Basic Experiment and Design of Electronics

Logic Circuits

Ho Kyung Kim, Ph.D.
hokyung@pusan.ac.kr
School of Mechanical Engineering
Pusan National University
Digital IC packages

- TTL (transistor-transistor logic)
  - High-power consumption, fast
  - 74 series
- CMOS (complementary metal-oxide-semiconductor)
  - Low-power consumption, slow
  - Weak to static
  - 40 series

![DIP (dual-in-line) package](image1)
![Flat-type](image2)
![Surface-mount package](image3)

7400
패키지 형식
N: Plastic DIP
J: Ceramic DIP
W: Flat Pack

기능에 따른 고유 번호

회로 타입
S: high speed Schottky
L: Low power
LS: Low power Schottky
H: High speed
F: Fast
HC: High speed CMOS(CMOS compatible)
HCT: High speed CMOS TTL(LSTTL compatible)
AC: Advanced
AS: Advanced Schottky
ALS: Advanced Low power Schottky

시리즈명
74: TTL
40: CMOS

제조회사
SN: Texas Instrument
MC: Motorola
DM: National Semiconductor
IM: Intersil
N: Signetics
MM: Monolithic Memories
P: Intel
H: Harris
F: Fairchild
AM: Advanced Micro Devices
CD: RCA
HD: Hitach
DN/MN: Mitsubishi
MB: Fujitsu
TC: Toshiba
HY: Hyundai
GD: GoldStar
K: Samsung
Outline

• Combinational logic circuits
  – Output depends on only the *present* inputs; not on the past inputs
  – Multiplex
  – ROM
  – Decoder
  – RAM
  – PLD

• Sequential logic circuits
  – Output depends on both the *present* and *past* inputs; hence having “memory” function
  – Flip-flops
  – Counters
Combinational logic circuits (modules)

- Multiplex
- ROM
- Decoder
- RAM
- PLD
Calculator

Key pad

Input

Decimal

Encoder

4 bits

BCD

CPU

4 bits

BCD

Decoder

7 bits

Output

7-segment display
• Half adder (HA)
  – 2 inputs: $X$ and $Y$
  – 3 outputs: $S$ (sum, LSB) and $C_{OUT}$ (carry, MSB)

$$C_{OUT} = XY$$

$$S = \bar{X}Y + X\bar{Y} = X \oplus Y$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$C_{OUT}$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
- Full adder (FA)
  - 3 inputs: $X$, $Y$, and $C_{IN}$
  - 2 outputs: $S$ and $C_{OUT}$

$$S = \overline{X}YC_{IN} + \overline{X}Y\overline{C}_{IN} + X\overline{Y}C_{IN} + XYC_{IN}$$
$$= \overline{X}(\overline{Y}C_{IN} + \overline{Y}\overline{C}_{IN}) + X(\overline{Y}C_{IN} + YC_{IN})$$
$$= \overline{X}(Y \oplus C_{IN}) + X(Y \oplus C_{IN})$$
$$= \overline{X} \cdot Z + X \cdot \overline{Z} = X \oplus Z$$
$$= X \oplus Y \oplus C_{IN}$$

$$C_{OUT} = \overline{XY}C_{IN} + XY\overline{C}_{IN} + XY\overline{C}_{IN} + XYC_{IN}$$
$$= \overline{Y}XYC_{IN} + X\overline{Y}C_{IN} + X\overline{Y}C_{IN} + XYC_{IN} + XY\overline{C}_{IN} + XYC_{IN}$$
$$= YC_{IN}(\overline{X} + X) + XC_{IN}(\overline{Y} + Y) + XY(\overline{C}_{IN} + C_{IN})$$
$$= XY + X\overline{C}_{IN} + YC_{IN}$$
$$= XY + C_{IN}(X + Y)$$
$$= XY + C_{IN}[X(Y + \overline{Y}) + Y(X + \overline{X})]$$
$$= XY(1 + C_{IN}) + C_{IN}(X\overline{Y} + \overline{X}Y)$$
$$= XY + C_{IN}(X \oplus Y)$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$C_{IN}$</th>
<th>$C_{OUT}$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
For the output $S$:

\[
\begin{array}{c|cccc}
\text{X} & \text{00} & \text{01} & \text{11} & \text{10} \\
\hline
\text{0} & 0 & 1 & 0 & 1 \\
\text{1} & 1 & 0 & 1 & 0 \\
\end{array}
\]

\[
S = \overline{X}YC_{IN} + \overline{Y}C_{IN} + \overline{X}C_{IN} + XYC_{IN} = X \oplus Y \oplus C_{IN}
\]

For the output $C_{OUT}$:

\[
\begin{array}{c|cccc}
\text{X} & \text{00} & \text{01} & \text{11} & \text{10} \\
\hline
\text{0} & 0 & 0 & 1 & 0 \\
\text{1} & 0 & 1 & 1 & 1 \\
\end{array}
\]

\[
C_{OUT} = XY + XC_{IN} + YC_{IN} \\
= XY + C_{IN}(X + Y) \\
= XY + C_{IN}[X(Y + \overline{Y}) + Y(X + \overline{X})] \\
= XY(1 + C_{IN}) + C_{IN}(X\overline{Y} + \overline{XY}) \\
= XY + C_{IN}(X \oplus Y)
\]

\[
\text{C}_{OUT}
\]

\[
\text{S}
\]
Multiplexers

- Selecting one of many inputs (also called **data selectors**)
- Consisting of $2^n$ **data** lines, $n$ **address** lines, 1 output, 1 **enable** control input

- Ex) 4-to-1 MUX

<table>
<thead>
<tr>
<th>$I_1$</th>
<th>$I_0$</th>
<th>$D_3$</th>
<th>$D_2$</th>
<th>$D_1$</th>
<th>$D_0$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$x$</td>
<td>$x$</td>
<td>0</td>
<td>$x$</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$x$</td>
<td>$x$</td>
<td>1</td>
<td>$x$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$x$</td>
<td>0</td>
<td>$x$</td>
<td>$x$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$x$</td>
<td>1</td>
<td>$x$</td>
<td>$x$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>1</td>
</tr>
</tbody>
</table>
• Read-only memory
• Holding information in storage ("memory") that cannot be altered but can be "read" by a logic circuit
• Consisting $2^m \times n$ cells
  – $m = \#$ of address lines
  – $n = \#$ of bits in each word stored in ROM
• When an address line is selected, the binary word corresponding to the address selected appears at the output
• c.f., EPROM (erasable programmable ROM)

• Ex) $2^2 \times 4$ ROM
• Ex) 8-word × 4-bit (or $2^2 × 4$) ROM

![Diagram of ROM with inputs A, B, C and outputs $F_0$, $F_1$, $F_2$, $F_3$. The table shows typical data stored in ROM (2^3 words of 4bits each).]
Decoder

- Identifying, recognizing, and detecting a particular code
- $N \times M$ decoder
  - $N$ inputs
    - $2^N$ input codes
    - Representing a binary number
    - Activating only the output that corresponds to that input number
  - $M$ outputs
    - Activated (HIGH) with only one of the $M$ outputs for each input code
    - LOW for the other outputs

- Ex) 3 × 8 decoder, 4 × 10 (BCD-to-decimal) decoder, BCD-to-7 segment decoder
• Ex) $2 \times 4$ decoder
- Ex) $3 \times 8$ decoder

$$
\begin{array}{cccccccccc}
\text{a} & \text{b} & \text{c} & \text{y}_0 & \text{y}_1 & \text{y}_2 & \text{y}_3 & \text{y}_4 & \text{y}_5 & \text{y}_6 & \text{y}_7 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
$$
• Ex) BCD-to-decimal decoder
  – 74LS42, 74HC42

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>BCD Input</th>
<th>Decimal Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0 0</td>
<td>0 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 0 0 1</td>
<td>1 0 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 0 1 0</td>
<td>1 1 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 0 1 1</td>
<td>1 1 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 1 0 0</td>
<td>1 1 1 1 0 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 1 0 1</td>
<td>1 1 1 1 1 0 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 1 1 0</td>
<td>1 1 1 1 1 1 0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 1 1 1</td>
<td>1 1 1 1 1 1 1 0 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 0 0 1</td>
<td>1 1 1 1 1 1 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 0 1 0</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 0 1 1</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 1 0 0</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 0 1</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 1 0</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

*active-LOW outputs*
Decoder and RAM

- Commonly used for address decoding or memory expansion
- Ex) 2-to-4 decoder

```
+---+---+---+---+
| A | B | Y0| Y1|
+---+---+---+---+
| G |   | Y2| Y3|
+---+---+---+---+
```

Inputs | Outputs
--- | ---
Enable | Select | Y0 | Y1 | Y2 | Y3
--- | --- | --- | --- | --- | ---
1 | x | 1 | 1 | 1 | 1
0 | 0 | 0 | 0 | 1 | 1 | 1
0 | 0 | 1 | 1 | 0 | 1 | 1
0 | 1 | 0 | 1 | 1 | 0 | 1
0 | 1 | 1 | 1 | 1 | 0 | 0

- SRAM (static random access, or read and write, memory)
Encoder

• Opposite to the decoding process
• Only one of input lines is activated at a given time
• Producing an $N$-bit output code
- Ex) $8 \times 3$ decoder

\[ \begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
X & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
X & X & 1 & 0 & 0 & 0 & 0 & 0 \\
X & X & X & 1 & 0 & 0 & 0 & 0 \\
X & X & X & X & 1 & 0 & 0 & 0 \\
X & X & X & X & X & 1 & 0 & 1 \\
X & X & X & X & X & X & 1 & 1 \\
\end{array} \]

<table>
<thead>
<tr>
<th>$y_0$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
<th>$y_4$</th>
<th>$y_5$</th>
<th>$y_6$</th>
<th>$y_7$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
• Ex) Decimal-to-BCD encoder

Switch on 3: ABCD = 0011
Switch on 7: ABCD = 0111
PLD

• Programmable logic device
  – PROM (programmable read-only memory)
  – PLA (programmable logic array)
  – PAL/GAL (programmable array logic/generic array logic)
• Arrays of gates (e.g., AND and OR gates) having interconnections that can be programmed to perform a specific logical function
  – Programming language: hardware description languages (HDLs)
• Used for various digital logic designs
• Timing diagram
Sequential logic circuits

- Combinational logic circuits provide outputs that are based on a combination of present inputs only

- Sequential logic circuits depend on present and past input values (it memorizes!)
  - Being able to store information
Flip-flops

- Basic information storage device in a digital circuit
- Many different varieties of flip-flops
  - $RS$ FF
  - $D$ FF
  - $JK$ FF
  - $T$ FF

- Common characteristics
  - Bistable device
    - Remaining in one of two stable states (0 and 1) until appropriate conditions cause FF to change state
    - Memory element
  - Two outputs; complement ($\bar{Q}$) and uncomplement ($Q$) outputs

- Synchronous operation by a "clock" signal
- Asynchronous operation
  - Independent of the clock
  - Level sensitive (⇒ “Latch”)
RS flip-flop

- Two inputs ($S$ set and $R$ reset), two outputs ($Q$ and $\overline{Q}$, called the state of FF)

Requiring the FF to set and reset at the same time!

<table>
<thead>
<tr>
<th>$S$</th>
<th>$R$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Present state</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Disallowed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S$</th>
<th>$R$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Time delays!
<table>
<thead>
<tr>
<th>R</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Waveform diagram](image-url)
Ex) Initial state $Q = 0$ (then, $\bar{Q} = 1$); apply $S = 1$

- $Q = \bar{S} \cdot \bar{Q} = 0 \cdot 1 = 1 \Rightarrow \text{SET}$
  - $\bar{Q}$ becomes 0; $Q = 0 \cdot 0 = 1 \Rightarrow \text{still SET}$
  - Cross-coupled feedback from outputs $Q$ and $\bar{Q}$ to the input of the NAND gates is such that the set condition sustains itself
• RS FF with enable ($E$), preset ($P$), and clear ($C$) inputs
  – $R$ or $S$ is effective only when $E = 1$
    • Synchronizing signal
  – Direct inputs $P$ and $C$ allow the user to preset or clear the FF at any time (asynchronous operation)
    • $S = 1$ (preset) when $P = 1$
    • $Q = 0$ (cleared) when $C = 1$
• **Delay latch (or delay element)**
  
  – An extension of RS FF
  – Always $R = S$
    
    • SET whenever $E = 1$
    • Prohibiting $R = S = 1$; eliminating $R$ input
  – Once $E = 0$, FF is **latched** to the previous value of the input ("memory") and **delays** the output by one clock count w.r.t. the input

<table>
<thead>
<tr>
<th>$E$</th>
<th>$D$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$X$</td>
<td>No change</td>
</tr>
</tbody>
</table>
D flip-flop

- An extension of data latch with two RS FFs
- Changing state only on the positive edge of the clock (leading or positive edge-triggered)
- Similarly, trailing or negative edge-triggered D FF

<table>
<thead>
<tr>
<th>$D$</th>
<th>$CLK$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\uparrow$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\uparrow$</td>
<td>1</td>
</tr>
</tbody>
</table>

indicating “leading edge-trigger”
• Note that "C" implies the "control" signal
JK flip-flop

- Same as RS FF except that \( J = K = 1 \) states

\[
\begin{array}{c|c|c}
J & K & Q \\
0 & 0 & \text{No change} \\
0 & 1 & \text{Reset} \\
1 & 0 & \text{Set} \\
1 & 1 & \text{Toggle} \\
\end{array}
\]
indicating “trailing edge-triggerC

<table>
<thead>
<tr>
<th>$J_n$</th>
<th>$K_n$</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$Q_n$ (no change)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 (reset)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 (set)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\bar{Q}_n$ (toggle)</td>
</tr>
</tbody>
</table>
• Master/slave FF
  – Delayed output by the width of clock pulse
T flip-flop

- JK FF with its inputs tied together

Waveforms for (toggle mode) $T$ flip-flop ($T = 1$)
• 클럭이 들어올 때마다 상태가 바뀌는 회로
• 출력신호가 정확히 T 입력신호 주파수의 절반

![D FF](image1)

![JK FF](image2)

![J K FF](image3)
3-bit binary up counter

- Force a RESET

![Timing diagram and timing table for a 3-bit binary up counter]
Decade counter

- Count from 0 to 9 and then RESET
- Impractical due to propagation delays
Ripple counter

- Consists of a cascade of 3 JK FFs

<table>
<thead>
<tr>
<th>Input</th>
<th>$Q_3$</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Ripple-up counter

- Asynchronous counter
  - T FF
  - \( n \) serial cascades = \( (2^n - 1) \) counter
  - Slow
Ripple-down counter

- Asynchronous counter

![Diagram of a ripple-down counter with three flip-flops labeled Q0, Q1, and Q2, and a clock input. The Q0 (LSB) output changes with each clock pulse, starting from 1 and decreasing to 0, as shown in the timing chart.](image)
Synchronous counter

- Parallel counter
- Fast
- Complex
Divider circuit
Synchronous counter
Ring counter
Parallel register

- The load input (clock) simultaneously transfers the parallel input binary word $b_3b_2b_1b_0$ (store!)
Shift register