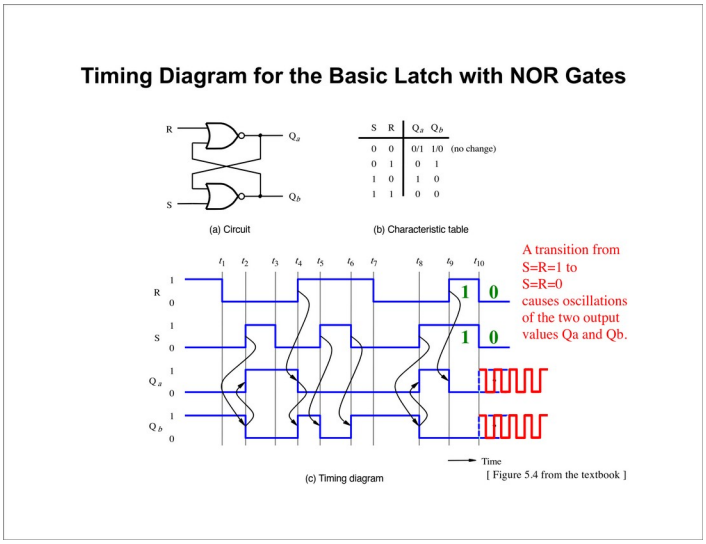
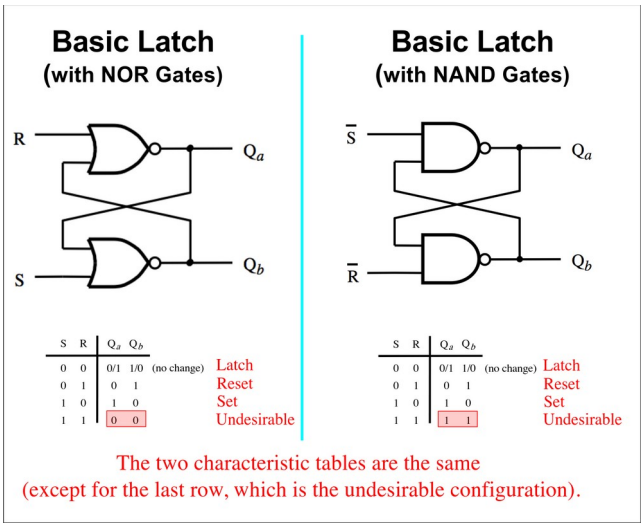
# Latches

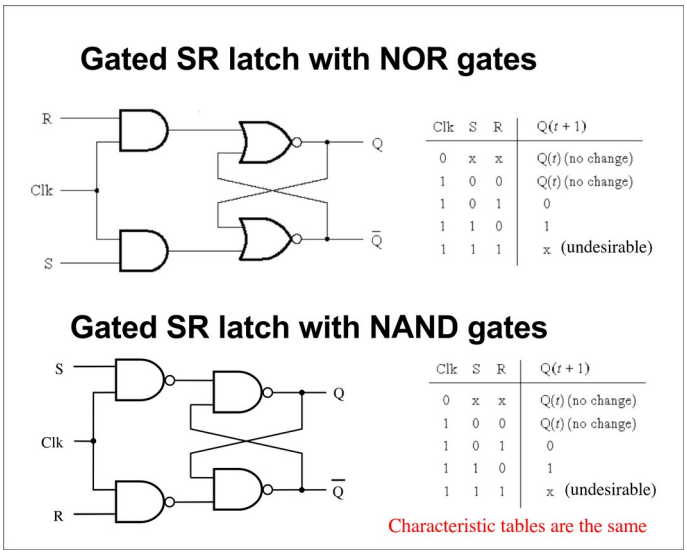
Basic NOR Latch:




Basic NAND Latch:

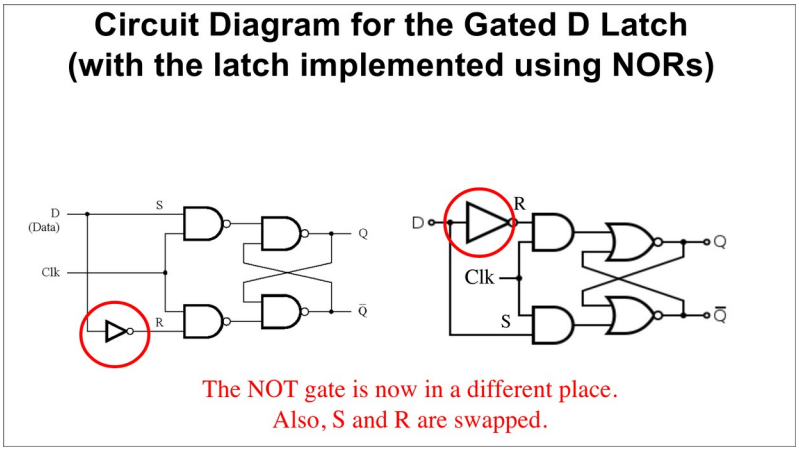


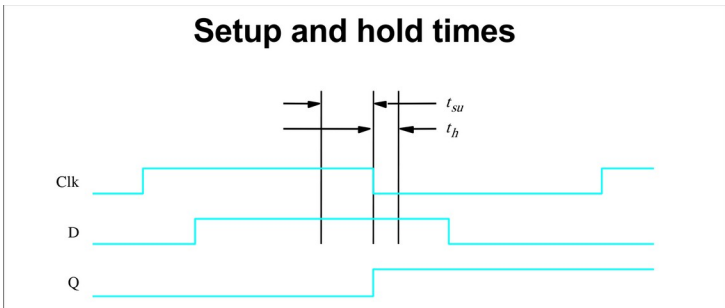

Gated SR Latches:




***A transition from S=R=1 to S=R=0 causes the output of the SR latch to oscillate between 0 and 1, until random factors (gate delays & wire lengths) cause it to settle on an unpredictable value**

Gated D Latch with NAND/XOR Gates:

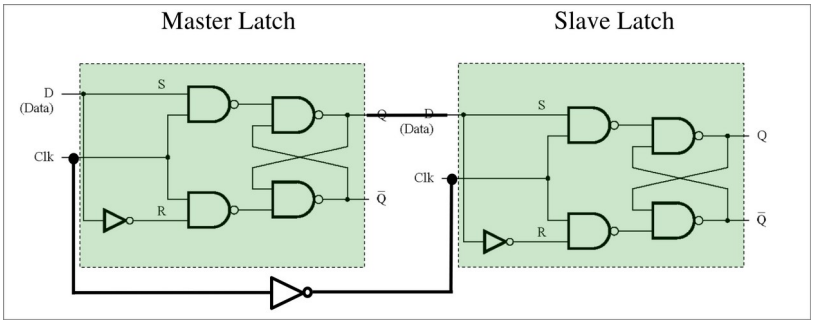


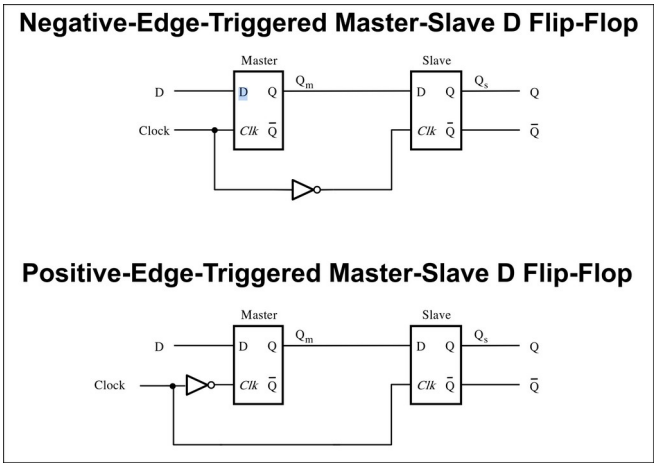

Setup time ($t_{su}$) – the minimum time that the D signal must be stable prior to the negative edge of the Clock signal.

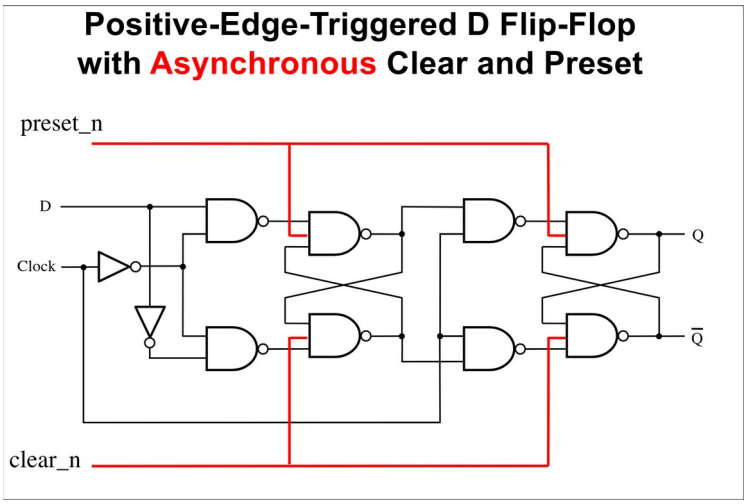
Hold time ($t_h$) – the minimum time that the D signal must remain stable after the negative edge of the Clock signal.

Master-Slave D Flip-Flop:

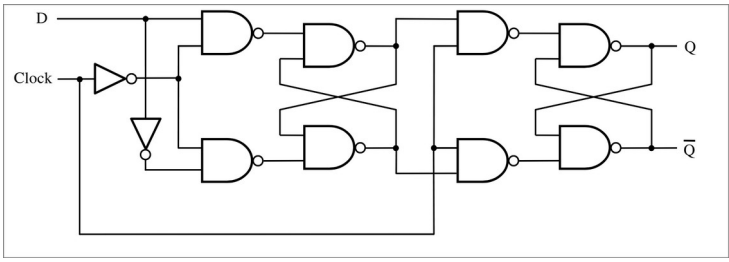



Pos/Neg Edge-Triggered D Flip-Flops:

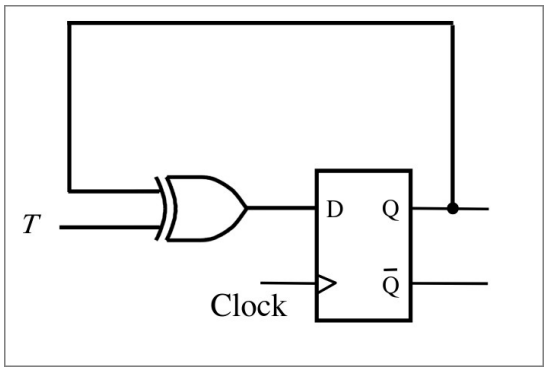


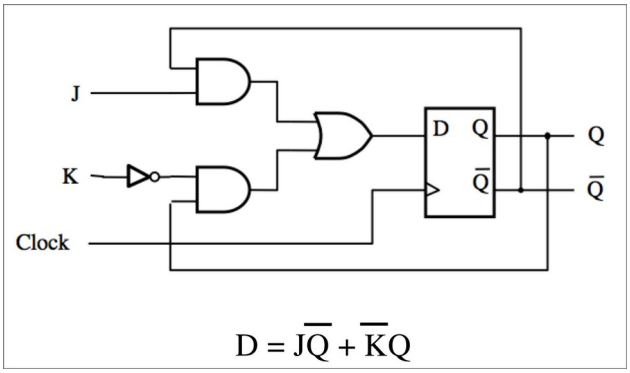

Complete Wiring Diagram for a
Positive-Edge-Triggered D Flip-Flop:



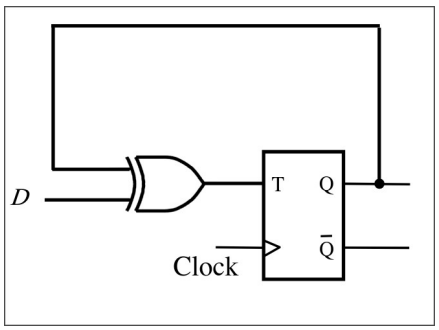

T Flip-Flop w/ D Flip-Flop:



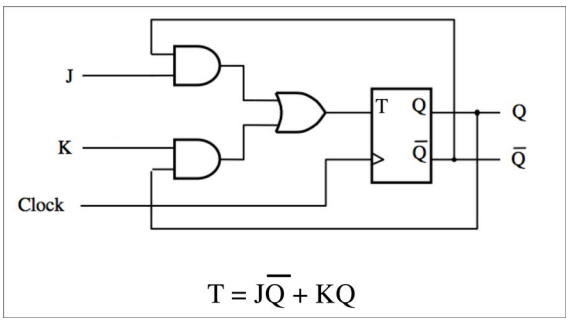

JK Flip-Flop w/ D Flip-Flop:



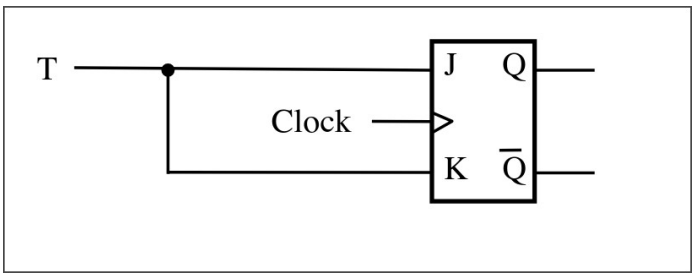

D Flip-Flop w/ T Flip-Flop:



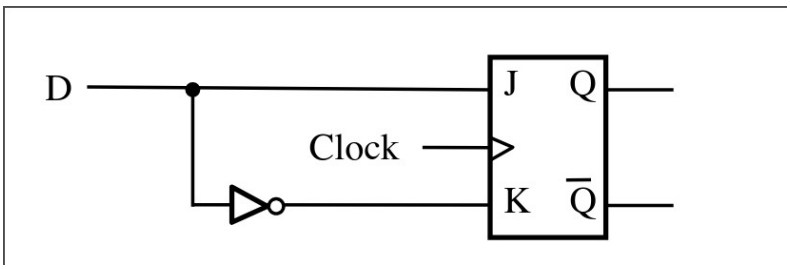

JK Flip-Flop w/ T Flip-Flop



T Flip-Flop w/ JK Flip-Flop:

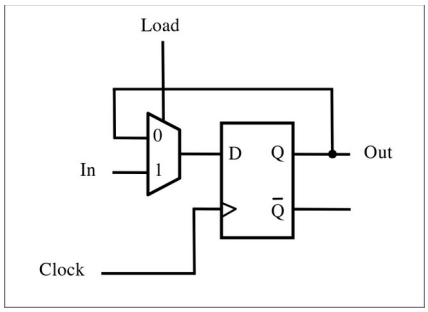



D Flip-Flop w/ JK Flip-Flop:

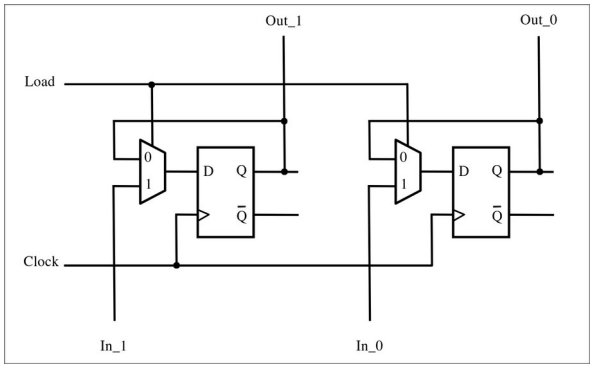



# Registers

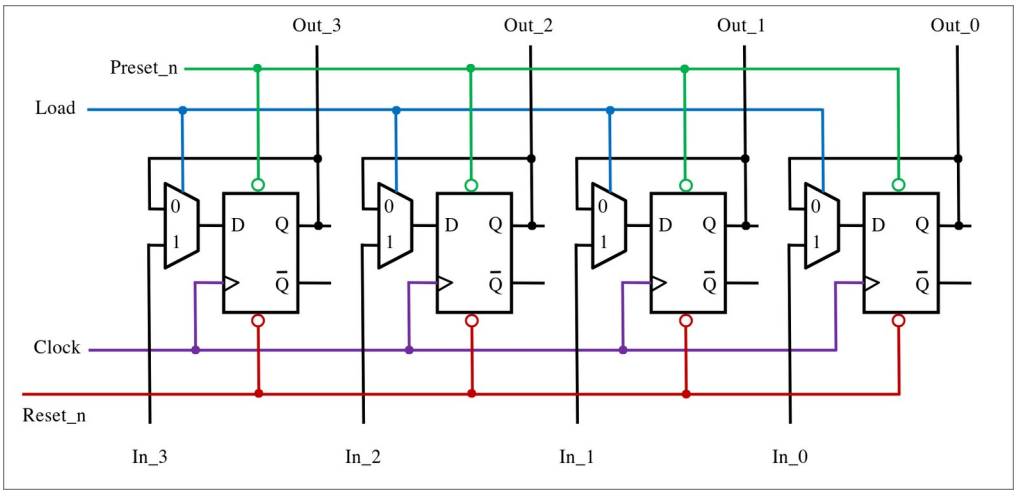
1-bit Parallel Access Register

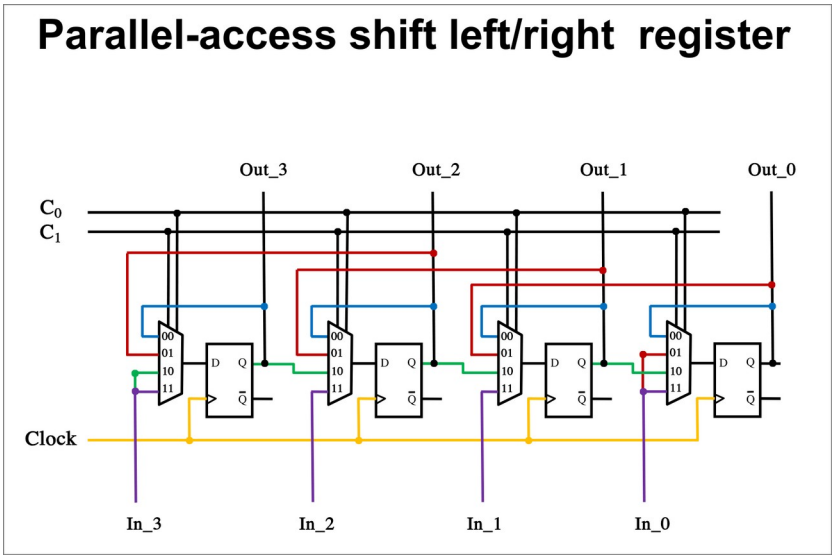



2-bit Parallel Access Register




4-bit Parallel Access Register w/ Async Preset/Clear





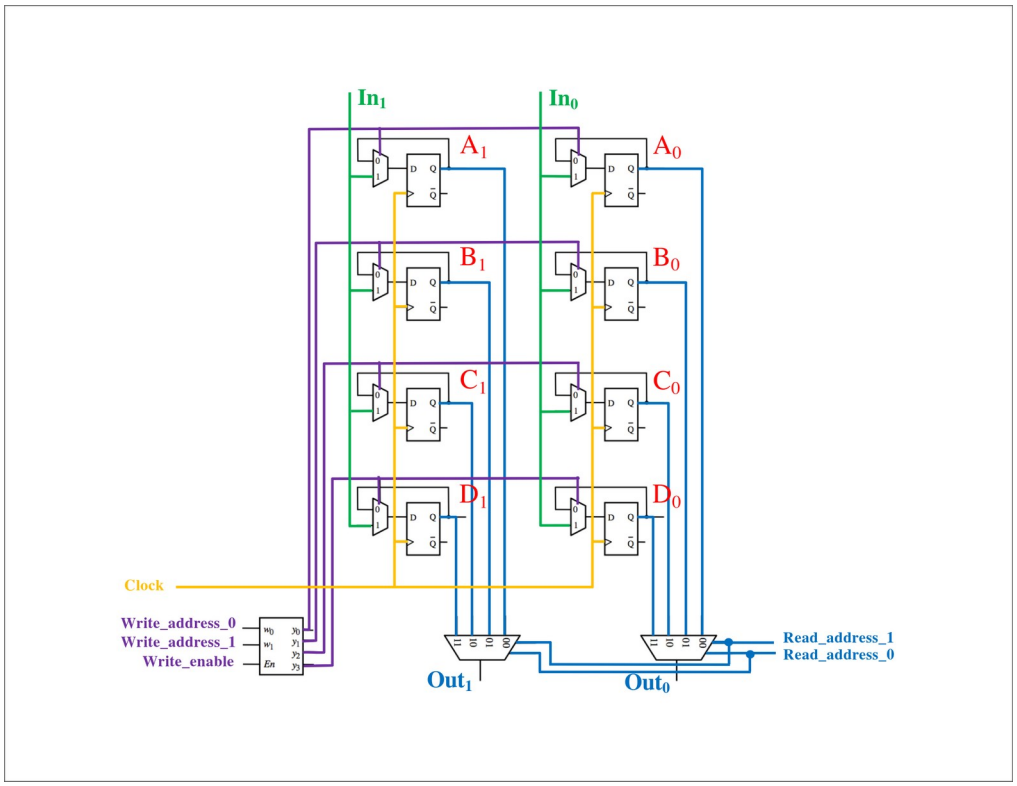

**C{1,0} = 00 : do nothing**

**C{1,0} = 01 : shift left**

**C{1,0} = 10 : shift right**

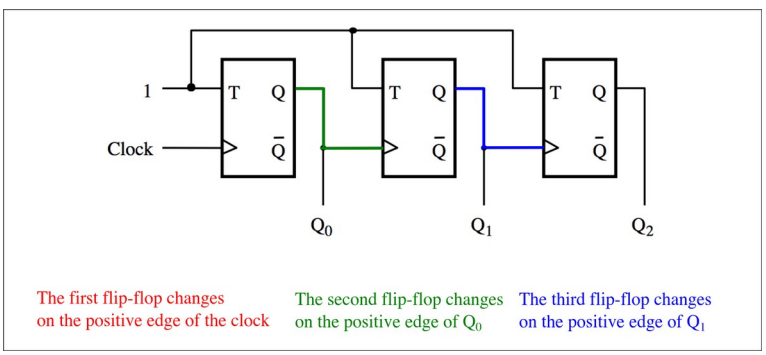
**C{1,0} = 11 : load from input**

Register File:

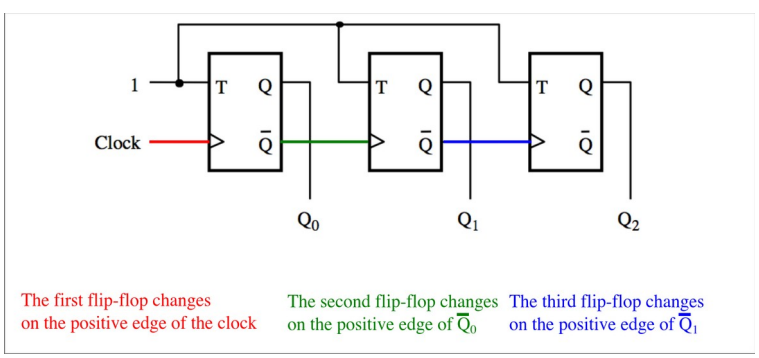



# Counters

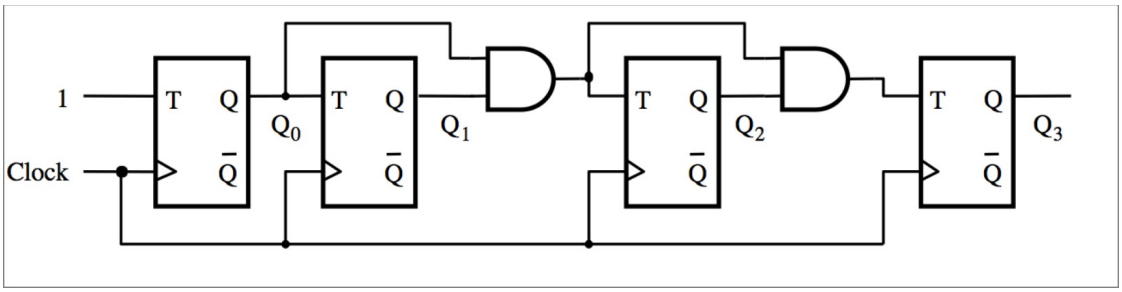
3-bit Down Counter



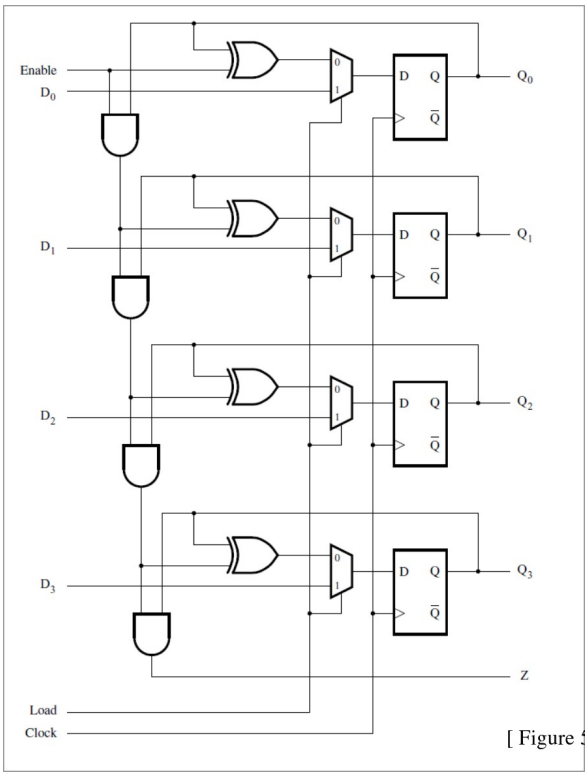

3-bit Up Counter



4-bit **Synchronous** Up Counter

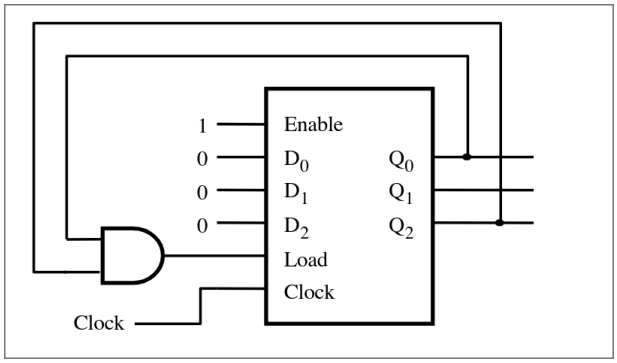



Synchronous 4-bit Up Counter w/ Parallel Load using D Flip-Flops

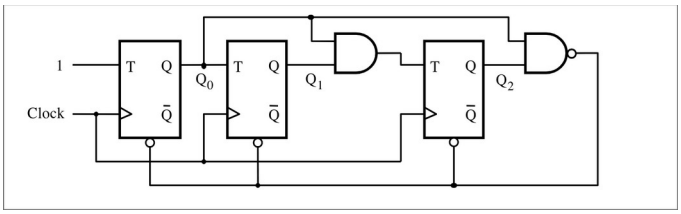



[ Figure 5

Mod-6 Up Counter w/ Synchronous Reset



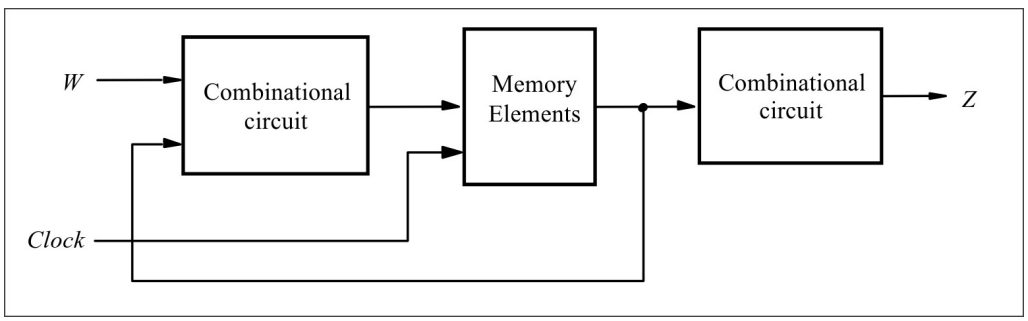

Mod-6 Up Counter w/ Async Reset




*For a counter that doesn't count to a power of 2, a sync/async reset on all flip-flops should occur when the max value is reached
(e.g. above we reset on Q2*Q0 = 101 = 6)

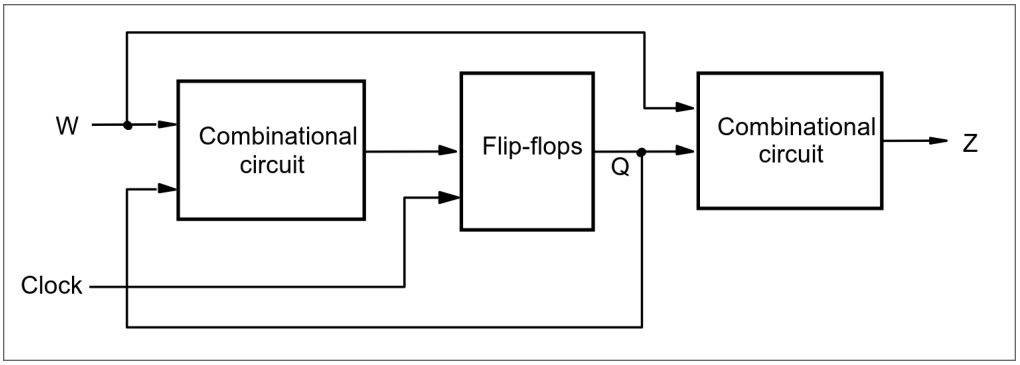
# Finite State Machines

Moore Implementation



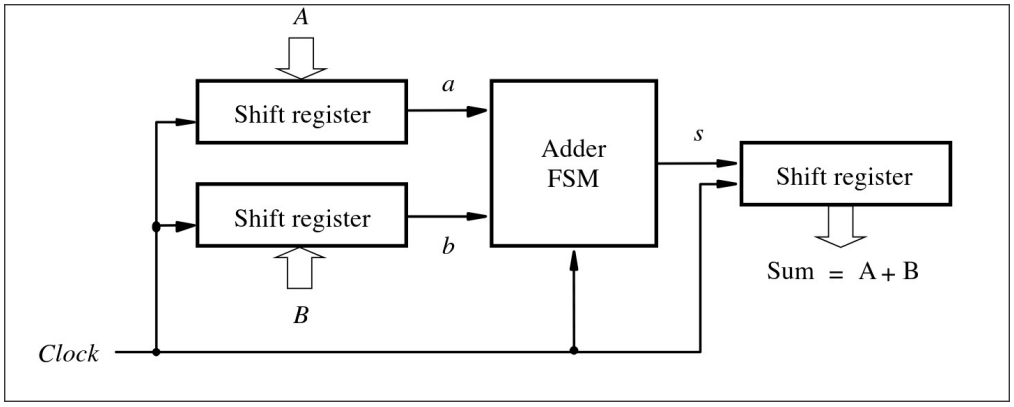

*The output depends only on the current state

Mealy Machine

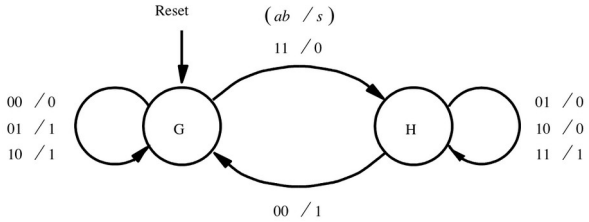



*The output depends on the current state **and** the current inputs

Serial Adder



Mealy Implementation of FSM

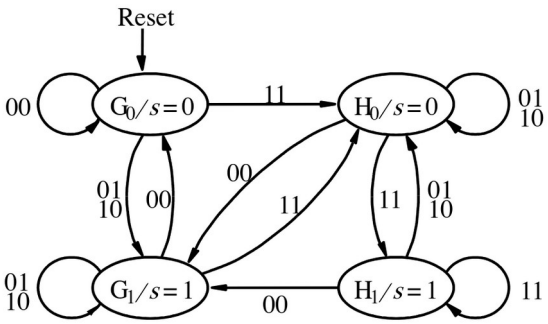



**State table for the serial adder FSM**

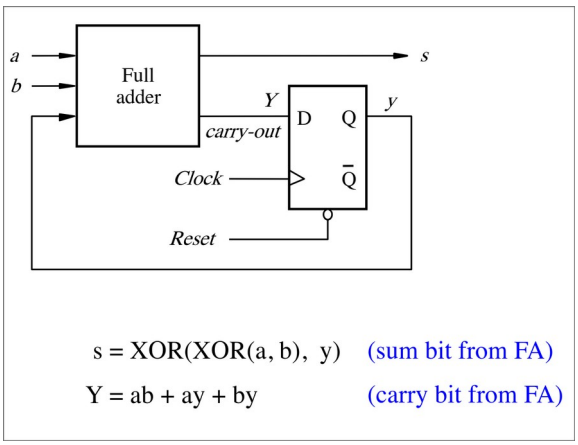
| Present state | Next state | | | | Output *s* | | | |
|---|---|---|---|---|---|---|---|---|
| | *ab* =00 | 01 | 10 | 11 | 00 | 01 | 10 | 11 |
| G | G | G | G | H | 0 | 1 | 1 | 0 |
| H | G | H | H | H | 1 | 0 | 0 | 1 |

Moore Machine Implementation of FSM

| Present state | Next state | | | | Output |
|---|---|---|---|---|---|
| | *ab* =00 | 01 | 10 | 11 | *s* |
| $G_0$ | $G_0$ | $G_1$ | $G_1$ | $H_0$ | 0 |
| $G_1$ | $G_0$ | $G_1$ | $G_1$ | $H_0$ | 1 |
| $H_0$ | $G_1$ | $H_0$ | $H_0$ | $H_1$ | 0 |
| $H_1$ | $G_1$ | $H_0$ | $H_0$ | $H_1$ | 1 |



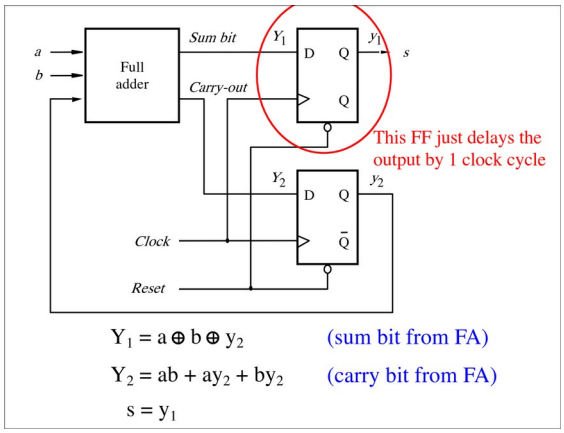

Mealy FSM Circuit Diagram




s = XOR(XOR(a, b), y) (sum bit from FA)

Y = ab + ay + by (carry bit from FA)

Moore FSM Circuit Diagram

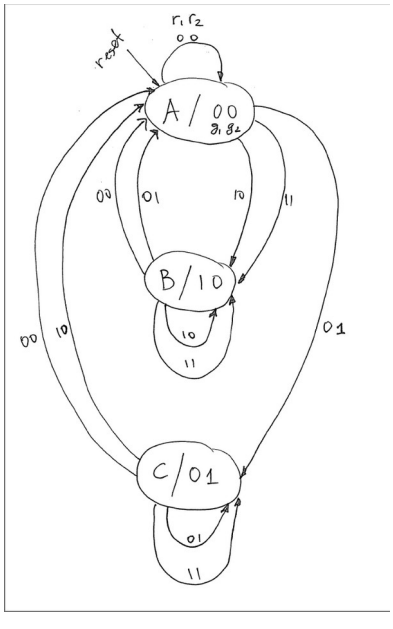



$Y_1 = a \oplus b \oplus y_2$ (sum bit from FA)

$Y_2 = ab + ay_2 + by_2$ (carry bit from FA)

$s = y_1$

Arbiter FSM Diagram




Arbiter State Table

| | $r_1r_2$ = 00 | 01 | 10 | 11 | Output |
|---|---|---|---|---|---|
| A | A | C | B | B | 00 |
| B | A | A | B | B | 10 |
| C | A | C | A | C | 01 |

Arbiter State-Assigned Table

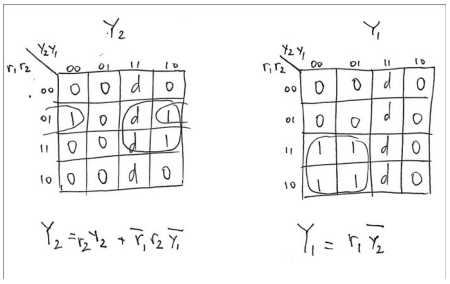
| | Present state $y_2y_1$ | Next state $r_1r_2$=00 | 01 | 10 | 11 | Output $g_1g_2$ |
|---|---|---|---|---|---|---|
| | | $Y_2Y_1$ | | | | |
| A | 00 | 00 | 01 | 10 | 01 | 00 |
| B | 01 | 00 | 00 | 01 | 01 | 10 |
| C | 10 | 00 | 10 | 00 | 10 | 01 |
| | 11 | dd | dd | dd | dd | dd |

Arbiter Output Expressions

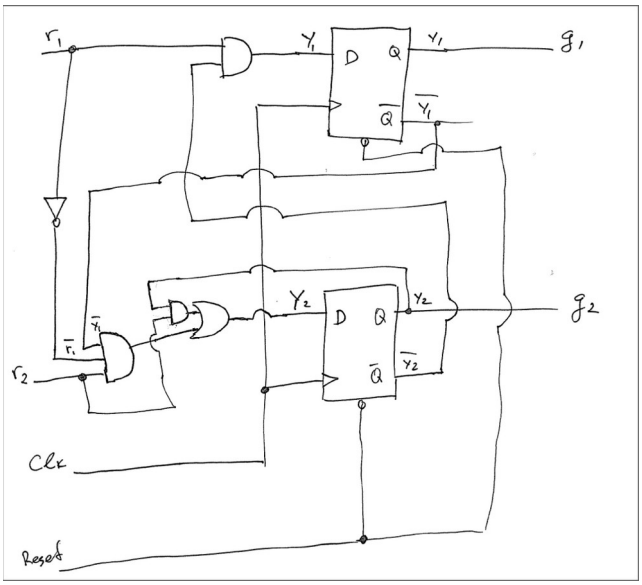
Output expressions

$g_1 = Y_1$

$g_2 = Y_2$



Arbiter Circuit Diagram




# State Minimization:

- 1. Group all states by their outputs
- 2. Ensure that all for all k (every set of inputs), for each group G, each state in G has the same k-successor (next state for input k). If not, then partition G by the next-state that k-leads to.
- 3. Repeat (2.) until no more partitions are made

# SSC Circuit Analysis:

- 1. Find the logic expressions for all next state variables and all outputs.
- 2. Derive the state-assigned table from these logic expressions
- 3. Derive the state table from the state-assigned table
- 4. Derive the state diagram from the state table

Register Machine Instructions

- INC reg, next_step
- DEB reg, next_step, branch_to_if_zero
- END

Example Register Machine Program

| STEP | INSTRUCTION | REGISTER | GO TO STEP | [BRANCH TO STEP] |
|---|---|---|---|---|
| 1. | Deb | 1 | 2 | 3 |
| 2. | Inc | 2 | 1 | |
| 3. | End | | | |