UNIT 5

Registers & Counters

Registers and Counter

- The filp-flops are essential component in clocked sequential circuits.
- Circuits that include filp-flops are usually classified by the function they perform. Two such circuits are registers and counters.
- An n-bit register consists of a group of n flip-flops capable of storing n bits of binary information.

Registers

- In its broadest definition, a register consists a group of flip-flops and gates that effect their transition.
  - The flip-flops hold the binary information.
  - The gates determine how the information is transferred into the register.
- Counters are a special type of register.
- A counter goes through a predetermined sequence of states.

Registers

- Fig 6-1 shows a register constructed with four D-type filpflops.
- “Clock” triggers all flip-folps on the positive edge of each pulse.
- “Clear” is useful for clearing the register to all 0’s prior to its clocked operation.

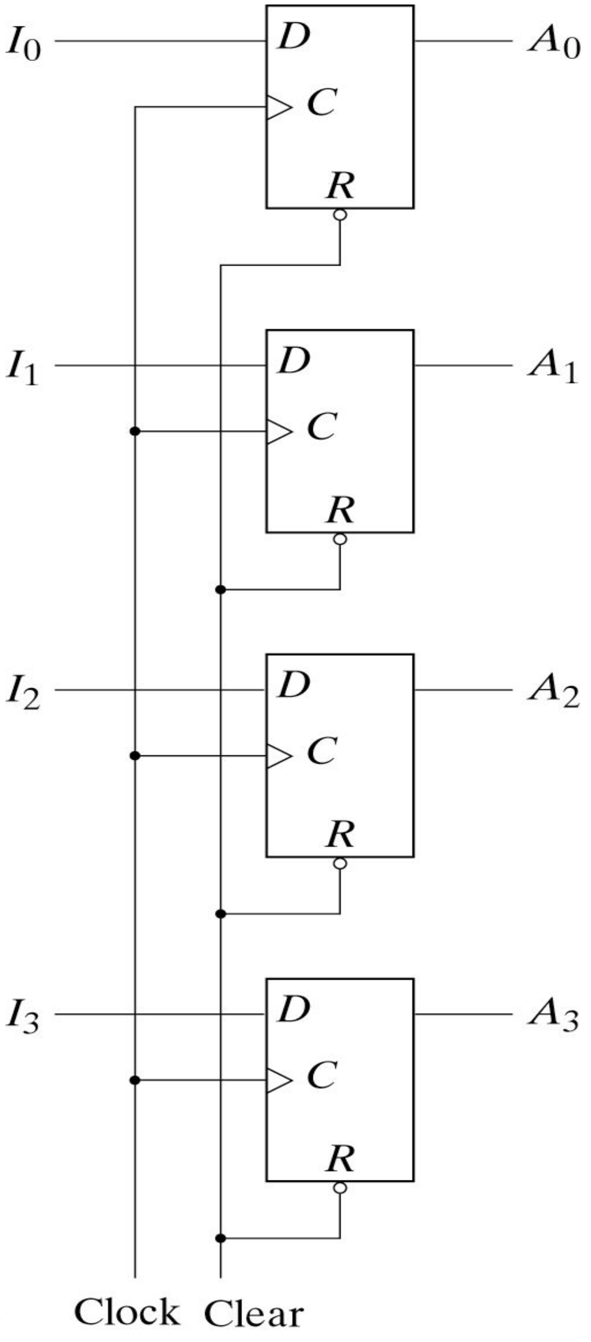


Fig. 6-1 4-Bit Register

Register with Parallel Load

- A clock edge applied to the C inputs of the register of Fig. 6-1 will load all four inputs in parallel.
- For synchronism, it is advisable to control the operation of the register with the D inputs rather than controlling the clock in the C inputs of the flip-flops.
- A 4-bit register with a load control input that is directed through gates and into the D inputs of the flip-flops si shown in Fig. 6-2.

Register with Parallel Load

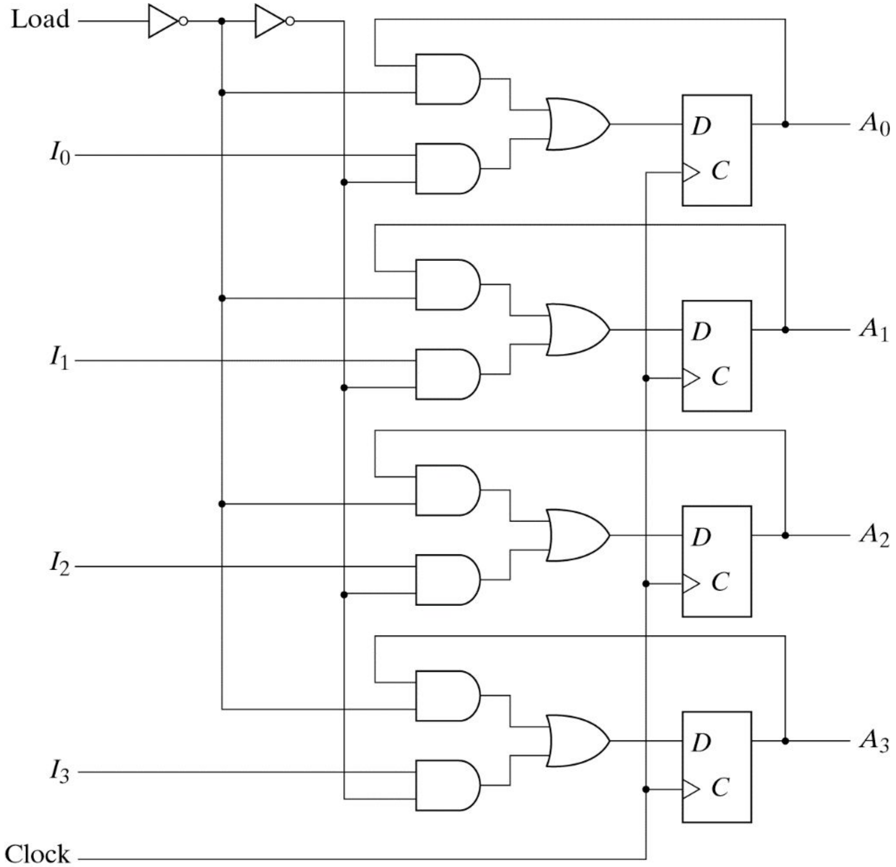


Fig. 6-2 4-Bit Register with Parallel Load

Register with Parallel Load

- When the load input is 1 , the data in the four inputs are transferred into the register with next positive edge of the clock.
- When the load input is 0 ,the outputs of the flip-flops are connected to their respective inputs.
- The feedback connection from output to input is necessary because the D flip-flops does not have a “no change” condition.

Shift Registers

- A register capable of shifting its binary information in one or both direction is called a *shift register*.
- All flip-flops receive common clock pulses, which activate the shift from one stage to the next.
- The simplest possible shift register is one that uses only flip-flops, as shown in Fig. 6-3.

Shift Registers

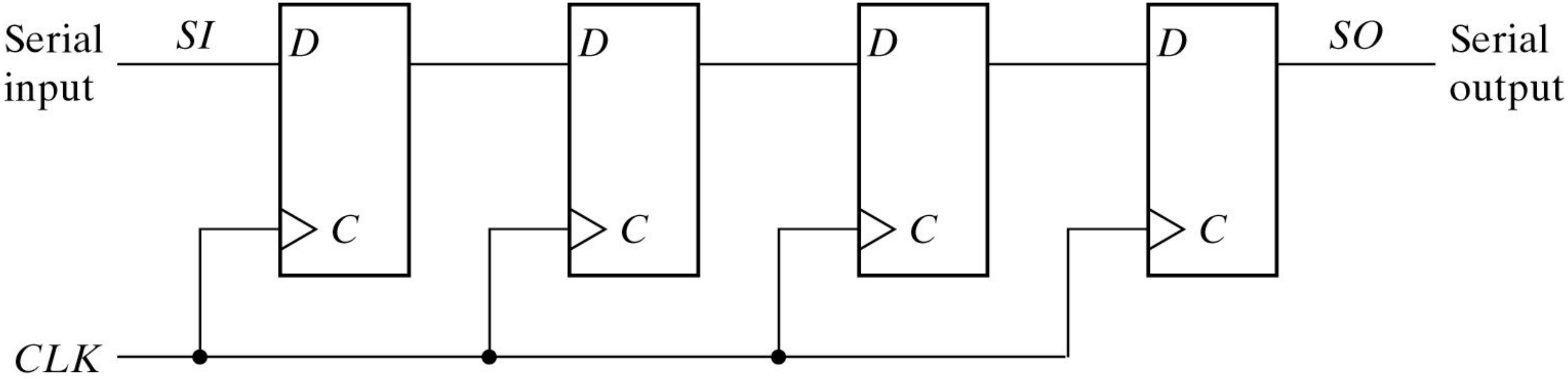


Fig. 6-3 4-Bit Shift Register

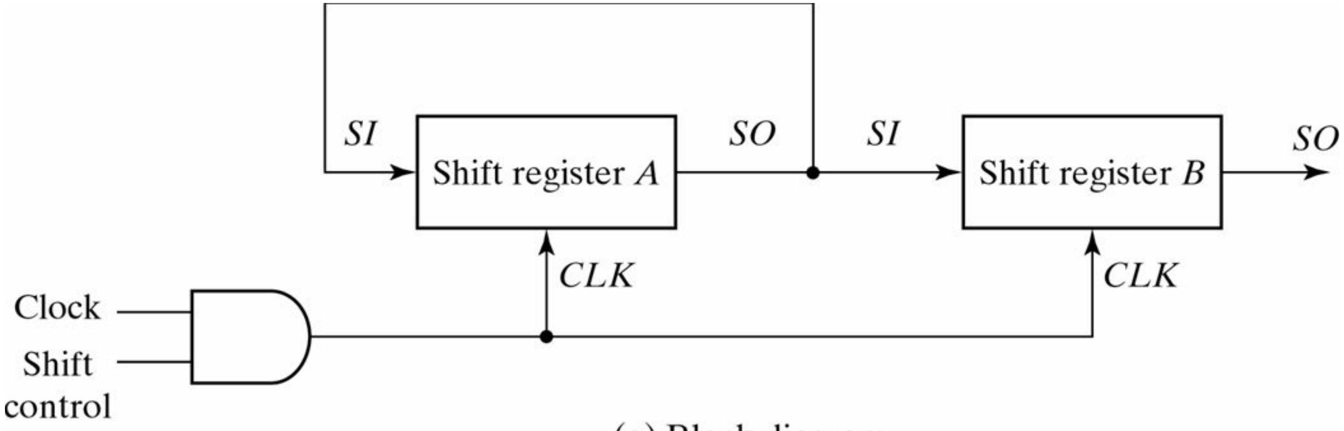
Shift Registers

- Each clock pulse shifts the contents of the register one bit position to the right.
- The *serial input* determines what goes into the leftmost flip-flop during the shift.
- The *serial output* is taken from the output of the rightmost flip-flop.

Serial Transfer

- A digital system is said tp operate in a serial mode when information is transferred and manipulated one bit at a time.
- This in contrast to parallel transfer where all the bits of the register are transferred at the same time.
- The serial transfer us done with shift registers, as shown in the block diagram of Fig. 6-4(a).

Serial Transfer

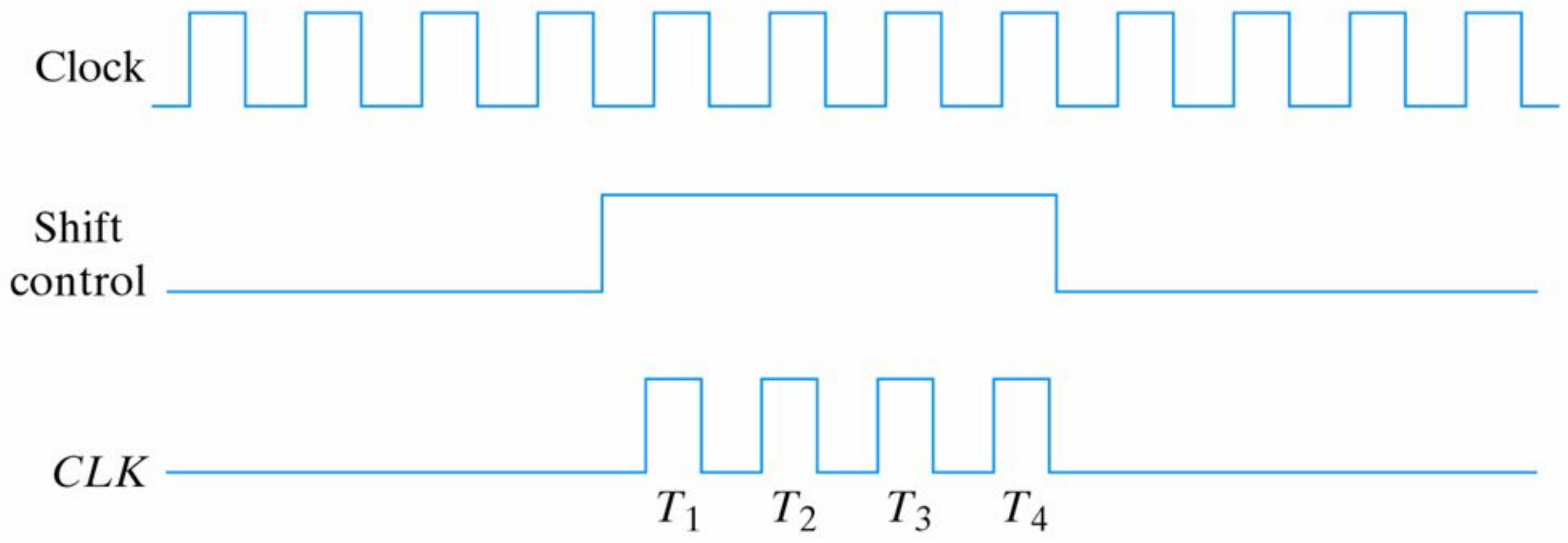


(a) Block diagram

Serial Transfer

- To prevent the loss of information stored in the source register, the information in register A is made to circulate by connecting the serial output to its serial input.
- The shift control input determines when and how many times the registers are shifted. This is done with an AND gate that allows clock pulses to pass into the CLK terminals only when the shift control is active. [Fig. 6-4(a)].

Serial Transfer



(b) Timing diagram

Fig. 6-4 Serial Transfer from Register *A* to register *B*

Serial Transfer

- The shift control signal is synchronized with the clock and changes value just after the negative edge of the clock.
- Each rising edge of the pulse causes a shift in both registers. The fourth pulse changes the shift control to 0 and the shift registers are disabled.

Serial Transfer

Table 6-1

Serial-Transfer Example

Timing Pulse	Shift Register A	Shift Register B
Initial value	1 0 1 1	0 0 1 0
After T1	1 1 0 1	1 0 0 1
After T2	1 1 1 0	1 1 0 0
After T3	0 1 1 1	0 1 1 0
After T4	1 0 1 1	1 0 1 1

Serial Transfer

- In the parallel mode, information is available from all bits can be transferred simultaneously during one clock pulse.
- In the serial mode, the registers have a single serial input and a single serial output. The information us transferred one bit at a time while the registers are shifted in the same direction.

Serial Addition

- Operations in digital computers are usually done in parallel because this is a faster mode of operation.
- Serial operations are slower, but have the advantage of requiring less equipment.
- The two binary numbers to be added serially are stored in two shift registers.
- Bits are added one pair at a time through a single full adder. [Fig. 6-5]

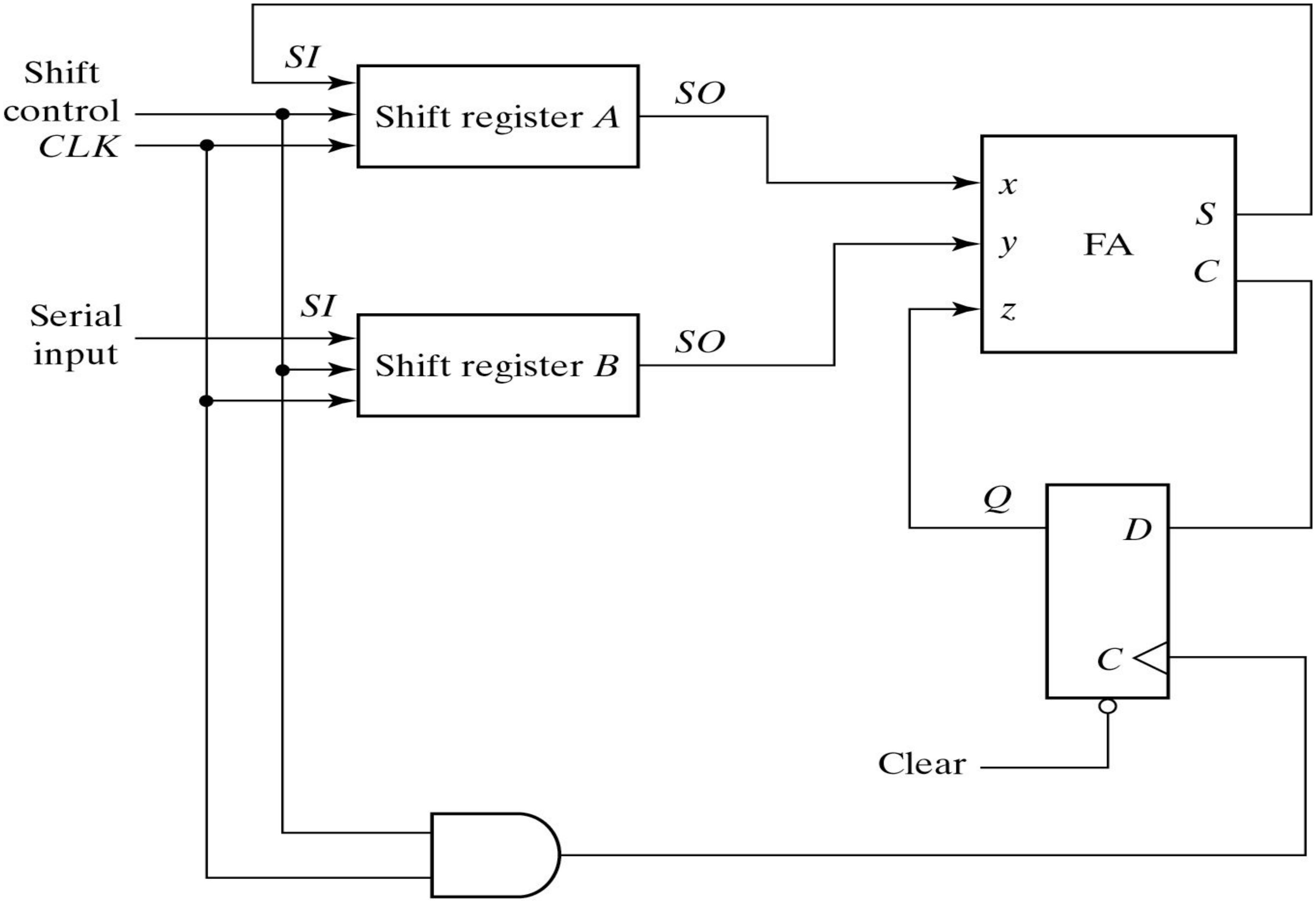


Fig. 6-5 Serial Adder

Serial Addition

- By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both the augend and sum bits.
- The carry out of the full adder is transferred to a D flip-flop.
- The output of the D flip-flop is then used as carry input for the next pair of significant bits.

Serial Addition

- To show that serial operations can be designed by means of sequential circuit procedure, we will redesign the serial adder using a state table.
- The serial outputs from registers are designated by x and y.
- The sequential circuit proper has two inputs, x and y, that provide a pair of significant bits, an output S that generates the sum bit, and flip-flop Q for storing the carry. [Table. 6-2]

Serial Addition

Table 6-2

State Table for serial Adder

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	X	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

Serial Addition

- The two flip-flop input equations and the output equation can be simplified by means of map to obtain
  - JQ=xy
  - KQ=$x'y'$=$(x+y)'$
  - S=$x \oplus y \oplus$Q

Serial Addition

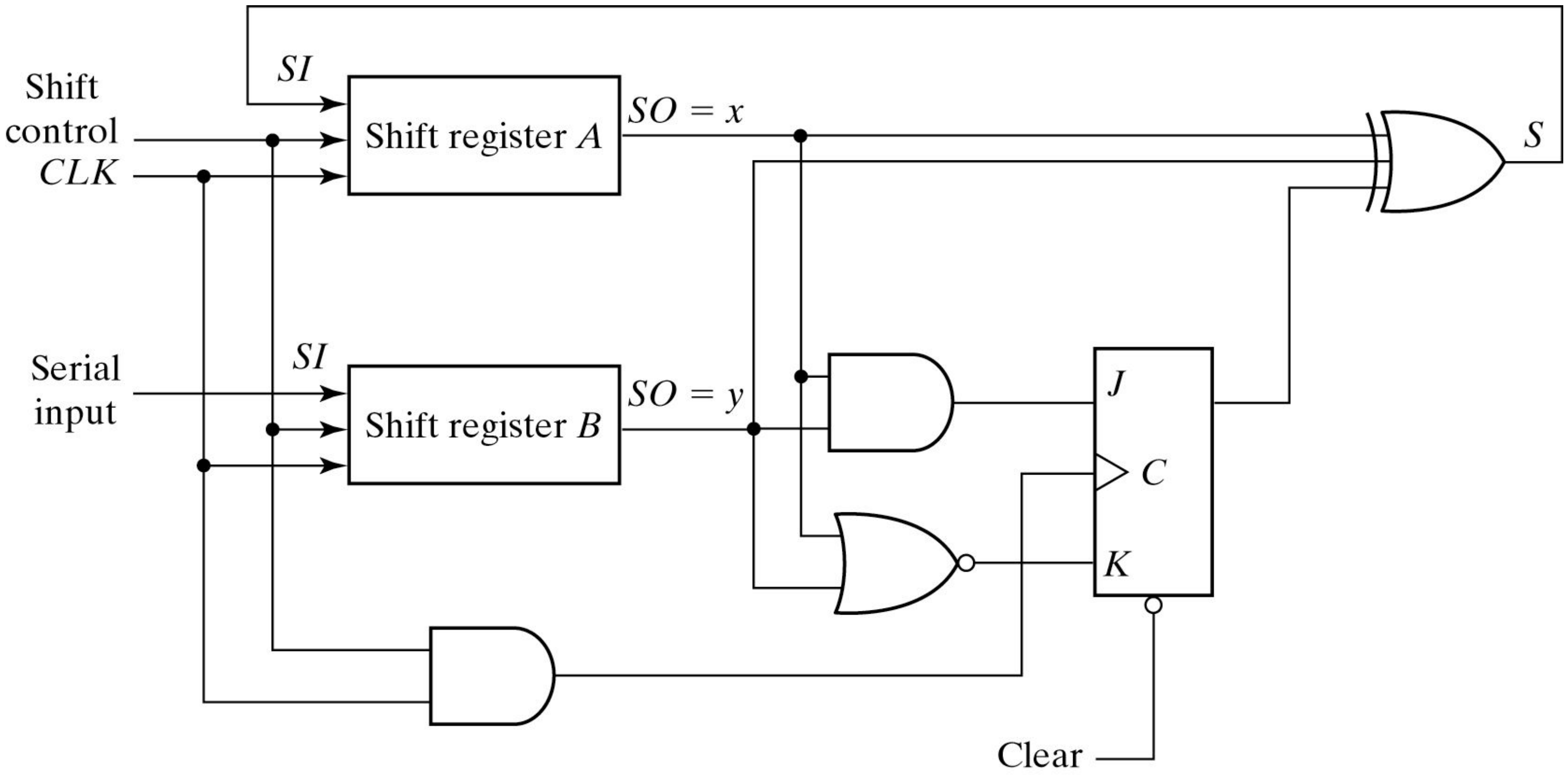


Fig. 6-6 Second form of Serial Adder

Universal Shift Register

- A *clear* control to clear the register to 0.
- A *clock* input to synchronize the operations.
- A *shift-right* control to enable the shift operation and the *serial input* and *output* lines associated with the shift right.
- A *shift-left* control to enable the shift operation and the *serial input* and *output* lines associated with the shift left.

Universal Shift Register

- A *parallel-load* control to enable a parallel transfer and the *n* input lines associated with the parallel transfer.
- *n* parallel output lines.
- A control state that leaves the information in the register unchanged in the presence of the clock.
- If the register has both shifts and parallel load capabilities, it is referred to as a *universal shift register.*

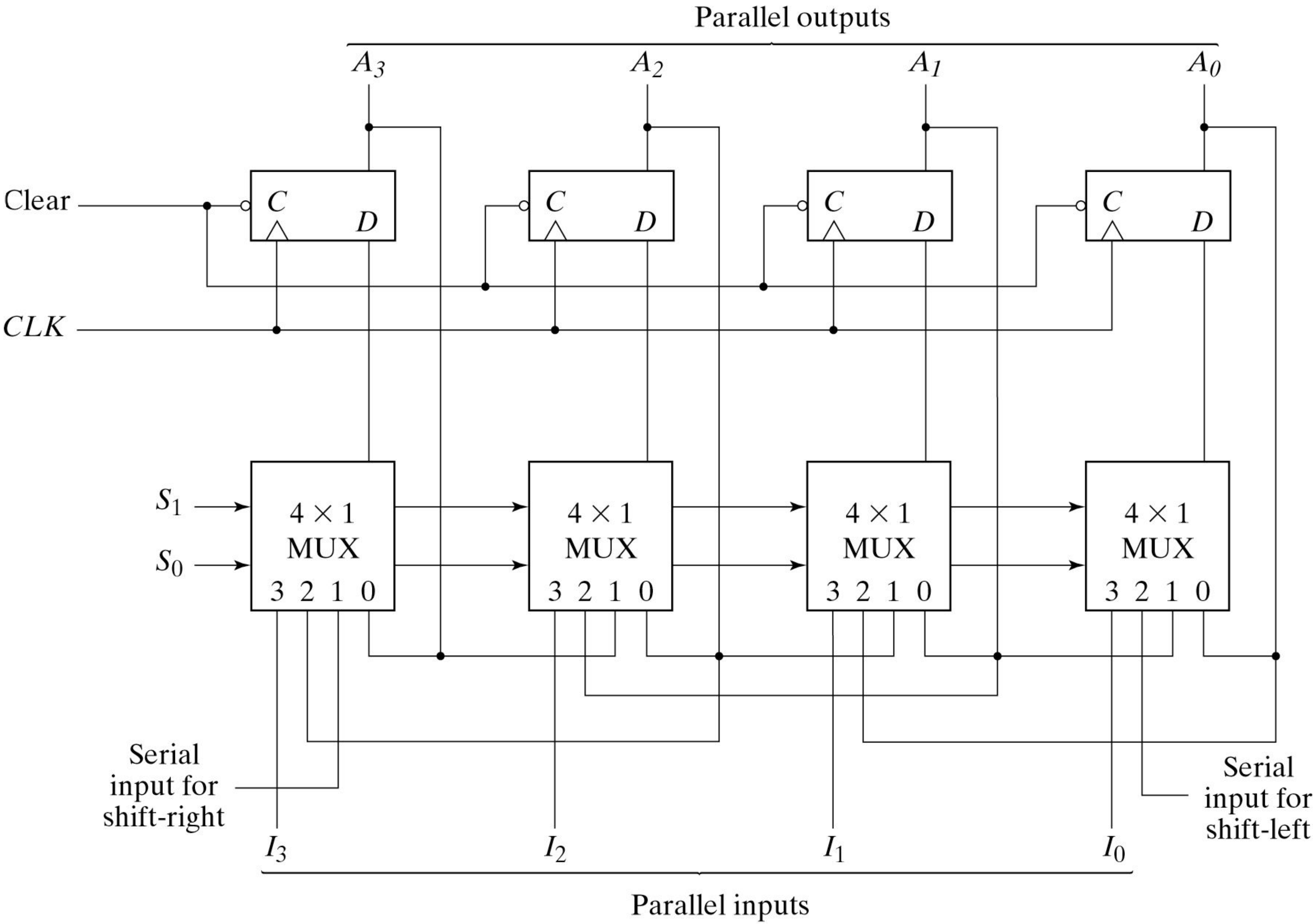


Fig. 6-7 4-Bit Universal Shift Register

Universal Shift Register

Table 6-3
Function Table for the Register of Fig. 6-7

Mode Control		***Register* Operation**
S_1	S_0	
0	0	No Change
0	1	Shift right
1	0	Shift Left
1	1	Parallel load

Universal Shift Register

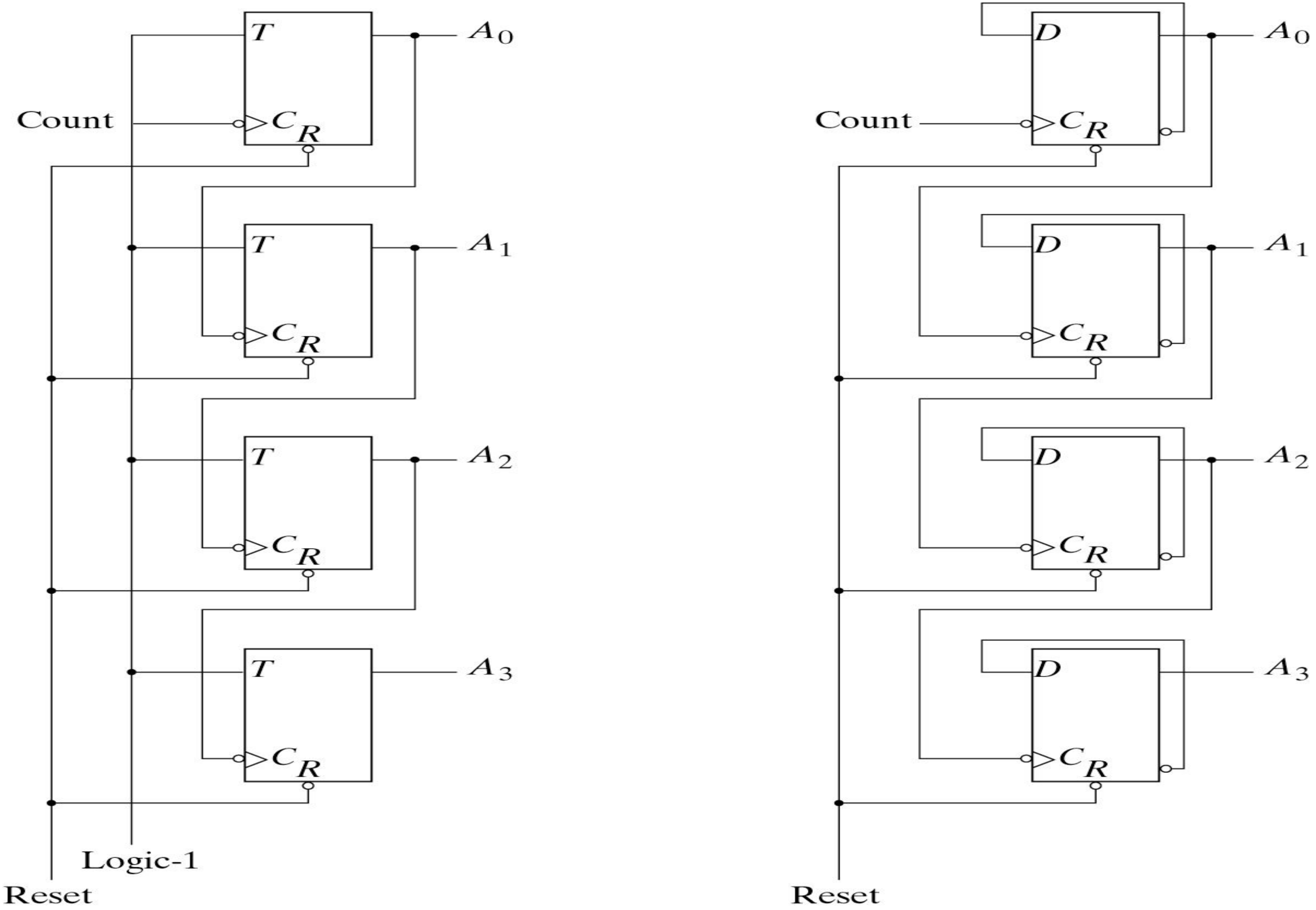
- Shift registers are often used to interface digital system situated remotely from each other.
- If the distance is far, it will be expensive to use n lines to transmit the n bits in parallel.
- Transmitter performs a parallel-to-serial conversion of data and the receiver does a serial-to-parallel conversion.

Ripple Counters

- A register that goes through a prescribed sequence of states upon the application of input pulse is called a counter.
- A counter that follows the binary number sequence is called a binary counter.
- Counters are available in two categories
    - Ripple counters
    - Synchronous counters

Binary Ripple Counter

- The output of each flip-flop is connected to the C input of the next flip-flop in sequence.
- The flip-flop holding the last significant bit receives the incoming count pulse.
- A complementing flip-flop can be obtained from:
  - JK flip-flop with the J and K inputs tied together.
  - T flip-flop
  - D flip-flop with the complement output connected to the D input. [Fig. 6-8]



(a) With T flip-flops

(b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

Binary Ripple Counter

Table 6-3

Function Table for the Register of Fig. 6-7

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

BCD Ripple Counter

- A decimal counter follows a sequence of ten states and returns to 0 after the count of 9.
- This is similar to a binary counter, except that the state after 1001 is 0000.
- The operation of the counter can be explained by a list of conditions for flip-flop transitions.

BCD Ripple Counter

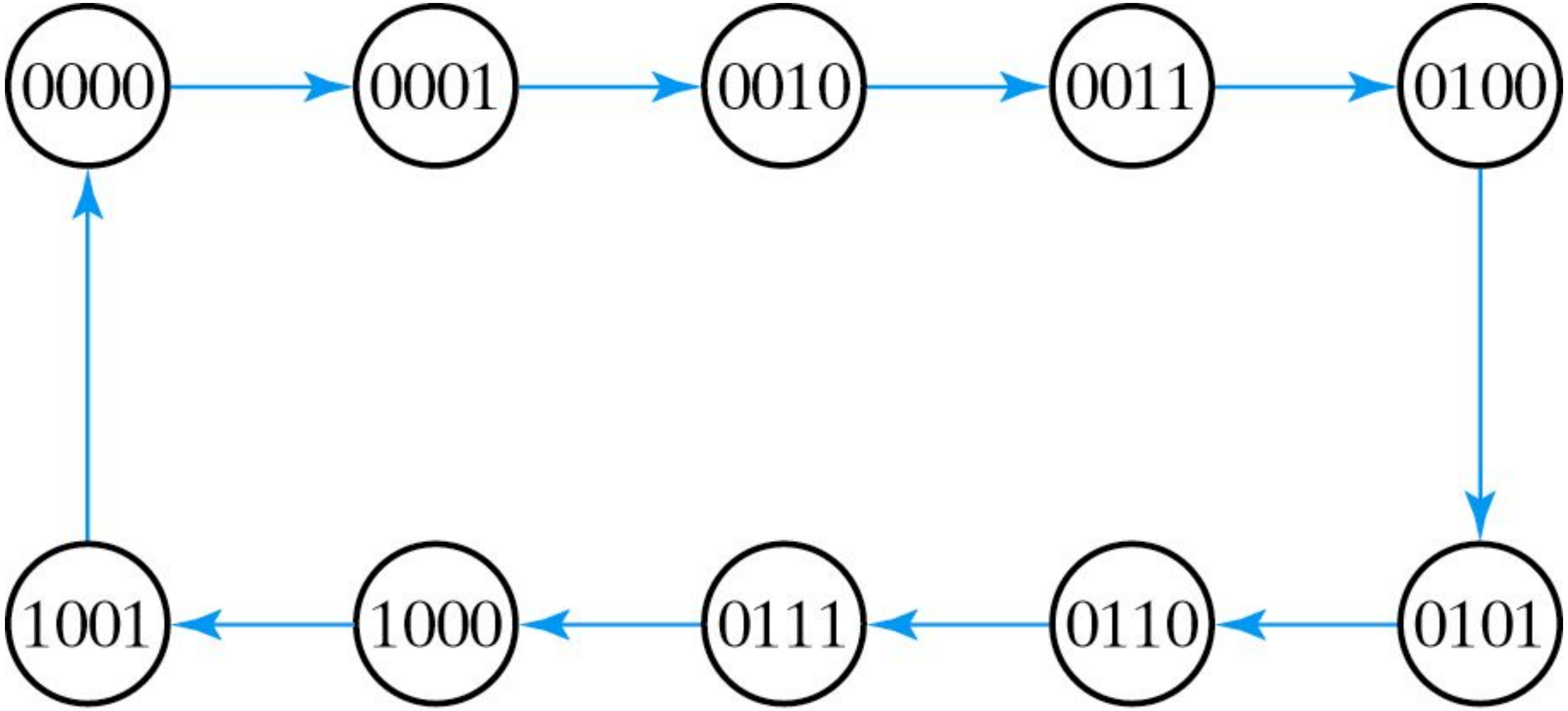


Fig. 6-9 State Diagram of a Decimal BCD-Counter

BCD Ripple Counter

- The four outputs are designated by the letter symbol Q with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code.

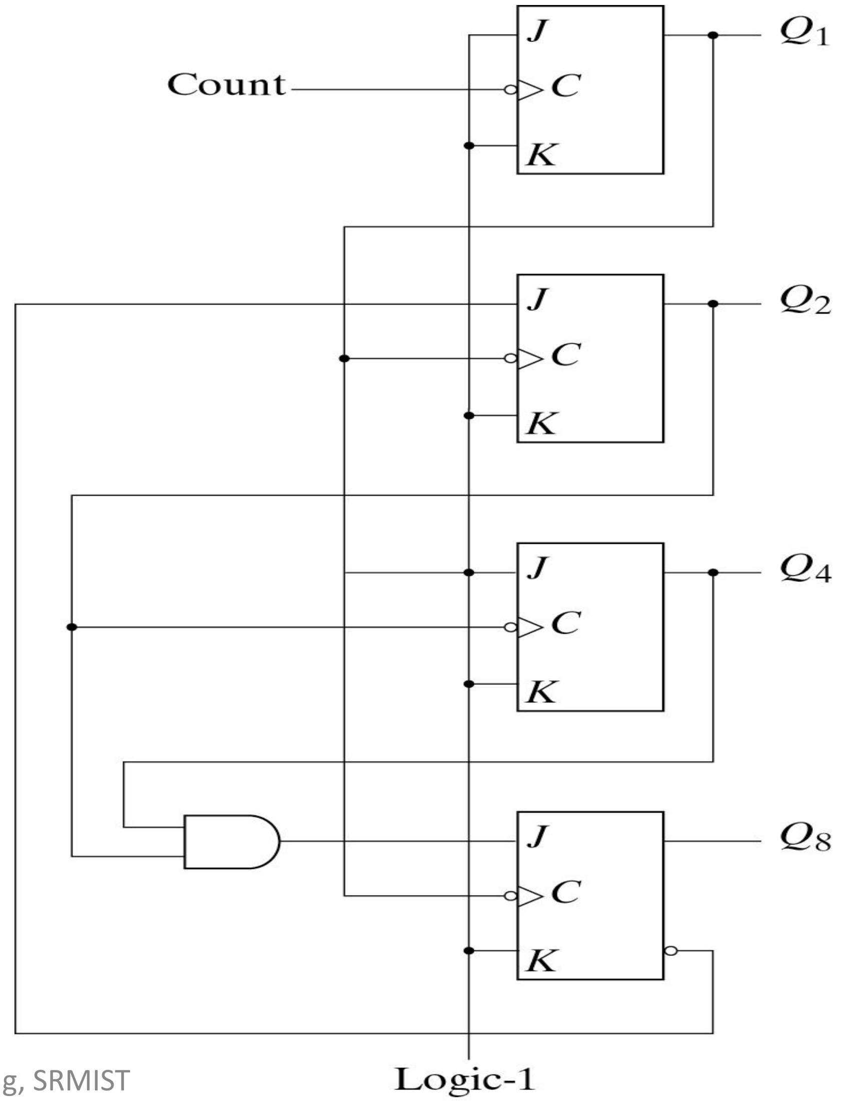


Fig. 6-10 BCD Ripple Counter

BCD Ripple Counter

- The BCD counter of [Fig. 6-9] is a decade counter.
- To count in decimal from 0 to 999, we need a three-decade counter. [Fig. 6-11]
- Multiple decade counters can be constructed by connecting BCD counters ic cascade, one for each decade.

BCD Ripple Counter

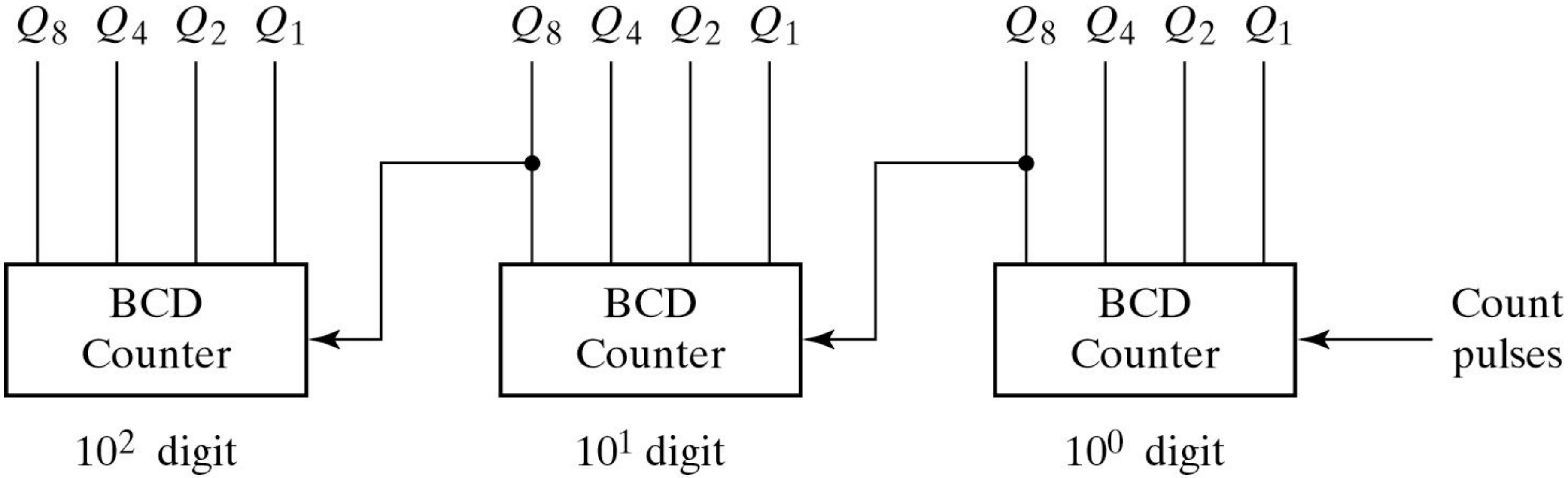


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

Synchronous Counters

- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.
- A common clock triggers all flip-flops simultaneously rather than one at a time in succession as in a ripple counter.

Binary Counter

- The design of a synchronous binary counter is so simple that is no need to go through a sequential logic design process.
- Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flop and gates

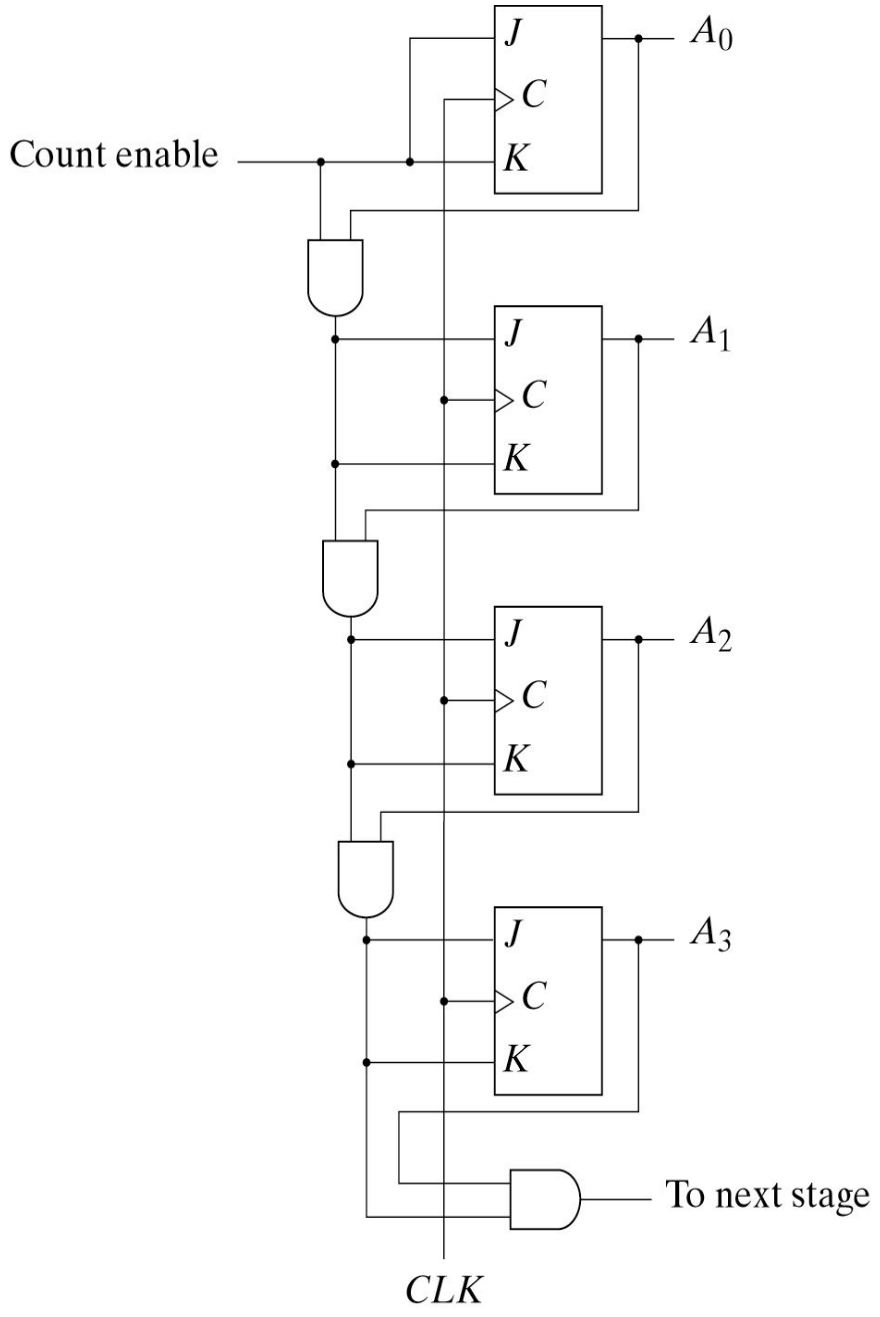


Fig. 6-12 4-Bit Synchronous Binary Counter

Presettable Up-Down Binary Counter

- The two operations can be combined in one circuit to form a counter capable of counting up or down.
- It has an up control input and down control input.

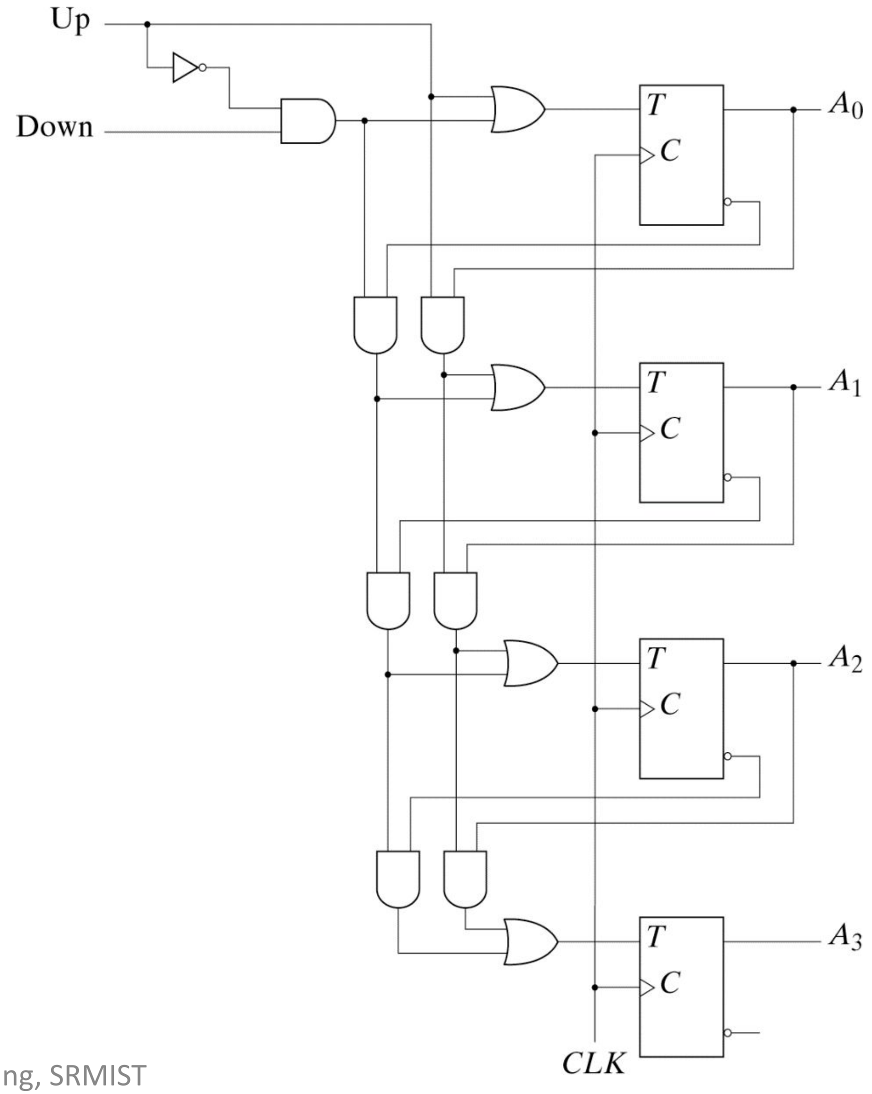


Fig. 6-13 4-Bit Up-Down Binary Counter

Decade/BCD Counter

- Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern as in a straight binary count.
- To derive the circuit of a BCD synchronous counter, it is necessary to go through a sequential circuit design procedure.

BCD Counter

Table 6-5

State Table for BCD Counter

Present State				Next State				Output	Flip-Flop inputs			
Q8	**Q4**	**Q2**	**Q1**	**Q8**	**Q4**	**Q2**	**Q1**	**Y**	**TQ8**	**TQ4**	**TQ2**	**TQ1**
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	0	1

BCD Counter

- The flip flop input equations can be simplified by means of maps. The simplified functions are
  - T_{Q1}=1
  - T_{Q2}=Q_8’Q_1
  - T_{Q4}=Q_2Q_1
  - T_{Q8}=$Q_8Q_1+Q_4Q_2Q_1$
  - y=Q_8Q_1
- The circuit can be easily drawn with four T flip-flops, five AND gates, and one OR gate.

Binary Counter with Parallel Load

- Counters employed in digital systems quite often require a parallel load capability for transferring an initial binary number into the counter prior to count operation.
- The input load control when equal to 1 disables the count operation and causes a transfer of data from the four data inputs into the four flip-flops [Fig. 6-14]

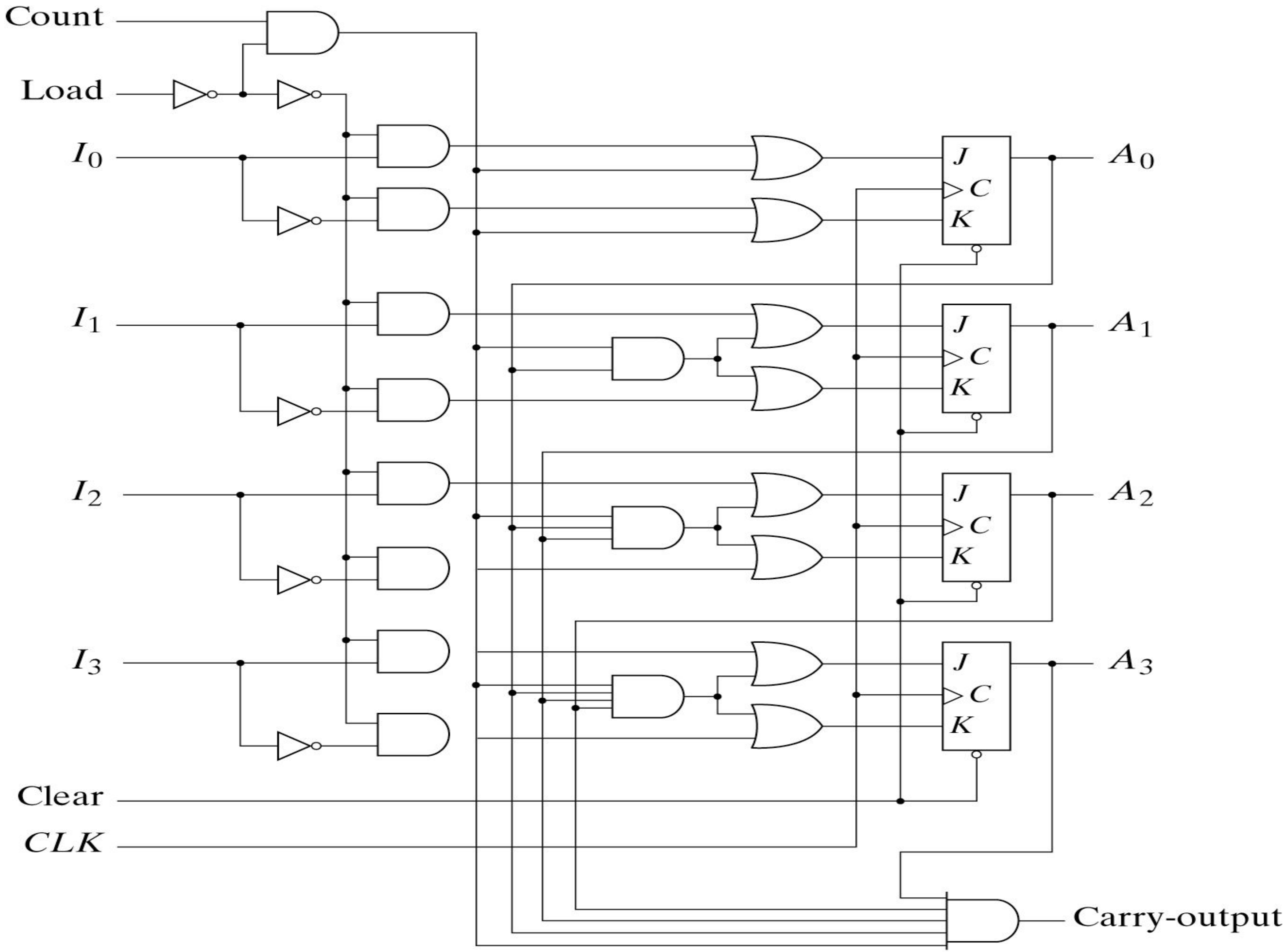


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Binary Counter with Parallel Load

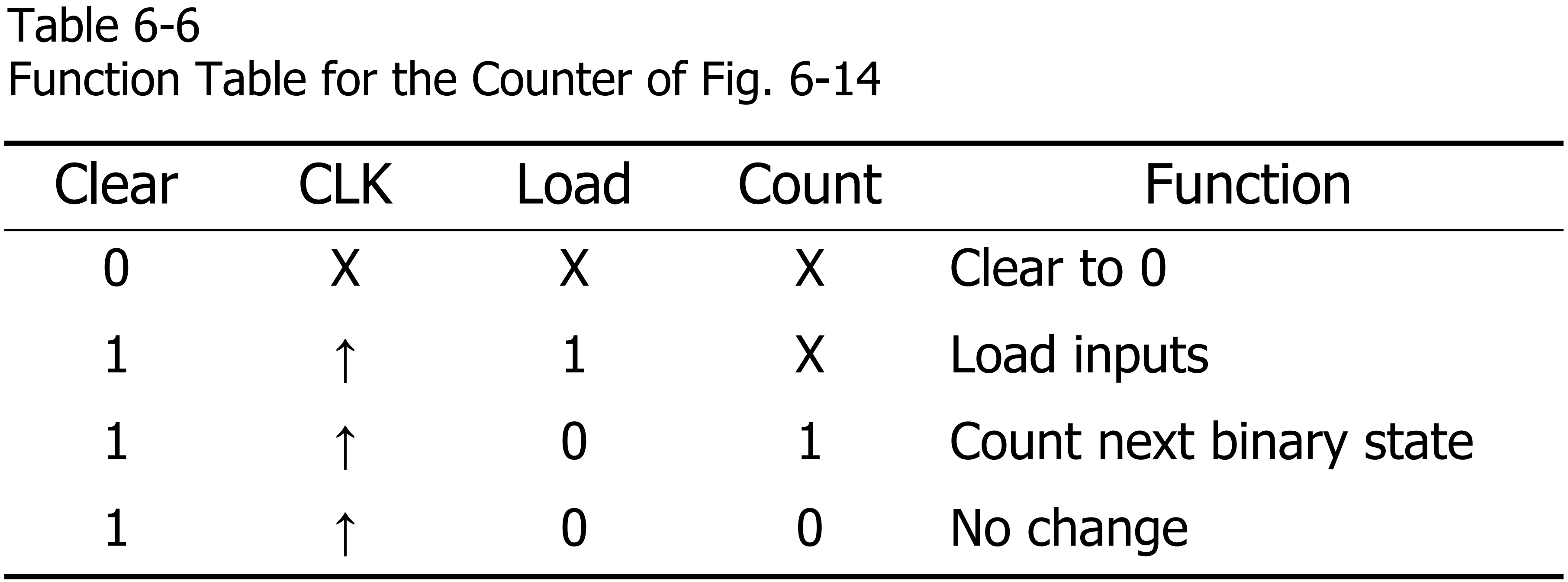


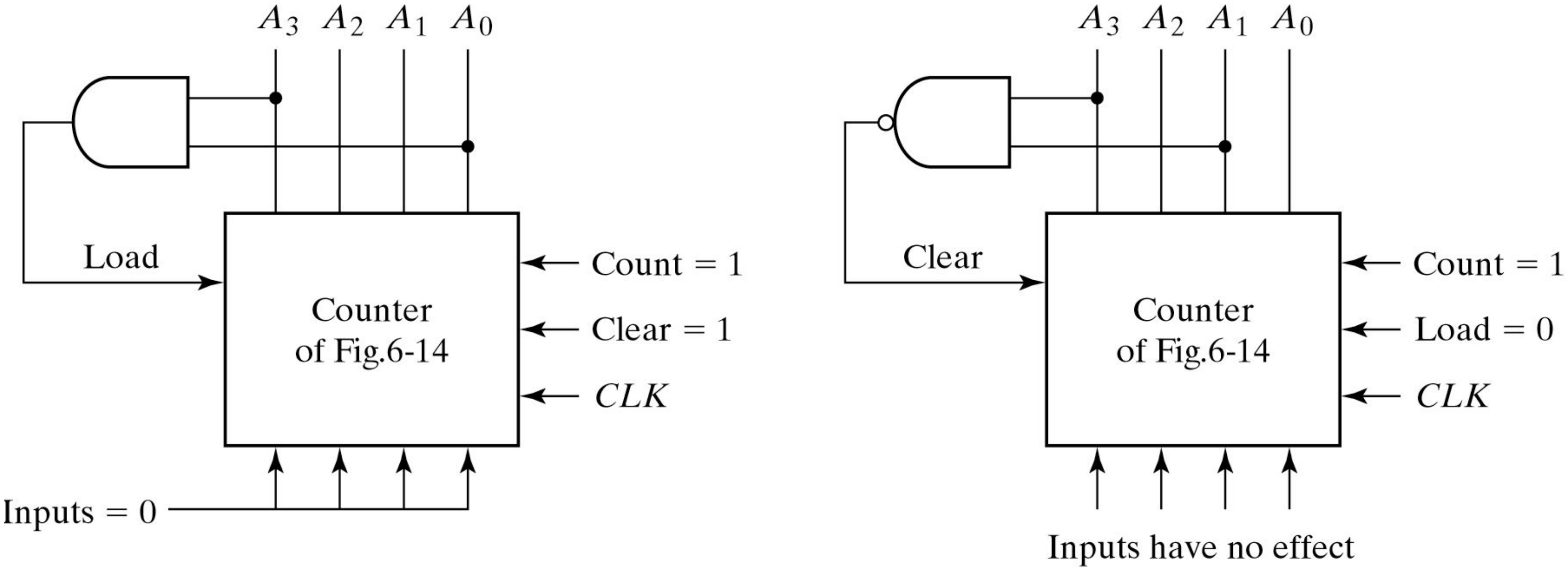
Table 6-6
Function Table for the Counter of Fig. 6-14

Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Binary Counter with Parallel Load

- A counter with parallel load can be used to generate any desire count sequence.
- [Fig.6-15] shows two ways in which a counter with parallel load is used to generate the BCD count.

Binary Counter with Parallel Load



(a) Using the load input

(b) Using the clear input

Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

Modulus Counters

- **Modulus Counters**, or simply MOD counters, are defined based on the number of states that the counter will sequence through before returning back to its original value.
- For example, a 2-bit counter that counts from 00_2 to 11_2 in binary, that is 0 to 3 in decimal, has a modulus value of 4 (00 → 1 → 10 → 11, and return back to 00) so would therefore be called a modulo-4, or mod-4, counter.
- Note also that it has taken four clock pulses to get from 00 to 11.

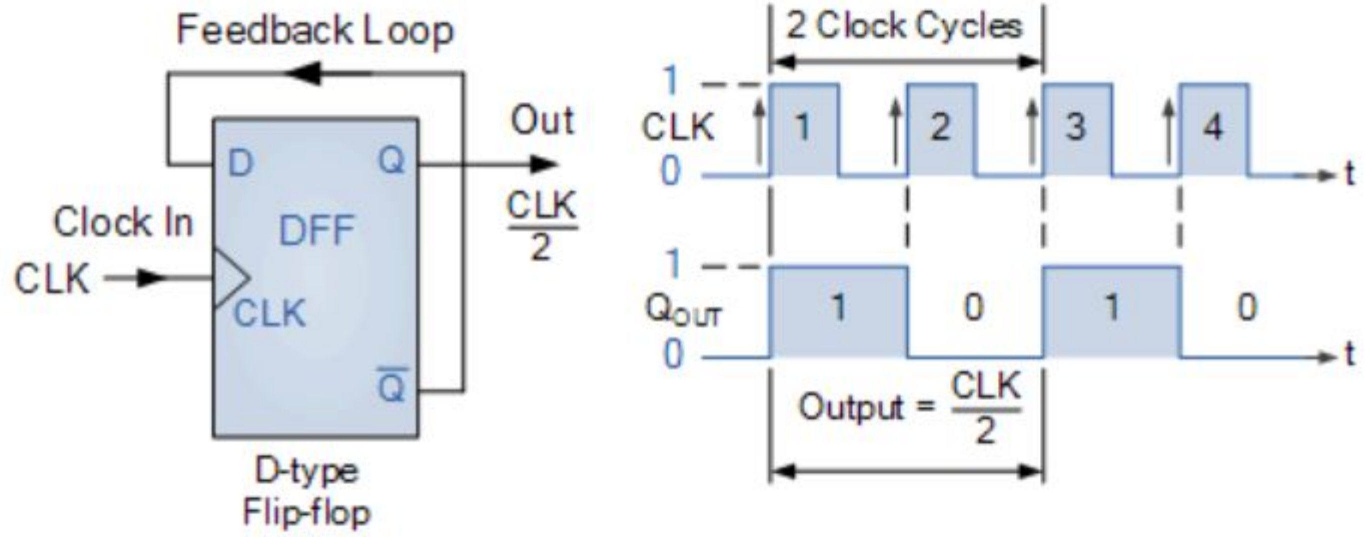
- Example: there are only two bits, (n = 2) then the maximum number of possible output states (maximum modulus) for the counter is: $2^n = 2^2$ or 4.
- Therefore, a “Mod-N” counter will require “N” number of flip-flops connected together to count a single data bit while providing 2^n different output states, (n is the number of bits). Note that N is always a whole integer value.
- MOD counters have a modulus value that is an integral power of 2, that is, 2, 4, 8, 16 and so on to produce an n-bit counter depending on the number of flip-flops used, and how they are connected, determining the type and modulus of the counter.

Design

- MOD counters are made using “flip-flops” and a single flip-flop can produce a count of 0 or 1, giving a maximum count of 2.
- There are different types of flip-flop designs we could use, the S-R, the J-K, J-K Master-slave, the D-type or even the T-type flip-flop to construct a counter.

Divide-by-Two Counter

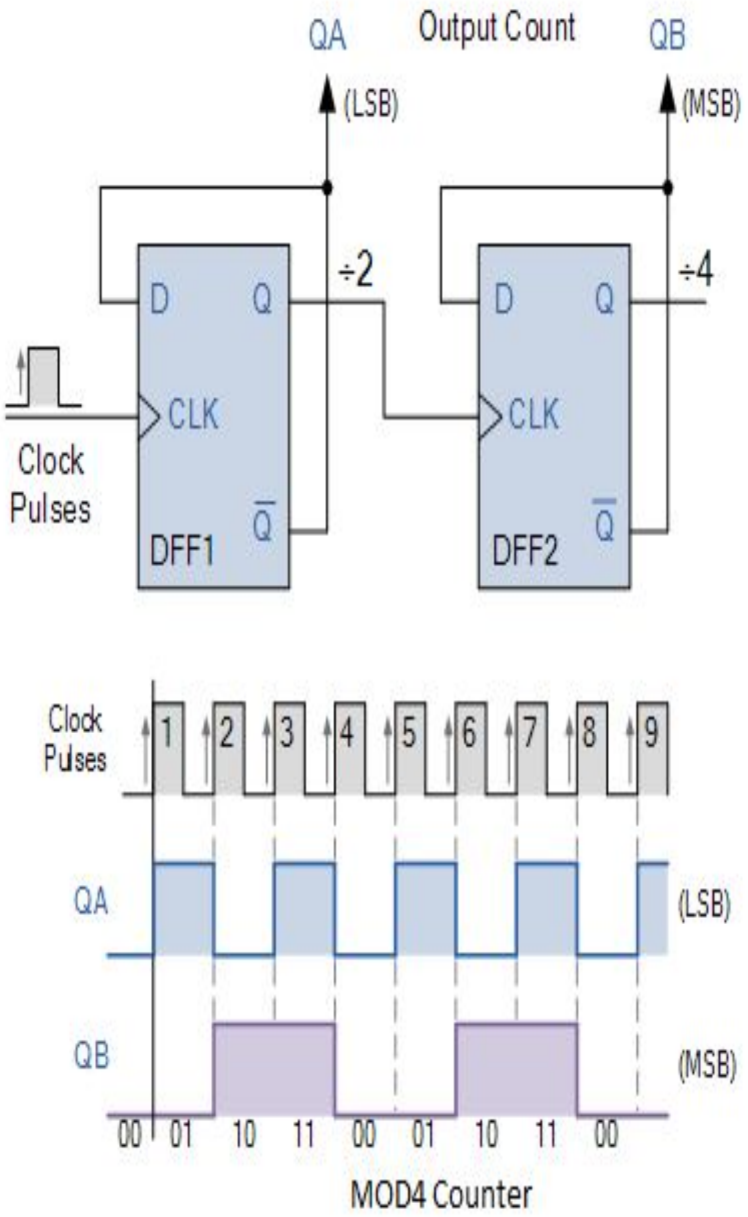
- The edge-triggered D-type flip-flop is a useful and versatile building block to construct a MOD counter or any other type of sequential logic circuit.
- By connecting the Q output back to the “D” input as shown, and creating a feedback loop, we can convert it into a binary divide-by-two counter using the clock input only as the Q output signal is always the inverse of the Q output signal.



- The timing diagrams show that the “Q” output waveform has a frequency exactly one-half that of the clock input, thus the flip-flop acts as a frequency divider.
- If we added another D-type flip-flop so that the output at “Q” was the input to the second DFF, then the output signal from this second DFF would be one-quarter of the clock input frequency, and so on.
- So for an “n” number of flip-flops, the output frequency is divided by 2n, in steps of 2.

MOD 4 Counter

- If a single flip-flop can be considered as a modulo-2 or MOD-2 counter, then adding a second flip-flop would give us a MOD-4 counter allowing it to count in four discrete steps.
- The overall effect would be to divide the original clock input signal by four.
- Then the binary sequence for this 2-bit MOD-4 counter would be: 00, 01, 10, and 11 as shown.



- The switching transitions of QA, QB and CLK in the above timing diagram are shown to be simultaneous even though this connection represents an asynchronous counter.
- In reality there would be a very small switching delay between the application of the positive going clock (CLK) signal, and the outputs at QA and QB.

Truth table

- when QA = 0 and QB = 0, the count is 00.
- After the application of the clock pulse, the values become QA = 1, QB = 0, giving a count of 01 and after the next clock pulse, the values become QA = 0, QB = 1, giving a count of 10.
- Finally the values become QA = 1, QB = 1, giving a count of 11.
- The application of the next clock pulse causes the count to return back to 00, and thereafter it counts continuously up in a binary sequence of: 00, 01, 10, 11, 00, 01 …etc.
- Then we have seen that a MOD-2 counter consists of a single flip-flop and a MOD-4 counter requires two flip-flops,allowing it to count in four discrete steps.

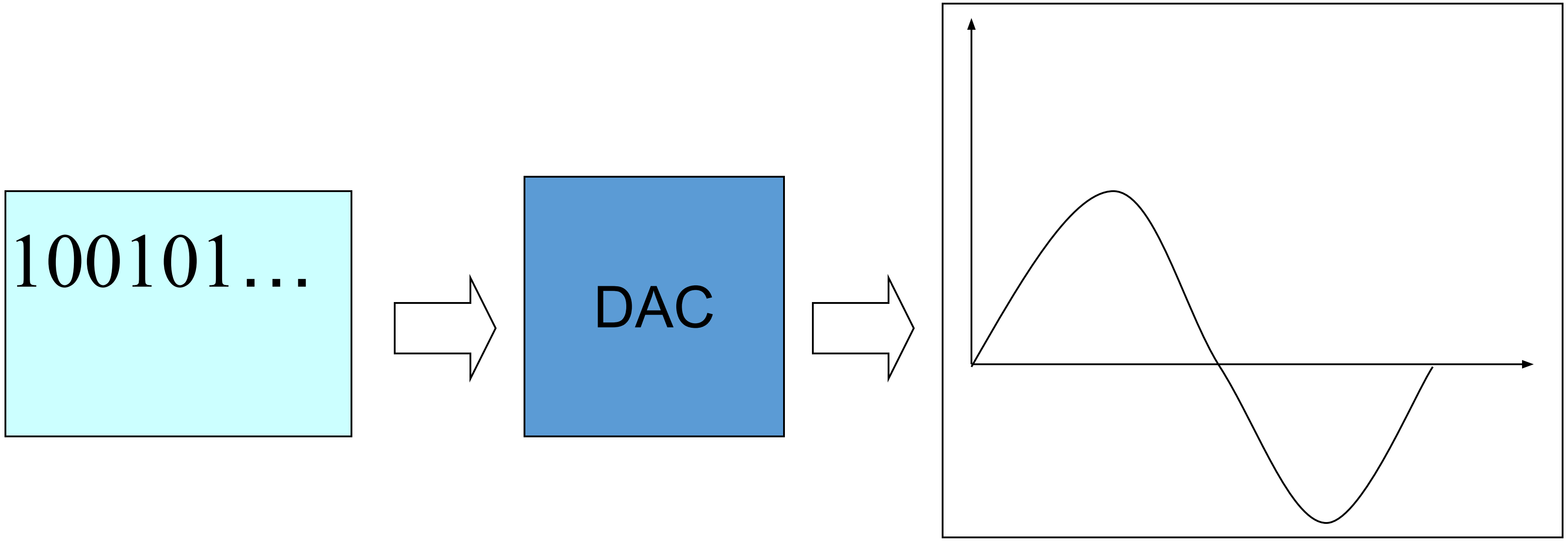
- We could easily add another flip-flop onto the end of a MOD-4 counter to produce a MOD-8 counter giving us a 23 binary sequence of counting from 000 up to 111, before resetting back to 000.
- A fourth flip-flop would make a MOD-16 counter and so on, in fact we could go on adding extra flip-flops for as long as we wanted.

Modulo-5 Counter

- Suppose we want to design a MOD-5 counter, how could we do that.
- First we know that “m = 5”, so 2^n must be greater than 5.
- As $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and 8 is greater than 5, then we need a counter with three flip-flops (N = 3) giving us a natural count of 000 to 111 in binary (0 to 7 decimal).

What is a DAC?

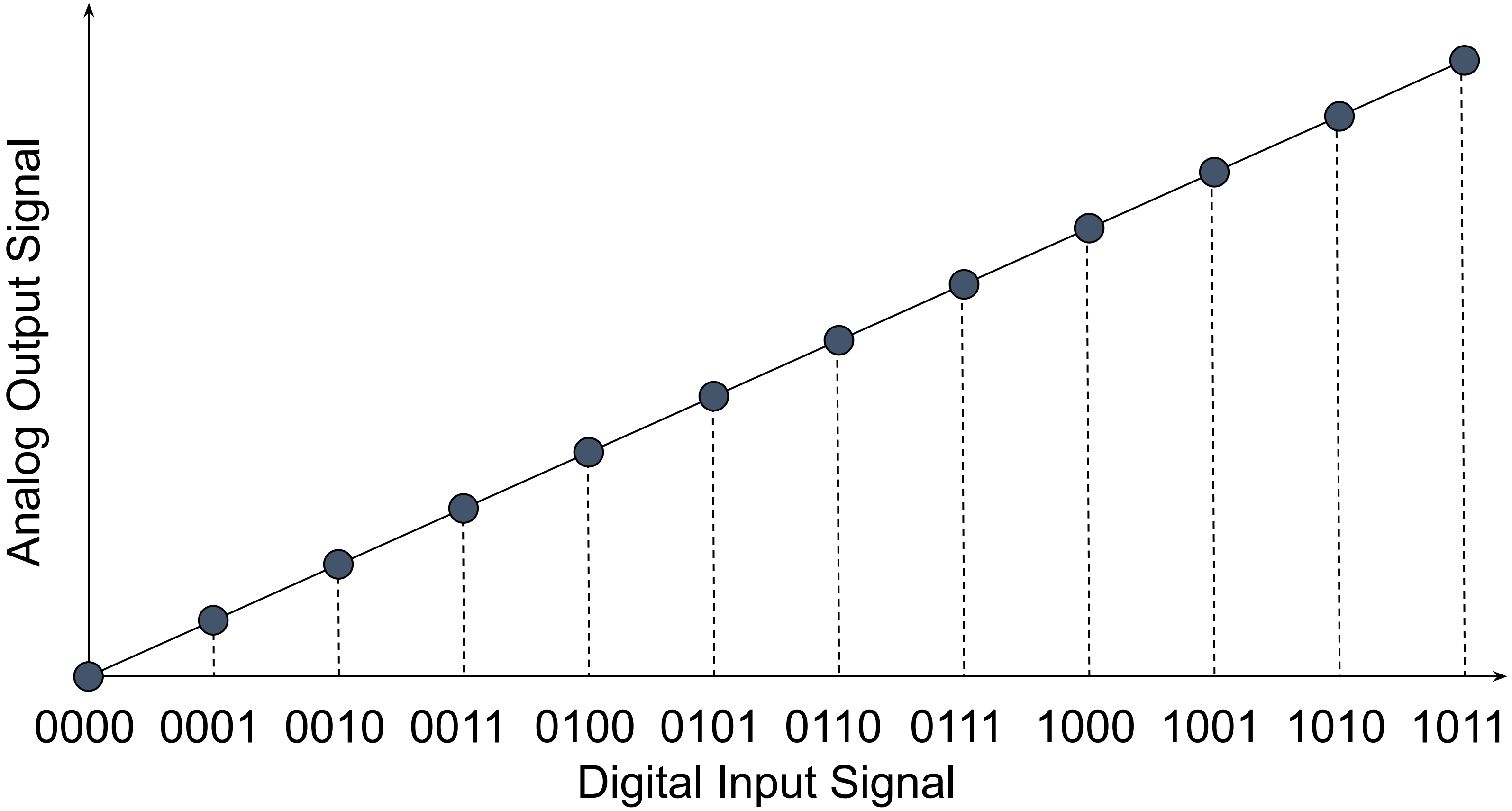
- A digital to analog converter (DAC) converts a digital signal to an analog voltage or current output.



100101…

DAC

What is a DAC?


Analog Output Signal
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
Digital Input Signal

Types of DACs

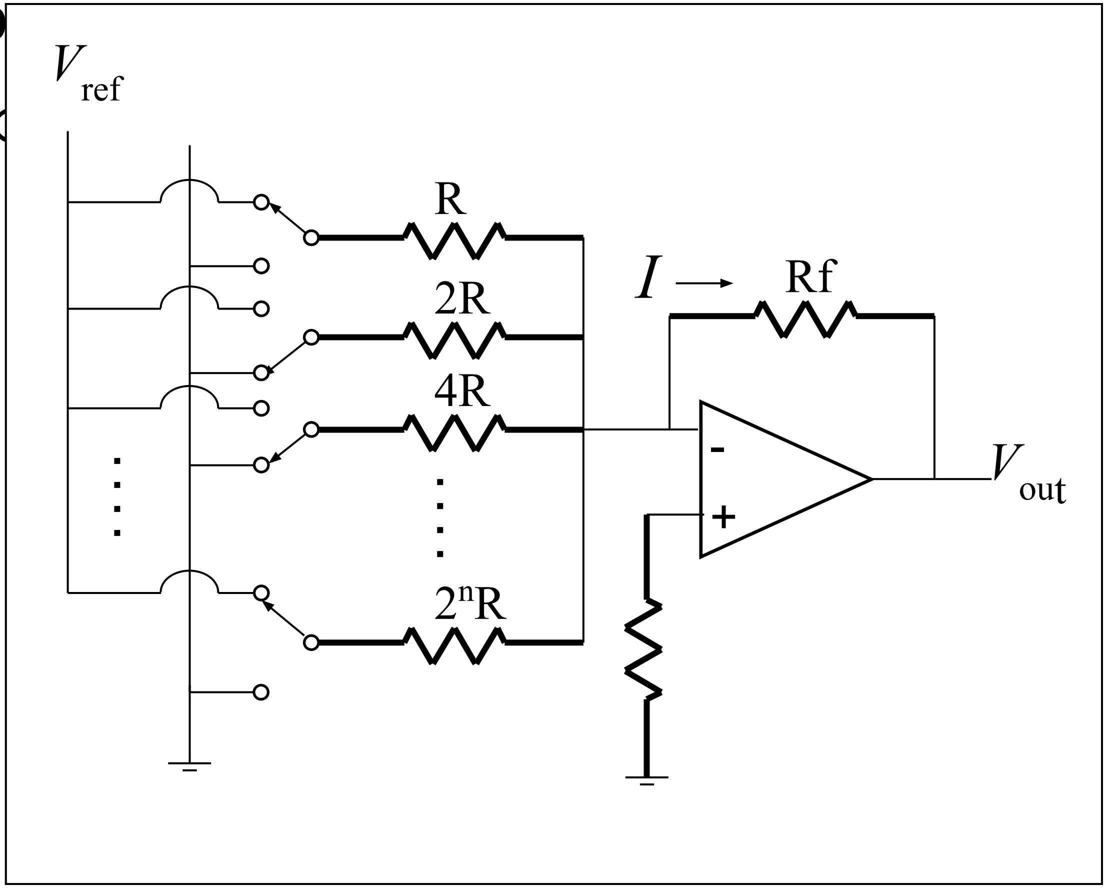
- Many types of DACs available.
- Usually switches, resistors, and op-amps used to implement conversion
- Two Types:
  - Binary Weighted Resistor
  - R-2R Ladder

Binary Weighted Resistor

- Utilizes a summing op-amp circuit
- Weighted resistors are used to distinguish each bit from the most significant to the least significant
- Transistors are used to switch between V_{ref} and ground (bit high or low)

Binary Weighted Resistor

- Assume Ideal Op-amp
- No current into op-amp
- Virtual ground at inverting input
- $V_{out} = -IR_f$



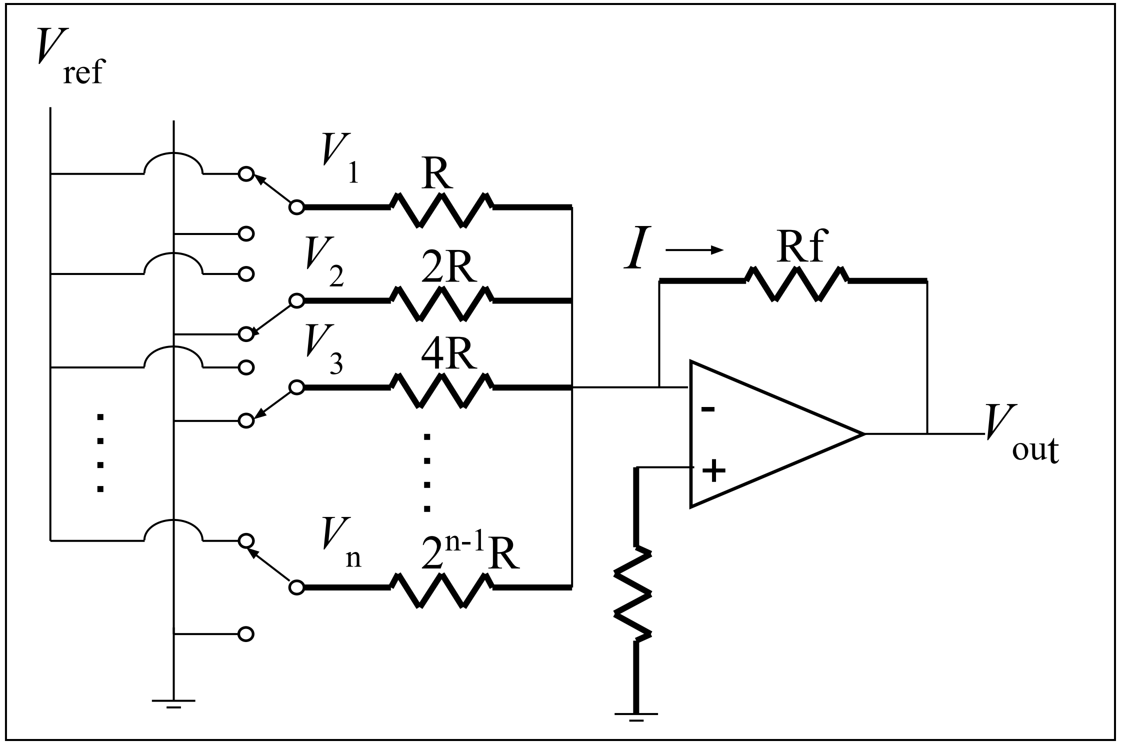
Binary Weighted Resistor

Voltages V_1 through V_n are either V_{ref} if corresponding bit is high or ground if corresponding bit is low

V_1 is most significant bit

V_n is least significant bit

$$V_{\text{out}} = -IR_{\text{f}} = -R_{\text{f}}\left(\frac{V_1}{R} + \frac{V_2}{2R} + \frac{V_3}{4R} + \cdots + \frac{V_{\text{n}}}{2^{\text{n-1}}R}\right)$$



MSB

LSB

Binary Weighted Resistor

If $R_f = R/2$

$$V_{\text{out}} = -IR_{\text{f}} = -\left(\frac{V_1}{2} + \frac{V_2}{4} + \frac{V_3}{8} + \cdots \quad \frac{V_{\text{n}}}{2^{\text{n}}} \right)$$

For example, a 4-Bit converter yields

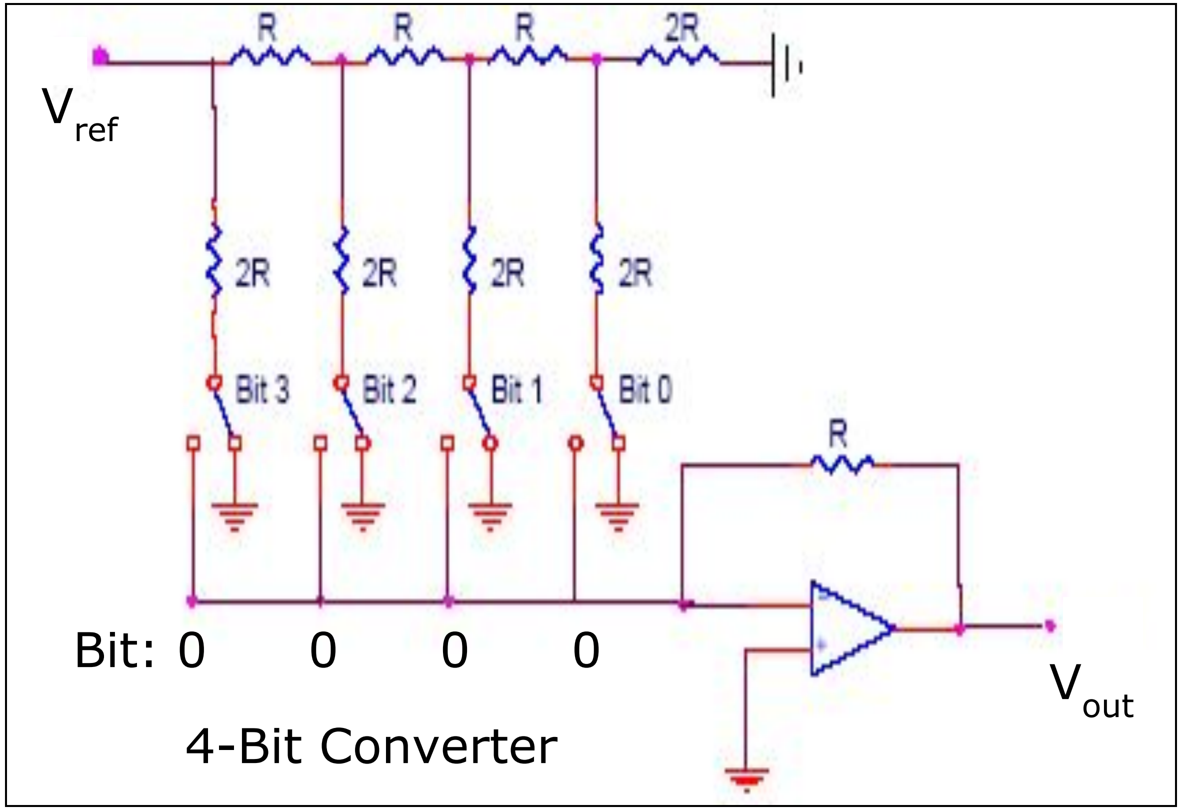
$$V_{\text{out}} = -V_{\text{ref}} \left(b_3 \frac{1}{2} + b_2 \frac{1}{4} + b_1 \frac{1}{8} + b_0 \frac{1}{16} \right)$$

Where b_3 corresponds to Bit-3, b_2 to Bit-2, etc.

Binary Weighted Resistor

- Advantages
  - Simple Construction/Analysis
  - Fast Conversion
- Disadvantages
  - Requires large range of resistors (2000:1 for 12-bit DAC) with necessary high precision for low resistors
  - Requires low switch resistances in transistors
  - Can be expensive. Therefore, usually limited to 8-bit resolution.

R-2R Ladder



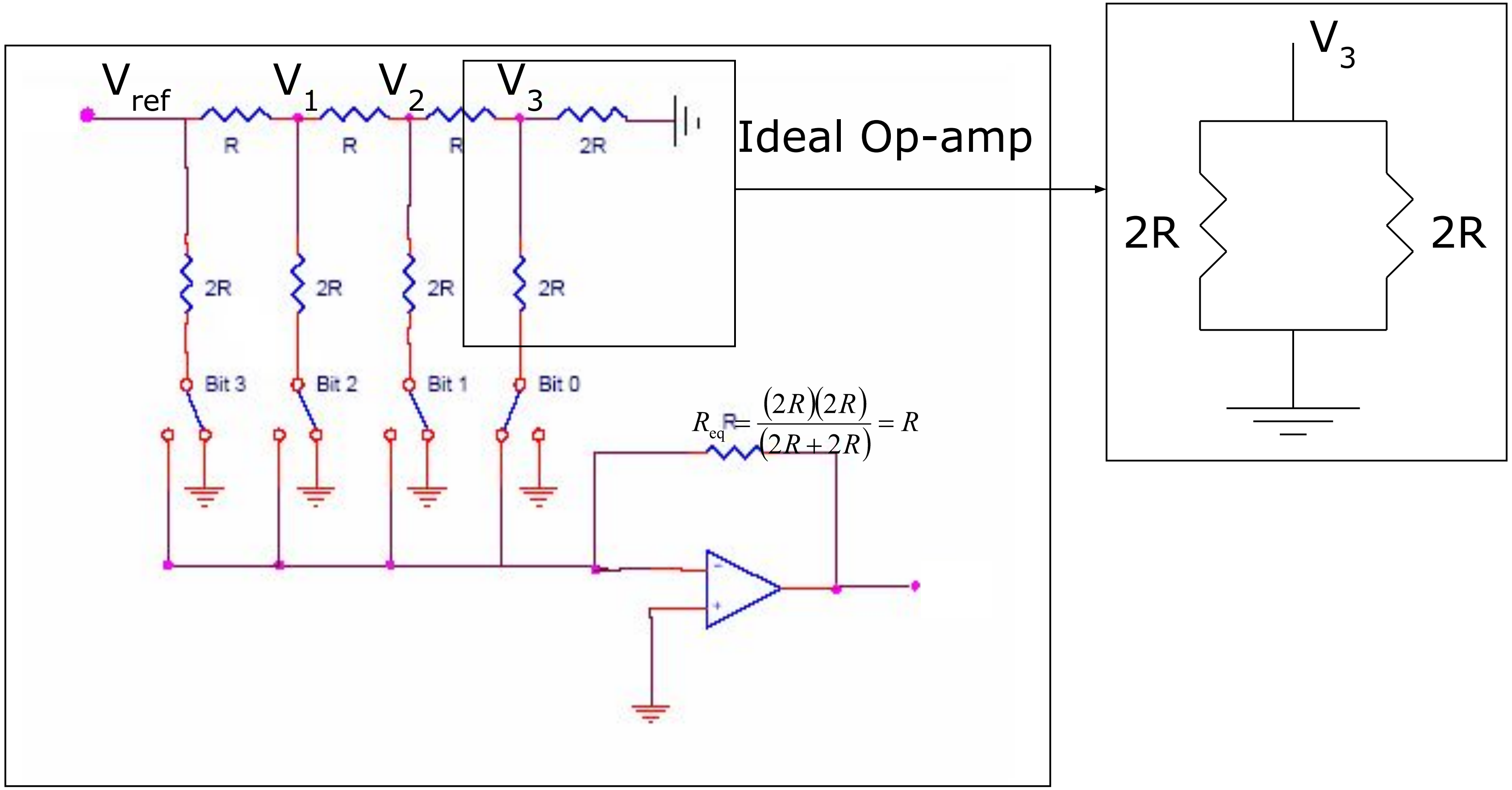
4-Bit Converter

Each bit corresponds to a switch:

If the bit is high, the corresponding switch is connected to the inverting input of the op-amp.

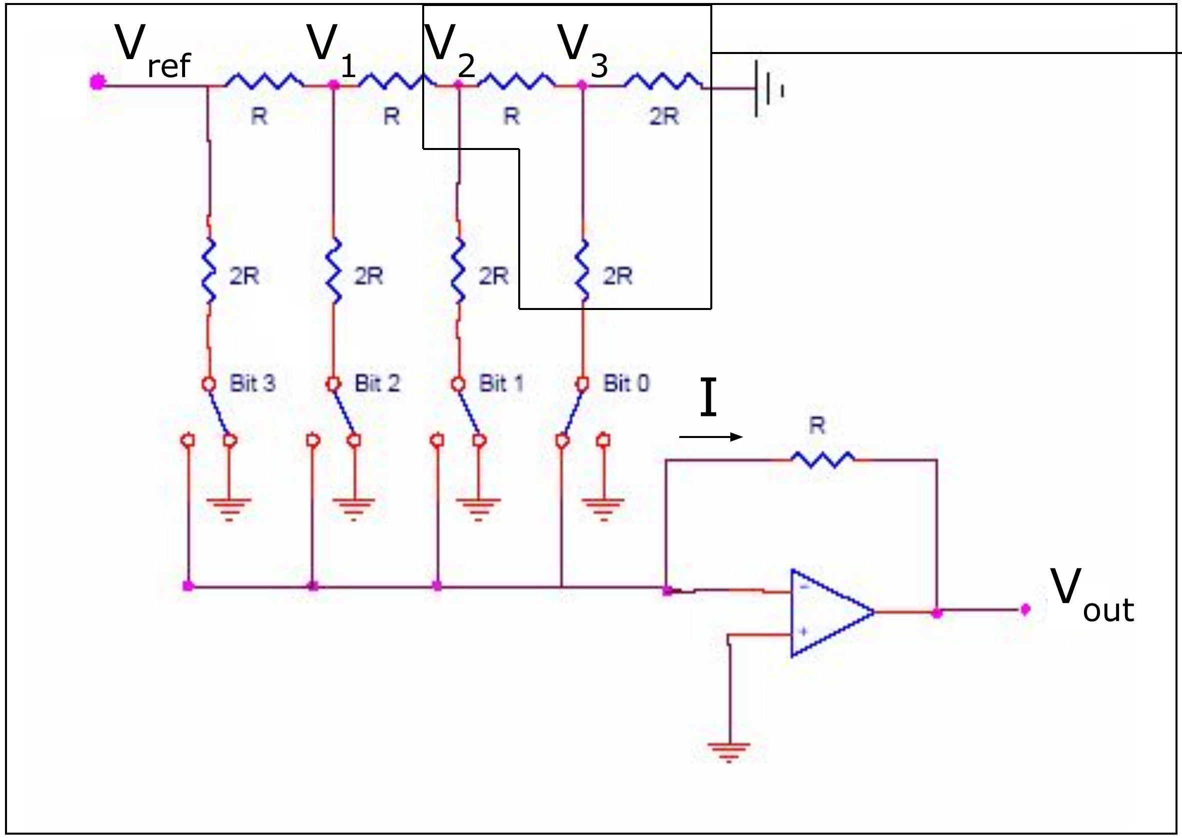
If the bit is low, the corresponding switch is connected to ground.

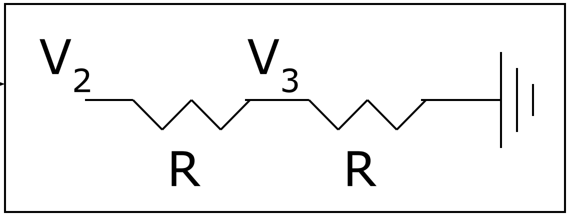
R-2R Ladder



V_{ref}
V_1
V_2
V_3
R
R
R
2R
2R
2R
2R
2R
Bit 3
Bit 2
Bit 1
Bit 0
Ideal Op-amp
R_{eq} R $= \frac{(2R)(2R)}{(2R+2R)} = R$
-
+
V_3
2R
2R

R-2R Ladder





$$V_3 = \left(\frac{R}{R+R}\right)V_2 = \frac{1}{2}V_2$$

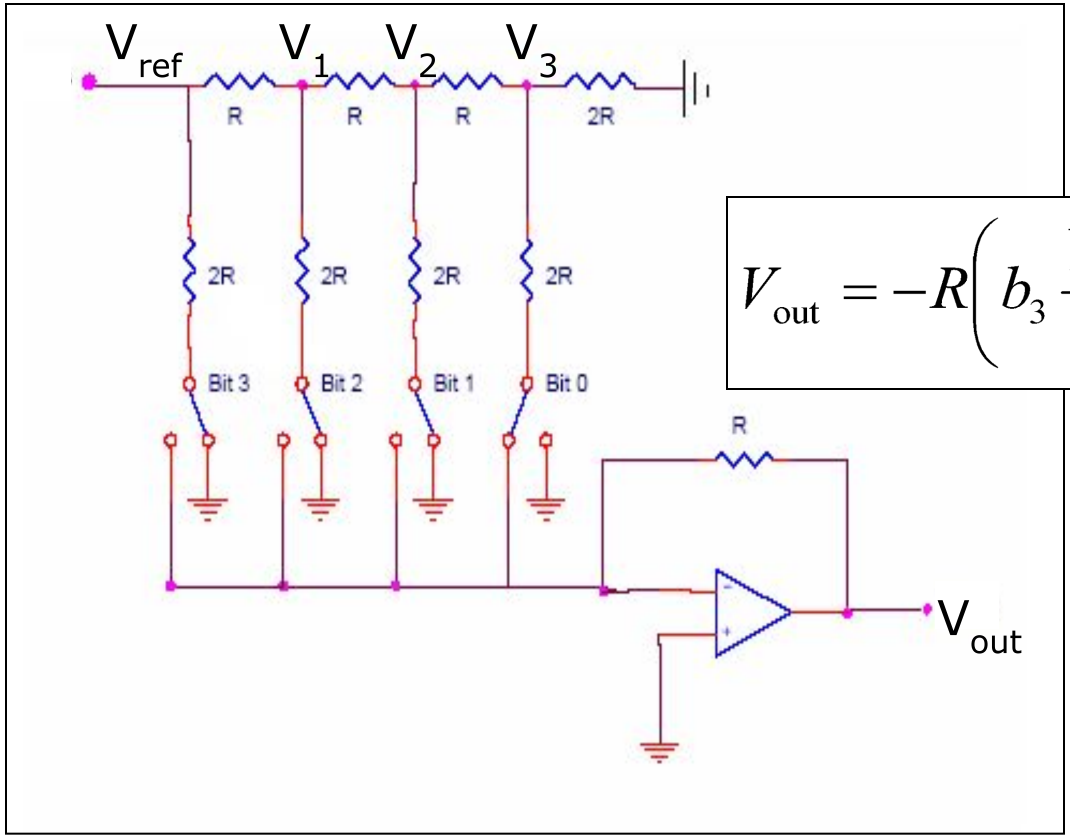
Likewise,

$$V_2 = \frac{1}{2}V_1$$

$$V_1 = \frac{1}{2}V_{\text{ref}}$$

$$V_{\text{out}} = -IR$$

R-2R Ladder



Results:

$$V_3 = \frac{1}{8}V_{\text{ref}},\ V_2 = \frac{1}{4}V_{\text{ref}},\ V_1 = \frac{1}{2}V_{\text{ref}}$$

$$\boxed{V_{\text{out}} = -R\left(b_3\frac{V_{\text{ref}}}{2R} + b_2\frac{V_{\text{ref}}}{4R} + b_1\frac{V_{\text{ref}}}{8R} + b_0\frac{V_{\text{ref}}}{16R}\right)}$$

Where b_3 corresponds to bit 3, b_2 to bit 2, etc.

If bit n is set, b_n=1

If bit n is clear, b_n=0

R-2R Ladder

For a 4-Bit R-2R Ladder

$$V_{\text{out}} = -V_{\text{ref}}\left(b_3\frac{1}{2} + b_2\frac{1}{4} + b_1\frac{1}{8} + b_0\frac{1}{16}\right)$$

For general n-Bit R-2R Ladder or Binary Weighted Resister DAC

$$\boxed{V_{\text{out}} = -V_{\text{ref}}\sum_{i=1}^{n} b_{n-i}\frac{1}{2^i}}$$

R-2R Ladder

- Advantages
  - Only two resistor values (R and 2R)
  - Does not require high precision resistors
- Disadvantage
  - Lower conversion speed than binary weighted DAC

Specifications of DACs

- Resolution
- Speed
- Linearity
- Settling Time
- Reference Voltages
- Errors

Resolution

- Smallest analog increment corresponding to 1 LSB change
- An N-bit resolution can resolve 2^N distinct analog levels
- Common DAC has a 8-16 bit resolution

$$\text{Resolution} = V_{LSB} = \frac{V_{ref}}{2^N}$$

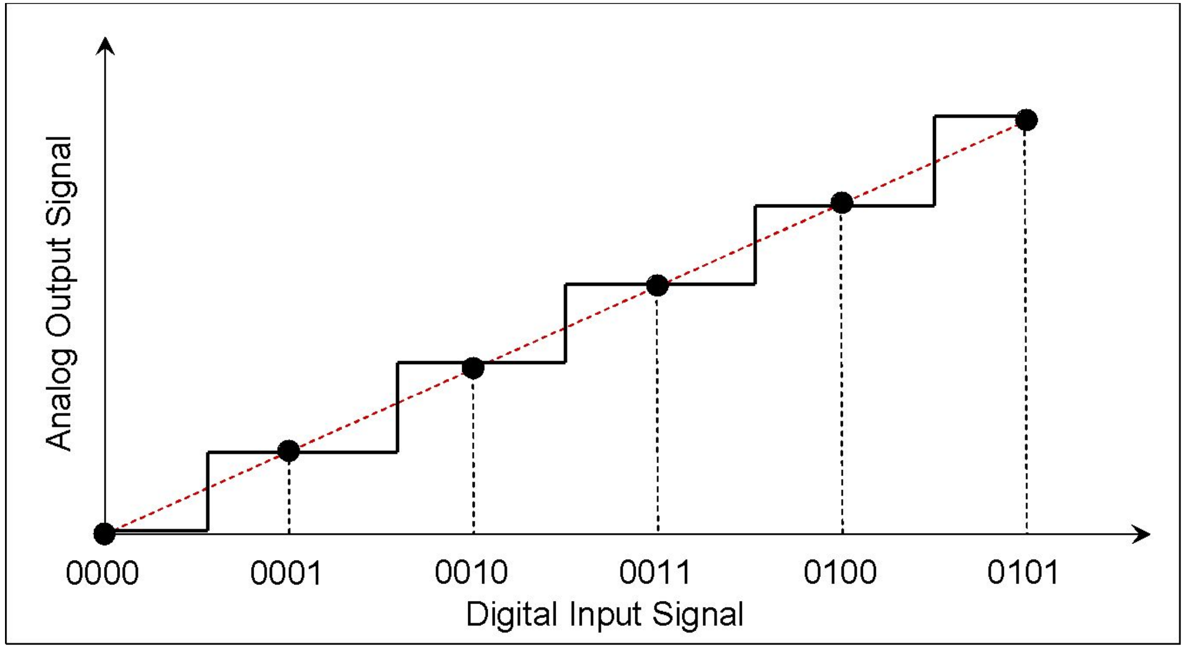
$$\text{where } N = number\ of\ bits$$

Speed

- Rate of conversion of a single digital input to its analog equivalent
- Conversion rate depends on
    - clock speed of input signal
    - settling time of converter
- When the input changes rapidly, the DAC conversion speed must be high.

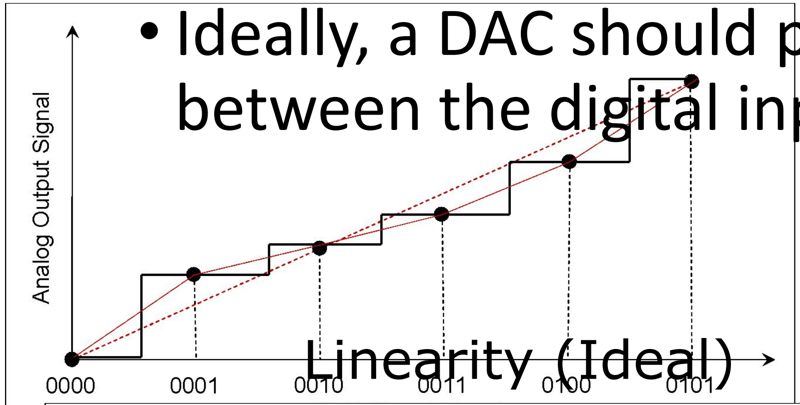
Linearity

- The difference between the desired analog output and the actual output over the full range of expected values



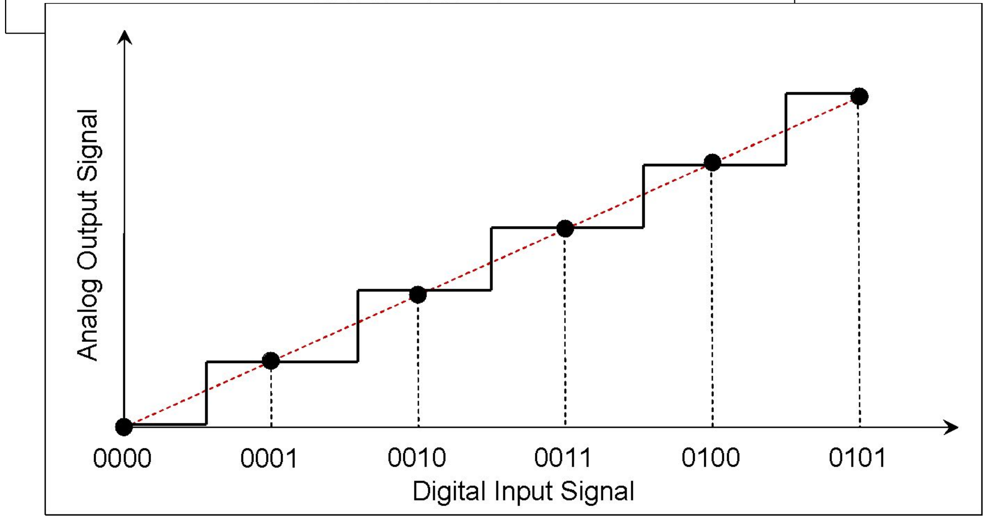
Linearity

- Ideally, a DAC should produce a linear relationship between the digital input and analog output



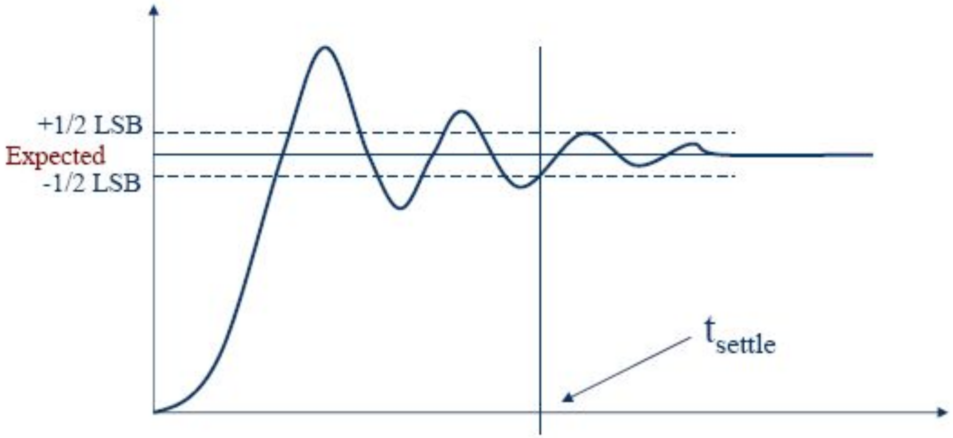
Linearity (Ideal)

Non-Linearity



Settling Time

- Time required for the output signal to settle within +/- ½ LSB of its final value after a given change in input scale
- Limited by slew rate of output amplifier
- Ideally, an instantaneous change in analog voltage would occur when a new binary word enters into DAC



Reference Voltages

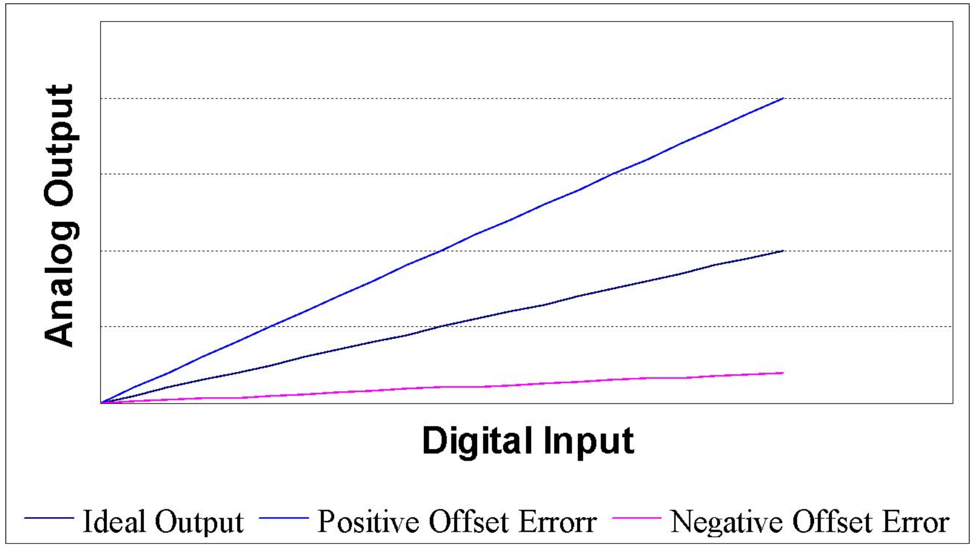
- Used to determine how each digital input will be assigned to each voltage division
- Types:
  - Non-multiplier DAC: Vref is fixed
  - Multiplier DAC: Vref provided by external source

Types of Errors Associated with DACs

- Gain
- Offset
- Full Scale
- Resolution
- Non-Linearity
- Non-Monotonic
- Settling Time and Overshoot

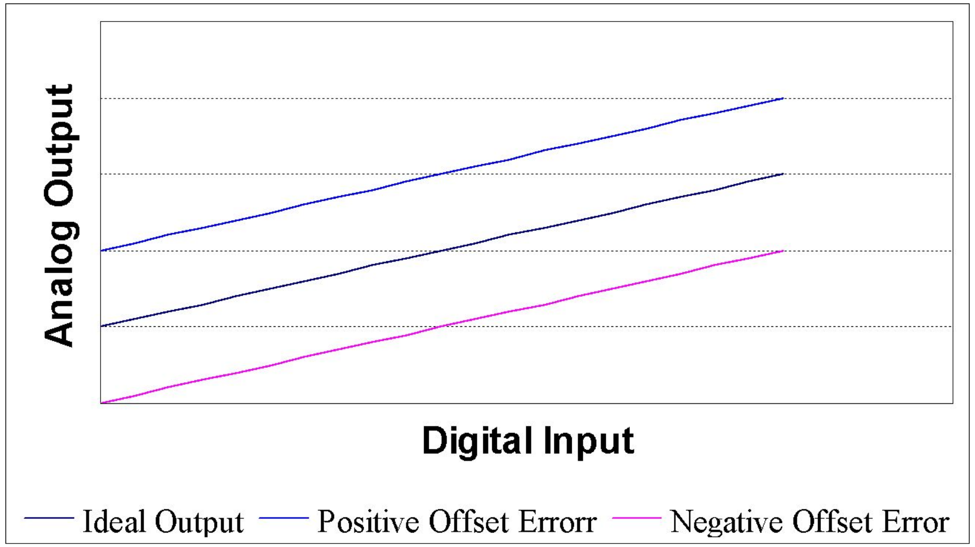
Gain Error

- Occurs when the slope of the actual output deviates from the ideal output



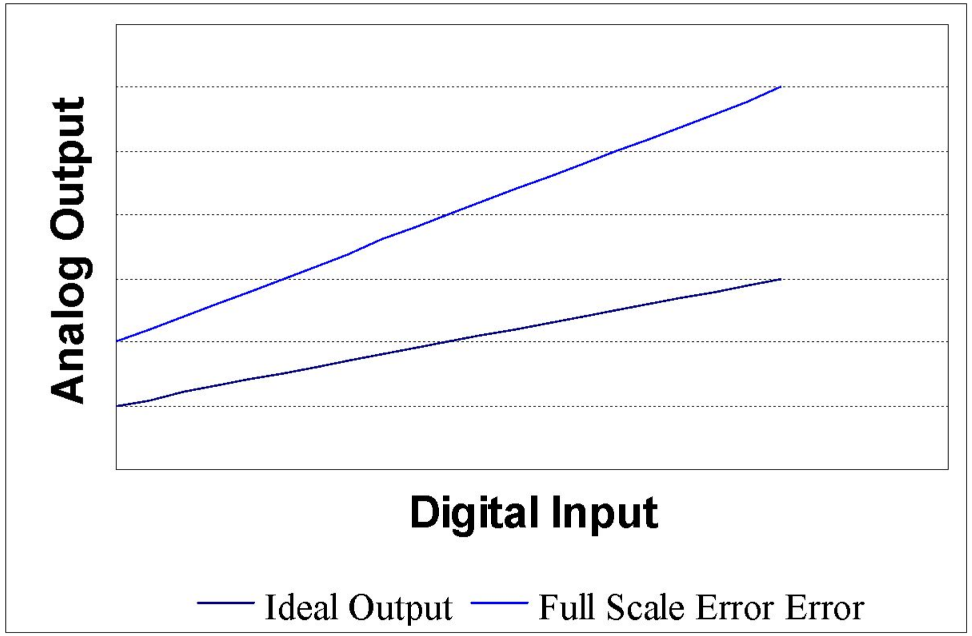
Offset Error

- Occurs when there is a constant offset between the actual output and the ideal output



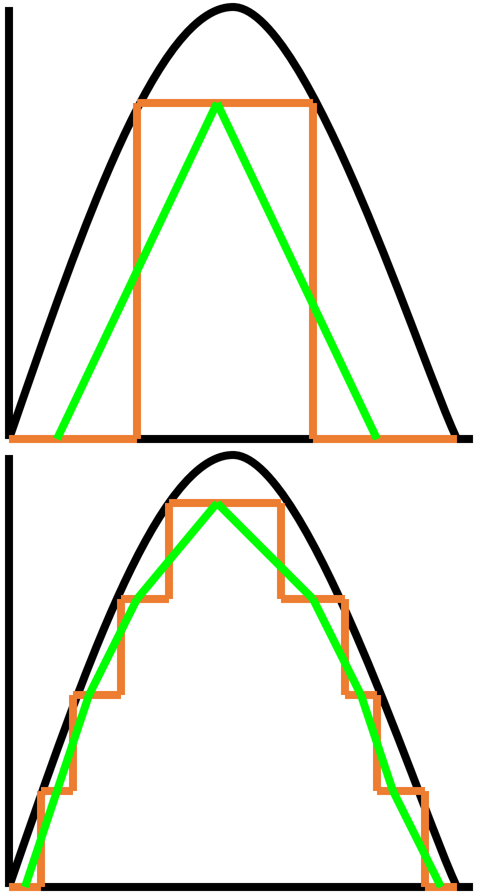
Full Scale Error

- Occurs when the actual signal has both gain and offset errors



Resolution Error

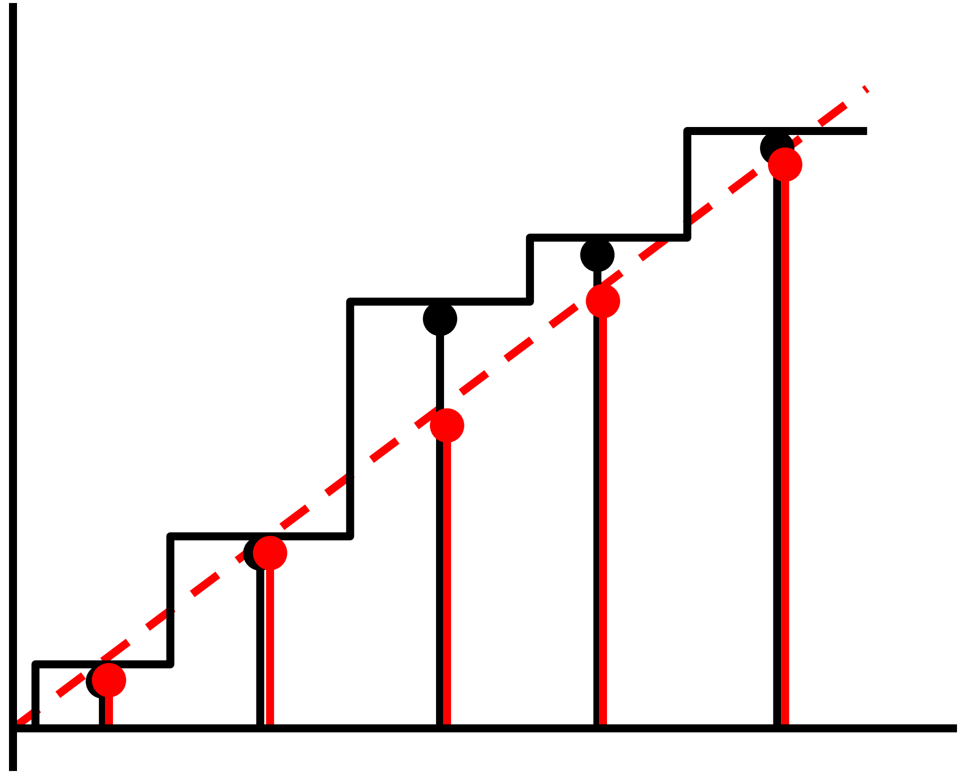
- Poor representation of ideal output due to poor resolution
- Size of voltage divisions affect the resolution



Non-Linearity Error

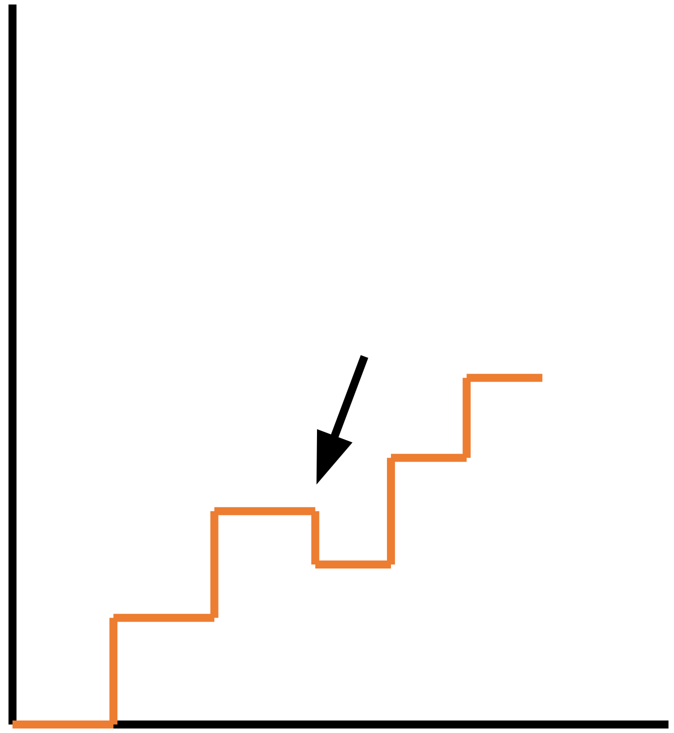
- Occurs when analog output of signal is non-linear
- Two Types
  - Differential – analog step-sizes changes with increasing digital input (measure of largest deviation; between successive bits
  - Integral – amount of deviation from a straight line after offset and gain errors removed; on concurrent bits

Non-Linearity Error, cont.



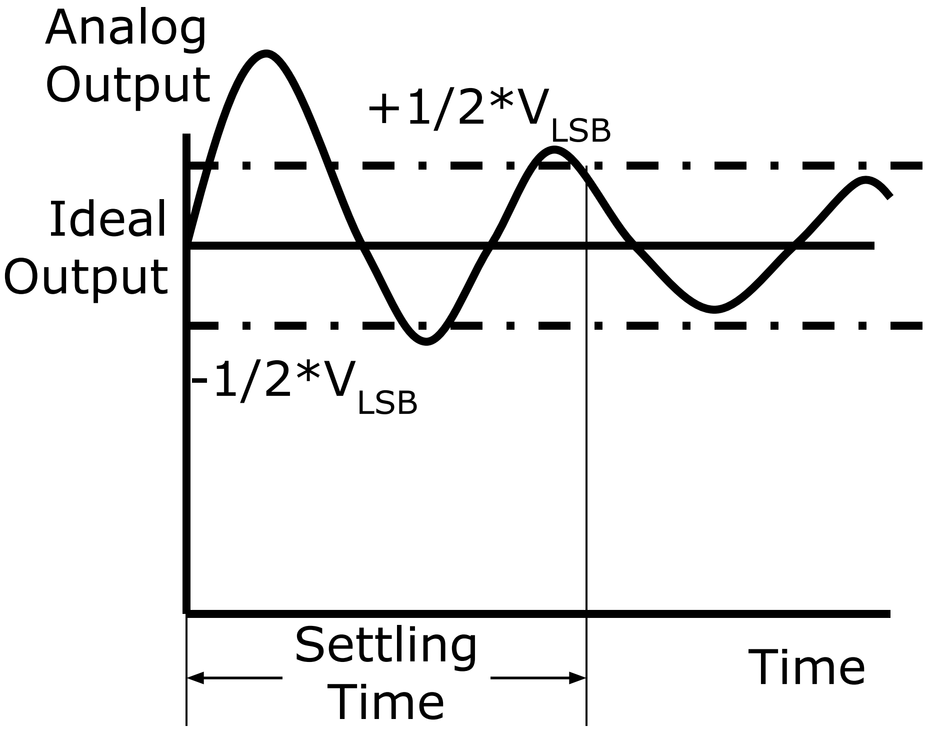
Non-Monotonic Error

- Occurs when an increase in digital input results in a decrease in the analog output



Settling Time and Overshoot Error

- Settling Time – time required for the output to fall with in +/- ½ V_{LSB}
- Overshoot – occurs when analog output overshoots the ideal output

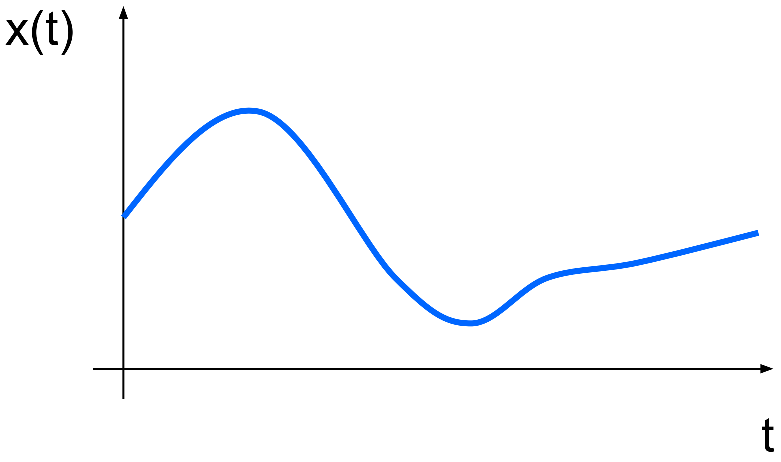


Applications

- Digital Motor Control
- Computer Printers
- Sound Equipment (e.g. CD/MP3 Players, etc.)
- Electronic Cruise Control
- Digital Thermostat

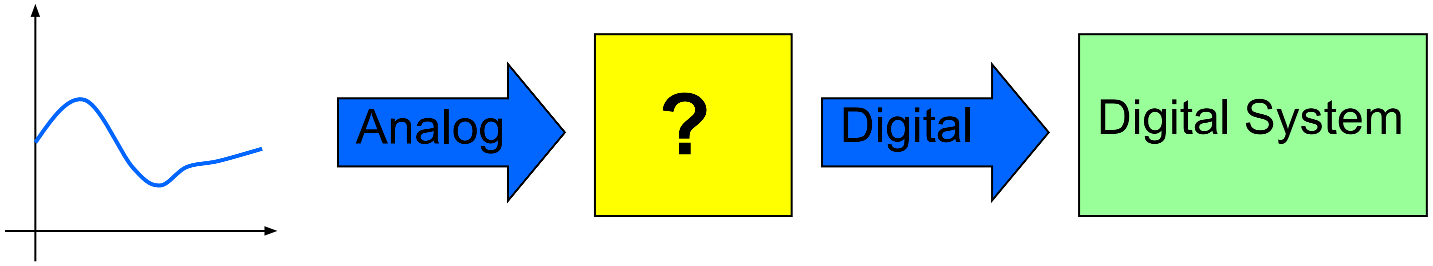
ANALOG TO DIGITAL CONVERSION

- Most signals we want to process are analog
- i.e.: they are continuous and can take an inifinity of values



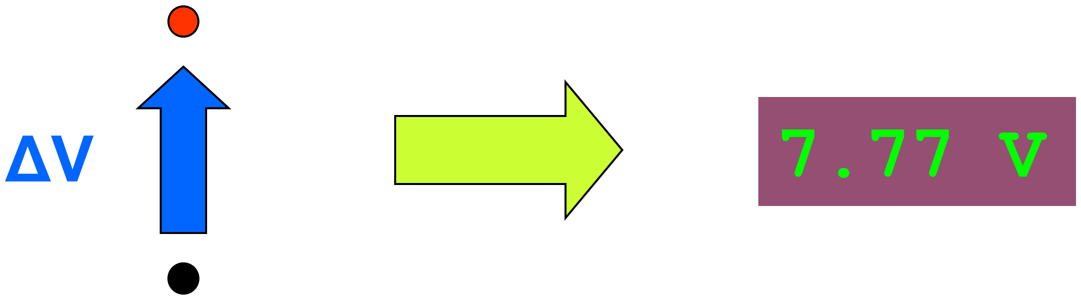
Definition

- Digital systems require discrete digital data
- ADC converts an analog information into a digital information



Examples of use

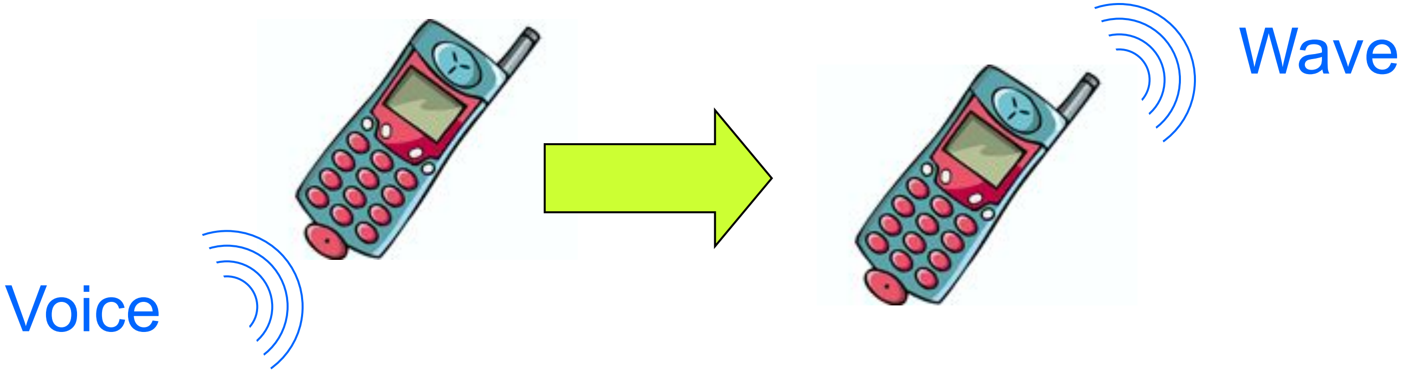
- **Voltmeter**



ΔV

7.77 V

- **Cell phone (microphone)**



Wave

Voice

Conversion process

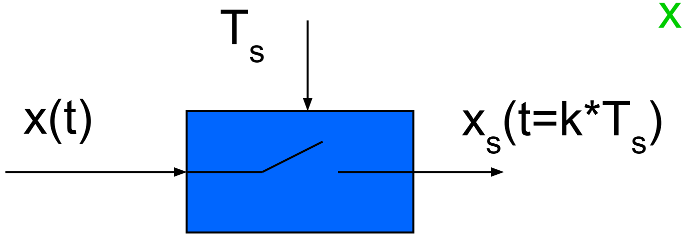
3 steps:

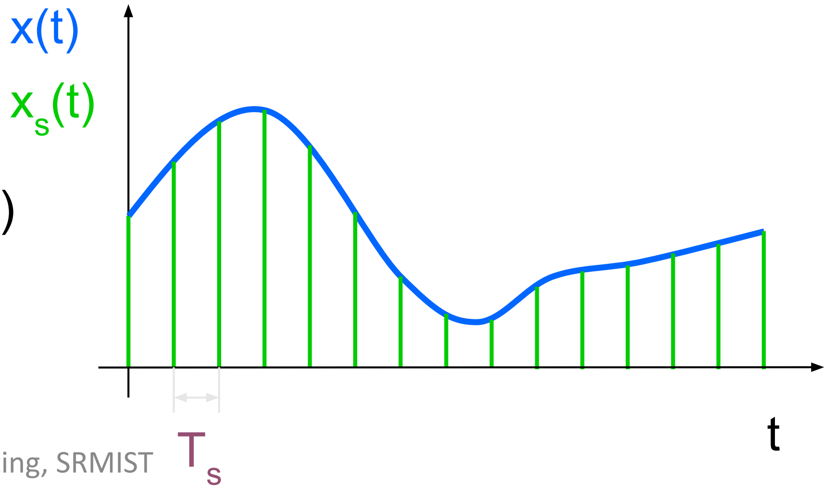
- Sampling
- Quantification
- Coding

These operations are all performed in a same element: the A to D Converter

Conversion process: Sampling

- Digital system works with discrete states
- The signal is only defined at determined times
- The sampling times are proportional to the sampling period (**T_s**)





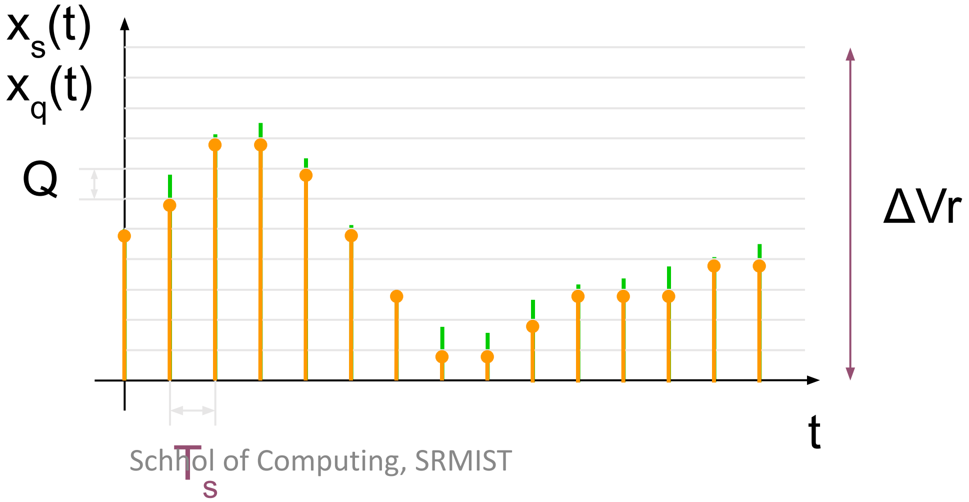
Conversion process: Quantification

The signal can only take determined values

Belonging to a range of conversion (**ΔV_r**)

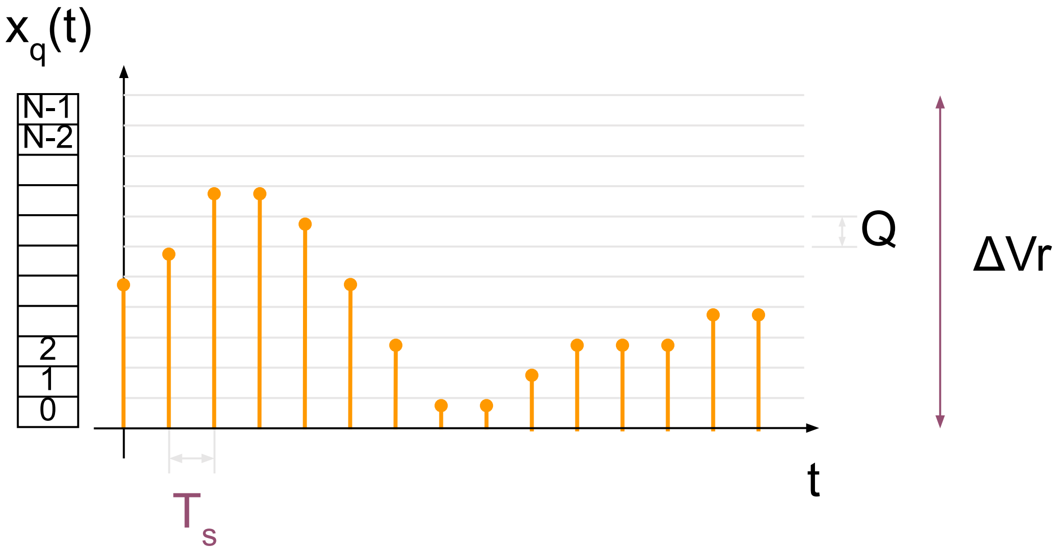
- Based on number of bit combinations that the converter can output
- Number of possible states:

  $N=2^n$ where n is number of bits
- Resolution: **$Q= \Delta V_r/N$**



Conversion process: Coding

- Assigning a unique digital word to each sample
- Matching the digital word to the input signal

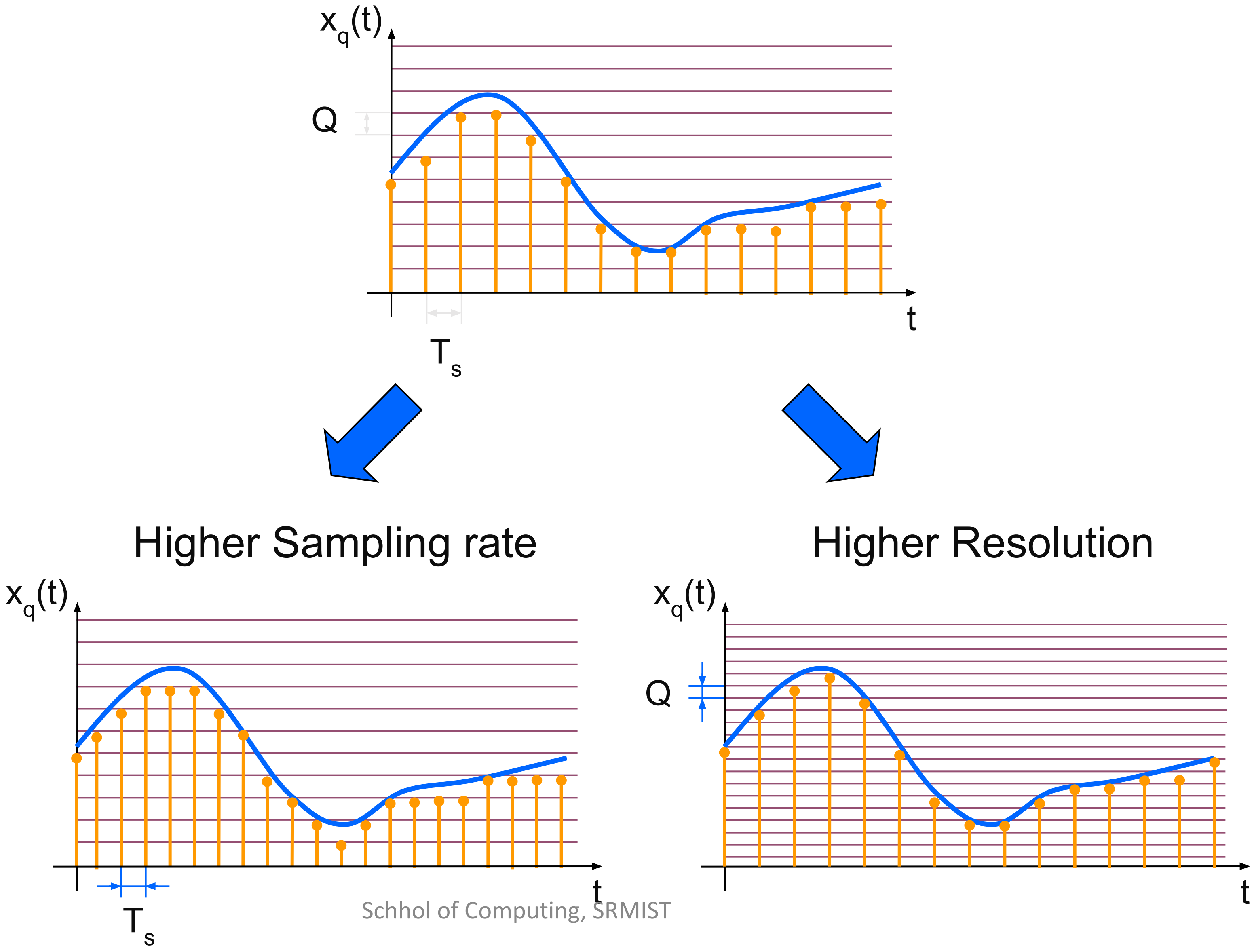


Accuracy

The accuracy of an ADC can be improved by increasing:

- The sampling rate (T_s)
- The resolution (Q)

Accuracy



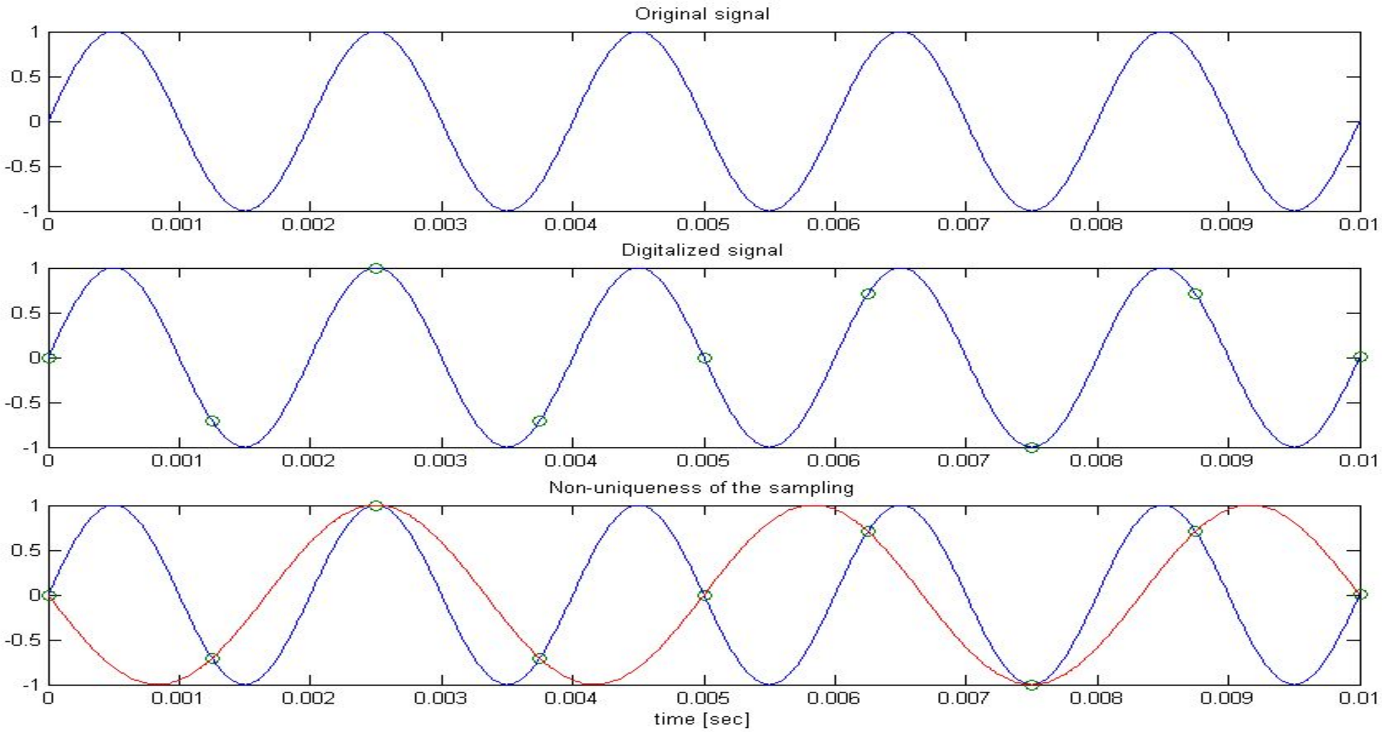
Higher Sampling rate

Higher Resolution

Sampling rate

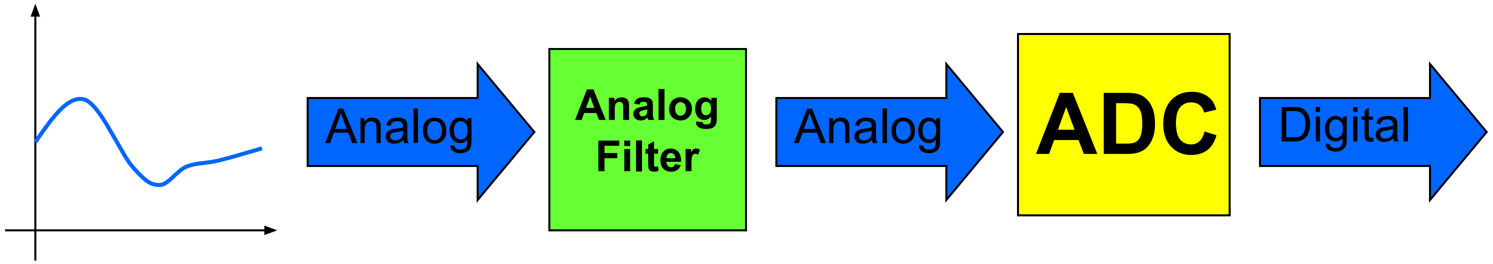
Nyquist-Shannon theorem: Minimum sampling rate should be at least twice the highest data frequency of the analog signal

$f_s>2_*f_{max}$



Sampling rate

- Analog signals are composed of an infinity of harmonics
- Need to limit the frequency band to its useful part
- Use of an analog filter

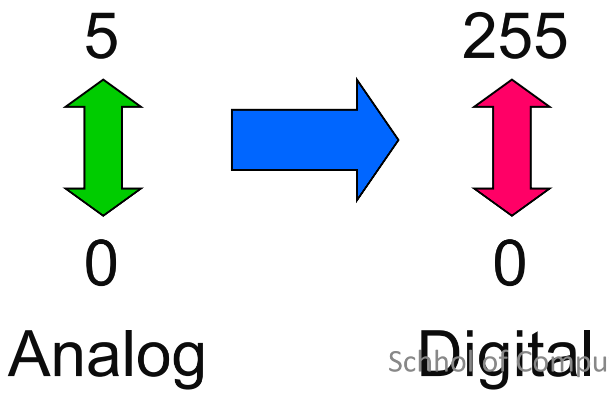


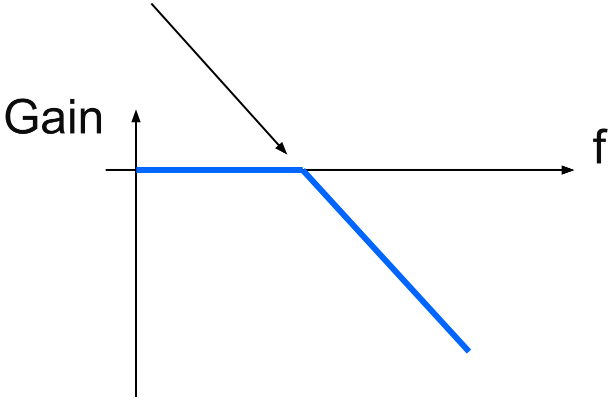
In practice: $f_s \approx (3\ldots5)_* f_{\text{filter}}$

Example

- 8 bits converter: n=8
- Range of conversion: ΔVr=5V
- Sampling time: T_s=1ms

- Number of possible states: $N=2^8=256$
- Resolution: Q=ΔVr/N=19.5 mV
- Analog Filter: $f_{filter} \approx fs/5 = 200$ Hz

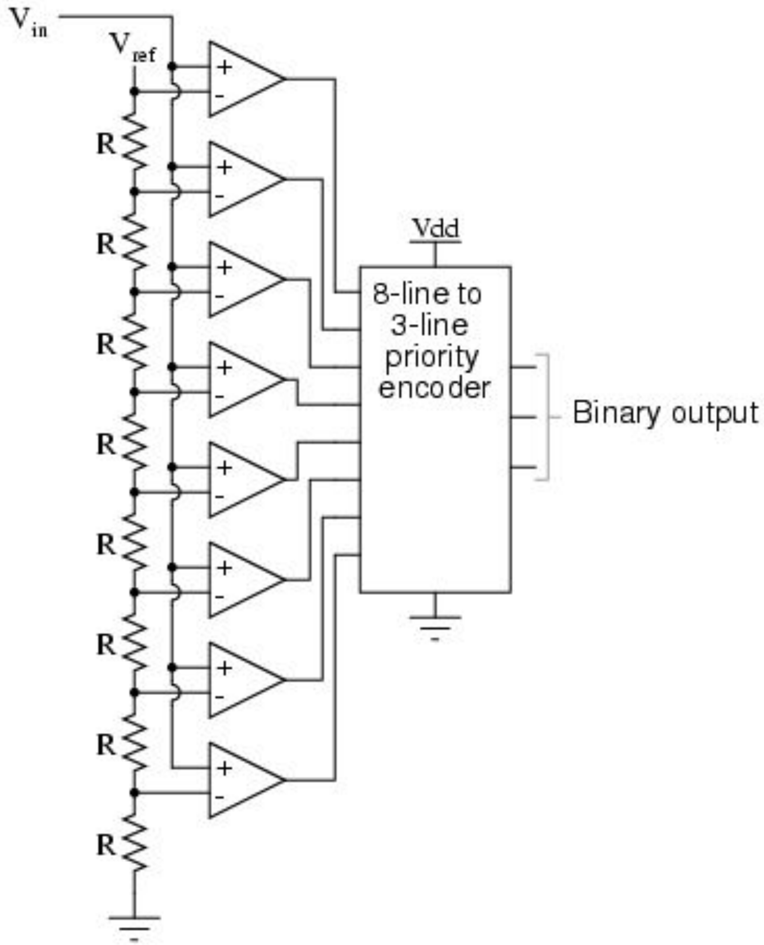




Types of ADCs

- Flash ADC
- Sigma-delta ADC
- Dual slope converter
- Successive approximation converter

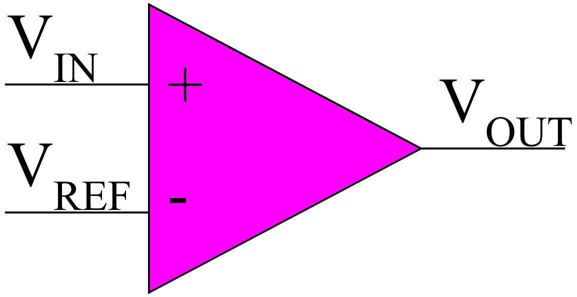
Flash ADC



- “parallel A/D”
- Uses a series of comparators
- Each comparator compares V_{in} to a different reference voltage, starting w/ V_{ref} = 1/2 lsb

Flash ADC

Comparator is one use of an Op-Amp



If	Output
$V_{IN} > V_{REF}$	High
$V_{IN} < V_{REF}$	Low

Flash ADC

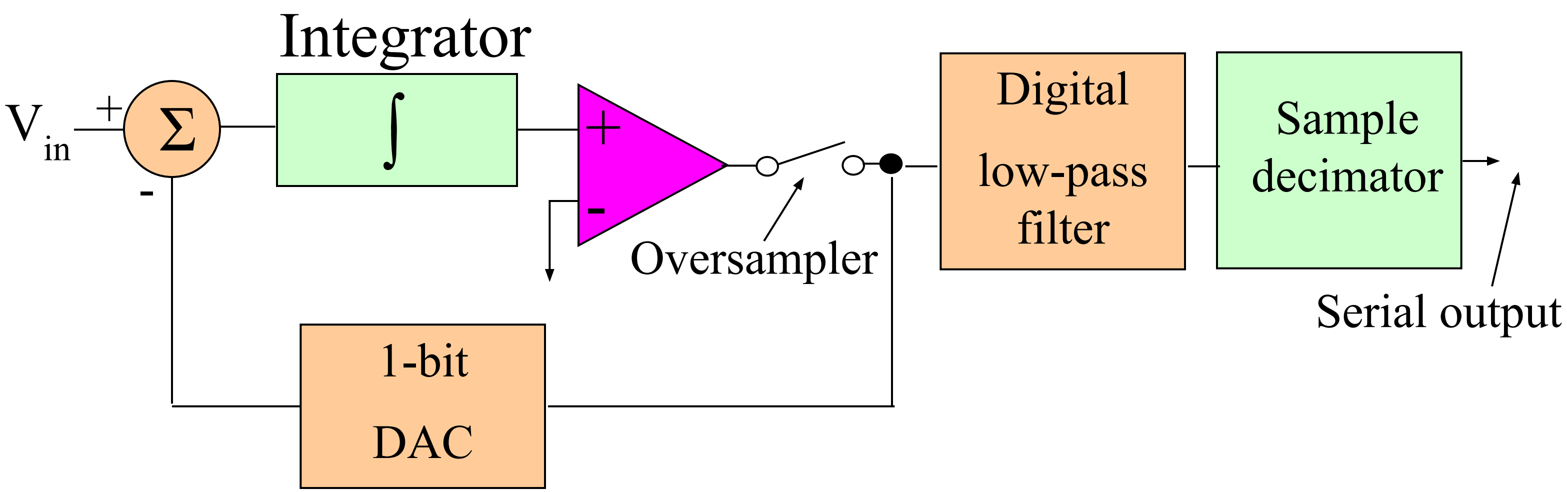
Advantages

- Very fast

Disadvantages

- Needs many parts (255 comparators for 8-bit ADC)
- Lower resolution
- Expensive
- Large power consumption

Sigma-Delta ADC



- Oversampled input signal goes in the integrator
- Output of integration is compared to GND
- Iterates to produce a serial bitstream
- Output is serial bit stream with # of 1's proportional to V_{in}

Sigma-Delta ADC

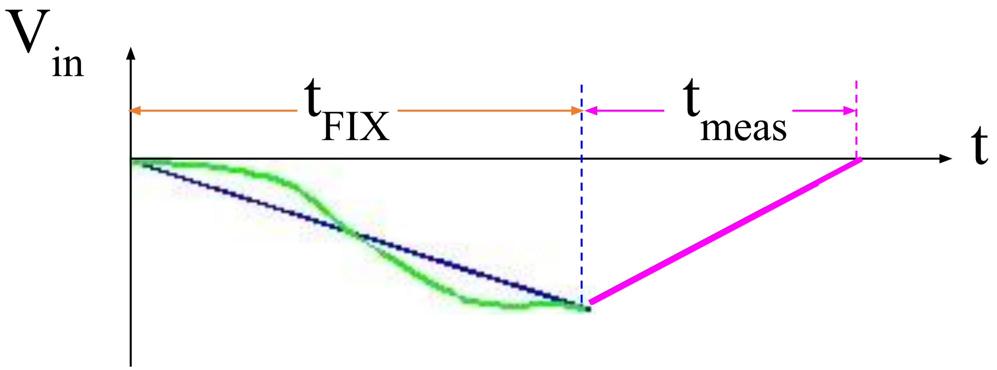
Advantages

- High resolution
- No precision external components needed

Disadvantages

- Slow due to oversampling

Dual Slope converter



- The sampled signal charges a capacitor for a fixed amount of time
- By integrating over time, noise integrates out of the conversion.
- Then the ADC discharges the capacitor at a fixed rate while a counter counts the ADC's output bits. A longer discharge time results in a higher count.

Dual Slope converter

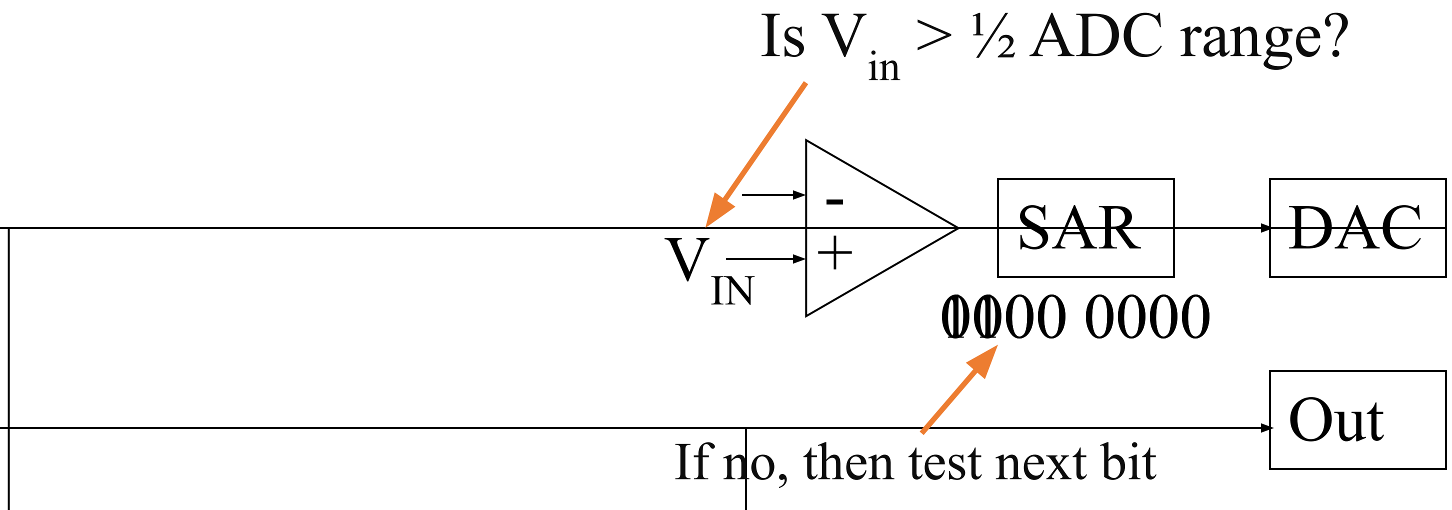
Advantages

- Input signal is averaged
- Greater noise immunity than other ADC types
- High accuracy

Disadvantages

- Slow
- High precision external components required to achieve accuracy

Successive Approximation



Is V_{in} > ½ ADC range?

V_{IN}

-

+

SAR

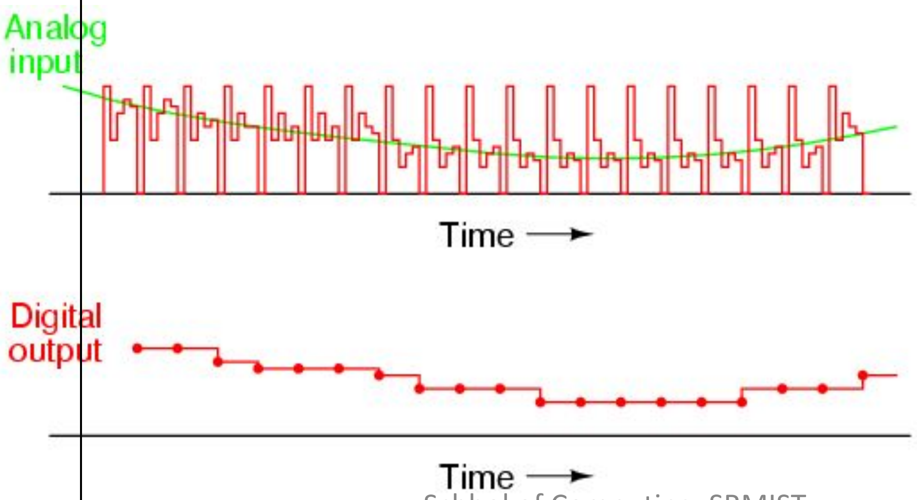
DAC

0000 0000

Out

If no, then test next bit

- Sets MSB
- Converts MSB to analog using DAC
- Compares guess to input
- Set bit
- Test next bit



Successive Approximation

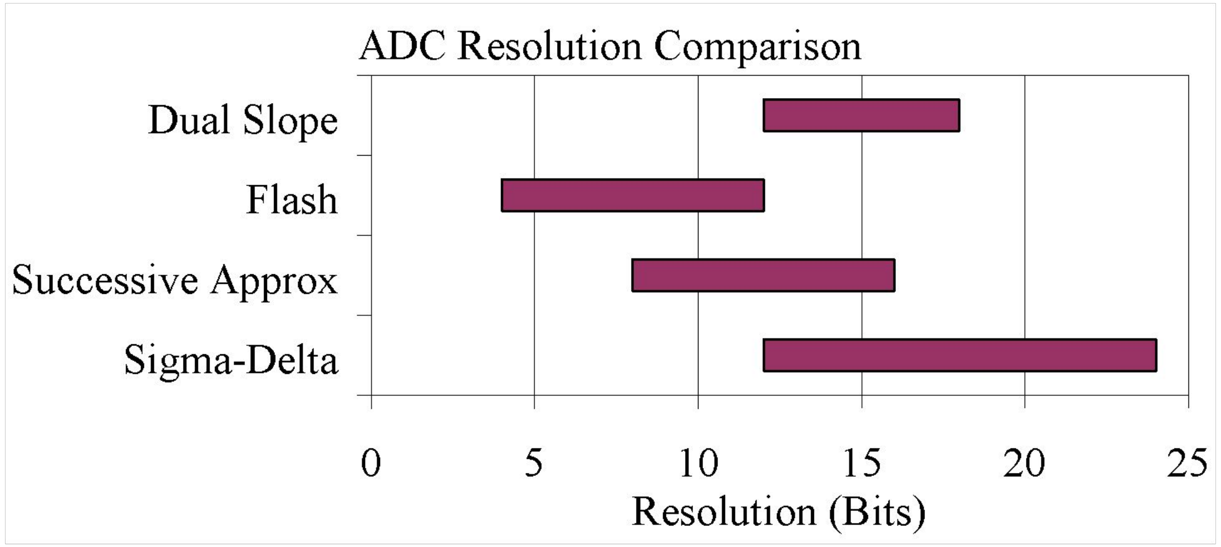
Advantages

- Capable of high speed
- Medium accuracy compared to other ADC types
- Good tradeoff between speed and cost

Disadvantages

- Higher resolution successive approximation ADCs will be slower
- Speed limited ~5Msps

ADC Types Comparison



Type	Speed (relative)	Cost (relative)
Dual Slope	Slow	Med
Flash	Very Fast	High
Successive Appox	Medium – Fast	Low
Sigma-Delta	Slow	Low

Shift Register

```
//Behavioral description of universal shift register Fig. 6-7 and Table 6-3
Module shftreg (s1,s0,Pin,lfin,rtin,A,CLK,Clr);
  input s1,s0                //Select inputs
  input lfin, rtin;               //Serial inputs
  input CLk, clr;            //Clock and Clear
  input [3:0] Pin;           //Parallel input
  output [3:0] A;            //Register output
  reg [3:0] A;
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4' b0000'
      else
        case ({s1,s0})            //No change
       2' b00: A = A;             //Shift right
       2' b01: A = {rtin,A[3:1]}; //Shift left
       2' b10: A = {A[2:0],lfin}; //Parallel load input
       2' b11: A =Pin;
     endcase
endmodule
```

Shift Register

```
//Structural description of Universal shift register(see Fig.6-7)
module SHFTREG (I,select,lfin,rtin,A,CLK,Clr);
   input [3:0] I;              //Parallel input
   input [1:0] select;         //Mode select
   input lfin,rtin,CLK,Clr;   //Serial inputs,clock,clear
   output [3:0] A;             //Parallel output
 //Instantiate the four stages
   stage ST0 (A[0],A[1],lfin,I[0],A[0],select,CLK,Clr);
   stage ST1 (A[1],A[2],A[0],I[1],A[1],select,CLK,Clr);
   stage ST2 (A[2],A[3],A[1],I[2],A[2],select,CLK,Clr);
   stage ST3 (A[3],rtin,A[2],I[3],A[3],select,CLK,Clr);
endmodule
```

Shift Register

```
//One stage of shift register
module stage(i0,i1,i2,i3,Q,select,CLK,Clr);
   input i0,i1,i2,i3,CLK,Clr;
   input [1:0] select;
   output Q;
   reg Q;
   reg D;
//4x1 multiplexer
   always @ (i0 or i1 or i2 or i3 or select)
         case (select)
            2'b00: D = i0;
            2'b01: D = i1;
            2'b10: D = i2;
            2'b11: D = i3;
         endcase
//D flip-flop
   always @ (posedge CLK or negedge Clr)
         if (~Clr) Q = 1'b0;
         else Q = D;
endmodule
```

Synchronous Counter

```
//Binary counter with parallel load See Figure 6-14 and Table 6-6
module counter (Count,Load,IN,CLK,Clr,A,CO);
  input Count,Load,CLK,Clr;
  input [3:0] IN;                //Data input
  output CO;                     //Output carry
  output [3:0] A;                //Data output
  reg [3:0] A;
  assign CO = Count & ~Load & (A == 4'b1111);
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4'b0000;
    else if (Load)  A = IN;
    else if (Count) A = A + 1'b1;
    else A = A;                  // no change, default condition
endmodule
```

Ripple Counter

```
//Ripple counter (See Fig. 6-8(b))
module ripplecounter (A0,A1,A2,A3,Count,Reset);
   output A0,A1,A2,A3;
   input Count,Reset;
//Instantiate complementing flip-flop
   CF F0 (A0,Count,Reset);
   CF F1 (A1,A0,Reset);
   CF F2 (A2,A1,Reset);
   CF F3 (A3,A2,Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
   output Q;
   input CLK,Reset;
   reg Q;
   always @ (negedge CLK or posedge Reset)
      if (Reset) Q = 1'b0;
      else  Q = #2 (~Q);     // Delay of 2 time units
endmodule
```

Ripple Counter

```
//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   ripplecounter RC (A0,A1,A2,A3,Count,Reset);
always
    #5 Count = ~Count;
initial
 begin
      Count = 1'b0;
      Reset = 1'b1;
   #4 Reset = 1'b0;
   #165 $finish;
 end
endmodule
```