ESE 570 STATIC SEQUENTIAL CMOS LOGIC CELLS
Classes of Logic Circuits

COMBINATIONAL (non-regenerative)
- BISTABLE: two stable op. pts. 
  - Latch – level triggered.
  - Flip-Flop – edge triggered.
- MONOSTABLE: one stable op. pt.
  - One-shot – single pulse output
- ASTABLE: no stable op. pt.
  - Ring Oscillator

SEQUENTIAL (regenerative)

Combinational Circuits:
- a. Current Output(s) depend ONLY on Current Inputs.
- b. Suited to problems that can be solved using truth tables.

Sequential Circuits or State Machines:
- a. Current Output(s) depend on Current Inputs and Past Inputs via State(s).
- b. Suited to problems that are solved by completing several steps using current inputs and past outputs in a specific order or a sequential manner.
Functions Using Sequential Operations

Sequential Operations

Combinational Operations

Data Transfer
  - Serial
  - Parallel

Arithmetic Logic Unit
  - Microprocessor
  - Microcomputer

Functions Using Sequential Operations

Flip-Flops/Latches
  - to accomplish
  - may use memory cells such as

Binary Counting
  - to accomplish
  - and

Decade Counting

Frequency Division

Frequency Counting

Shift Register

Data Transfer & Sequential Operations

may use memory cells such as

to accomplish

which may be

or

which is a component of a

which is the core of a
Sequential Circuit (or State Machine) Construct

-> Memory is used to Store Past Values of State(s) and Output(s).
-> Asynchronous Sequential Circuit – no clock, outputs change after inputs change
-> Synchronous Sequential Circuit – clock, outputs change with clock event
Moore FSM

Inputs $X_j$

Combinational Logic

State Register

Comb Log Outputs $Y_k$

Clock

Y $k$'s are solely function of current states.

Y $k$'s change in sync with state clock.

State Feedback

Mealy FSM

Inputs $X_j$

Combinational Logic

State Register

Clock

Y $k$ Outputs

Y $k$'s function of inputs & current states.

Y $k$'s change when inputs change.

Y $k$'s are asynchronous

State Feedback
Static Bistable Sequential Circuits

Basic Cross-coupled Inverter pair

![Diagram of cross-coupled inverter pair]

BISTABLE BEHAVIOR

V_{o1} / V_{in} < 1

INV1 VTC

V_{o1} / V_{in} >> 1

INV2 VTC

V_{i1} V_{o2}

ENERGY

V_{o2} / V_{in} < 1

Kenneth R. Laker, University of Pennsylvania, updated 25Mar15
Basic Bistable Cross-coupled Inverter Pair has no means to apply input(s) to change the circuit's State.

STATIC: $V_{DD}$ and GND are required to maintain a stable state.
Basic Sequential Circuits (Cells)

Latch
- Asynchronous or synchronous
- If synchronous, clock input is level sensitive.
- If synchronous, output can change multiple times during a clock cycle.
- If synchronous, output changes while clock is active.

Flip-Flop
- Synchronous
- Involves two synchronous latches.
- Output is edge sensitive, i.e. Output only changes on rising (or falling) edge of clock.
- Output can change only once during a clock cycle.
- Output changes on clock transition.
Asynchronous Latch Circuits

Full CMOS Asynchronous SR Latch

NOR Based Latch

STATE OF LATCH can be EXTERNALLY SWITCHED between the 2 STABLE STATES

SET STATE: \( S_{t1} = 1, R_{t1} = 0 \Rightarrow Q_{t1} = 1, \overline{Q_{t1}} = 0 \)

RESET STATE: \( S_{t1} = 0, R_{t1} = 1 \Rightarrow Q_{t1} = 0, \overline{Q_{t1}} = 1 \)

HOLD: \( S_{t1} = 0, R_{t1} = 0 \Rightarrow Q_{t1} = Q_{t0}, \overline{Q_{t1}} = \overline{Q_{t0}} \)

(two cross-coupled Inverters)
(M2, MP2 and M3, MP3)

NOT ALLOWED: \( S_{t1} = 1, R_{t1} = 1 \rightarrow Q_{t1} = 0, \overline{Q_{t1}} = 0 \) is forbidden state
or not allowed state
Let at $t = t_0$: $Q_{t_0} = 0, \overline{Q_{t_0}} = V_{DD}$ -> initial state

At $t = t_1 > t_0$:

1. $S_{t_1} = 1$ => $M1$ ON, $MP1$ OFF => $\overline{Q_{t_1}} = 0$

2. $R_{t_1} = 0$ and $\overline{Q_{t_1}} = 0$ => $M4$ OFF, $M3$ OFF, $MP3$ ON, $MP4$ ON => $Q_{t_1} = 1$

3. $Q_{t_1} = 1$ => $M2$ ON, $MP2$ OFF => $\overline{Q_{t_1}} = 0$ -> set state
Asynchronous CMOS NOR SR Latch Operation - cont.

**RESET OP:**
\[ R = 1, \; S = 0 \]

Let at \( t = t_0 \): \( Q_{t_0} = 1, \; \overline{Q}_{t_0} = 0 \) \( \rightarrow \) initial state

At \( t = t_1 > t_0 \)
1. \( \overline{R}_{t_1} = 1 \rightarrow M4 \text{ ON, MP4 OFF} \rightarrow Q_{t_1} = 0 \)

2. \( S_{t_1} = 0 \) and \( Q_{t_1} = 0 \rightarrow M1 \text{ OFF, M2 OFF, MP1 ON, MP2 ON} \rightarrow \overline{Q}_{t_1} = V_{DD} \)

3. \( \overline{Q}_{t_1} = V_{DD} \rightarrow M3 \text{ ON, MP3 OFF} \rightarrow Q_{t_1} = 0 \) \( \rightarrow \) reset state
HOLD OP:
\( S = 0, \quad R = 0 \)

At \( t = t_1 > t_0 \)

1. \( S_{t_1} = 0 \Rightarrow M1 \text{ OFF, } MP1 \text{ ON} \); \( R_{t_1} = 0 \Rightarrow M4 \text{ OFF, } MP4 \text{ ON} \)

2a. \( Q_{t_1} = Q_{t_0} = 1 \) \quad \( \overline{Q}_{t_1} = \overline{Q}_{t_0} = 0 \Rightarrow M2 \text{ ON, } MP2 \text{ OFF, } M3 \text{ OFF, } MP3 \text{ ON} \)

or

2b. \( Q_{t_1} = Q_{t_0} = 0, \quad \overline{Q}_{t_1} = \overline{Q}_{t_0} = 1 \Rightarrow M2 \text{ OFF, } MP2 \text{ ON, } M3 \text{ ON, } MP3 \text{ OFF} \)
Asynchronous CMOS NOR SR Latch Operation - cont.

HOLD OP:
\[ S = 0, \quad R = 0 \]

At \( t = t_1 > t_0 \)

1. \( S_{t_1} = 0 \) => M1 OFF, MP1 ON; \( R_{t_1} = 0 \) => M4 OFF, MP4 ON

2a. \( Q_{t_1} = Q_{t_0} = 1 \), \( \overline{Q}_{t_1} = \overline{Q}_{t_0} = 0 \) => M2 ON, MP2 OFF, M3 OFF, MP3 ON

or

2b. \( Q_{t_1} = Q_{t_0} = 0 \), \( \overline{Q}_{t_1} = \overline{Q}_{t_0} = 1 \) => M2 OFF, MP2 ON, M3 ON, MP3 OFF
Asynchronous CMOS NOR SR Latch Operation - cont.

\[ C_Q = C_{\text{load-NR2Q}} = 2C_{\text{n-int}} + 3C_{\text{p-int}} + C_{\text{ext}} \]

\[ C_{\bar{Q}} = C_{\text{load-NR2\bar{Q}}} = 2C_{\text{n-int}} + 3C_{\text{p-int}} + C_{\text{ext}} \]

Estimate time to simultaneously switch \( Q \) & \( \bar{Q} \): solution of two coupled differential equations.

**Conservative Estimate:** Assume \( Q \) & \( \bar{Q} \) switch in sequence

\[ \tau_{\text{rise},Q}(SR-Latch) \approx \tau_{\text{rise},Q}(NR2) + \tau_{\text{fall},\bar{Q}}(NR2) \]

at \( t = 0: S \rightarrow 1, R \rightarrow 0 \)
Asynchronous Latch Circuits - cont.

"ACTIVE HIGH"

<table>
<thead>
<tr>
<th>$S_{t1}$</th>
<th>$R_{t1}$</th>
<th>$Q_{t1}$</th>
<th>$\overline{Q}_{t1}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$Q_{t0}$</td>
<td>$\overline{Q}_{t0}$</td>
<td>hold</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>set*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>reset*</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>NOT allowed</td>
</tr>
</tbody>
</table>

*Data is written by over powering the feedback loop using $S, R$ inputs.
Asynchronous CMOS NAND SR Latch Circuit

ASYNCHRONOUS NAND BASED SR LATCH

![Circuit Diagram]

Is NAND OR NOR SR LATCH PREFERRED?

<table>
<thead>
<tr>
<th>$S_{t1}$</th>
<th>$R_{t1}$</th>
<th>$Q_{t1}$</th>
<th>$\overline{Q}_{t1}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NOT allowed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$Q_{t0}$</td>
<td>$\overline{Q}_{t0}$</td>
<td>hold</td>
</tr>
</tbody>
</table>

$t = t_1 > t = t_0$

“ACTIVE LOW”
Asynchronous CMOS NAND SR Latch Circuit - cont.

\[ t_1 > t_0 \]

<table>
<thead>
<tr>
<th>( S_{t_1} )</th>
<th>( R_{t_1} )</th>
<th>( Q_{t_1} )</th>
<th>( \overline{Q}_{t_1} )</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NOT allowed set</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( Q_{t_0} )</td>
<td>( \overline{Q}_{t_0} )</td>
<td></td>
</tr>
</tbody>
</table>

SR- Latches
+ Simplest form of latch
- Asynchronous
- Not Allowed Input Sequence
CMOS Level Sensitive Synchronous Latches

SR LATCH: Synchronization is introduced through clock CK.

NAND SR LATCH

When CK = 0, S, R HAVE NO INFLUENCE OF Q, \overline{Q} => HOLD

HOLD STATE: CK = 0, S = x, R = x => S' = 1, R' = 1 => Q_{n+1} = Q_n, \overline{Q}_{n+1} = \overline{Q}_n

SET STATE: CK = 1, S = 1, R = 0 => S' = 0, R' = 1 => Q_{n+1} = 1, \overline{Q}_{n+1} = 0

RESET STATE: CK = 1, S = 0, R = 1 => S' = 1, R' = 0 => Q_{n+1} = 0, \overline{Q}_{n+1} = 1

NOT ALLOWED: CK = 1, S = 1, R = 1 => S' = 0, R' = 0

IS CK = 1, S = 0, R = 0 a HOLD STATE?

“ACTIVE HIGH”
CMOS Level Sensitive Synchronous Latches - cont.

**HOLD STATE:** \( CK = 0, S = x, R = x \) => \( Q_{n+1} = Q_n, \quad Q_{n+1}^c = Q_n^c \)

**SET STATE:** \( CK = 1, S = 1, R = 0 \) => \( Q_{n+1} = 1, \quad Q_{n+1}^c = 0 \)

**RESET STATE:** \( CK = 1, S = 0, R = 1 \) => \( Q_{n+1} = 0, \quad Q_{n+1}^c = 1 \)

**HOLD STATE:** \( CK = 1, S = 0, R = 0 \)

**NOT ALLOWED:** \( CK = 1, S = 1, R = 1 \)

---

**When “GLITCH” on S (or R) occurs during \( CK = 1 \), Q is SET (or RESET):**

**LEVEL SENSITIVE:** When \( CK = 1 \), any changes in S, R will effect Q.
CMOS Level Sensitive Synchronous Latches - cont.

Another Gate Level schematic of a Clocked NAND SR Flip Flop

When \( CK = 1 \), \( S' = R' = 1 \) independent of the values of \( S \) and \( R \) => HOLD

Set and Reset operations only occur when \( CK = 0 \).

<table>
<thead>
<tr>
<th>CK</th>
<th>S</th>
<th>R</th>
<th>( Q_{n+1} )</th>
<th>( \overline{Q}_{n+1} )</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NOT allowed</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>( Q_n )</td>
<td>( \overline{Q}_n )</td>
<td>hold</td>
</tr>
</tbody>
</table>

“ACTIVE LOW”
CMOS Level Sensitive Synchronous Latches - cont.

CMOS NAND Based Clocked SR Latch

CMOS NAND Based Clocked SR Latch

+ Synchronous operation
- Level Sensitive
- Not Allowed Input Sequence

<table>
<thead>
<tr>
<th>CK</th>
<th>S</th>
<th>R</th>
<th>$Q_{n+1}$</th>
<th>$\overline{Q}_{n+1}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NOT allowed set</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>reset</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>$Q_n$</td>
<td>$\overline{Q}_n$</td>
<td></td>
</tr>
</tbody>
</table>

“ACTIVE LOW”

Kenneth R. Laker, University of Pennsylvania, updated 25Mar15
When $CK = 1$, the latch state $Q$ and $\bar{Q}$ is independent of inputs $S$ and $R$, and the latch is in Hold operation.
When $\text{CK} = 0$, the state $Q$ and $\bar{Q}$ depend on inputs $S$ and $R$, and the schematic reduces to that of a NAND based SR latch.
CMOS Level Sensitive Synchronous Latches - cont.

NAND BASED JK Synchronous Latch

CK = 1 => active

ELIMINATES THE NOT ALLOWED INPUT COMBINATIONS

S = R ≠ 0 for all values of J, K, CK

<table>
<thead>
<tr>
<th>S_{i1}</th>
<th>R_{i1}</th>
<th>Q_{i1}</th>
<th>Q_{i1}'</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NOT allowed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q_{i0}</td>
<td>Q_{i0}'</td>
<td>hold</td>
</tr>
</tbody>
</table>
NAND JK Synchronous Latch Operation

CK = 0 => Hold
i.e. S = R = 1
independent of J, K

CK = 1 => active

“ACTIVE HIGH”

CK

J

SR LATCH

CK

R

S

Q

\overline{Q}

\begin{align*}
\text{CK} = 1 & \Rightarrow \text{active} \\
\text{CK} = 0 & \Rightarrow \text{Hold}
\end{align*}

\begin{align*}
\text{i.e. } S = R = 1 & \text{ independent of } J, K
\end{align*}

\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
J & K & \text{Q}_n & \overline{\text{Q}}_n & S & R & \text{Q}_{n+1} & \overline{\text{Q}}_{n+1} & \text{Operation} \\
\hline
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & \text{hold} \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & \text{hold} \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & \text{reset (hold)} \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & \text{reset} \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \text{set} \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & \text{set (hold)} \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \text{toggle} \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & \text{toggle} \\
\hline
\end{tabular}

The not-allowed \( S, R \) values \( S = R = 0 \) do not occur for any values of \( J, K, CK \).

\[
\begin{array}{c|c|c|c|c|c|c|c|}
\hline
S_n & R_n & Q_n & \overline{Q}_n & \text{Operation} \\
\hline
0 & 0 & 0 & 0 & \text{NOT allowed} \\
0 & 1 & 1 & 0 & \text{set} \\
1 & 0 & 0 & 1 & \text{reset} \\
1 & 1 & Q_{n+1} & \overline{Q}_{n+1} & \text{hold} \\
\hline
\end{array}
\]

not desirable, but the state \( Q_{n+1}, \overline{Q}_{n+1} \) is not forbidden
### NAND Based JK Synchronous Latch in Toggle Mode

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q_n</th>
<th>(\bar{Q}_n)</th>
<th>S</th>
<th>R</th>
<th>Q_{n+1}</th>
<th>(\bar{Q}_{n+1})</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>toggle</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>toggle</td>
</tr>
</tbody>
</table>

CK = 1

**OSC**

**TO PREVENT OSCILLATION WHEN J = K = 1:**

\[ \tau_{JKP} > T_1 \]

\(\tau_{JKP} = \text{INPUT-OUTPUT PROP DELAY OF JK FLIP FLOP} \)

(CK 1 \(\rightarrow\) 0 BEFORE Q, \(\bar{Q}\) CAN SWITCH 2\textsuperscript{nd} TIME)

(high speed clock may be impractical)
Clocked Negative Edge Triggered SR Flip-Flop (FF)

- Start with $\text{CK} = 0 \Rightarrow X = X' = 1$, Latch 1 is Hold $\Rightarrow Q_1, \overline{Q_1}$ are independent of $S, R$.
- Changes in $S, R$ can't change the state of Latch 1 nor the state $Q_2, \overline{Q_2}$ of Latch 2.

- When $\text{CK} = 1$, inputs $S, R$ control the state of Latch 1.
- Inverted $\overline{\text{CK}} = 0$ applied to Latch 2 ($\Rightarrow Y = Y' = 1$) and Latch 2 is Hold and state $Q_2, \overline{Q_2}$ are independent of $Q_1, \overline{Q_1}$.
- When $\text{CK} = 1$ changes to $S, R$ are tracked by Latch 1, but not reflected in the state $Q_2, \overline{Q_2}$ of Latch 2.

- When $\text{CK} = 0$, the state of Latch 1 $Q_1, \overline{Q_1}$ are independent of inputs $S, R$.
- Inverted $\overline{\text{CK}} = 1$ enables the Held state of Latch 1 to effect the state $Q_2, \overline{Q_2}$ of Latch 2.
- $\text{CK} \rightarrow 1$ to $0$ is the the falling (negative) edge of the $\text{CK}$ signal.

<table>
<thead>
<tr>
<th>$X_t$</th>
<th>$X'_t$</th>
<th>$Q_{t_1}$</th>
<th>$\overline{Q}_{t_1}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NOT allowed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$Q_{t_0}$</td>
<td>$\overline{Q}_{t_0}$</td>
<td>hold</td>
</tr>
</tbody>
</table>

+ Synchronous Op
+ Not Level Sensitive
- $S, R = 1$ Not Allowed
Clocked Negative Edge Triggered JK Flip-Flop

- Complex circuit, requiring 42 transistors
- Note: SR FF circuit requires 32 transistors

+ Synchronous Operation
+ No Not-Allowed States
+ Not Level Sensitive
+ No $Q_2$, $\overline{Q}_2$ Oscillation when $J = K = 1$; i.e. toggle of JK-Latch1 is not seen by JK-Latch2

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>$Q_{1_{n-1}}$</th>
<th>$\overline{Q}<em>{1</em>{n-1}}$</th>
<th>S</th>
<th>R</th>
<th>$Q_{1_n}$</th>
<th>$\overline{Q}_{1_n}$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>reset (hold)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>set (hold)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>toggle</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>toggle</td>
</tr>
</tbody>
</table>

CK = 1
Static CMOS D-LATCH

Gate level implementation by modifying a NAND SR Latch.

CK  D  S'  R'  Q_{n+1}  \overline{Q}_{n+1}
1  1  0  1    1  0    SR-Set
1  0  1  0    0  1    SR-Reset
0  x  0  0    Q_n  Q_n    SR-Hold

+ NO TOGGLE
+ NO NOT-ALLOWED INPUTS
**Transistor level implementation using transmission gates requires fewer transistors**
CMOS TG D-LATCH Operation

Since when $CK = 1$ output $Q = D$, and tracks $D$ until $CK = 0$, the D-Latch is referred to positive level triggered.

When $CK \to 1$ to $0$, the $Q = D$ is captured, held (or stored) in the Latch.
D-LATCH Timing Requirements

$t_{\text{setup}}$ - time before the NEG(pos)-CLK edge the D-input has to be stable.

$t_{\text{hold}}$ - time after NEG(pos)-CLK edge that the D-input has to remain stable.

$t_{\text{clock-to-Q}}$ - Delay from the NEG(pos)-CLK edge to new stable value of Q output.
D-Latch Metastability and Synchronization Failures

If data and clock do not satisfy the setup & hold time constraints of a latch, then synchronization failure may occur. This due to inherent analog nature of storage elements.

METASTABLE STATE - indeterminate state between "1" & "0", i.e. latch, perfectly balanced between making decision for "1" or "0". In practice noise will eventually arbitrarily push the flip-flop output to “0” or “1”.

Example: register entering metastable state (shown for negative edge case)
CMOS D Edge Triggered Flip-Flop

Negative Level Triggered D Latch

Positive Level Triggered D Latch

Positive Edge Triggered D Flip-Flop = Negative D-Latch + Positive D-Latch

Negative Edge Triggered D Flip-Flop = Positive D-Latch + Negative D-Latch
CMOS D Flip-Flop – Positive Edge Triggered

1. CLK = 0: master $Q_m$ tracks input D; slave $Q_s$ = previous $D_{n-1}$ sample ($Q_s$ is unaffected by variations in D).

2. CLK = 0 -> 1: master stores $Q_m = D_n$ (new D sample).

3. CLK = 1: master passes $Q_m = D_n$ to slave output $Q_s$ ($Q_m$ and $Q_s$ are unaffected by variations in D).

4. CLK = 1 -> 0: slave locks in new $D_n$.

5. CLK = 0: master $Q_m$ begins tracking D. ($Q_s$ is unaffected by variations in D)

6. CLK = 0 -> 1: master stores $Q_m = D_{n+1}$. 
Impact of Non-ideal Clock on D-Latch Operation
Two-Phase Clocked D-Latch
Practical CMOS D Edge Triggered Flip-Flop

\[ \phi_1 \quad Q_m \quad Q_m \quad \phi_2 \quad Q_s \quad Q_s \]

NOT

Practical

actical
CMOS Dynamic D Latch

Positive level-sensitive

1. NO FEEDBACK REGENERATIVE FEEDBACK LOOP
2. STATES STORED ON SOFT NODES

\( C_x \) is usually a parasitic capacitance
Comparison CMOS Static & Dynamic D Latches

Static D-Latch

Dynamic D-Latch

Data bit stored in bistable-loop when $\phi_1 = 1 \rightarrow 0$

Data bit stored on $C_x$ when $CK = 1 \rightarrow 0$
CMOS Static & Dynamic D FFs