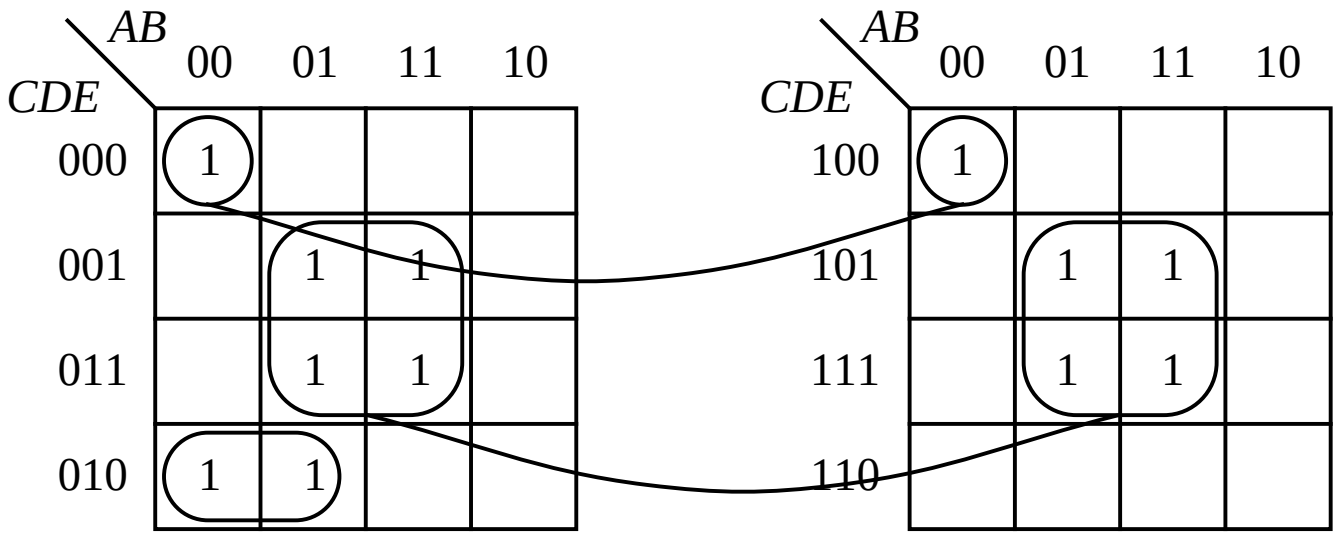
CMSC 313 Lecture 21

- Last time: Karnaugh Map Examples

- Higher Dimension Karnaugh Maps
- Quine-McCluskey Algorithm
- Introduction to Sequential Logic
- Flip-Flops

Five-Variable K-Map

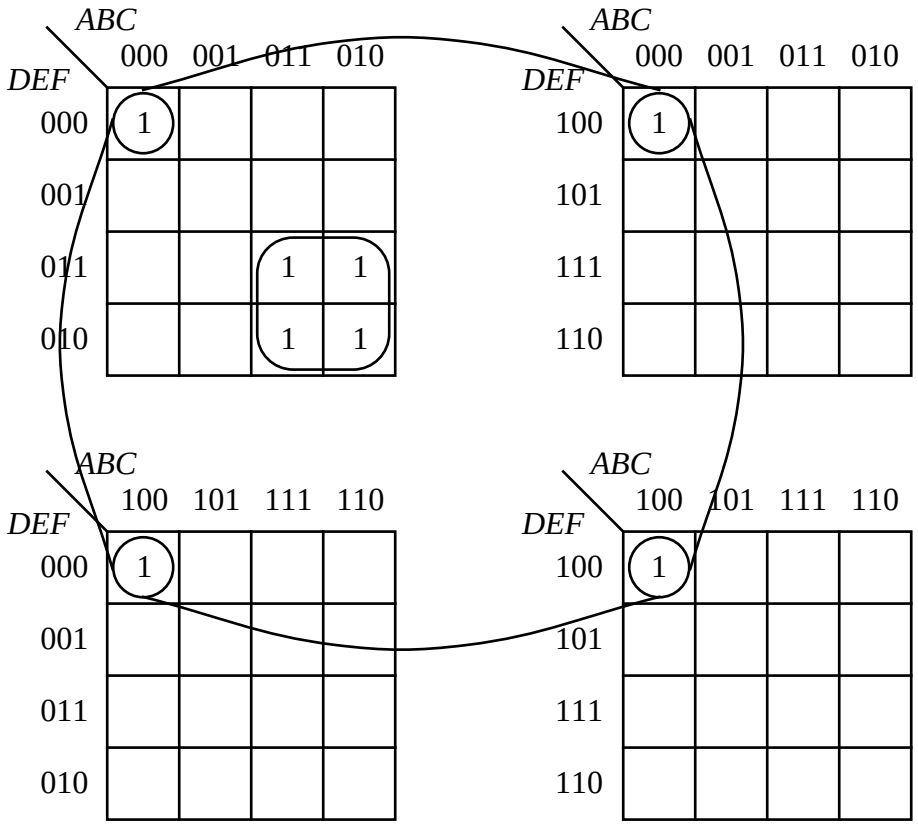
- **Visualize two 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.**



$$F = \overline{A}\,\overline{C}\,D\,\overline{E} + \overline{A}\,\overline{B}\,\overline{D}\,\overline{E} + B\,E$$

Six-Variable K-Map

- **Visualize four 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.**



$$G = \overline{B}\,\overline{C}\,\overline{E}\,\overline{F} + \overline{A}\,B\,\overline{D}\,E$$

Truth Table with Don't Cares

- **A truth table repre-sentation of a single function with don't cares.**

A	*B*	*C*	*D*	*F*
0	0	0	0	*d*
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	*d*
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	*d*

Tabular (Quine-McCluskey) Reduction

- **Tabular reduction begins by grouping minterms for which *F* is nonzero according to the number of 1's in each minterm. Don't cares are considered to be nonzero.**
- **The next step forms a consensus (the logical form of a cross product) between each pair of adjacent groups for all terms that differ in only one variable.**

Initial setup

A	*B*	*C*	*D*	
0	0	0	0	√
0	0	0	1	√
0	0	1	1	√
0	1	0	1	√
0	1	1	0	√
1	0	1	0	√
0	1	1	1	√
1	0	1	1	√
1	1	0	1	√
1	1	1	1	√

(a)

After first reduction

A	*B*	*C*	*D*	
0	0	0	_	*
0	0	_	1	√
0	_	0	1	√
0	_	1	1	√
_	0	1	1	√
0	1	_	1	√
_	1	0	1	√
0	1	1	_	*
1	0	1	_	*
_	1	1	1	√
1	_	1	1	√
1	1	_	1	√

(b)

After second reduction

A	*B*	*C*	*D*	
0	_	_	1	*
_	_	1	1	*
_	1	_	1	*

(c)

Table of Choice

- **The prime implicants form a set that completely covers the function, although not necessarily minimally.**
- **A *table of choice* is used to obtain a minimal cover set.**

Prime Implicants	Minterms						
	0001	0011	0101	0110	0111	1010	1101
0 0 0 _	√						
* 0 1 1 _				√	√		
* 1 0 1 _						√	
0 _ _ 1	√	√	√		√		
_ _ 1 1		√			√		
* _ 1 _ 1			√		√		√

Reduced Table of Choice

- **In a reduced table of choice, the essential prime implicants and the minterms they cover are removed, producing the *eligible set*.**
- ***F = $\bar{A}$BC + A$\bar{B}$C + BD + $\bar{A}$D***

Eligible Set		Minterms	
		0001	0011
X	0 0 0 _	√	
Y	0 _ _ 1	√	√
Z	_ _ 1 1		√

Set 1	Set 2
0 0 0 _	0 _ _ 1
_ _ 1 1	

Multiple Output Truth Table

- **The power of tabular reduction comes into play for multiple functions, in which minterms can be shared among the functions.**

Minterm	A	B	C	F_0	F_1	F_2
m_0	0	0	0	1	0	0
m_1	0	0	1	0	1	0
m_2	0	1	0	0	0	1
m_3	0	1	1	1	1	1
m_4	1	0	0	0	1	0
m_5	1	0	1	0	0	0
m_6	1	1	0	0	1	1
m_7	1	1	1	1	1	1

Multiple Output Table of Choice

$F_0(A,B,C) = \bar{A}\bar{B}\bar{C} + BC$

$F_1(A,B,C) = \bar{A}C + A\bar{C} + BC$

$F_2(A,B,C) = B$

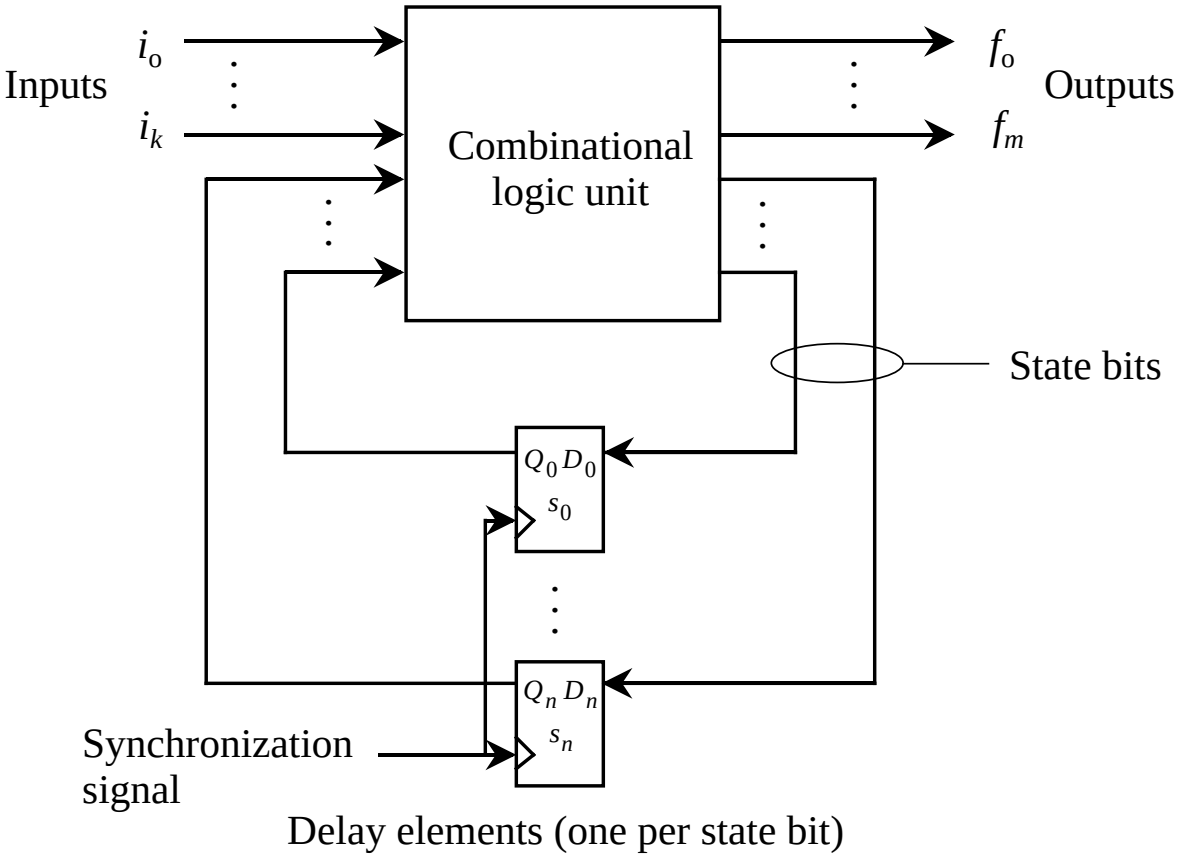
Prime Implicants \ Min-terms		$F_0(A,B,C)$			$F_1(A,B,C)$					$F_2(A,B,C)$			
		m_0	m_3	m_7	m_1	m_3	m_4	m_6	m_7	m_2	m_3	m_6	m_7
F_0	* 0 0 0	√											
F_1	* 0 _ 1				√	√							
F_1	* 1 _ 0						√	√					
F_2	* _ 1 _									√	√	√	√
$F_{1,2}$	1 1 _							√	√			√	√
$F_{0,1,2}$	* _ 1 1		√	√		√			√		√		√

Sequential Logic

- The combinational logic circuits we have been studying so far have no memory. The outputs always follow the inputs.
- There is a need for circuits with memory, which behave differently depending upon their previous state.
- An example is a vending machine, which must remember how many and what kinds of coins have been inserted. The machine should behave according to not only the current coin inserted, but also upon how many and what kinds of coins have been inserted previously.
- These are referred to as *finite state machines*, because they can have at most a finite number of states.

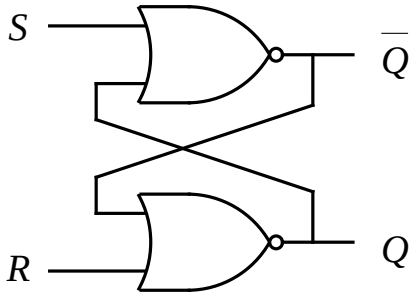
Classical Model of a Finite State Machine

- **An FSM is composed of a combinational logic unit and delay elements (called *flip-flops*) in a feedback path, which maintains state information.**

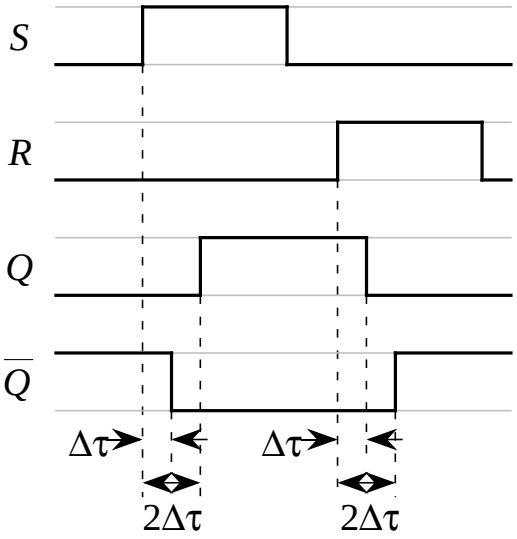


S-R Flip-Flop

- **The S-R flip-flop is an active high (positive logic) device.**

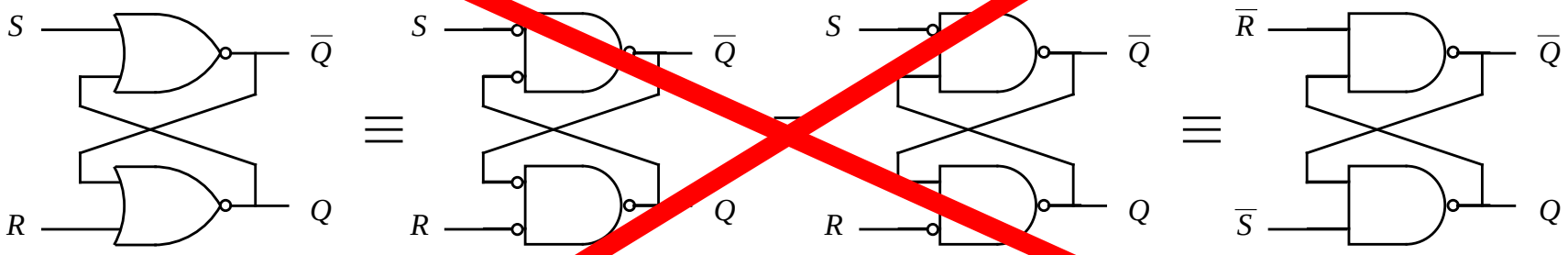


Q_t	S_t	R_t	Q_{i+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	(disallowed)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	(disallowed)

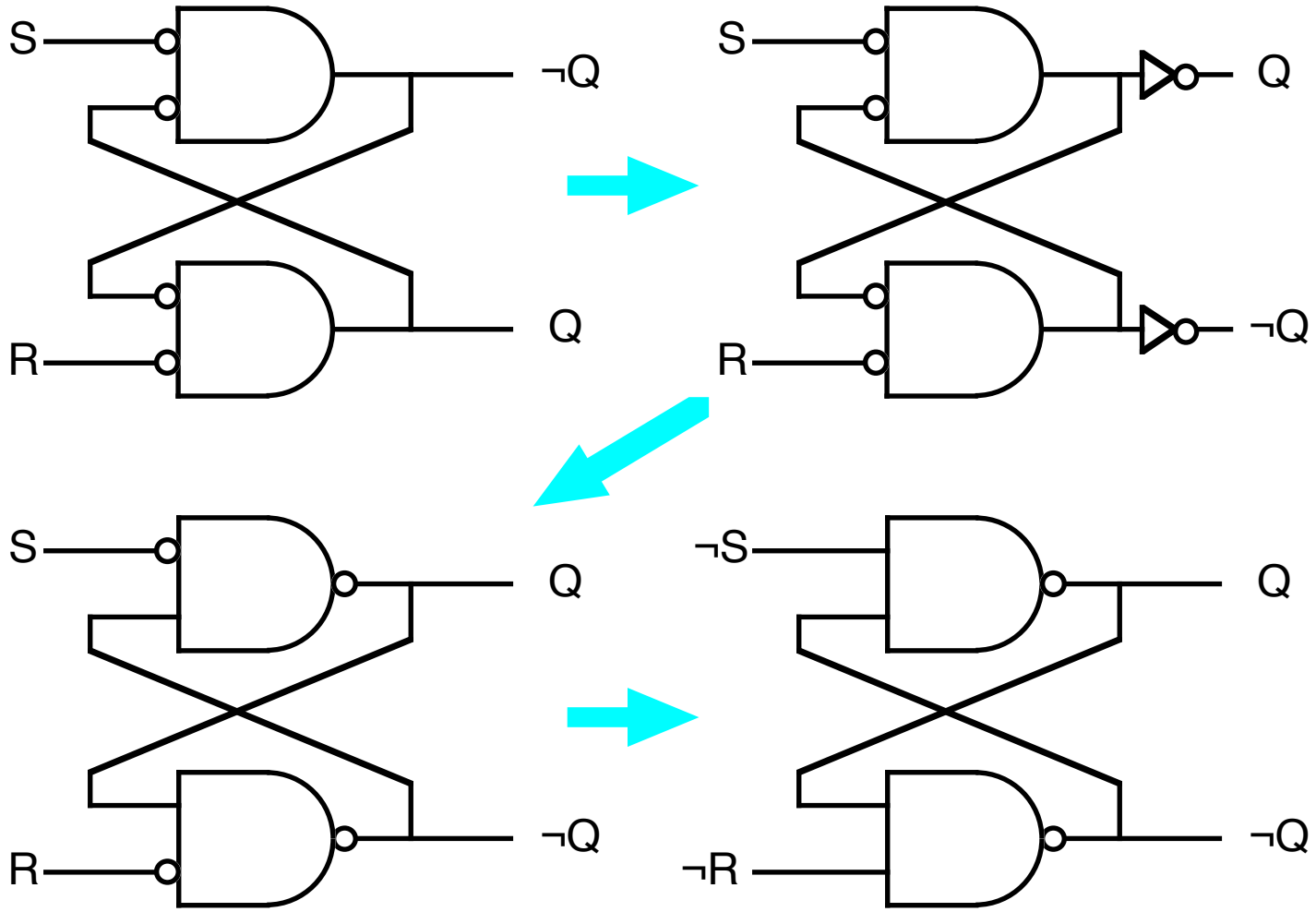


Timing Behavior

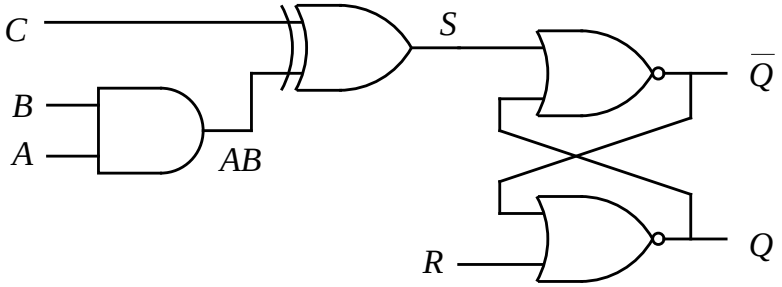
NAND Implementation of S-R Flip-Flop

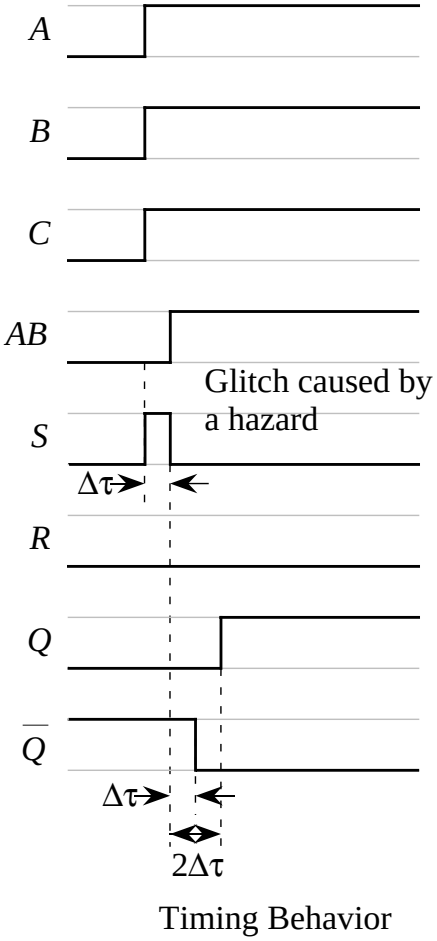


SR Latch using NAND GATES



A Hazard

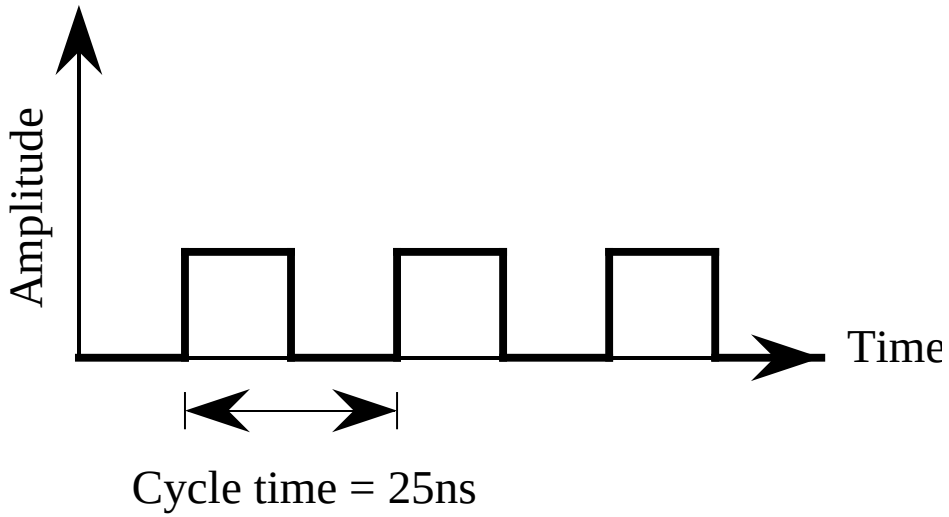




Timing Behavior

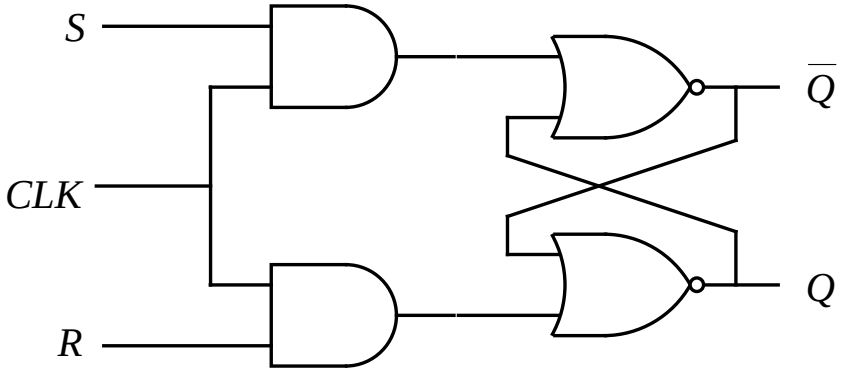
- **It is desirable to be able to "turn off" the flip-flop so it does not respond to such hazards.**

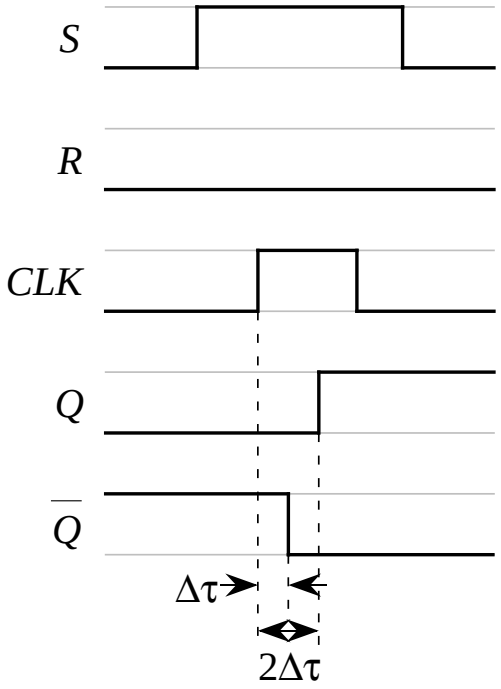
A Clock Waveform: The Clock Paces the System



- **In a positive logic system, the “action” happens when the clock is high, or positive. The low part of the clock cycle allows propaga-tion between subcircuits, so their inputs settle at the correct value when the clock next goes high.**

Clocked S-R Flip-Flop



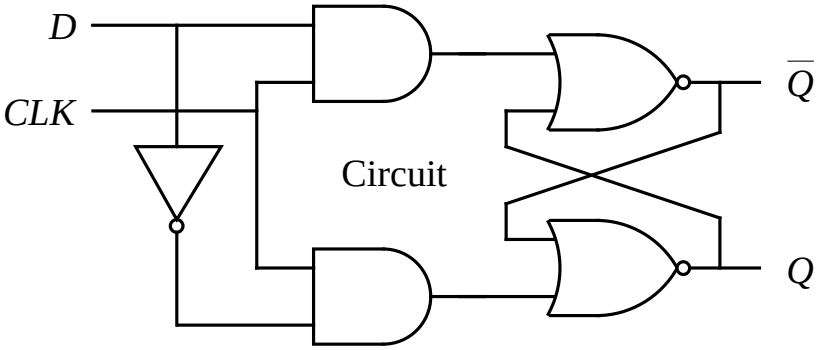


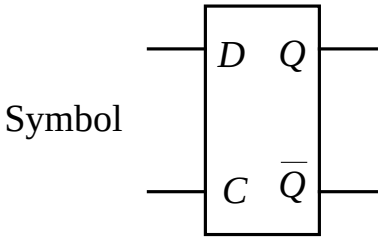
Timing Behavior

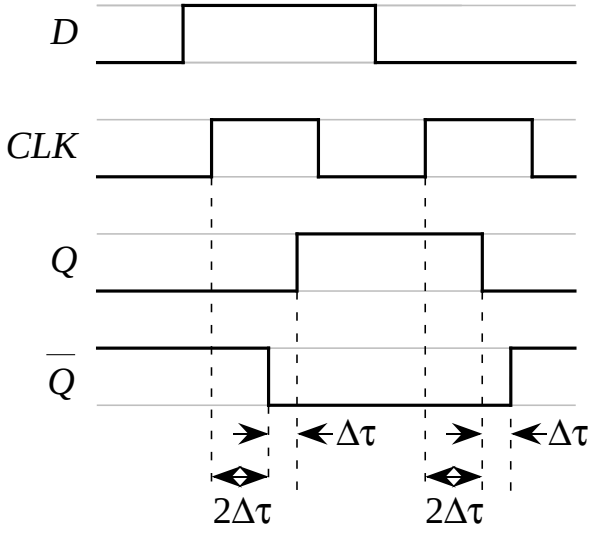
- **The clock signal, CLK, enables the S and R inputs to the flip-flop.**

Clocked D Flip-Flop

- **The clocked D flip-flop, sometimes called a *latch*, has a potential problem: If D changes while the clock is high, the output will also change. The *Master-Slave* flip-flop (next slide) addresses this problem.**



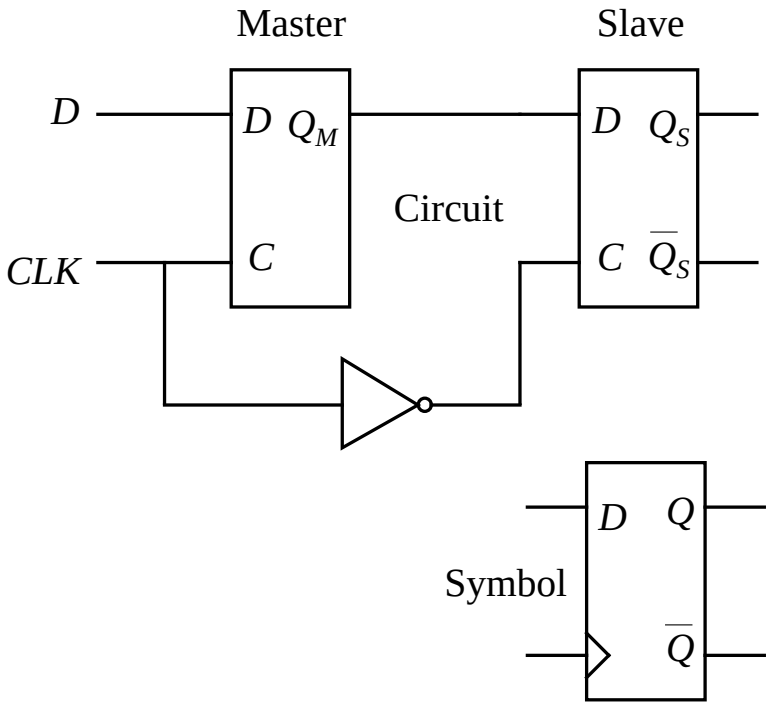


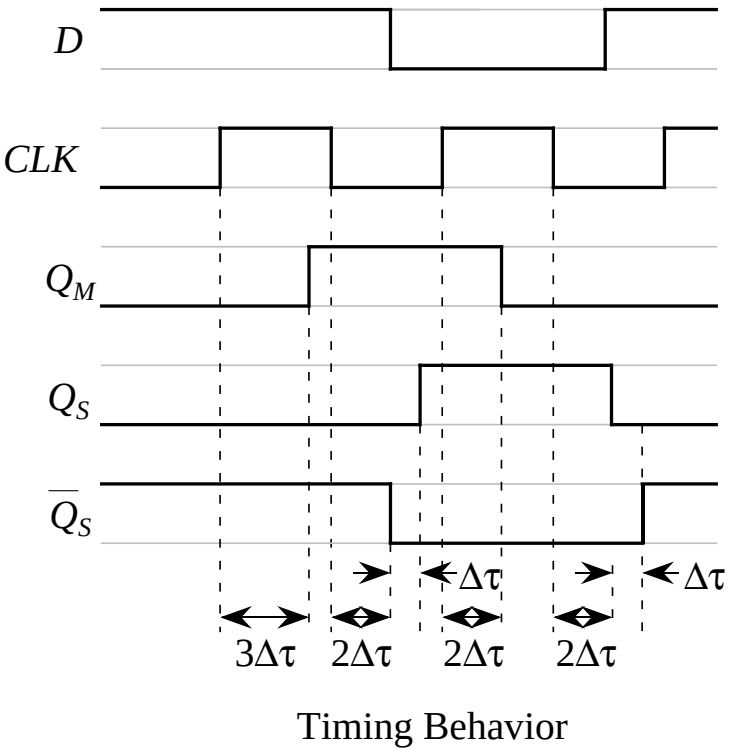


Timing Behavior

Master-Slave Flip-Flop

- **The rising edge of the clock loads new data into the master, while the slave continues to hold previous data. The falling edge of the clock loads the new master data into the slave.**





Timing Behavior

Next Time

- Edge-triggered Flip Flops
- Introduction to Finite State Machines