Introduction to Digital Systems

Lecture 9 review: (11.1-11.3)

- Digital Building Blocks
- Logic Blocks
- Flips-Flips

Today: (12)

- Bits and bytes
- Small digital systems
- Digital signal processing
- Introduction to computer architecture

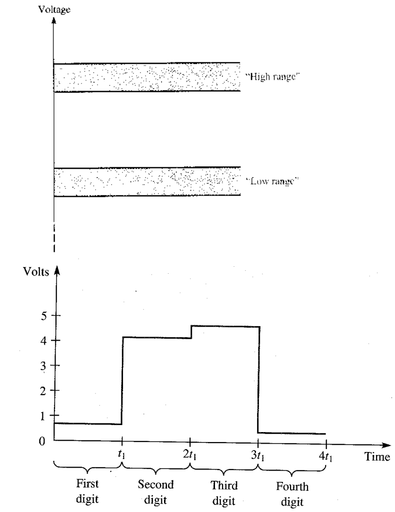
Digital Building Blocks

- Logical 1 ≡ 5V
- Logical 0 ≡ 0V

(negative logic)

- Logical 1 ≡ 0V
- Logical 0 ≡ 5V

There can be different voltage value for different digital circuit device.
Logical 1 ≡ 3.3V or
Logical 1 ≡ 2V



Digital Blocks

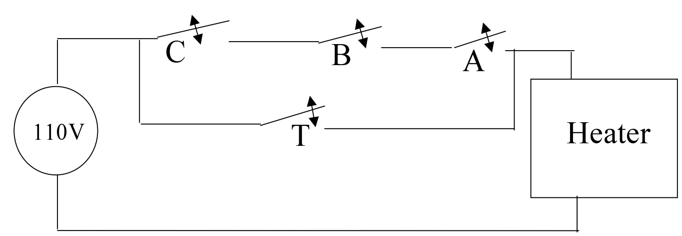
- 2 Types of Digital Blocks
  - Combination logic blocks (e.g. AND gate) chapter 11.2
  - Sequential blocks (e.g. Flips Flops) chapter 11.3
- Logic blocks have one/several inputs and one output

Digital Control Example: Hot Tub Controller

Algorithm: Turn on tub heater if Temp less than desired Temp ($T < T_{set}$) ***and*** motor is on ***and*** key switch to activate hot tub is closed. Suppose there is also a "test switch" which temporarily activates heater when depressed.

Approach: Series switches : C = key switch, B = relay closed if motor is on, A = bimetallic thermostatic switch. T = Test switch.

Simple Schematic Diagram of Possible Circuit:



Evaluation of Logical Expressions with "Truth Tables"

Truth Table for Heater Algorithm

A	B	C	T	H
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Logical Expressions to express Truth Tables

We need a notation for logic expressions.

Standard logic notation and logic gates:

AND: "dot" Examples: $X = A \cdot B$; $Y = A \cdot B \cdot C$

OR : "+ sign" Examples: $W = A+B$; $Z = A+B+C$

NOT: "bar over symbol for complement" Example: $Z = \overline{A}$

With these basic operations we can construct any logical expression.

Digital Heater Control Example (cont.)

Logical Expression : To create logical values we will define a closed switch as "True", ie boolean **1** (and thus an open switch as **0**).

Heater is on (H=**1**) if (A **and** B **and** C are **1) or** T is **1**

- Logical Statement: H = 1 if A and B and C are 1 or T is 1.
- Remember we use "dot" to designate logical "and" and "+" to designate logical "or" in switching algebra. So how can we express this as a Boolean Expression?
- Boolean Expression: $H = (A \cdot B \cdot C) + T$

The Important Logical Functions

The most frequent (i.e. important) logical functions are implemented as electronic "building blocks" or "gates".

We already know about **AND** , **OR** and **NOT** What are some others:

Combination of above: inverted AND = **NAND,**
inverted OR **= NOR**

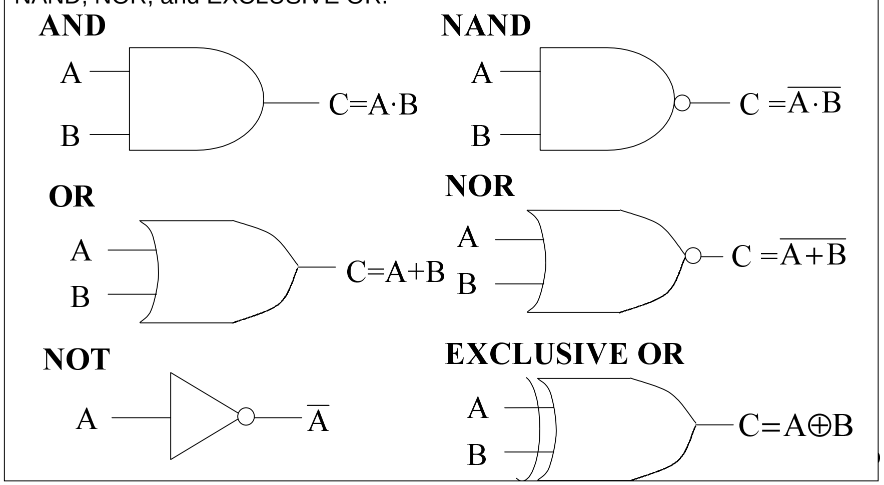
And one other basic function is often used: the "EXCLUSIVE OR" … which logically is "or except not and"

Some Important Logical Functions

- **"AND"** $A\cdot B$ (or $A\cdot B\cdot C$)
- **"OR"** $A+B$ (or $A+B+C+D\ldots$)
- **"INVERT" or "NOT"** $\overline{A}$ = not A or A − inverted
- **"not AND" = NAND** $\overline{AB}$ (only 0 when A and $B=1$)
- **"not OR" = NOR** $\overline{A+B}$ (only 1 when $A=B=0$)
- **exclusive OR = XOR** $A\oplus B$ (only 1 when A, B differ)
  i.e., $A+B$ **except** $A\cdot B$

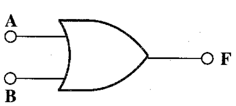
Logic Gates

These are circuits that accomplish a given logic function such as "OR". We will shortly see how such circuits are constructed. Each of the basic logic gates has a unique symbol, and there are several additional logic gates that are regarded as important enough to have their own symbol. The full set is: AND, OR, NOT, NAND, NOR, and EXCLUSIVE OR.



The basic gates in Digital Electronics

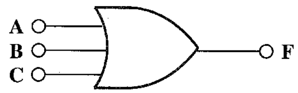
2 input OR gate. 3 input OR gate



(a)

Inputs A	B	Output F
0	0	0
0	1	1
1	0	1
1	1	1

(b)



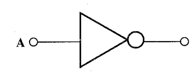
(a)

Input A	Input B	Input C	Output F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b)

The basic gates in Digital Electronics

•NOT/INVERSE/COMPLEMENT

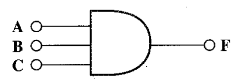


(a)

Input A	Output F
0	1
1	0

(b)

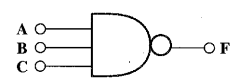
- 3 input AND gate



(a)

Input A	Input B	Input C	Output F
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

•3 input NAND (NOT-AND) gate

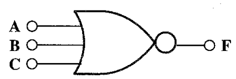


(a)

Input A	Input B	Input C	Output F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

The basic gates in Digital Electronics

- NOR (NOT-OR)



(a)

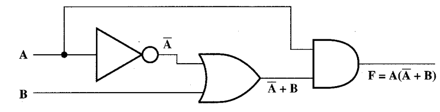
Input A	Input B	Input C	Output F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(b)

The basic gates in Digital Electronics

- F = A + B A OR B
- F = A • B A AND B
- $F = \overline{A+B}$ NOT (A OR B) alternative notation: (A+B)’
- $F = \overline{A \bullet B}$ NOT (A AND B) (A • B)’

Given $F = A(\overline{A}+B)$
+ draw the logic blocks
+ write the truth table



(a)

A	B	$\overline{A}$	$\overline{A}+B$	F
0	0	1	1	0
0	1	1	1	0
1	0	0	0	0
1	1	0	1	1

(b)

- 3 inputs: NOT A, A, B
- F1 = (NOT A) OR B
- F = A AND F1

Logic Synthesis

- So far, when given a logic blocks → generate the truth table

Can we reverse the process? Logic synthesis

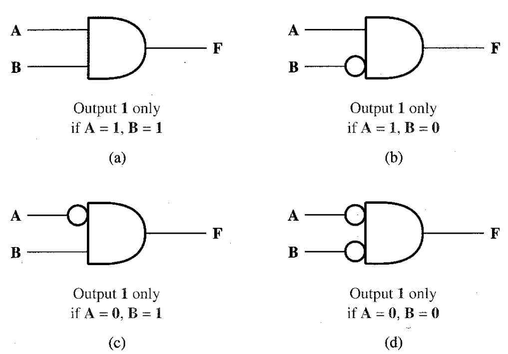
- We need a certain truth table → Find the logic blocks to implement it
- For example, a 2 inputs (A, B) and 1 output (F)

$F = AB$

$F = A\overline{B}$

$F = \overline{A}B$

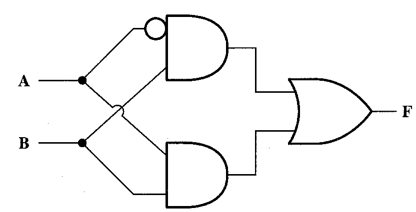
$F = \overline{AB}$



Logic Synthesis

- Next, we can combine these blocks AB, AB’, A’B, (AB)’ through an “OR” gate → AB + AB’…
- This is called the Sum-of-products method
- Example F = A’B + AB can be implemented by the following:

A	B	F
0	0	0
0	1	1
1	0	0
1	1	1



- On the other hand, F = A’B + AB can be simplified into F = (A’ + A) B = (1) B = B

Logic Synthesis (example)

- Use logic blocks to perform binary arithmetic (half adder):
- 2 binary inputs A and B, the result is 2 bits CD

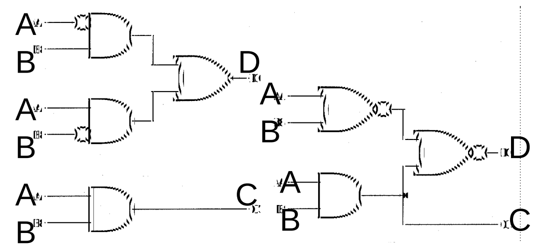
```
          A
+         B
-------------
        C D
```

The truth table is:

Input		Output	
A	B	C	D
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logic Synthesis (example)

- Using **Sum-of-product** method:
- We only tackle the C and D with a “one” in the truth table:
  - for output D = 1 (second row in the table), A = 0, B = 1 $\rightarrow F_1 = A'B$
  - for output D = 1 (third row in the table), A = 1, B = 0 $\rightarrow F_2 = AB'$
  - for output C to be equal to 1, A=1, B=1 $\rightarrow F_3 = AB$
- These 3 entries can be “realized” by the following:



Practical logic blocks

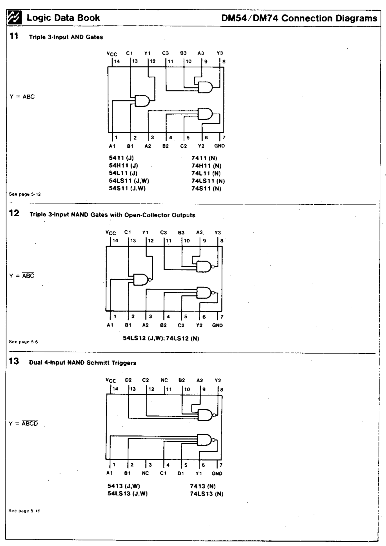


FIGURE 11.18(a) Typical SSI logic packages. (*Courtesy of National Semiconductor Corp.*)

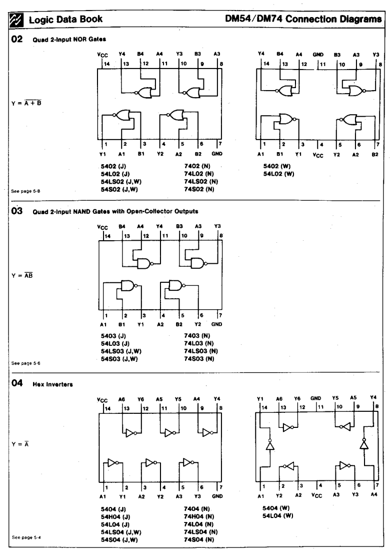


FIGURE 11.18(b)

Practical logic blocks: Electrical Characteristics

Electrical Characteristics over recommended operating free-air temperature range (unless otherwise noted).

Symbol	Parameter	Conditions			DM54/74 00, 04 10, 20, 30			DM54/74 H00, H04 H10, H20, H30			DM54/74 L00, L04 L10, L20, L30			DM54/74 LS00, LS04 LS10, LS20, LS30			DM54/74 S00, S04, S10 S20, S30, S133			Units
					Min	Typ (1)	Max	Min	Typ (1)	Max	Min	Typ (1)	Max	Min	Typ (1)	Max	Min	Typ (1)	Max	
V_{IH}	High Level Input Voltage				2			2			2			2			2			V
V_{IL}	Low Level Input Voltage			DM54			0.8			0.8			0.7			0.8			0.8	V
				DM74			0.8			0.8			0.7			0.8			0.8	
V_I	Input Clamp Voltage	V_{CC} = Min	$I_I = -8$ mA							−1.5			N/A							V
			$I_I = -12$ mA				−1.5						N/A							
			$I_I = -18$ mA										N/A			−1.5			−1.2	
I_{OH}	High Level Output Current						−400			−500			−200			−400			−1000	µA
V_{OH}	High Level Output Voltage	V_{CC} = Min, V_{IL} = Max I_{OH} = Max		DM54	2.4	3.4		2.4	3.5		2.4	3.3		2.5	3.4		2.5	3.4		V
				DM74	2.4	3.4		2.4	3.5		2.4	3.2		2.7	3.4		2.7	3.4		
I_{OL}	Low Level Output Current			DM54			16			20			2			4			20	mA
				DM74			16			20			3.6			8			20	
V_{OL}	Low Level Output Voltage	V_{CC} = Min V_{IH} = 2 V	I_{OL} = Max	DM54		0.2	0.4		0.2	0.4		0.15	0.3		0.25	0.4			0.5	V
				DM74		0.2	0.4		0.2	0.4		0.2	0.4		0.35	0.5			0.5	
			I_{OL} = 4 mA	DM74												0.4				
I_I	Input Current at Maximum Input Voltage	V_{CC} = Max	V_I = 5.5 V				1			1			0.1						1	mA
			V_I = 7 V													0.1				
I_{IH}	High Level Input Current	V_{CC} = Max	V_I = 2.4 V				40			50			10							µA
			V_I = 2.7 V													20			50	
I_{IL}	Low Level Input Current	V_{CC} = Max	V_I = 0.3 V										−0.18							mA
			V_I = 0.4 V	LS30												−0.4				
				Others			−1.6			−2						−0.36				
			V_I = 0.5 V																−2	
I_{OS}	Short Circuit Output Current	V_{CC} = Max (2)		DM54	−20		−55	−40		−100	−3		−15	−20		−100	−40		−100	mA
				DM74	−18		−55	−40		−100	−3		−15	−20		−100	−40		−100	
I_{CC}	Supply Current	V_{CC} = Max			See Table															

Note 1: All typical values are at V_{CC} = 5 V, T_A = 25°C.
Note 2: Not more than one output should be shorted at a time, and for DM54H/DM74H, DM54LS/DM74LS and DM54S/DM74S, duration of short circuit should not exceed one second.

SSI DM54/DM7400, 04, 10, 20, 30, S133 NAND Gates/Inverters

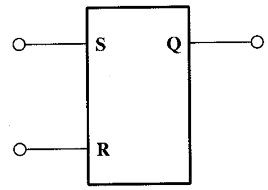
Flips Flops (Sequential Blocks)

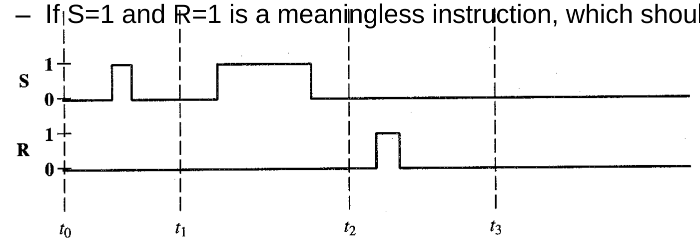
- Output of logic blocks depend on input at any instant*
- Output of sequential blocks depend on the previous input as well as the current input (in other word, it has “memory” effect)

3 types of Flips Flops: S-R Flip Flops, D Flip Flops, J-K Flip Flops

S-R Flips Flops

- Rule 1
  - If S=0 and R=0, Q does not change.
- Rule 2
  - If S=0 and R=1, then regardless of past history Q=0
- Rule 3
  - If S=1 and R=0, then regardless of past history Q=1
- Rule 4
  - If S=1 and R=1 is a meaningless instruction, which should not be used.

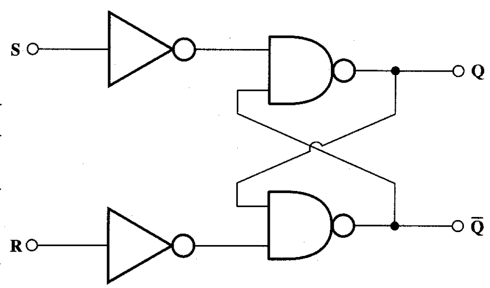




Realization of S-R Flips Flops

- Truth Table

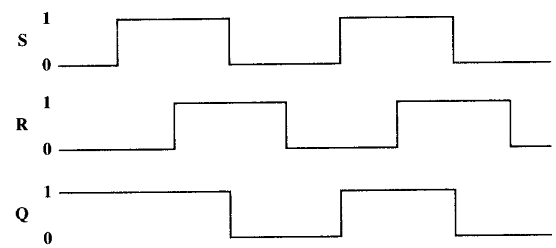
S	R	Q (guess)	$\overline{Q}$ (guess)	Q	$\overline{Q}$
0	0	1	0	1	0
0	0	0	1	0	1
~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~
1	0	1	0	1	0
0	1	0	1	0	1
1	1	1	1	1	1



•Why do we need to guess Q and Q’? The reason is that FF needs more than the current inputs, different current state of Q and Q’ will generate different Q, Q’ output. And we don’t know what is the current Q and Q’, so we can guess it in conjunction with the inputs to generate a truth table.
•S = 0 and R = 0 → no change in Q and Q’
•S = 1 → Q = 1, Q’ = 0
•R = 1 → Q = 0, Q’ = 1
•S = 1 and R = 1 → Q = 1, Q’ = 1 (don’t use)

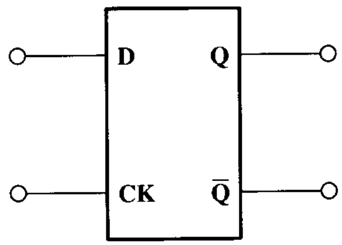
S-R Flips Flops (example)

- S = 0 and R = 0 → no change in Q and Q’
- S = 1 → Q = 1, Q’ = 0
- R = 1 → Q = 0, Q’ = 1
- S = 1 and R = 1 → Q = 1, Q’ = 1 (don’t use)



D Flips Flops

- Clocked Flip Flops
  - Flip Flops does not change state until an instruction is given thru the clock (CK)

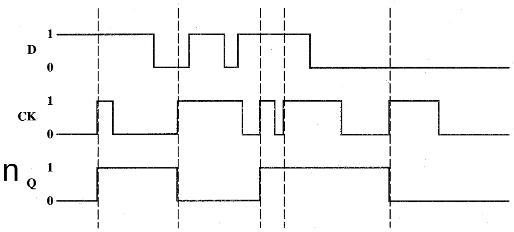


•Truth Table of D Flip Flops

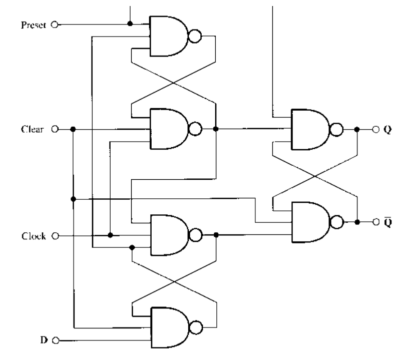
D	Q (before)	Q (after)
0	0	0
0	1	0
1	0	1
1	1	1

D Flips Flops (example & realization)

Propagation delay - a small delay between the change CLK and the time Q actually changed. (this is not shown in the timing diagram)



D Flip Flops realized using 6 NAND gates



16M RAM contains $16\times1024\times1024$ flip flops

Summary

- Digital blocks
  - Combinational blocks
    - output of a logic blocks depends on the inputs at any instant of time
    - Examples are: AND, OR, inverter (COMPLEMENT), NAND, NOR, XOR (exclusive or)
  - Sequential blocks
    - Sequential blocks remember inputs applied at earlier time
    - Examples are: S-R flip flops, D flip flops, J-K flip flops…
- Mathematical expressions for logical operations are written using Boolean algebra.
- The logic blocks is specified by a table showing the outputs that result from various combination of input. This table is known as the truth table.
- Any truth table can be realize by appropriate connection of inverters, AND gates, and one OR gate, using the sum-of-product method. However, this may not be the simplest realization of the truth table.
- Different flip flops exist: S-R FF, D FF, differing in ways instruction for storing information are applied. S-R FF can be built by 2 NAND gates, D clocked FF is built by more logic gates, but are simpler to use.

So Why Digital?

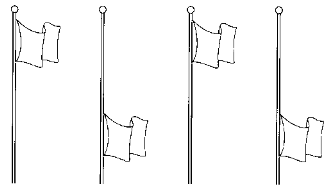
(For example, why CDROM audio vs vinyl recordings?)

- Digital signals can be transmitted, received, amplified, and re-transmitted with no degradation.
- Binary numbers are a natural method of expressing logical variables.
- Complex logical functions are easily expressed as binary functions (e.g., in control applications … see next page).
- Digital signals are easy to manipulate (as we shall see).
- With digital representation, we can achieve arbitrary levels of "dynamic range," that is, the ratio of the largest possible signal to the smallest than can be distinguished above the background noise
- Digital information is easily and inexpensively stored (in RAM, ROM, EPROM, etc.), again with arbitrary accuracy.

How do you represent number digitally?

10V in decimal becomes:

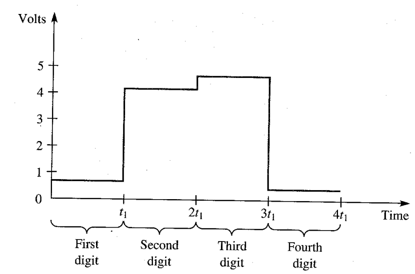
- "13" in base 7
- "14" in base 6
- "22" in base 4
- "1010" in Binary (base 2)

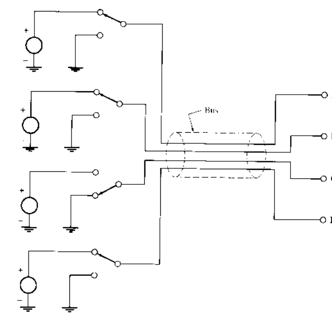


- A binary digit (bit) can store 0 or 1 and can be stored in a flip flop
- 4-bit can store number between 0000 to 1111 (16 different numbers)
- N-bit can store 2^N different numbers. (8 bit can store 256, 10 bit → 1024, 15 bit → 32768 ($2^5 \times 1024$)...

Binary transmission

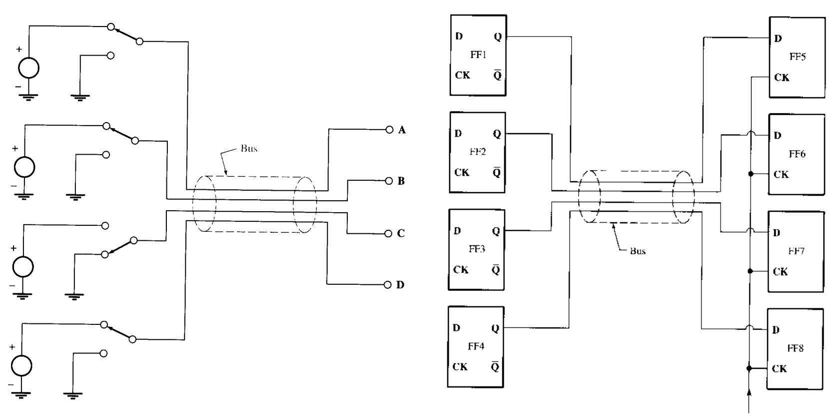
- Suppose we need to transmit 4 bit in binary.
- We can transmit 4 bit **serially** (one bit at a time) or **simultaneously** (four bit together)





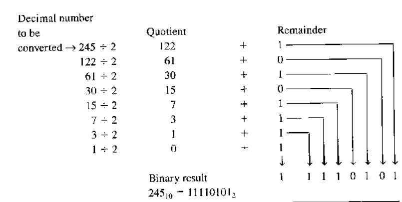
4-bit flip flops or registers

- 4 bits are transmitted simultaneously, but 4-bit byte is transmitted serially.
- FF5-FF8 are called the storage register or data latch.



Decimal, binary, and hexadecimal number

- Convert from decimal to binary
- $245_{10} = 11110101_2$



- Convert from binary to decimal

$11110101_2 = 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times 2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 128 + 64 + 32 + 16 + 0 + 4 + 0 + 1$

Decimal, binary, and hexadecimal number

- It is clumsy to write 11110101_2
- Can change it into hexadecimal (base 16)
- 11110101_2 = ? $_{16}$
- Convert from hexadecimal to binary?
- Convert from hexadecimal to decimal?
  - AEC_{16} = ? $_{10}$
  - AEC_{16} = ? $_2$
- Convert from decimal to hexadecimal?
  - 245_{10} = ? $_{16}$

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
⋮	⋮	⋮

Decimal, binary, and hexadecimal number

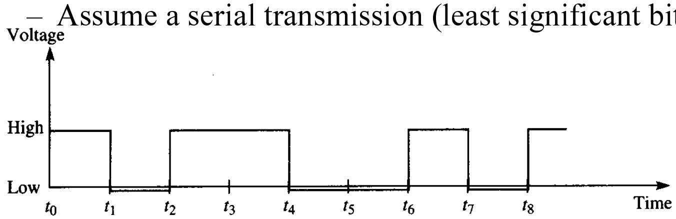
- Multiplication ?
- $9C_{16} \times A3_{16}$ =?

$9C_{16}$ = 1001 1100
$A3_{16}$ = 1010 0011

1001 1100
1 0011 100
1 0011 100
100 1110 0
110|0011|0101|0100
6 3 5 4 = 6354_{16}

Small digital systems

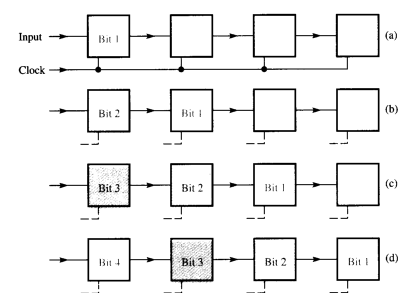
- Shift registers
  - Assume a serial transmission (least significant bit first)



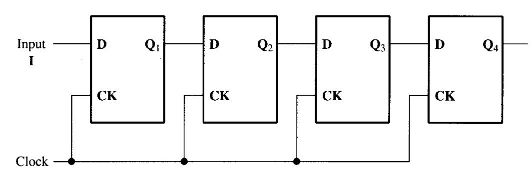
Represents 01001101 or $4D_{16}$
We need to look at the 8 bit in order to something about it.
What we need is shift registers.

Small digital systems (shift registers)

- Bit is shift to the right 1 clock cycle at a time

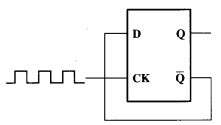


- A 4-bit shift register can be implemented by 4 D flips flops connected serially -- output Q1 of the 1st FF is connected to the input D2 of the 2nd FF.



Counter

1-bit counter



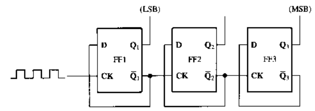
(a)

N	Q_N	D_N	Q_{N+1}
0	0	1	1
1	1	0	0
2	0	1	1
3	1	0	0
4	0	1	1

(b)

(c)

- 3-bit ripple counter



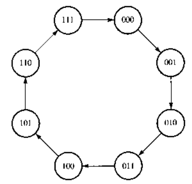
(a)

State table

N	Q_{1N}	Q_{2N}	Q_{3N}	$(Q_3Q_2Q_1)_N$
0	0	0	0	000
1	1	0	0	001
2	0	1	0	010
3	1	1	0	011
4	0	0	1	100
5	1	0	1	101
6	0	1	1	110
7	1	1	1	111
8	0	0	0	000

(b)

State diagram



(c)

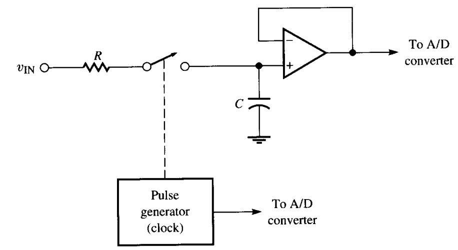
Digital Signal Processing

- Analog signal → digital signal

(need Analog to Digital converter or A/D converter)

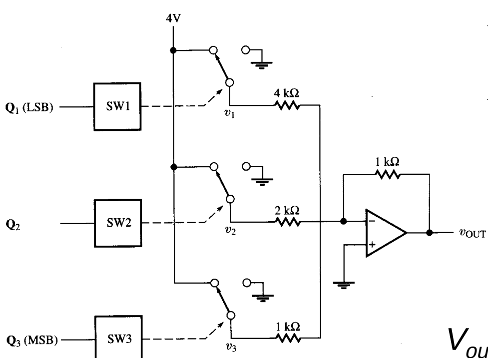
- Sampling
- Nyquist sampling theorem

$T_s < 1/(2f_{max})$



D/A converters

- 3 bit digital to analog (D/A) converter



$$\frac{v_{out}}{1000} + \frac{v_1}{4000} + \frac{v_2}{2000} + \frac{v_3}{1000} = 0$$

$$v_{out} = -\frac{1}{4}[v_1 + 2v_2 + 4v_3]$$

When Q_1=1, v_1=4v
else v_1=0v

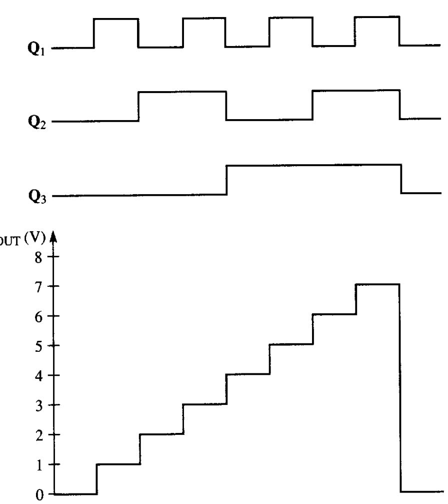
$$v_{out} = -[Q_1 + 2Q_2 + 4Q_3]$$

V_{out} is proportional to the values of Q_1, Q_2, Q_3 ...

D/A converters (continue)

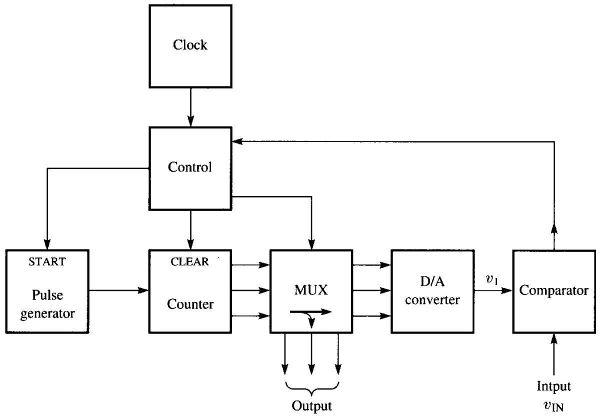
$$v_{out} = -[Q_1 + 2Q_2 + 4Q_3]$$

- If Q_1, Q_2, Q_3 are:
- Then V_{out} *will be*:



Analog to Digital (A/D) converters

- Clock
- Counter
- D/A convert
- Comparator
- Control



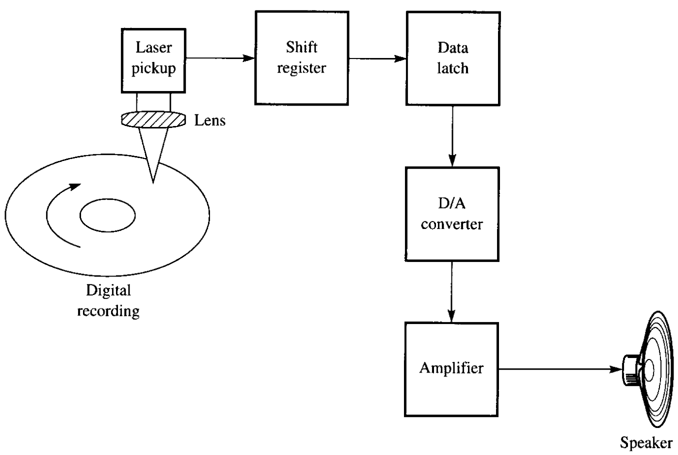
Percentage quantization error = $100 \times 1/(2^N-1)$, for a N-bit system
Example: 16 bit CD system has a quantization error of
0.001526% ≡ Signal/Noise ratio of = 20*log(error) = 96.3 dB

A/D converters (example)

- Suppose the speed of an 8-bit A/D converter is limited by the counter, which has a maximum speed of 4×10^7 count per second. Estimate the maximum number of A/D conversions per second that can be obtained.

Maximum speed (counter) of 4×10^7 count per second
Maximum number of count per signal (worse case) is 2^8-1=255 count
Minimum time per count = $255/(4\times10^7)$ = 6.375 µsec
Maximum number of conversion is the reciprocal of that
= 1/6.375 µsec) = 157,000 time per second.

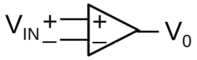
Compact Disc



Summary

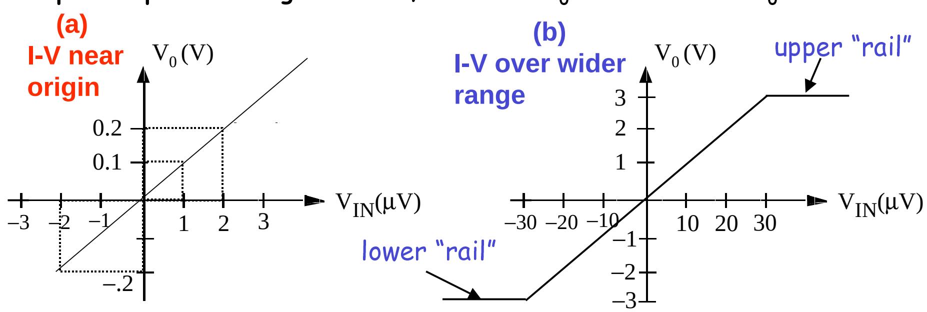
- A register is a set of N FF, which can store N-bit of info. 4-bit bytes are conveniently represented by hexadecimal digit. A number can be represented by binary, decimal, or hexadecimal form.
- A shift register is a small system that collects bytes of data when the bits are arriving serially.
- Counters are small systems that move thru a cycle of states, moving 1 step for each input.
- Usefully tool for analyzing sequential system are timing diagrams, states tables and state diagrams.
- Analog to digital and digital to analog converters are used as interfaces between analog and digital system.

WHAT ARE I-V CHARACTERISTICS OF AN ACTUAL HIGH-GAIN DIFFERENTIAL AMPLIFIER ? (optional)



- Circuit model gives the essential linear part
- But V_0 cannot rise above some physical voltage related to the positive power supply V_{CC} (" upper rail") $V_0 < V_{+RAIL}$
- And V_0 cannot go below most negative power supply, V_{EE} i.e., limited by lower "rail" $V_0 > V_{-RAIL}$

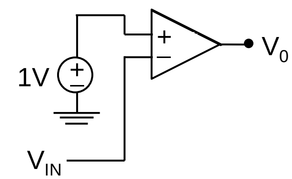
Example: Amplifier with gain of 10^5, with max V_0 of 3V and min V_0 of −3V.



OP-AMPS AND COMPARATORS (optional)

A very high-gain differential amplifier can function **either** in extremely linear fashion as a very nonlinear device – a comparator.

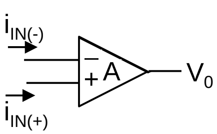
1V Comparator



Open loop or Positive Feedback: Comparator

EASY WAY TO GET ANSWER FOR OP-AMP CIRCUITS (optional)

"Ideal Op-Amp Technique":



(1) $V_+ \cong V_-$ Why? V_0 CANNOT → ∞, BUT A →∞ ⇒ $V_+ \to V_{(-)}$ in order that $V_0 = A\,(V_+ - V_-)$

Finite infinite must be zero

(2) $i_{IN+} \approx 0$, $i_{IN-} \approx 0$ Why? (a) R_{IN} large by design

(b) $V_+ \to V_{(-)} \Rightarrow$ voltage difference across $R_{IN} \to 0$

With ideal op-amp technique we can analyze all sorts of negative feedback circuits – see reader

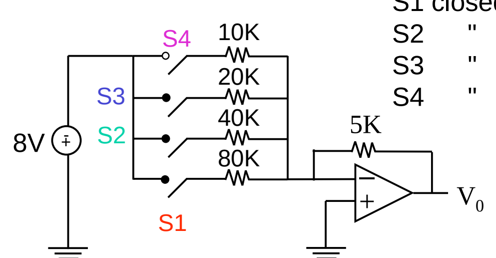
Examples: (1) unity gain follower
(2) amplifiers
(3) summing amplifiers

MULTI-BIT A/D AND D/A CONVERSION

Example: Digital representation of sound to analog (so you can hear it!) → D/A conversion

The summing junction op-amp provides a simple means of D/A conversion via **weighted-adder D/A converter**

S1 closed if LSB =1
S2 " if next bit = 1
S3 " if " " = 1
S4 " if MSB = 1



4-Bit D/A

Binary number	Analog output (volts)
0 0 0 0	0
0 0 0 1	.5
0 0 1 0	1
0 0 1 1	1.5
0 1 0 0	2
0 1 0 1	2.5
0 1 1 0	3
0 1 1 1	3.5
1 0 0 0	4
1 0 0 1	4.5
1 0 1 0	5
1 0 1 1	5.5
1 1 0 0	6
1 1 0 1	6.5
1 1 1 0	7
1 1 1 1	7.5

MSB LSB

Another circuit to do this is shown in reader (R-2R resistive ladder D/A converter

Yet another way (not shown) is to sum **charges** instead of current with capacitor networks