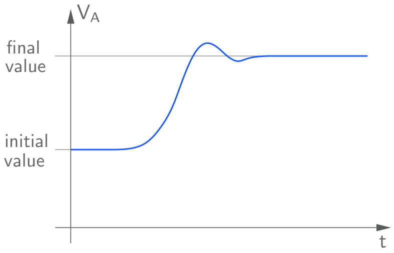




* When there is a change in the input binary number, the output V_A takes a finite time to settle to the new value.

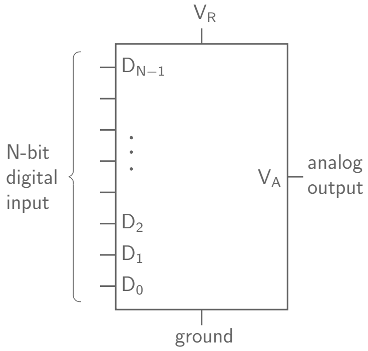
DAC: settling time

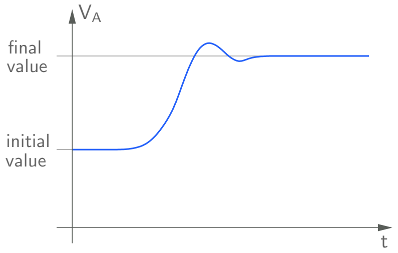




* When there is a change in the input binary number, the output V_A takes a finite time to settle to the new value.
* The finite settling time arises because of stray capacitances and switching delays of the semiconductor devices used within the DAC chip.

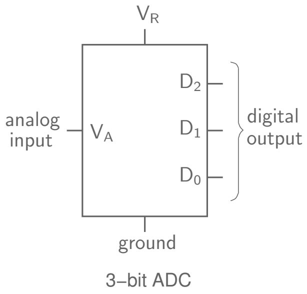
DAC: settling time

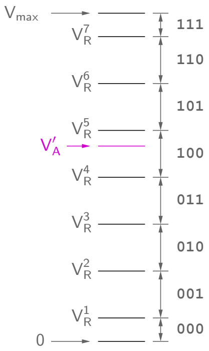




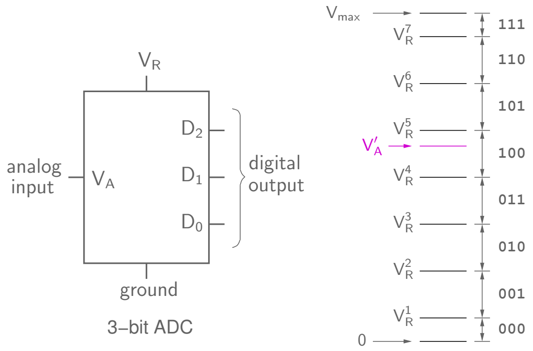
* When there is a change in the input binary number, the output V_A takes a finite time to settle to the new value.
* The finite settling time arises because of stray capacitances and switching delays of the semiconductor devices used within the DAC chip.
* Example: 500 ns to 0.2 % of full scale.

ADC: introduction



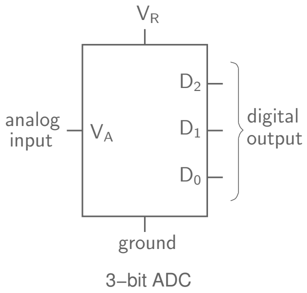


ADC: introduction

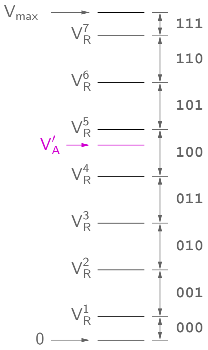


* If the input V_A is in the range $V_R^k < V_A < V_R^{k+1}$, the output is the binary number corresponding to the integer k. For example, for $V_A = V_A'$, the output is 100.

ADC: introduction

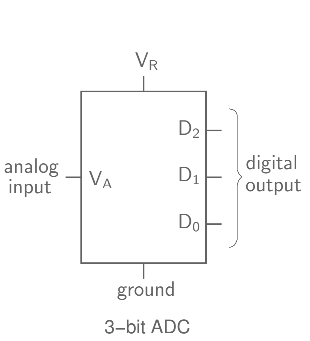


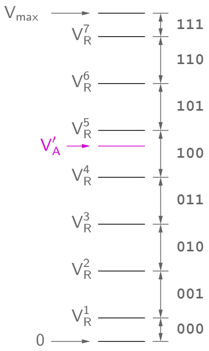
3–bit ADC



* If the input V_A is in the range $V_R^k < V_A < V_R^{k+1}$, the output is the binary number corresponding to the integer k. For example, for $V_A = V_A'$, the output is 100.
* We may think of each voltage interval (corresponding to 000, 001, etc.) as a "bin." In the above example, the input voltage V_A' falls in the 100 bin; therefore, the output of the ADC would be 100.

ADC: introduction



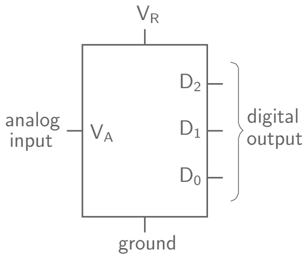


∗ If the input V_A is in the range $V_R^k < V_A < V_R^{k+1}$, the output is the binary number corresponding to the integer k. For example, for $V_A = V_A'$, the output is 100.

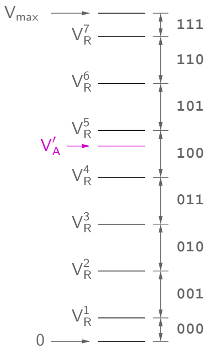
∗ We may think of each voltage interval (corresponding to 000, 001, etc.) as a "bin." In the above example, the input voltage V_A' falls in the 100 bin; therefore, the output of the ADC would be 100.

∗ Note that, for an N-bit ADC, there would be 2^N bins.

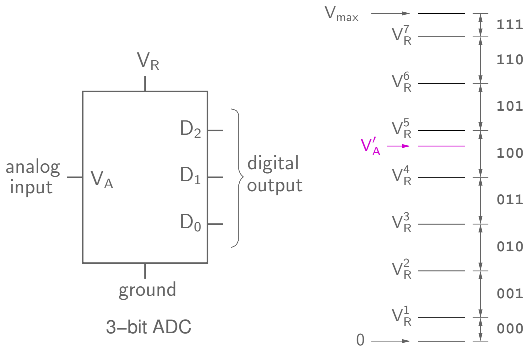
ADC: introduction



3–bit ADC

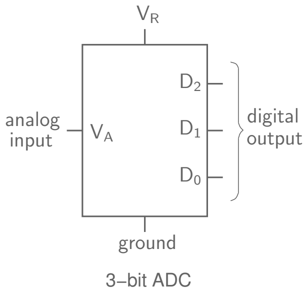


ADC: introduction

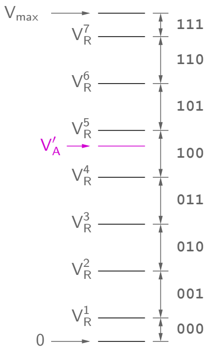


∗ The basic idea behind an ADC is simple:

ADC: introduction

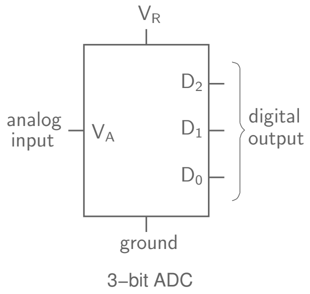


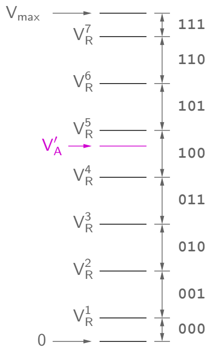
3–bit ADC



* The basic idea behind an ADC is simple:
  - Generate reference voltages V_R^1, V_R^2, etc.

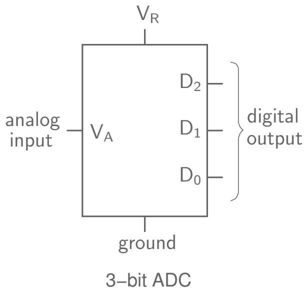
ADC: introduction



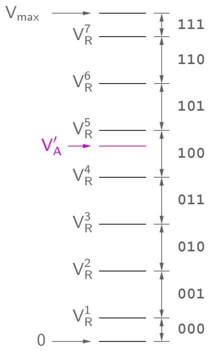


* The basic idea behind an ADC is simple:
  - Generate reference voltages V_R^1, V_R^2, etc.
  - Compare the input V_A with each of V_R^i to figure out which bin it belongs to.

ADC: introduction



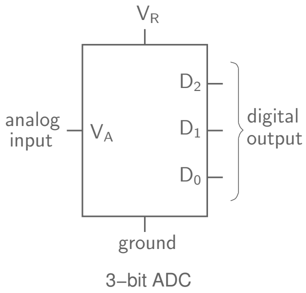
3–bit ADC

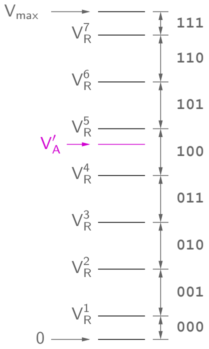


∗ The basic idea behind an ADC is simple:

- Generate reference voltages V_R^1, V_R^2, etc.
- Compare the input V_A with each of V_R^i to figure out which bin it belongs to.
- If V_A belongs to bin k (i.e., $V_R^k < V_A < V_R^{k+1}$), convert k to the binary format.

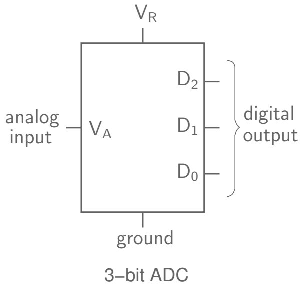
ADC: introduction

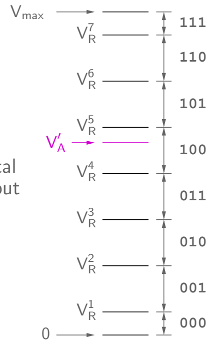


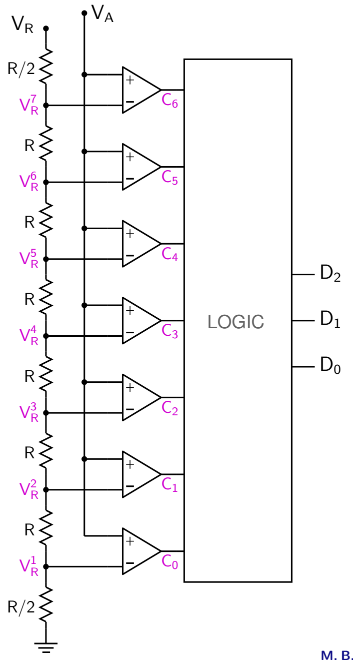


* The basic idea behind an ADC is simple:
  - Generate reference voltages V_R^1, V_R^2, etc.
  - Compare the input V_A with each of V_R^i to figure out which bin it belongs to.
  - If V_A belongs to bin k (i.e., $V_R^k < V_A < V_R^{k+1}$), convert k to the binary format.
* A "parallel" ADC does exactly that $\rightarrow$ next slide.

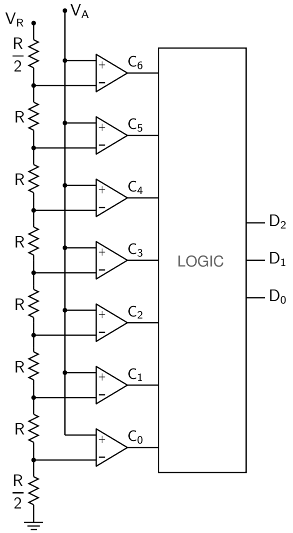
3-bit parallel (flash) ADC



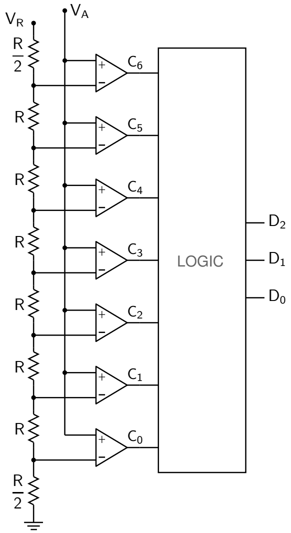




3-bit parallel (flash) ADC

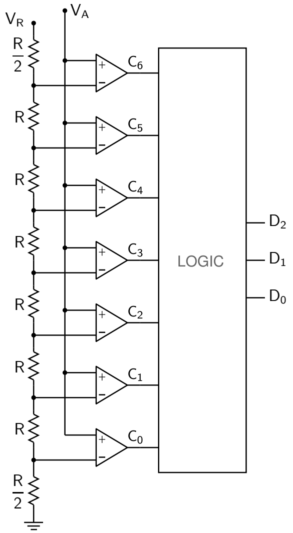


3-bit parallel (flash) ADC



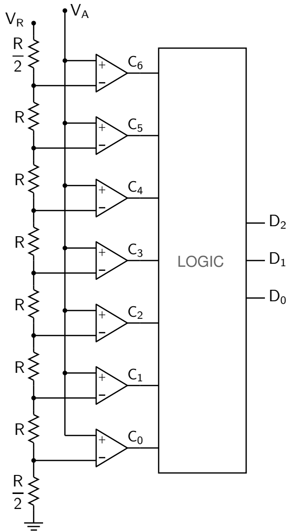
* Practical difficulty: As the input changes, the comparator outputs (C_0, C_1, etc.) may not settle to their new values at the same time. $\rightarrow$ ADC output will depend on when we sample it.

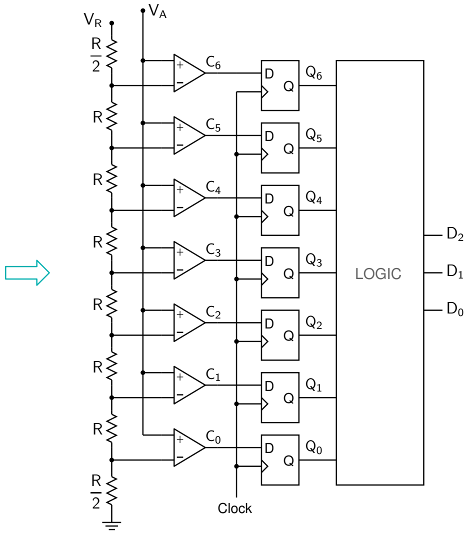
3-bit parallel (flash) ADC



* Practical difficulty: As the input changes, the comparator outputs (C_0, C_1, etc.) may not settle to their new values at the same time. $\rightarrow$ ADC output will depend on when we sample it.
* Add D flip-flops. Allow sufficient time (between the change in V_A and the active clock edge) so that the comparator outputs have already settled to their new values before they get latched in.

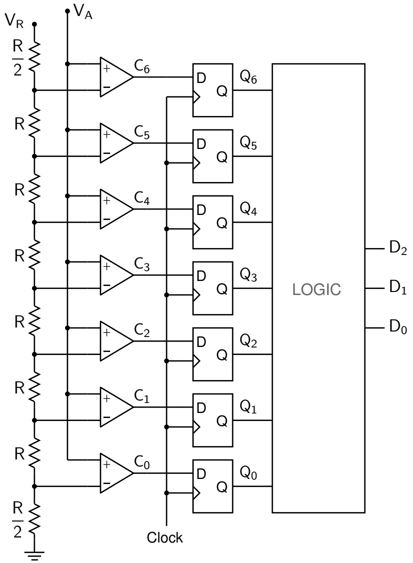
3-bit parallel (flash) ADC





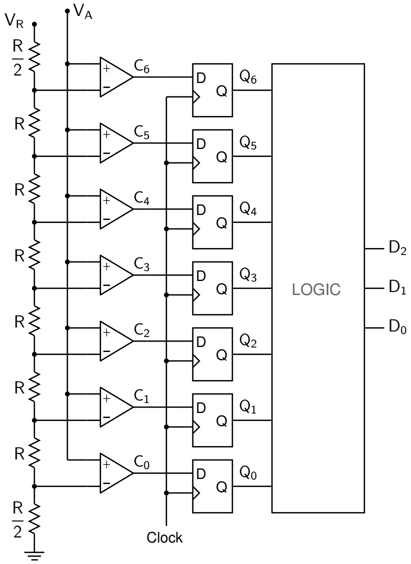
* Practical difficulty: As the input changes, the comparator outputs (C_0, C_1, etc.) may not settle to their new values at the same time. $\rightarrow$ ADC output will depend on when we sample it.
* Add D flip-flops. Allow sufficient time (between the change in V_A and the active clock edge) so that the comparator outputs have already settled to their new values before they get latched in.

Parallel (flash) ADC



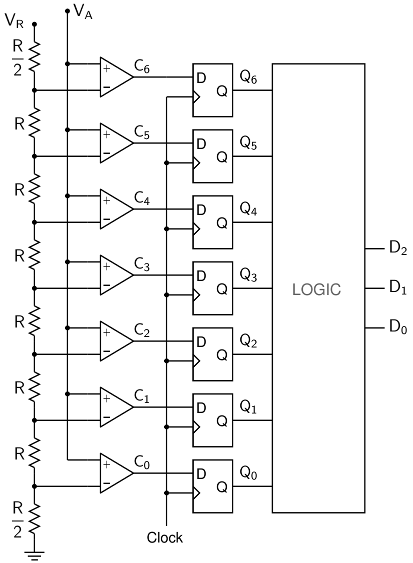
* In the parallel (flash) ADC, the conversion gets done "in parallel," since all comparators operate on the same input voltage.

Parallel (flash) ADC



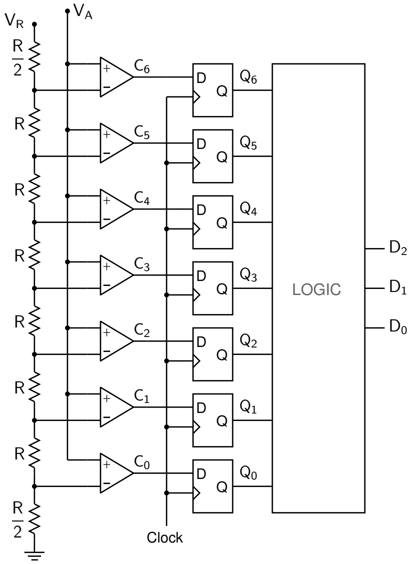
* In the parallel (flash) ADC, the conversion gets done "in parallel," since all comparators operate on the same input voltage.
* Conversion time is governed only by the comparator response time $\rightarrow$ fast conversion (hence the name "flash" converter).

Parallel (flash) ADC



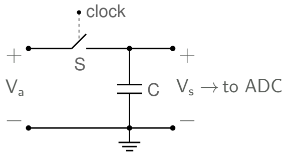
- ∗ In the parallel (flash) ADC, the conversion gets done "in parallel," since all comparators operate on the same input voltage.
- ∗ Conversion time is governed only by the comparator response time $\rightarrow$ fast conversion (hence the name "flash" converter).
- ∗ Flash ADCs to handle 500 million analog samples per second are commercially available.

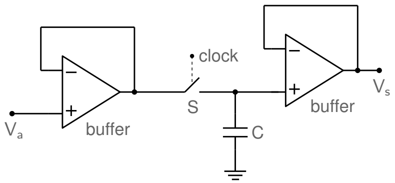
Parallel (flash) ADC

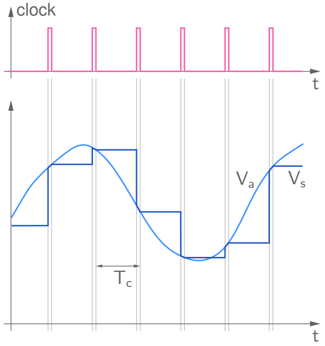


- ∗ In the parallel (flash) ADC, the conversion gets done "in parallel," since all comparators operate on the same input voltage.
- ∗ Conversion time is governed only by the comparator response time → fast conversion (hence the name "flash" converter).
- ∗ Flash ADCs to handle 500 million analog samples per second are commercially available.
- ∗ 2^N comparators are required for N-bit ADC → generally limited to 8 bits.

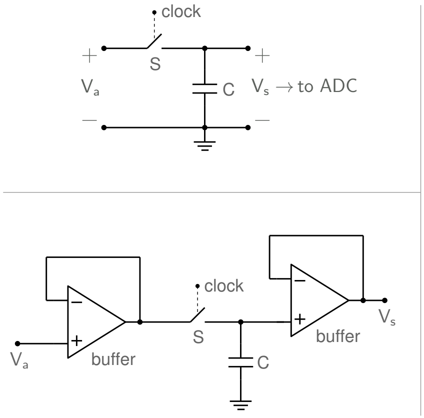
ADC: sampling of input signal

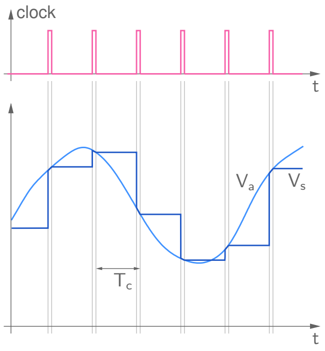






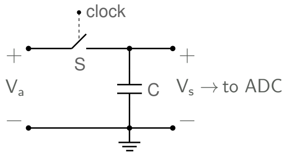
ADC: sampling of input signal

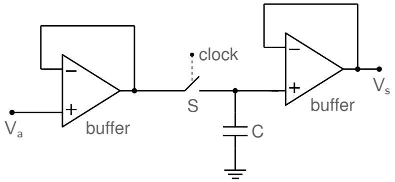


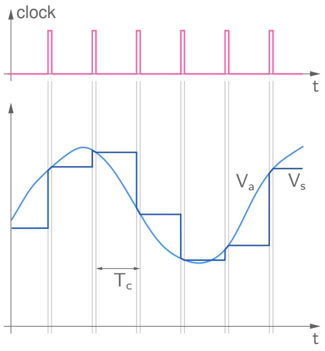


∗ An ADC typically operates on a "sampled" input signal ($V_s(t)$ in the figure) which is derived from the continuously varying input signal ($V_a(t)$ in the figure) with a "sample-and-hold" (S/H) circuit.

ADC: sampling of input signal



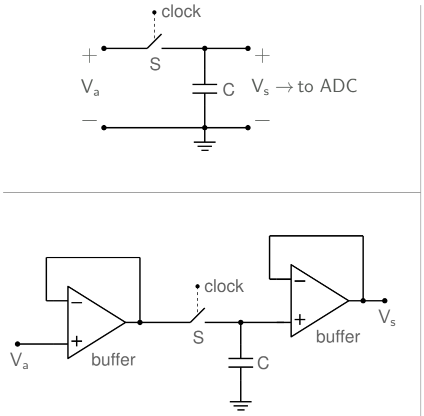


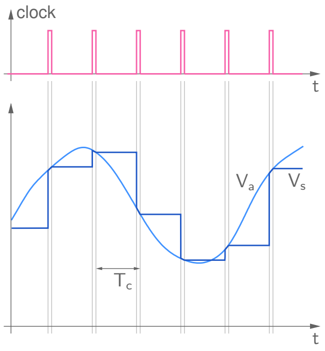


* An ADC typically operates on a "sampled" input signal ($V_s(t)$ in the figure) which is derived from the continuously varying input signal ($V_a(t)$ in the figure) with a "sample-and-hold" (S/H) circuit.
* The S/H circuit samples the input signal $V_a(t)$ at uniform intervals of duration T_c, the clock period.

M. B. Patil, IIT Bombay

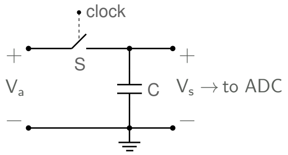
ADC: sampling of input signal

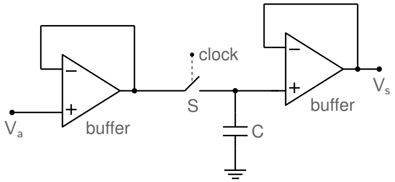


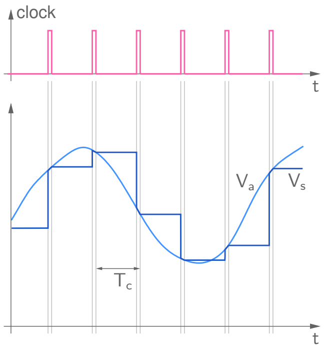


* An ADC typically operates on a "sampled" input signal ($V_s(t)$ in the figure) which is derived from the continuously varying input signal ($V_a(t)$ in the figure) with a "sample-and-hold" (S/H) circuit.
* The S/H circuit samples the input signal $V_a(t)$ at uniform intervals of duration T_c, the clock period.
* When the clock goes high, switch S (e.g., a FET or a CMOS pass gate) is closed, and the capacitor C gets charged to the signal voltage at that time. When the clock goes low, switch S is turned off, and C holds the voltage constant, as desired.

ADC: sampling of input signal

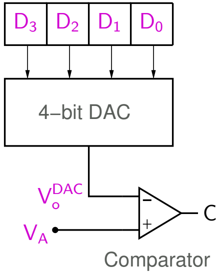




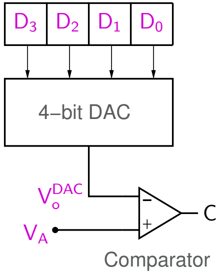


* An ADC typically operates on a "sampled" input signal ($V_s(t)$ in the figure) which is derived from the continuously varying input signal ($V_a(t)$ in the figure) with a "sample-and-hold" (S/H) circuit.
* The S/H circuit samples the input signal $V_a(t)$ at uniform intervals of duration T_c, the clock period.
* When the clock goes high, switch S (e.g., a FET or a CMOS pass gate) is closed, and the capacitor C gets charged to the signal voltage at that time. When the clock goes low, switch S is turned off, and C holds the voltage constant, as desired.
* Op-amp buffers can be used to minimise loading effects.

Successive Approximation ADC

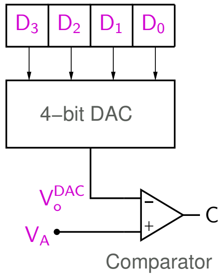


Successive Approximation ADC



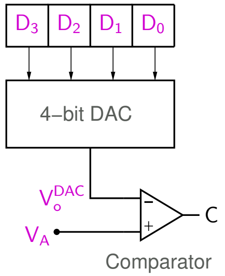
∗ Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.

Successive Approximation ADC



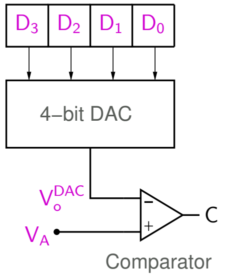
* Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.
  - Start with $D_3 D_2 D_1 D_0 = 0000$, $\mathtt{I} = 3$.

Successive Approximation ADC



* Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.
  - Start with $D_3D_2D_1D_0 = 0000$, $\mathtt{I} = 3$.
  - Set $D[\mathtt{I}] = 1$ (keep other bits unchanged).

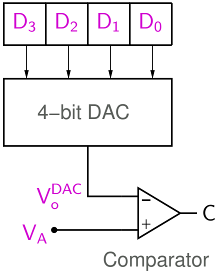
Successive Approximation ADC



* Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.

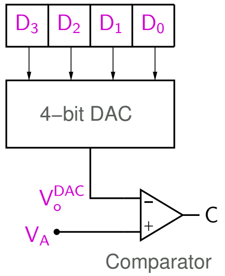
  - Start with $D_3 D_2 D_1 D_0 = 0000$, $\mathtt{I} = 3$.
  - Set $D[\mathtt{I}] = 1$ (keep other bits unchanged).
  - If $V_o^{DAC} > V_A$ (i.e., $C = 0$), set $D[\mathtt{I}] = 0$; else, keep $D[\mathtt{I}] = 1$.

Successive Approximation ADC



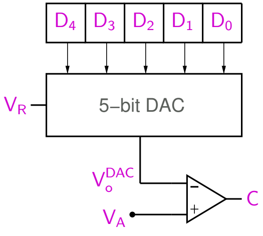
* Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.
  - Start with $D_3 D_2 D_1 D_0 = 0000$, $\mathtt{I} = 3$.
  - Set $D[\mathtt{I}] = 1$ (keep other bits unchanged).
  - If $V_o^{DAC} > V_A$ (i.e., $C = 0$), set $D[\mathtt{I}] = 0$; else, keep $D[\mathtt{I}] = 1$.
  - $\mathtt{I} \leftarrow \mathtt{I} - 1$; go to step 1.

Successive Approximation ADC

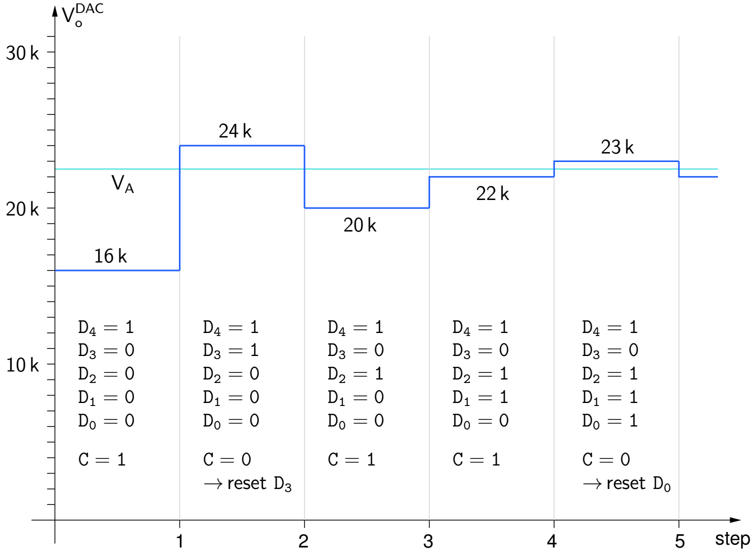


* Suppose we have a 4-bit DAC. We can use it to perform A-to-D conversion by successively setting the four bits as follows.
  - Start with $D_3D_2D_1D_0 = 0000$, $\mathtt{I} = 3$.
  - Set $D[\mathtt{I}] = 1$ (keep other bits unchanged).
  - If $V_o^{DAC} > V_A$ (i.e., $C = 0$), set $D[\mathtt{I}] = 0$; else, keep $D[\mathtt{I}] = 1$.
  - $\mathtt{I} \leftarrow \mathtt{I} - 1$; go to step 1.
* At the end of four steps, the digital output is given by $D_3D_2D_1D_0$.
  Example $\rightarrow$ next slide.

Successive Approximation ADC

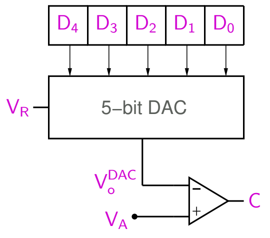


(Note: $k \propto V_R$)

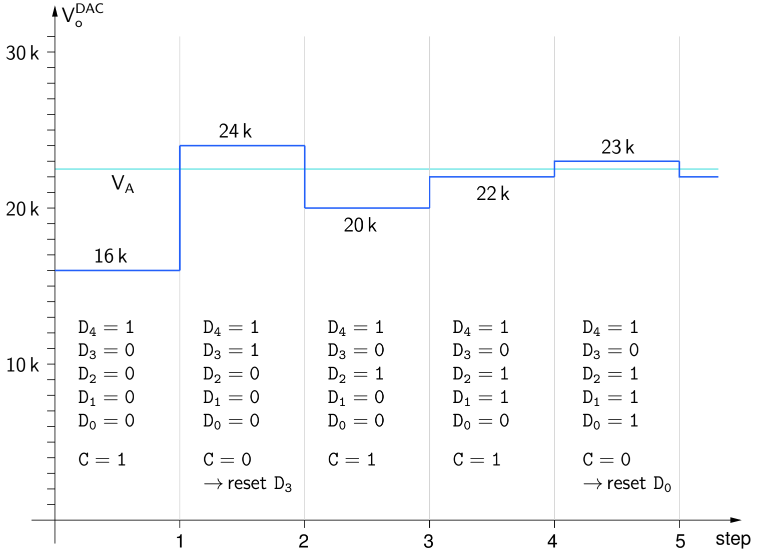


Step 1	Step 2	Step 3	Step 4	Step 5
$D_4 = 1$	$D_4 = 1$	$D_4 = 1$	$D_4 = 1$	$D_4 = 1$
$D_3 = 0$	$D_3 = 1$	$D_3 = 0$	$D_3 = 0$	$D_3 = 0$
$D_2 = 0$	$D_2 = 0$	$D_2 = 1$	$D_2 = 1$	$D_2 = 1$
$D_1 = 0$	$D_1 = 0$	$D_1 = 0$	$D_1 = 1$	$D_1 = 1$
$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 1$
$C = 1$	$C = 0$	$C = 1$	$C = 1$	$C = 0$
	$\rightarrow$ reset D_3			$\rightarrow$ reset D_0

Successive Approximation ADC



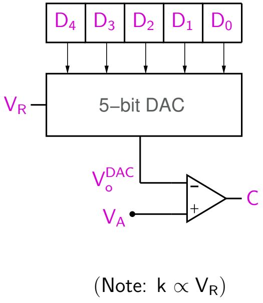
(Note: $k \propto V_R$)

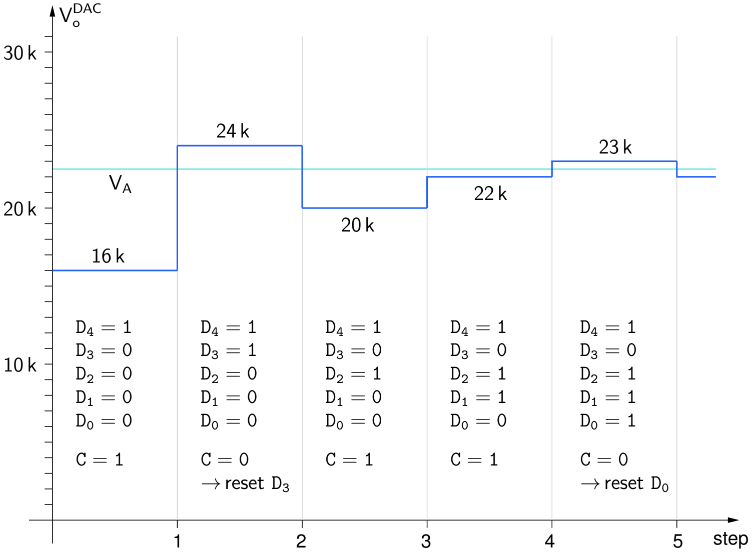


Step 1	Step 2	Step 3	Step 4	Step 5
$D_4 = 1$	$D_4 = 1$	$D_4 = 1$	$D_4 = 1$	$D_4 = 1$
$D_3 = 0$	$D_3 = 1$	$D_3 = 0$	$D_3 = 0$	$D_3 = 0$
$D_2 = 0$	$D_2 = 0$	$D_2 = 1$	$D_2 = 1$	$D_2 = 1$
$D_1 = 0$	$D_1 = 0$	$D_1 = 0$	$D_1 = 1$	$D_1 = 1$
$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 1$
$C = 1$	$C = 0$	$C = 1$	$C = 1$	$C = 0$
	$\rightarrow$ reset D_3			$\rightarrow$ reset D_0

* At the end of the 5th step, we know that the input voltage corresponds to 10110.

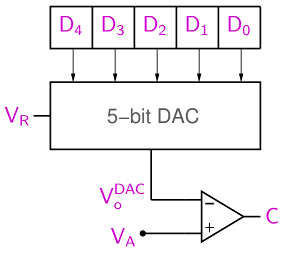
Successive Approximation ADC

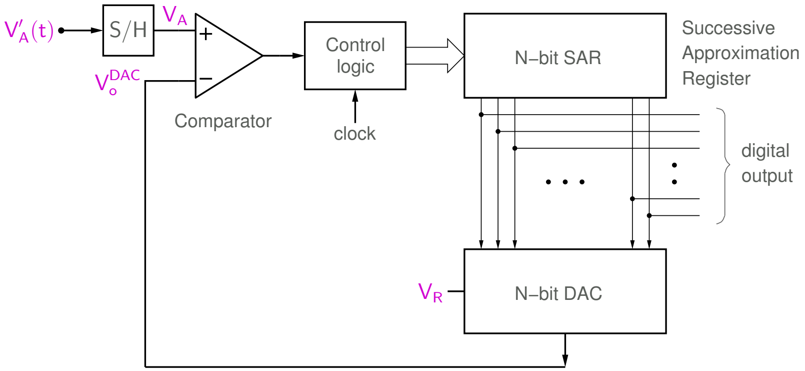




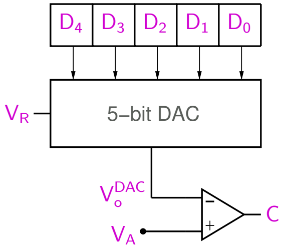
* At the end of the 5th step, we know that the input voltage corresponds to 10110.
* For the digital representation to be accurate up to $\pm\frac{1}{2}$ LSB, ΔV corresponding to $\frac{1}{2}$ LSB is added to V_A (see [Taub]).

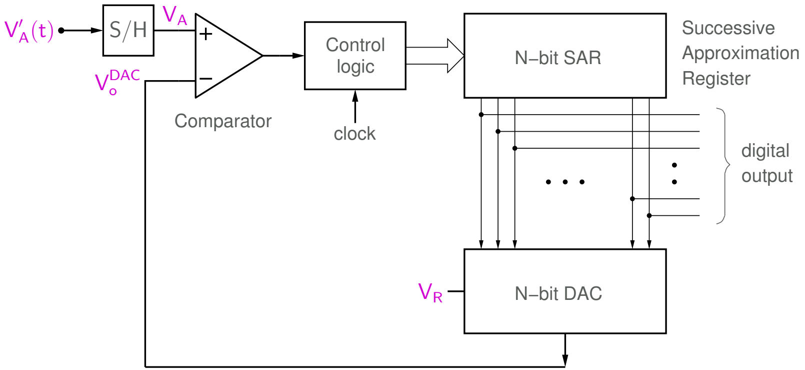
Successive Approximation ADC





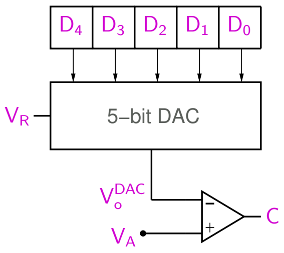
Successive Approximation ADC

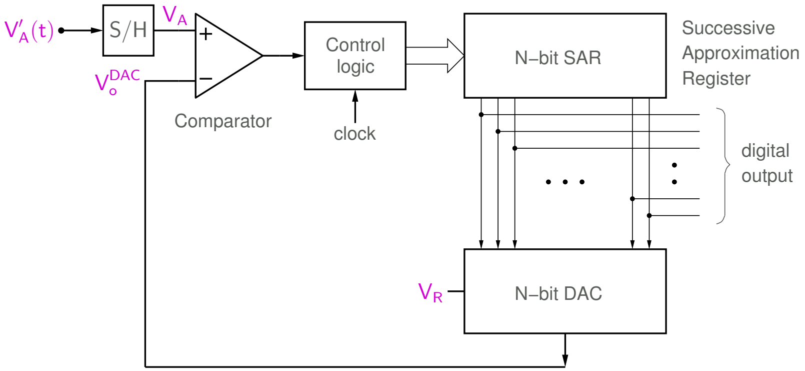




* Each step (setting SAR bits, comparison of V_A and V_o^{DAC}) is performed in one clock cycle $\rightarrow$ conversion time is N cycles, irrespective of the input voltage value V_A.

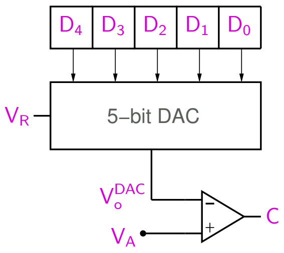
Successive Approximation ADC

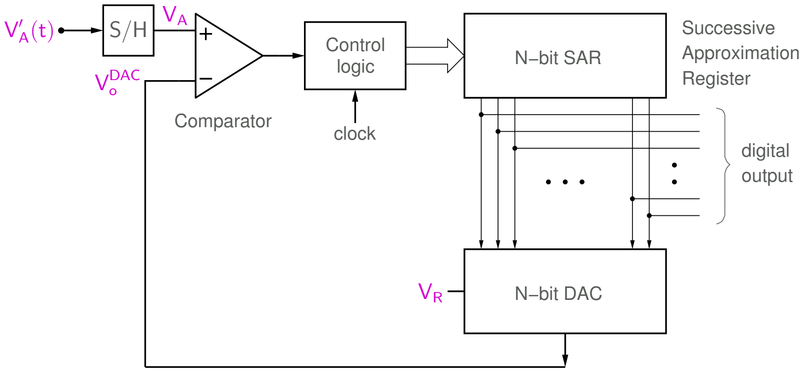




* Each step (setting SAR bits, comparison of V_A and V_o^{DAC}) is performed in one clock cycle → conversion time is N cycles, irrespective of the input voltage value V_A.
* S. A. ADCs with built-in or external S/H (sample-and-hold) are available for 8- to 16-bit resolution and conversion times of a few μsec to tens of μsec.

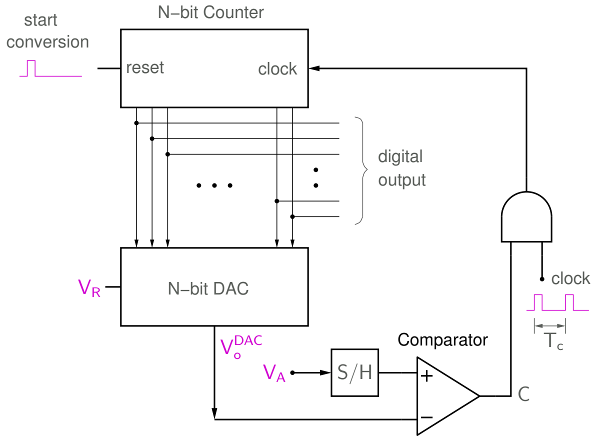
Successive Approximation ADC

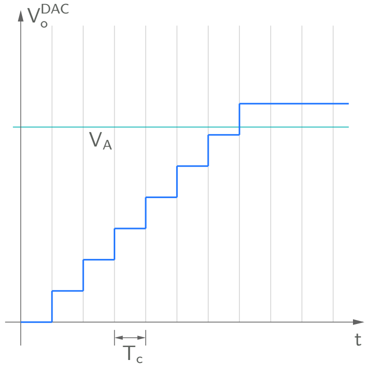




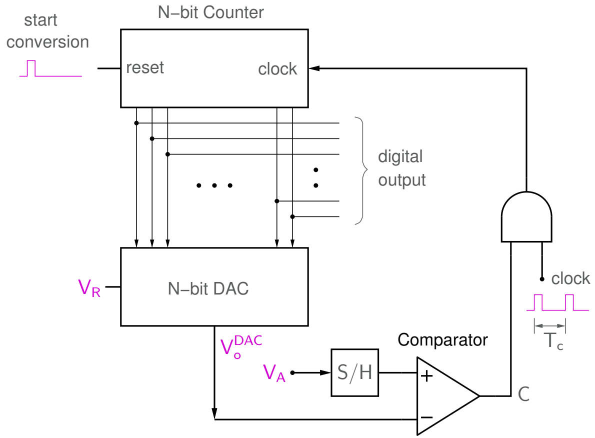
* Each step (setting SAR bits, comparison of V_A and V_o^{DAC}) is performed in one clock cycle $\rightarrow$ conversion time is N cycles, irrespective of the input voltage value V_A.
* S. A. ADCs with built-in or external S/H (sample-and-hold) are available for 8- to 16-bit resolution and conversion times of a few μsec to tens of μsec.
* Useful for medium-speed applications such as speech transmission with PCM.

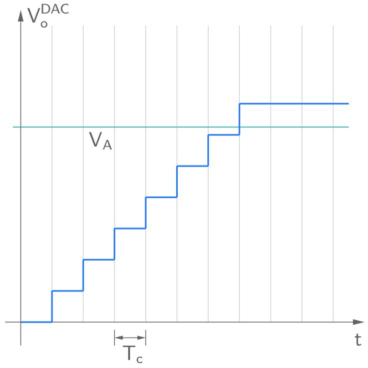
Counting ADC (digital-ramp ADC)





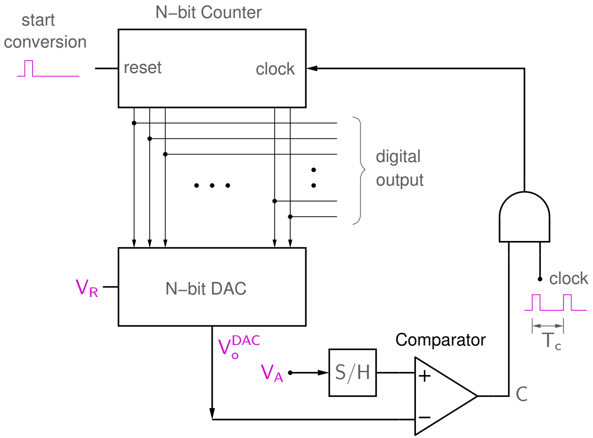
Counting ADC (digital-ramp ADC)

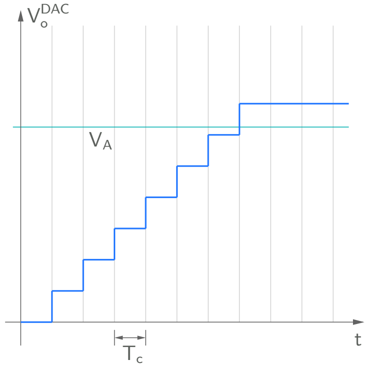




∗ The "start conversion" signal clears the counter; counting begins, and V_o^{DAC} increases with each clock cycle.

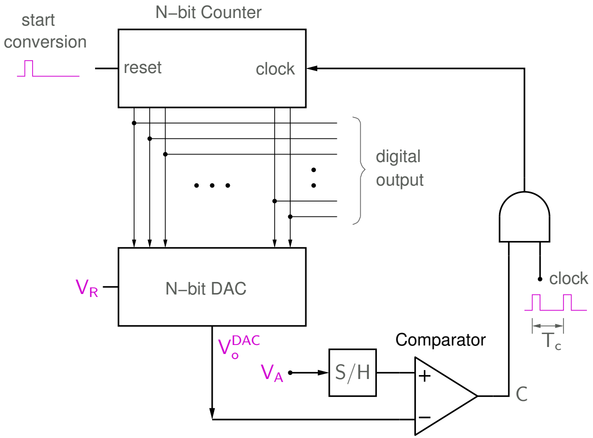
Counting ADC (digital-ramp ADC)

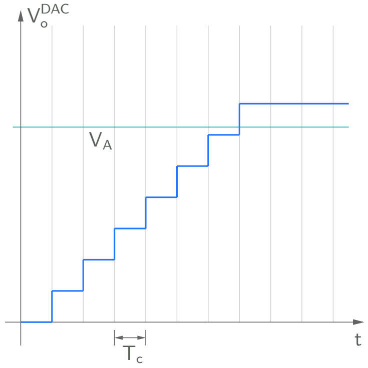




* The "start conversion" signal clears the counter; counting begins, and V_o^{DAC} increases with each clock cycle.
* When V_o^{DAC} exceeds V_A, C becomes 0, and counting stops.

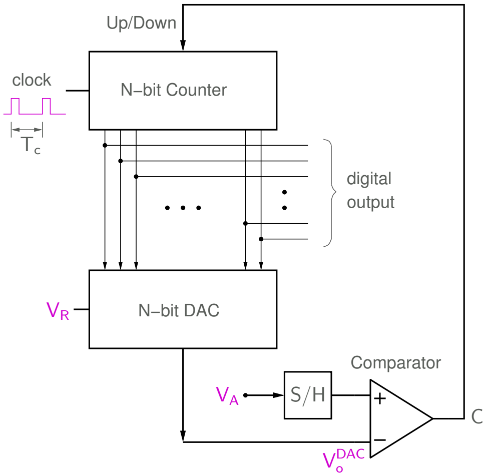
Counting ADC (digital-ramp ADC)

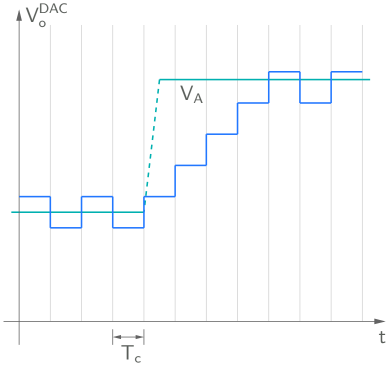




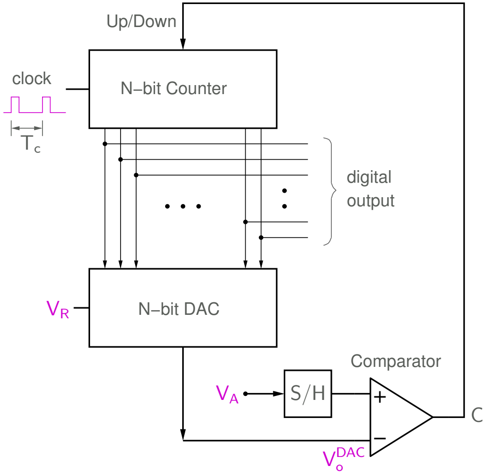
* The "start conversion" signal clears the counter; counting begins, and V_o^{DAC} increases with each clock cycle.
* When V_o^{DAC} exceeds V_A, C becomes 0, and counting stops.
* Simple scheme, but (a) conversion time depends on V_A, (b) slow (takes $(2^N - 1)$ clock cycles in the worst case) $\rightarrow$ tracking ADC

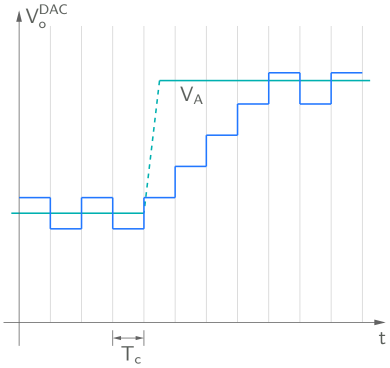
Tracking ADC





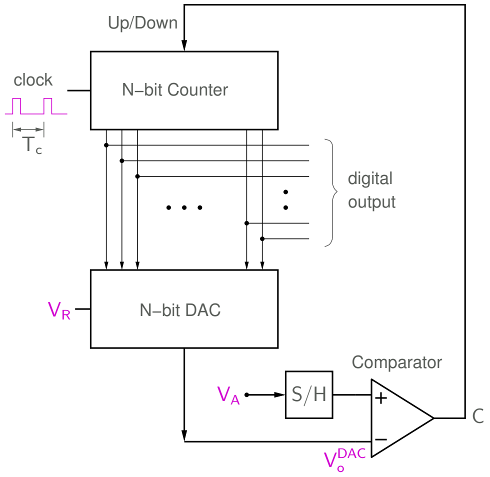
Tracking ADC

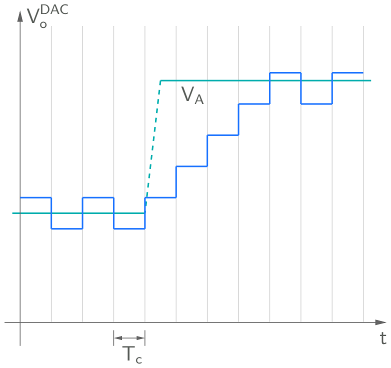




* The counter counts up if $V_o^{DAC} < V_A$; else, it counts down.

Tracking ADC

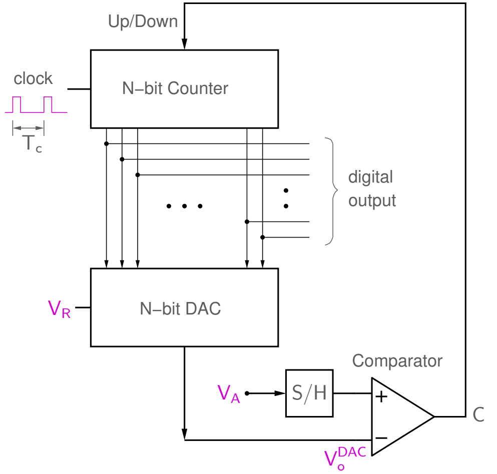


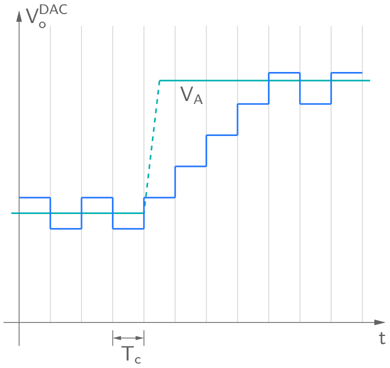


∗ The counter counts up if $V_o^{DAC} < V_A$; else, it counts down.

∗ If V_A changes, the counter does not need to start from $000 \cdots 0$, so the conversion time is less than that required by a counting ADC.

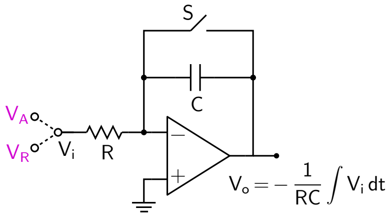
Tracking ADC

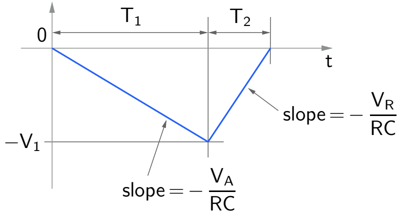




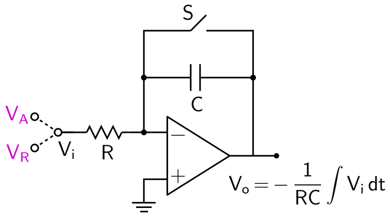
* The counter counts up if $V_o^{DAC} < V_A$; else, it counts down.
* If V_A changes, the counter does not need to start from $000 \cdots 0$, so the conversion time is less than that required by a counting ADC.
* used in low-cost, low-speed applications, e.g., measuring output from a temperature sensor or a strain gauge

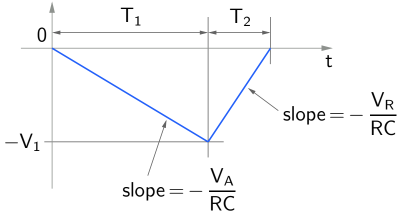
Dual-slope ADC





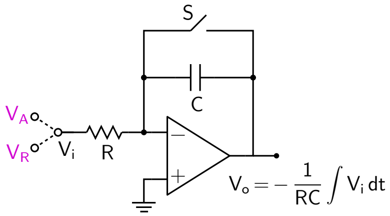
Dual-slope ADC

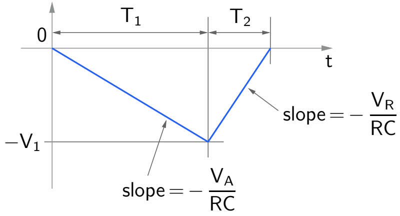




* $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.

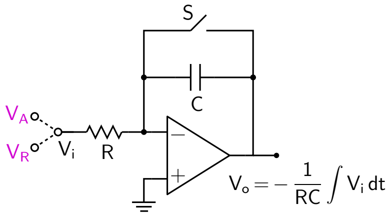
Dual-slope ADC

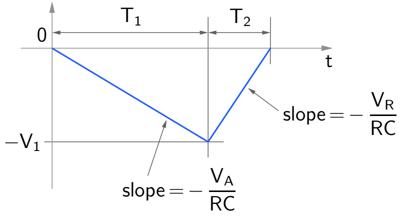




* $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.
* Integrate V_A (voltage to be converted to digital format, assumed to be positive) for a fixed interval T_1.

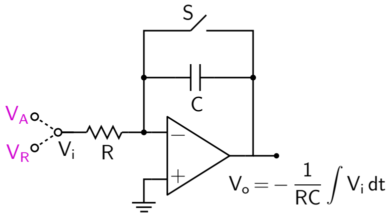
Dual-slope ADC

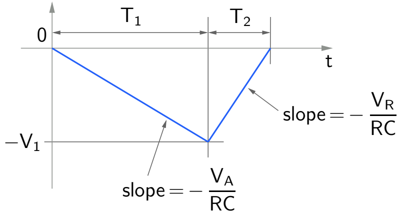




- $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.
- Integrate V_A (voltage to be converted to digital format, assumed to be positive) for a fixed interval T_1.
- At $t = T_1$, integrator output reaches $-V_1 = -\,V_A\,\dfrac{T_1}{RC}$.

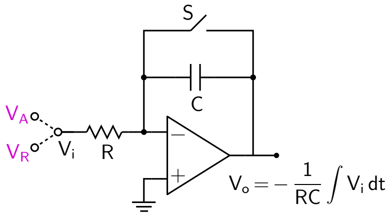
Dual-slope ADC

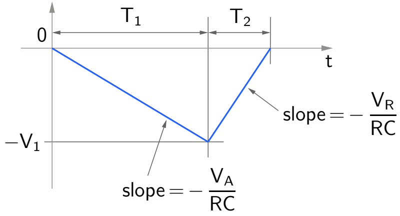




- $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.
- Integrate V_A (voltage to be converted to digital format, assumed to be positive) for a fixed interval T_1.
- At $t = T_1$, integrator output reaches $-V_1 = -\,V_A\,\frac{T_1}{RC}$.
- Now apply a reference voltage V_R (assumed to be negative, with $|V_R| > V_A$), and integrate until V_o reaches 0 V.

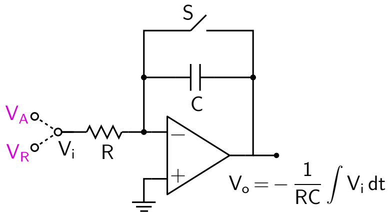
Dual-slope ADC

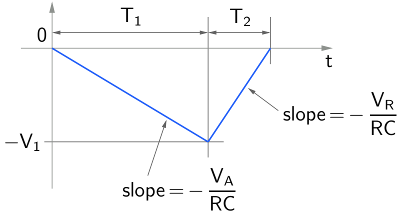




* $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.
* Integrate V_A (voltage to be converted to digital format, assumed to be positive) for a fixed interval T_1.
* At $t = T_1$, integrator output reaches $-V_1 = -\,V_A\,\dfrac{T_1}{RC}$.
* Now apply a reference voltage V_R (assumed to be negative, with $|V_R| > V_A$), and integrate until V_o reaches 0 V.
* Since $V_1 = V_A\,\dfrac{T_1}{RC} = |V_R|\,\dfrac{T_2}{RC}$, we have $T_2 = T_1\,\dfrac{V_A}{|V_R|} \rightarrow T_2$ gives a measure of V_A.

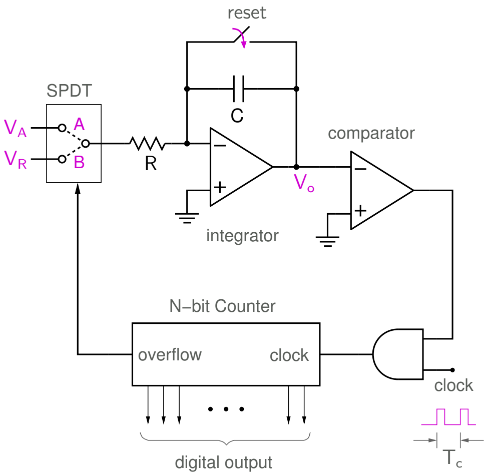
Dual-slope ADC

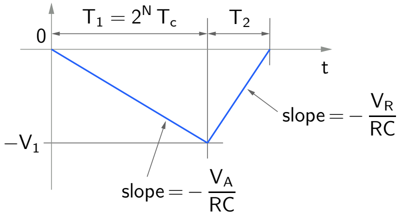




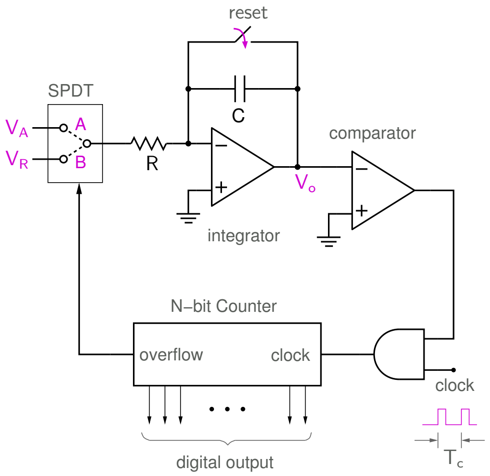
- $t = 0$: reset integrator output V_o to 0 V by closing S momentarily.
- Integrate V_A (voltage to be converted to digital format, assumed to be positive) for a fixed interval T_1.
- At $t = T_1$, integrator output reaches $-V_1 = -V_A\,\frac{T_1}{RC}$.
- Now apply a reference voltage V_R (assumed to be negative, with $|V_R| > V_A$), and integrate until V_o reaches 0 V.
- Since $V_1 = V_A\,\frac{T_1}{RC} = |V_R|\,\frac{T_2}{RC}$, we have $T_2 = T_1\,\frac{V_A}{|V_R|} \rightarrow T_2$ gives a measure of V_A.
- In the dual-slope ADC, a counter output – which is proportional to T_2 – provides the desired digital output.

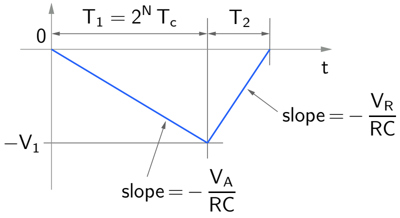
Dual-slope ADC





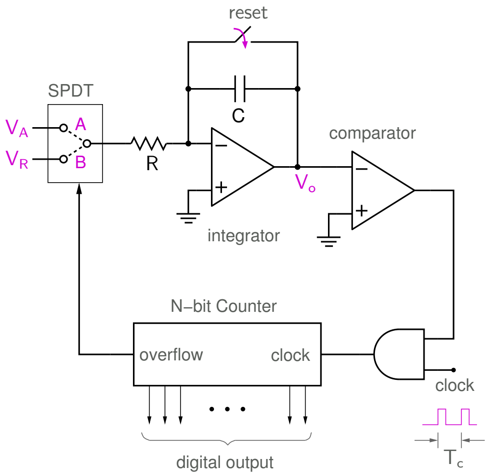
Dual-slope ADC

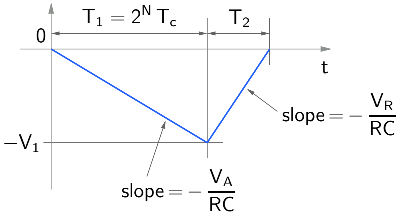




* Start: counter reset to $000 \cdots 0$, SPDT in position A.

Dual-slope ADC

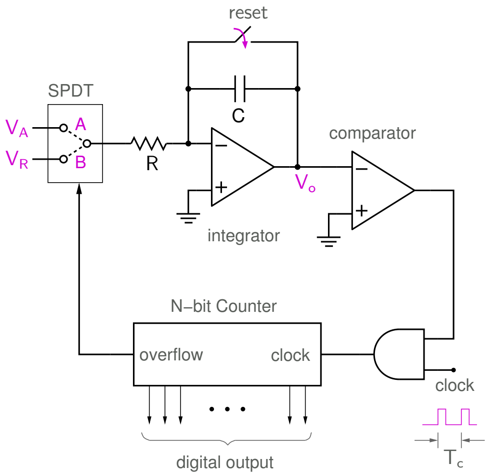


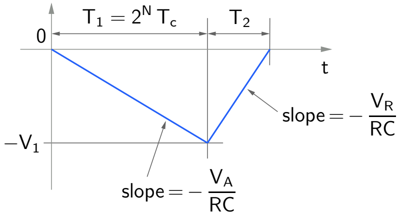


∗ Start: counter reset to 000· · · 0, SPDT in position A.

∗ Counter counts up to 2^N at which point the overflow flag becomes 1, and SPDT switches to position B $\rightarrow$ $T_1 = 2^N\, T_c$ where T_c is the clock period.

Dual-slope ADC





∗ Start: counter reset to 000· · · 0, SPDT in position A.

∗ Counter counts up to 2^N at which point the overflow flag becomes 1, and SPDT switches to position B $\rightarrow$ $T_1 = 2^N\, T_c$ where T_c is the clock period.

∗ The counter starts counting again from 000· · · 0, and stops counting when V_o crosses 0 V. The counter output gives T_2 in binary format.

References

* K. Gopalan, *Introduction to Digital Microelectronic Circuits*, Tata McGraw-Hill, New Delhi, 1998.
* H. Taub and D. Schilling, *Digital Integrated Electronics*, McGraw-Hill, 1977.