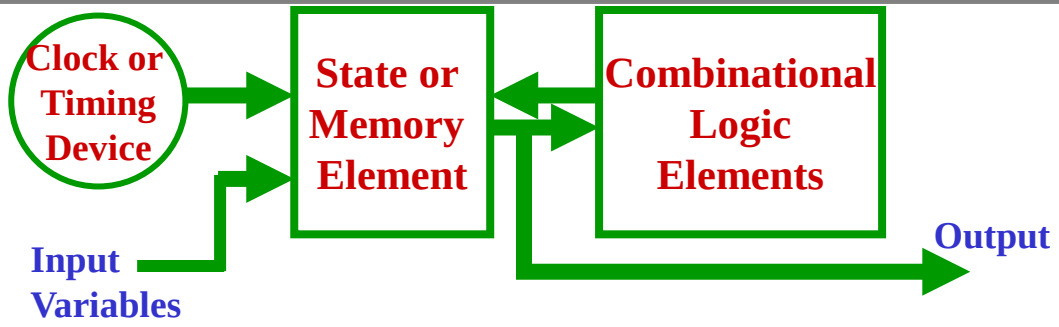
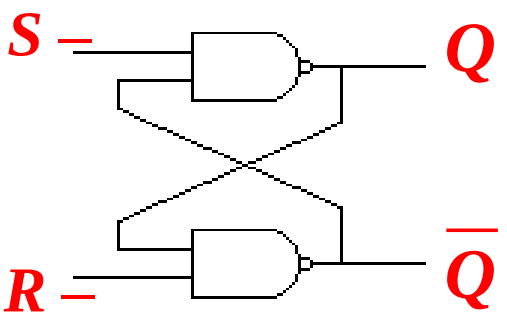
Sequential Logic and Clocked Circuits



- From combinational logic, we move on to sequential logic.
- Sequential logic differs from combinational logic in several ways:
  - Its outputs depend not only on logic inputs but also the internal state of the logic.
  - Sequential logic output does not necessarily change when an input changes, but is synchronized to some "triggering event."
  - Sequential logic is often synchronized or triggered by a series of regular pulses on a serial input line, which is referred to as a "clock." That is, the outputs normally change as a function of the timing element.
  - Sequential logic may (usually will) have combinational parts.

The Simple R-S Flip-Flop

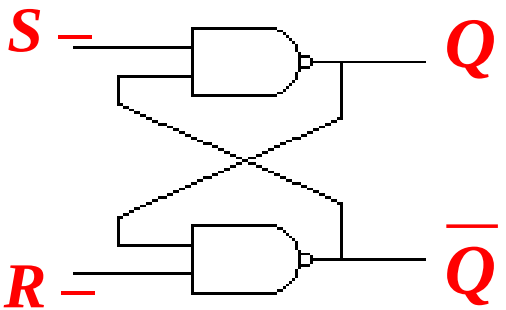
- **The simplest example of a sequential logic device is the R-S flip-flop (R-S FF).**
- **This is a non-clocked device that consisting of two cross-connected 2-input NAND gates (may also be made from other gates).**
- **Inputs are "negative-true" input logic (inputs are active at logic state 0).**
- **When $Q = 1$; $\overline{Q} = 0$, the R-S FF is "set."**
- **Likewise, when $Q = 0$; $\overline{Q} = 1$, the flip-flop is "reset."**
- **Q and $\overline{Q}$ will always have opposite states.**
- **The R-S FF is bistable. That is, it is "at rest" in either of its two states ("set" or "reset") until an input forces it to change.**



R-S Flip-Flop

The Simple R-S Flip-Flop (2)

- If the R-S FF is in the "set" state, it will not go "reset" until the Reset line goes "true" (in this case, to 0). Likewise, when "reset", it will not go "set" unless the Set line goes to 0.
- Note also that once "set," if Set goes to 0 more than once, the FF simply stays "set."
- Likewise, when "reset," more Reset's do not affect the circuit; it remains "reset."
- Thus the R-S FF has an output that depends not only on the inputs but the current state.



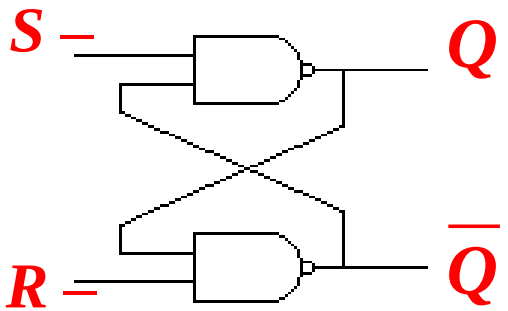
R-S Flip-Flop

Note: The triggering signal of an input (Set or Reset) is always assumed to be momentary. That is, a "Set" or "Reset" signal is considered to last a VERY short time (in the case of real circuits, this is nanoseconds or less).

R-S Flip-Flop Set Cycle

- Assume the ff is reset ($Q = 0$). Also, since the Set and Reset inputs are not active, both input are at 1.
- Thus Set = 1, Reset = 1, $Q = 0$; $\overline{Q} = 1$.
- Then the cycle is:

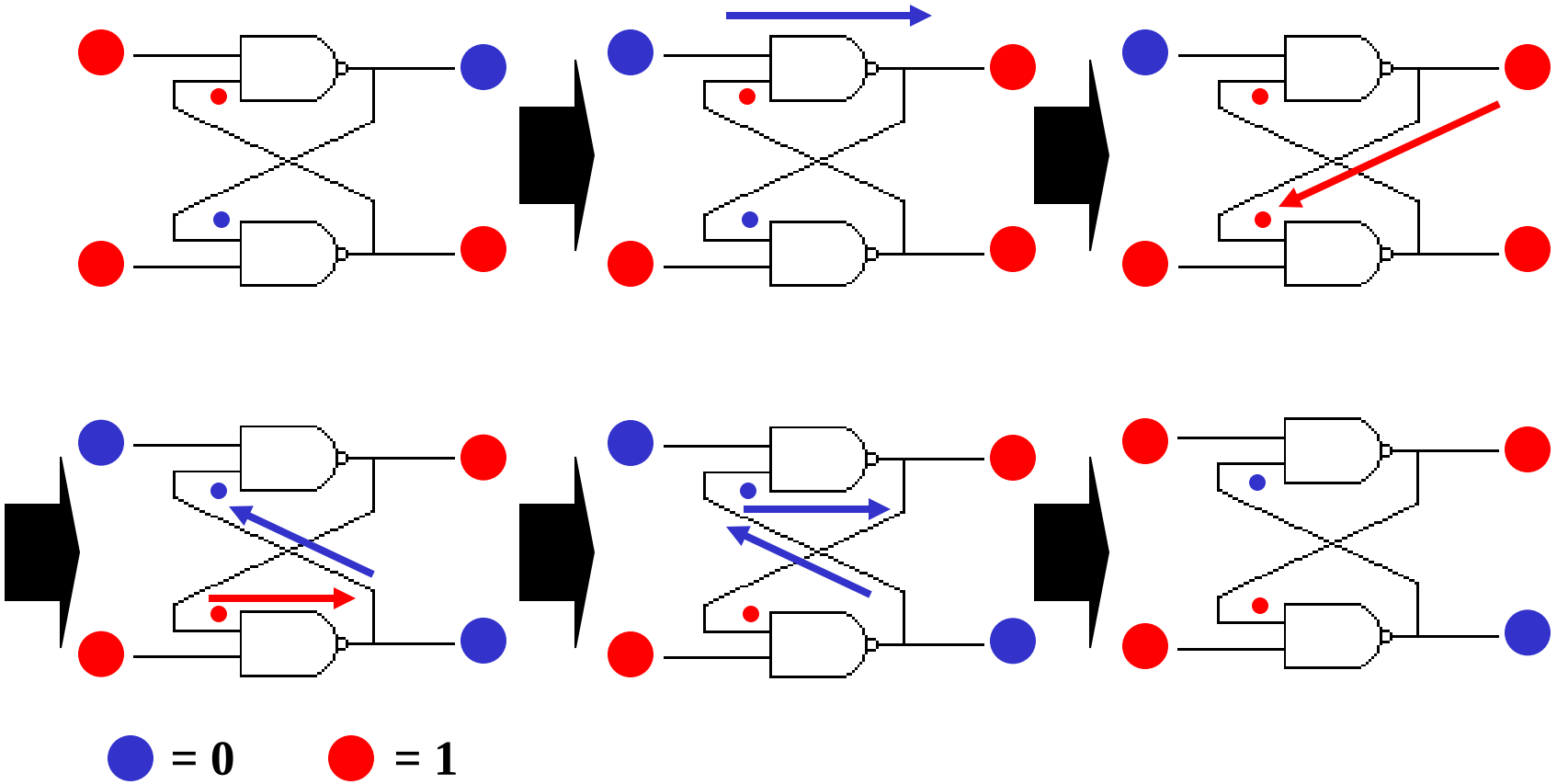
1. Set goes active (Set → 0).
2. Then Q must → 1 (output of a NAND = 1 if any input = 0).
3. Then both inputs to bottom NAND are 1, and $\overline{Q}$ → 0.
4. The other input to the upper NAND is now 0. Thus, when the Set signal goes back high, Q remains at 1, since the other input is still 0.



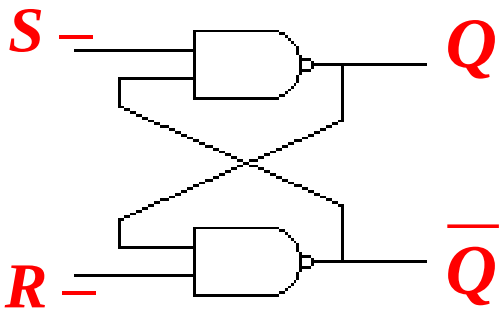
R-S Flip-Flop

5. Likewise, since both inputs to the lower NAND are now 1, then the value of Q remains 0.
6. The reverse cycle (set-to-reset) occurs in the identical way, except that the change is initiated by reset going low.

RS FF "Set" Cycle Pictorially



The Simple Latch or R-S Flip-Flop (4)



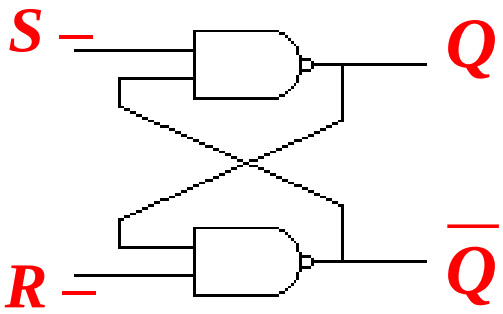
R-S Flip-Flop

- **The truth table for the R-S FF is relatively simple, as shown below:**

Inputs		Current Outputs		New Outputs	
S	*R*	*Q*	$\overline{Q}$	*Q*	$\overline{Q}$
1	1	1 or 0	0 or 1	Same	Same
0	1	0	1	1	0
0	1	1	0	Same	Same
1	0	1	0	0	1
1	0	0	1	Same	Same
0*	0*	1 or 0	0 or 1	1*	1*

* This is a race condition and not stable. The S-R input "0-0" is "forbidden."

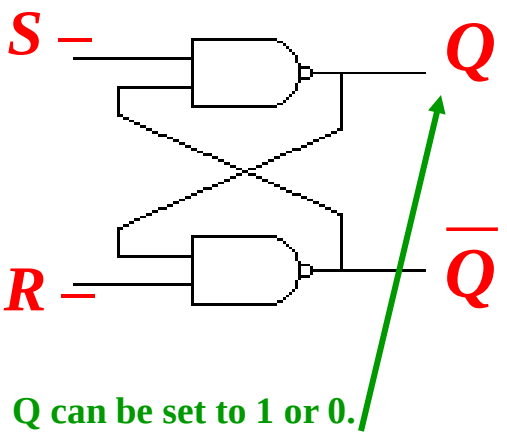
"Forbidden" RS FF Inputs



- As noted on the truth table, 0-input to both *R* and *S* is forbidden.
- Note the race condition that is triggered by *R*=*S*=0:
  - Then $Q = \overline{Q} = 1$, so both other 2-NAND inputs are 1.
  - If *S* and *R* go to 1 simultaneously, then all 4 inputs of the two 2-input NAND gates are 1 and both outputs go to 0(!)
  - The result is a "race" to see which output gets to 0 first, getting one 2-NAND input to 0, and therefore forcing that NAND output to 1. The result of the "race" cannot be predicted. Thus R and S = 0 together is forbidden, since the output state is not stable.

Flip-Flops and "Memory"

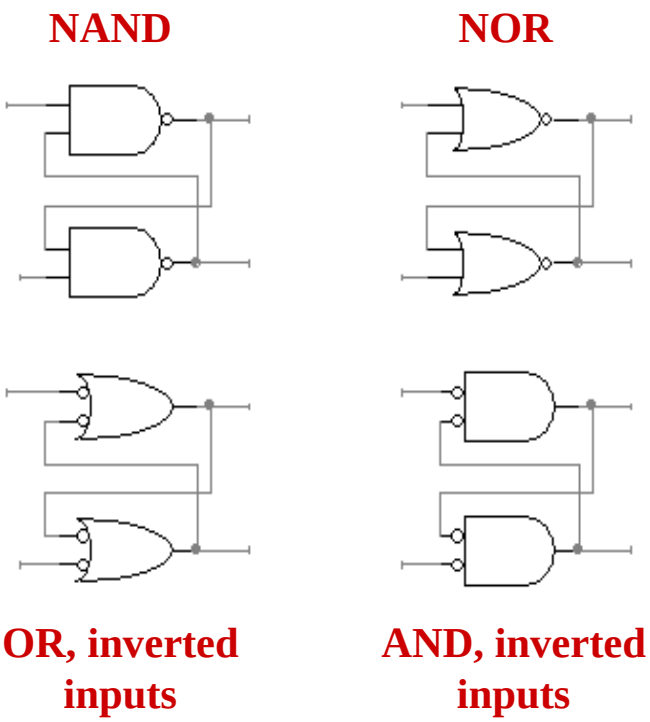
- **Many circuits in the modern computer are either based on or related to the R-S FF.**
- **If an RS FF has its *Q* output changed to 1 or 0, the output stays in that state until the opposite input is triggered.**
- **Thus the RS flip-flop or latch has the property of "remembering" a one-bit binary number: It can be set to 1 or 0.**
- **Thus, we can use flip-flops to store binary numbers in a computer.**
- **Groups of flip-flops that store large binary numbers are called registers.**



Q can be set to 1 or 0. Thus the output Q can be said to "remember" one bit.

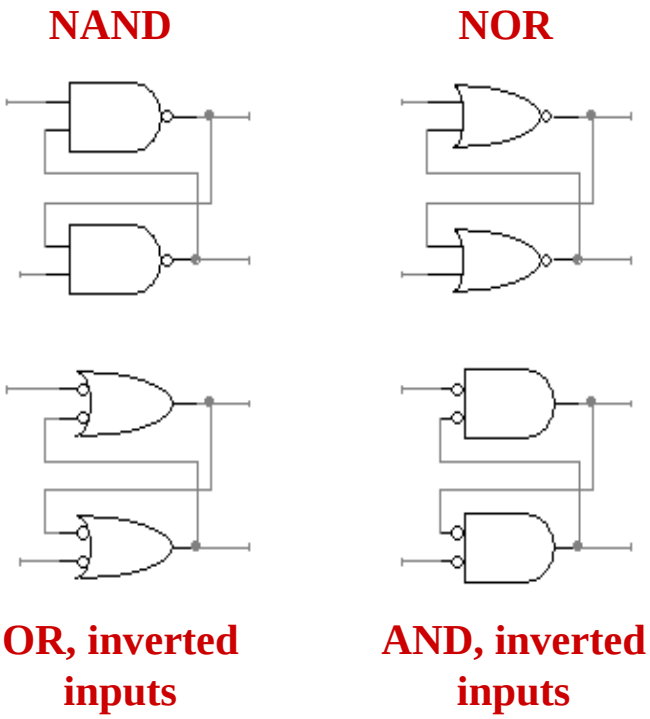
Simple Flip-Flops

- **Either two-input ANDs or ORs with may be used to create a simple RS FF.**
- **However, either the inputs or the output must be inverted.**
- **The examples to the left show the possible RS FF's that can be constructed from 2-input gates. → → → → → →**



Simple Flip-Flops (2)

- **Even though any of these four constructs can operate as a FF, their inputs are slightly different, and the labeling of R and S varies.**
- **However, there is a foolproof method to analyze any 2-input-port, 2-output FF to determine its correct operation.**

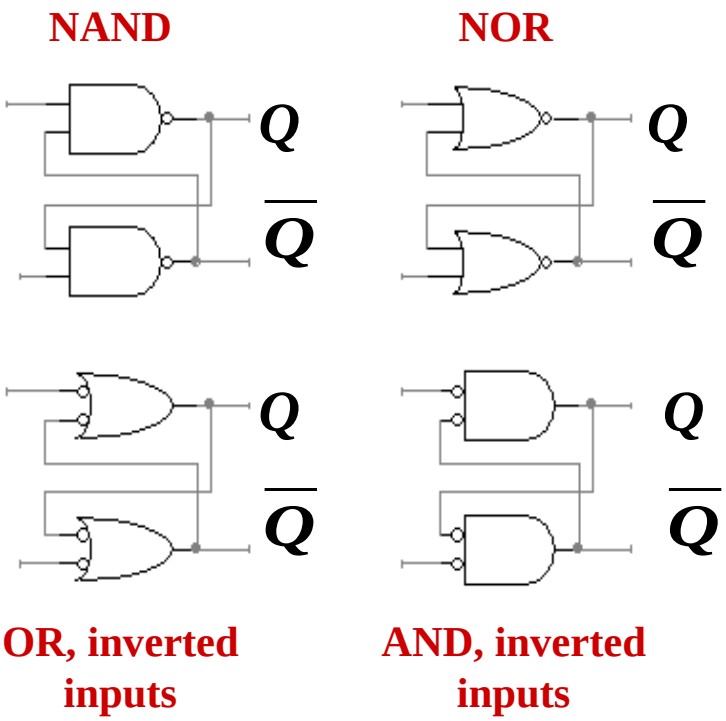


Analyzing Flip-Flop Operation

- **There is a 100%, absolutely-guaranteed method to analyze ANY of the basic flip-flops and determine its correct operation.**
- **It is a 3-step method that can easily show you how a 2-gate flip-flop operates—what inputs trigger it and how its states change.**
- **It depends on analyzing the flip-flop based on the fact that, from combinational logic theory, we know exactly how each of the four gate types shown earlier operates.**

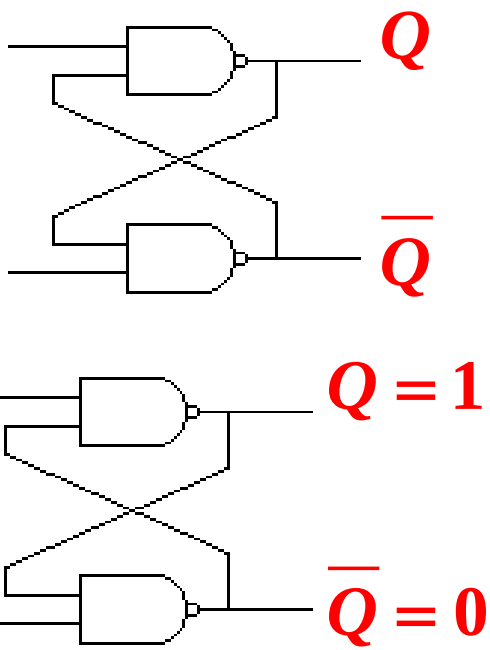
Analyzing Flip-Flop Operation (2)

- **Flip-flop analysis method:**
  1. **Remember that *Q* is always the top output. $\overline{Q}$ is always the bottom output.**
  2. **Three analysis steps:**
     - a) **Assume a state for the output (either Set or Reset).**
     - b) **Assume a quiescent state for the inputs (both will either be 0 or 1 when not activated).**
     - c) **When the quiescent state is validated, let one input be active and observe the result.**



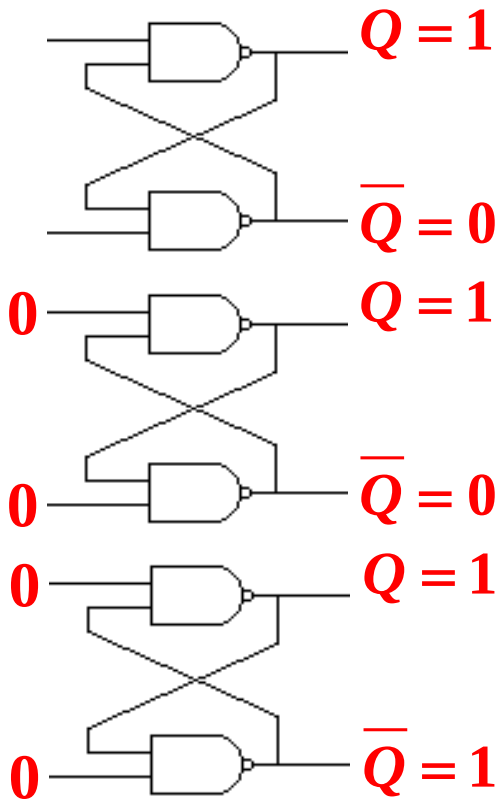
Analyzing Flip-Flop Operation (3)

- **Let re-analyze our original NAND-gate RS FF using the method:**
  1. **First, we remember that *Q* and *Q*-NOT are as shown—*Q* always on top.**
  2. **Now we need to assume a state. The choice is irrelevant, but let's assume the FF is SET.**
  3. **Thus the FF looks as shown in the second diagram, below.**



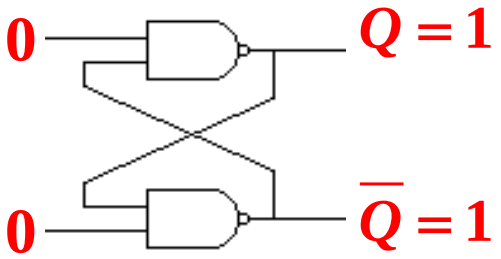
Analyzing Flip-Flop Operation (4)

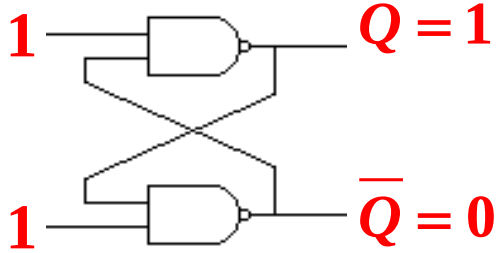
- **Remember: at this point, we do not know what input is "SET" and which is "RESET."**
  4. **Now, lets assume a state for both inputs when quiescent (inactive).**
  5. **Again, the choice doesn't matter, but lets assume both inputs are at logic 0. (second diagram).**
  6. **Whoa! If we do that, then (remembering how a NAND works), both FF outputs must be 1 (third diagram)!**



Analyzing Flip-Flop Operation (5)

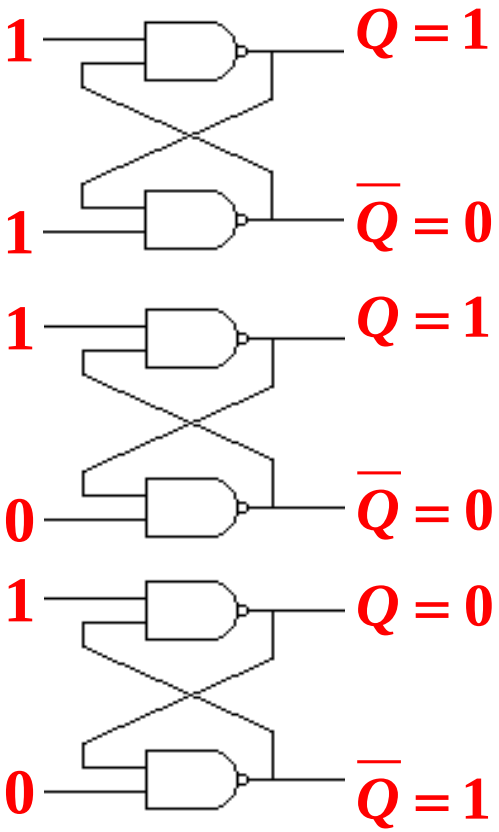
- **Analysis continued:**
  7. **Since an RS FF is always SET or RESET, (both stable states), the inactive state for the inputs cannot be 0, as the RS FF state is not possible when the inputs are inactive.**
  8. **Therefore the inputs must be logic 1 in the inactive state (second diagram).**
  9. **Checking, we see that with both inputs at logic 1, the RS FF output is stable, as we assumed. Had we assumed that the inputs were logic 1 originally, we would have verified that level, as the outputs of the RS FF would have been proper.**





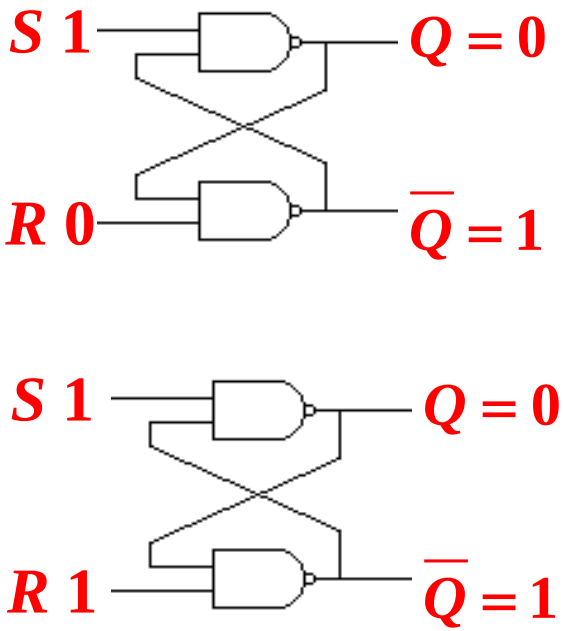
Analyzing Flip-Flop Operation (6)

- **Analysis continued:**
  10. **Having completed Step 2, we know that the flip-flop is stable, and that its inputs are logic 1 when quiescent or inactive.**
  11. **For the third step, we now let one of the inputs become active. Again, the choice is arbitrary, but lets let the bottom input → 0 (second diagram).**
  12. **We note immediately that any NAND gate with a 0 input has an output of 1.**
  13. **Then the upper NAND has two 1-inputs. Thus its output → 0 (third diagram).**



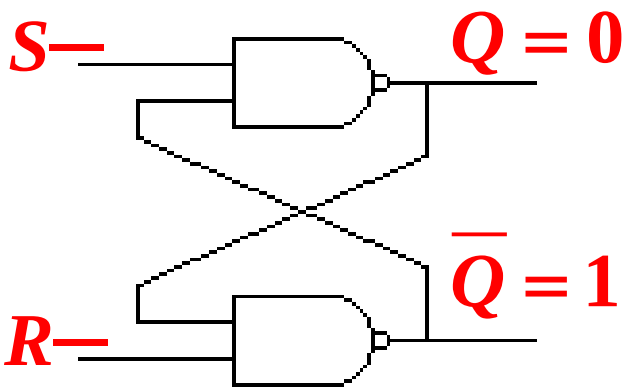
Analyzing Flip-Flop Operation (7)

- The RS FF has been RESET. Thus the bottom input must be RESET and the top SET (top diagram).
- Note that the lower NAND gate will NOT change its output, as the output is 1 whether one or both inputs are 0.
- Also note that the RESET signal is a pulse. It will quickly end. But the FF stays RESET (bottom diagram).



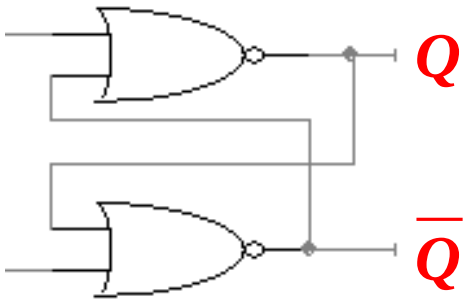
Analyzing Flip-Flop Operation (8)

- **We have successfully analyzed the NAND-gate, RS FF.**
  - **Its inputs are normally quiescent or inactive at logic 1.**
  - **They activate by going to logic 0.**
  - **The upper input activates the SET condition.**
  - **The lower input activates RESET.**
  - **Since the inputs go to 0 (i.e., in the negative direction) to activate, we label them R— and S— (diagram).**



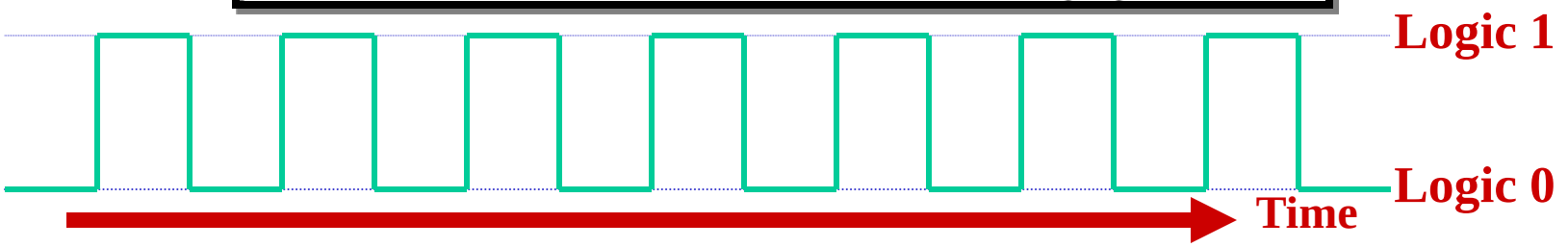
Exercise 1

- This exercise can improve your understanding of RS flip-flops AND our analysis method.
- The circuit below is one of the RS flip-flops that we saw earlier.
- Q and $\overline{Q}$ are labeled. Complete the truth table and label R and S (including their polarity).



Inputs		Current Outputs		New Outputs	
1	2	Q	$\overline{Q}$	Q	$\overline{Q}$
0	0	1 or 0	0 or 1		
0	1	0	1		
0	1	1	0		
1	0	1	0		
1	0	0	1		
1	1	1 or 0	1 or 0		

Clocked Circuits (1)

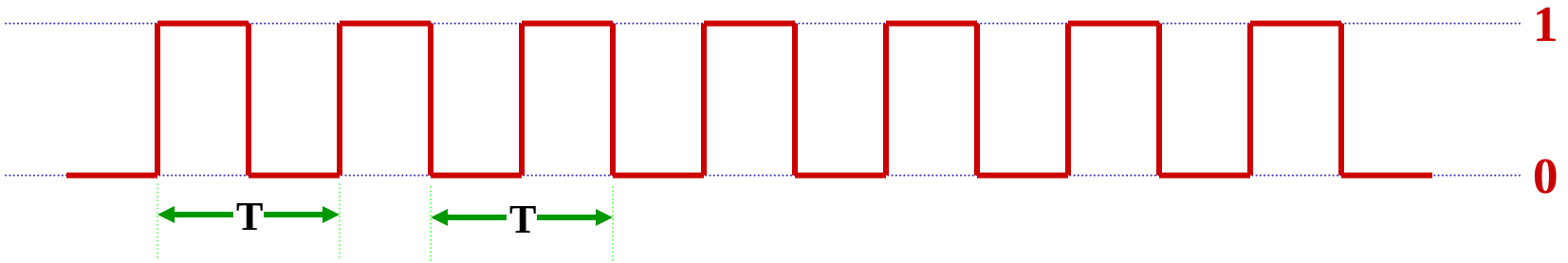


Stream of Clock Pulses ("Square Wave" or "Pulse Train")

- The majority of all sequential logic circuits are clocked logic circuits.
- Clocked circuits are circuits that are regulated by a clocking element, ("square wave"), which determines when state changes occur.
- In a clocked sequential circuit, in general, the circuit can only change states on a "tick" of the clock element.
- We refer to a circuit as a "clocked circuit" when sequential elements in the circuit change states in synchronization to a train of pulses.
- Such a "pulse train" is shown below.
- The clock pulses change regularly from 0 to 1 and back.

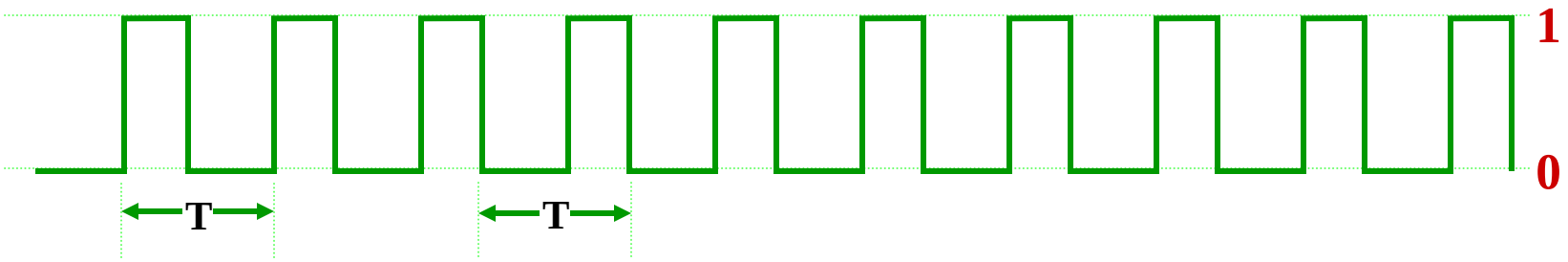
Clocked Circuits (2)

- **The period T of a sequential logic clock is the distance between identical points on the pulse train, e.g., two rising edges or two falling edges.**
  - **The clock has only two states: 0 and 1.**
  - **The clock alternates between the states.**
- **The amount of time that the clock spends in each of the two states is called the duty cycle.**
- **The clock below has a 50/50 duty cycle; it spends equal time each period in the 1 and 0 states.**



Clocked Circuits (3)

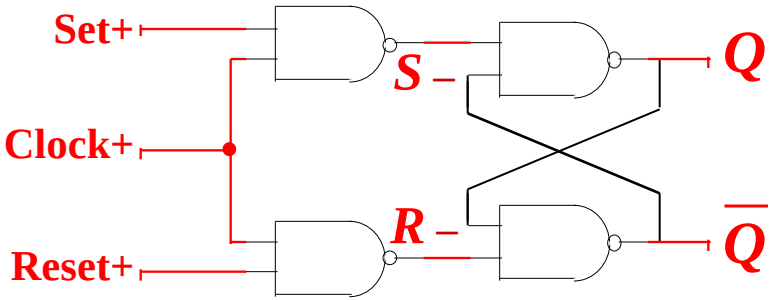
- The clock below does not have a 50/50 duty cycle. It stays in the "1" state about 35% of the time, and in the "0" state about 65% of the time.
- Thus we say that the clock has a "35/65" duty cycle. In the same way, a clock can have a 70/30 cycle time (i.e., it stays in the "1" state 70% of the time), and so forth.
- Note that the period T is defined in the same way as before.



Clocked Flip-Flops

- **All ff's have the same basic configuration:**
  - **Both true and false outputs ("*Q*" and "*Q-not*").**
  - **"Set" is when *Q*=1.**
  - **Triggered by "set" and "reset" inputs.**
- **The most useful ff's are not simple asynchronous (non-clocked) ff's, however, but synchronous ("clocked") ff's.**
- **Clocked ff's are very similar to non-clocked ff's -- the main difference is that in addition to a "set" or "reset" input to cause the outputs to change, there must also be the presence of a clock signal in its true state (normally 1).**
- **Thus clocked ff's do not change states, regardless of the set or reset inputs, until the "clock ticks."**

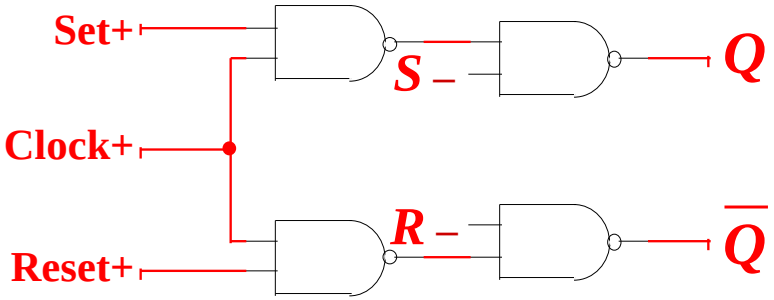
The Clocked R-S Flip-Flop



Clocked R-S Flip-Flop

- The simplest clocked ff is the clocked R-S FF, shown above (NAND version).
- In addition to the set and reset inputs, the clock input is present.
- Since when clock is low (0), neither set or reset input affect the circuit, we say that the clock "gates" the set or reset signal to the RS FF.
- In this case, the set or reset input must be high (1) to set or reset the ff when the clock goes true (0 ⟶ 1).
- Having set and reset 1 at the same time is forbidden as for the RS FF; simultaneous set and reset true causes a race condition when clock is high.

Clocked R-S Flip-Flop Truth Table



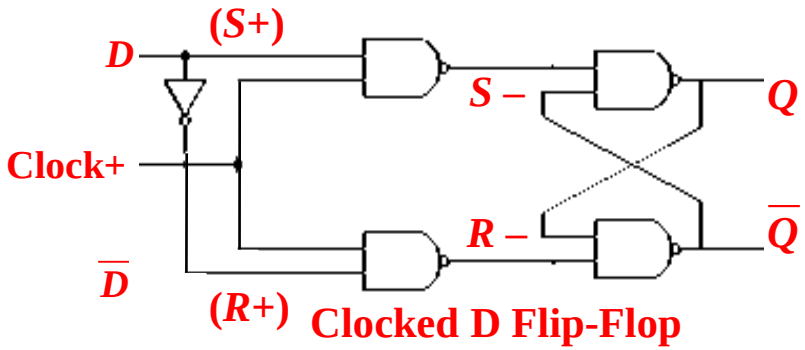
Clocked R-S Flip-Flop

Inputs			Current Outputs		New Outputs	
Clock	S	R	Q	$\overline{Q}$	Q	$\overline{Q}$
0	X	X	1 or 0	0 or 1	Same	Same
1	0	0	1 or 0	0 or 1	Same	Same
1	1	0	0	1	1	0
1	1	0	1	0	Same	Same
1	0	1	1	0	0	1
1	0	1	0	1	Same	Same
1	1*	1*	1 or 0	0 or 1	1*	1*

* This is a race condition and not stable, as for the non-clocked RS FF.

The D Flip-Flop

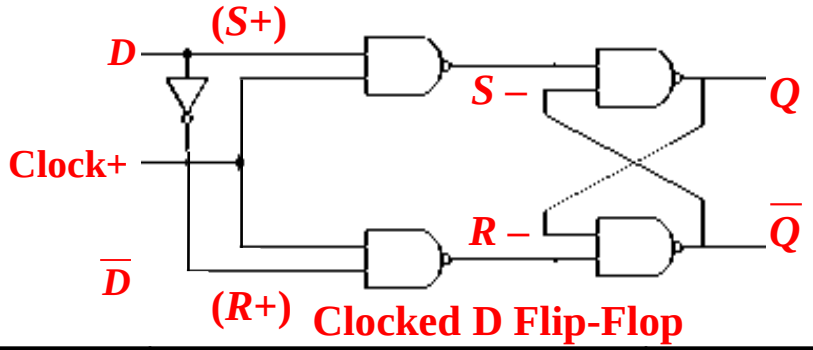
- The D FF is a bistable circuit with only one input plus the clock. The term "D" refers to "data."
- Since the D FF is limited to one input, there is no chance that a race condition will occur.



Clocked D Flip-Flop

- The clocked D FF can be created simply by replacing the "reset" input with an inverted "set" input in the clocked RS FF.
- In this configuration, we can see that if the D input = 1, when the clock is high, the ff will go into the "set" state ("set" input=1, "reset"=0).
- When the D input = 0, then the ff goes into the "reset" state when clock goes high ("set" input=0, "reset"=1). When clock = 0, the ff is idle.
- There is no "forbidden" state here -- the D FF output is defined (see truth table, next page) for ALL inputs.

D Flip-Flop Truth Table

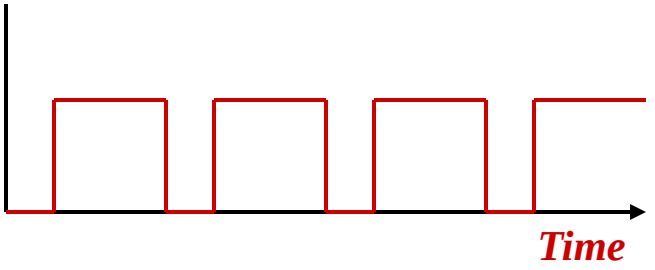


Clocked D Flip-Flop

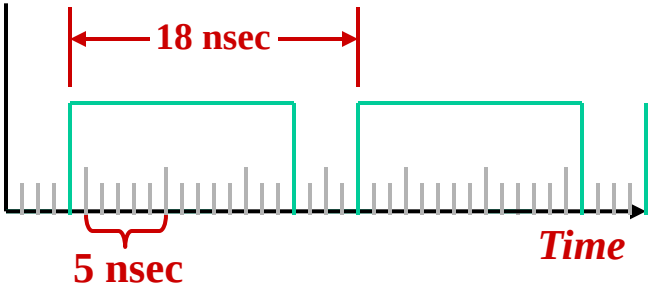
Inputs		Current Outputs		New Outputs	
Clock	D	Q	$\overline{Q}$	Q	$\overline{Q}$
0	X	1 or 0	1 or 0	Same	Same
1	1	0	1	1	0
1	1	1	0	Same	Same
1	0	1	0	0	1
1	0	0	1	Same	Same

Exercise 2

1. **Draw a representation of a 70-30 duty cycle clock (pulse train).**



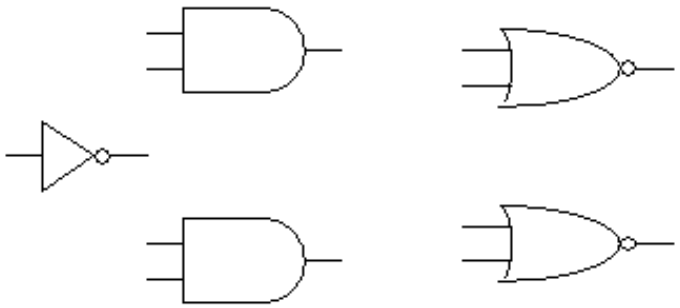
2. **What is the period of the clock shown, to the nearest nsec? What is the duty cycle (nearest %)?**



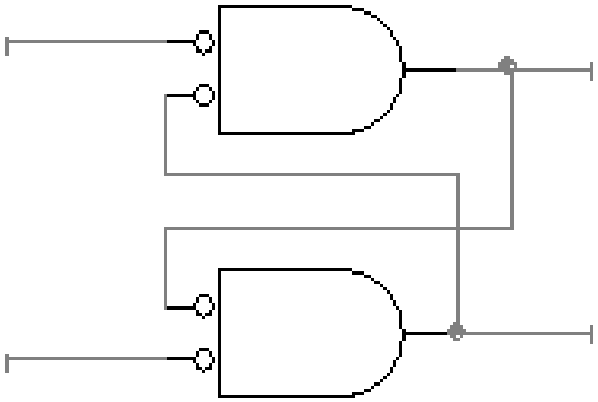
Period = 18 nsec; duty cycle ≈ 78/22

3. **Using the gates shown on the next page, make a clocked D FF. Label inputs and outputs.**

Exercise 2 (continued)

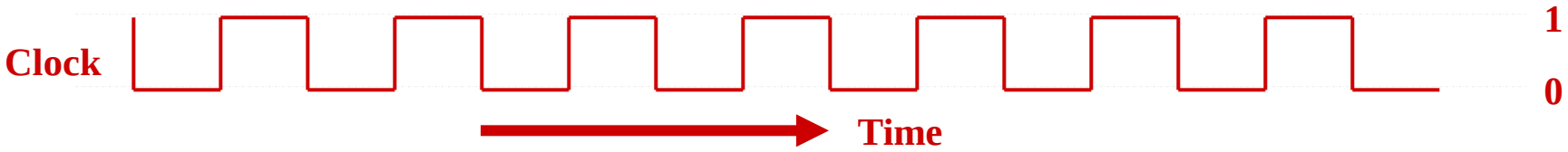


New Information Quiz



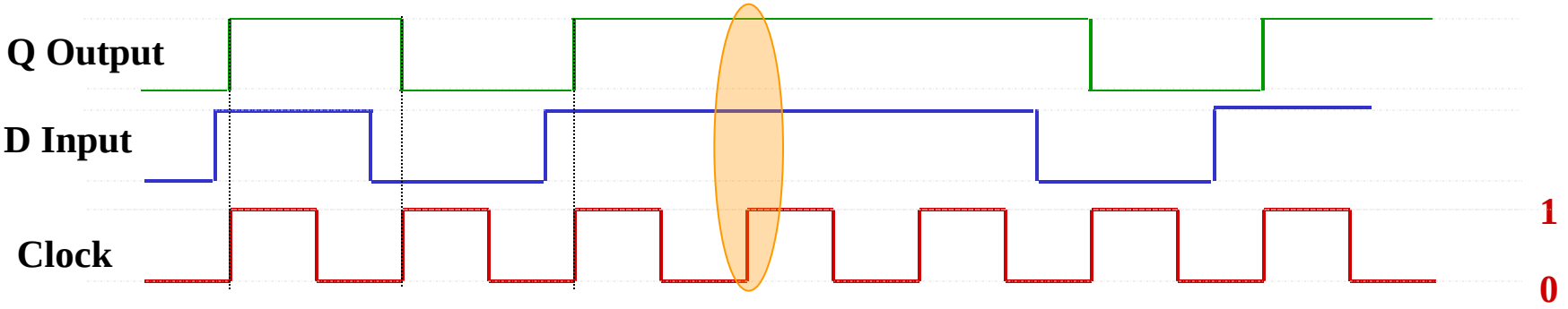
- **This is also one of the four RS FF types we saw earlier. Determine the inputs in the same way the example was worked.**
- **Label the signals Set, Reset, Q, Q-not, and show the input polarities (indicating whether the signals are active high or low).**

Timing Diagrams



- **It is important when designing sequential circuits to understand the timing relationships between circuit elements.**
- **This is normally done by plotting the "transitions" -- the changes between logic "1" and logic "0" -- of the elements of interest.**
- **All event timing is normally related to the clock.**
- **In any fundamental timing diagram, unless specifically instructed to do otherwise, always start with the clock (it goes best at the diagram bottom).**
- **Always assume a "50/50" clock, if duty cycle not specified.**

Timing Diagrams (Continued)

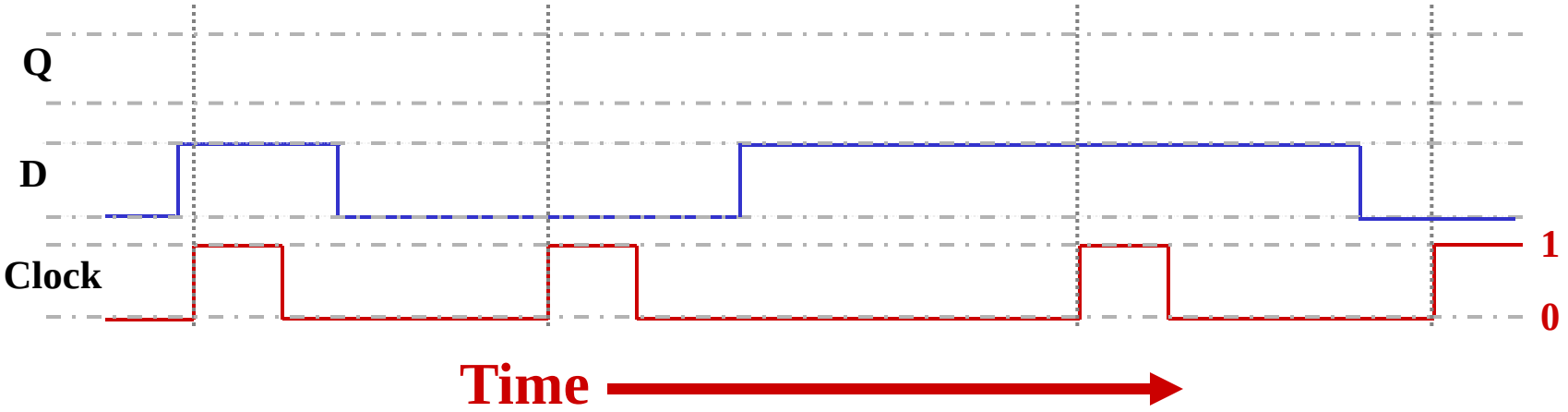


- To illustrate D FF timing by plotting the D FF "Q" output:
  - Remember that a D FF output is normally triggered when clock = 1.
  - Assume D FF is originally "reset." Then on the rising edge of the clock, the D FF output goes to "set" (1), when the D input is 1.
- On successive clock pulses, the D FF "Q" changes as D changes.
- Principle: Output "Q" of a simple D FF tracks D when the clock ticks!

Note: The clock is shown above running continuously. In some cases, the clock may "tick" only some times.

Exercise 3

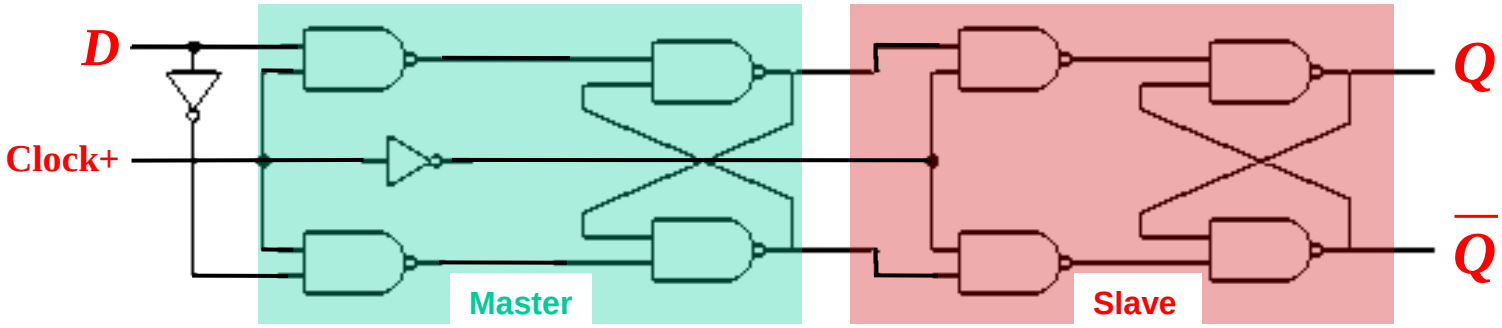
- **The incomplete timing diagram below shows the clock input as the basis for the diagram and a D FF input, but not the output. In this case, the clock does not run continuously, but on a sporadic basis. Based on the discussion so far, plot the timing of the Q output of the D FF (assume the D FF starts out in the reset condition). Note that you do not need to see the ff diagram itself to do the plot.**



"Master-Slave," or Delay Flip-Flops

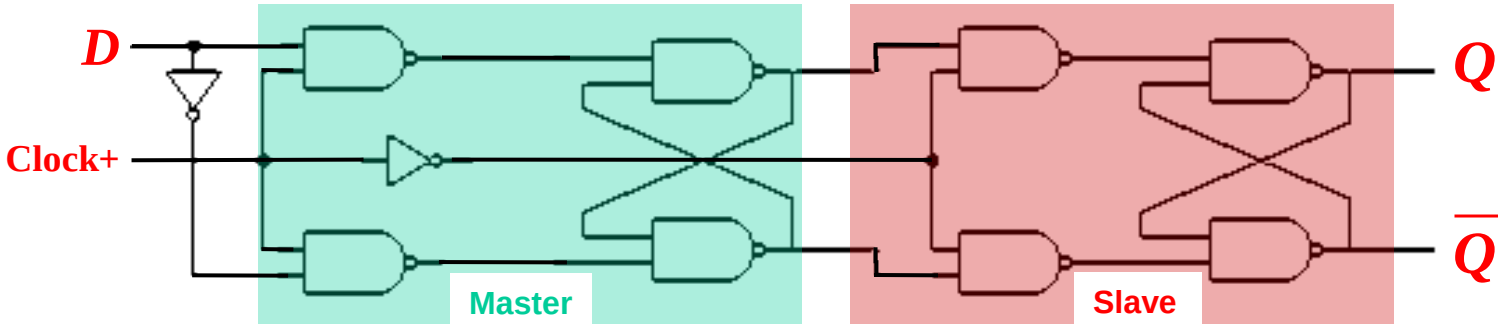
- **It is often desirable to have a flip-flop whose output does not change immediately when its internal state is altered from "set" (Q = 1) to "reset" (Q = 0), or vice-versa.**
- **This sort of ff is called a "master-slave" or "delay" ff.**
- **The idea behind the master-slave ff is to have a "master" (i.e., controlling) ff change states on one edge of a clock pulse (normally the leading edge) and have a second ff connected to the first change to the same state as the "master" on the trailing edge, or "backside" of a clock pulse.**
- **In this way, the internal state of the ff changes one-half clock cycle prior to the time in which the changed state appears on the circuit outputs.**

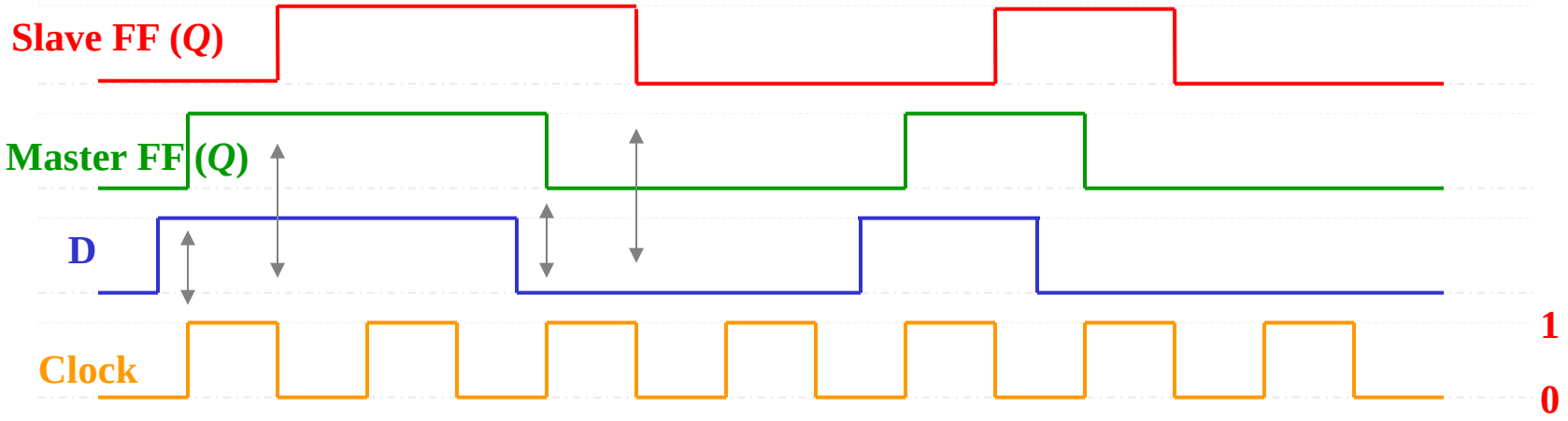
The Master-Slave D Flip-Flop



- D FF converted to master-slave type.
- The slave (basically a clocked RS FF) always mirrors the state of the master.
- The slave circuit changes state 1/2 cycle after the master.
- The device still operates as a D FF; no indeterminate state.

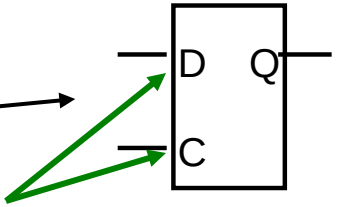
Timing of Master-Slave D Flip-Flop



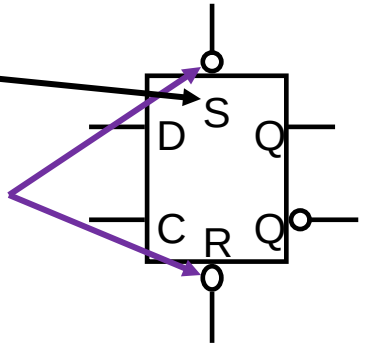


D Flip-Flop Symbols

- **Flip-flop detail is not usually shown in diagrams.**
- **One symbol for a D FF is shown to the right.**
- **There is no small circle on either input. Therefore, "1" is the active state (when clock and D = 1, output will → 1).**
- **D FF's with asynchronous set and reset are also available.**
- **Circles on S and R inputs mean that set and reset are "negative-true" signals (active at level 0).**
  - **"Q-not" output is also available.**
- **Set and reset have the same problems discussed before: If S = R = 0, output may be indeterminate.**

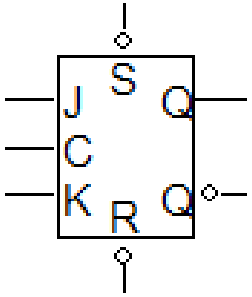


Simple D FF



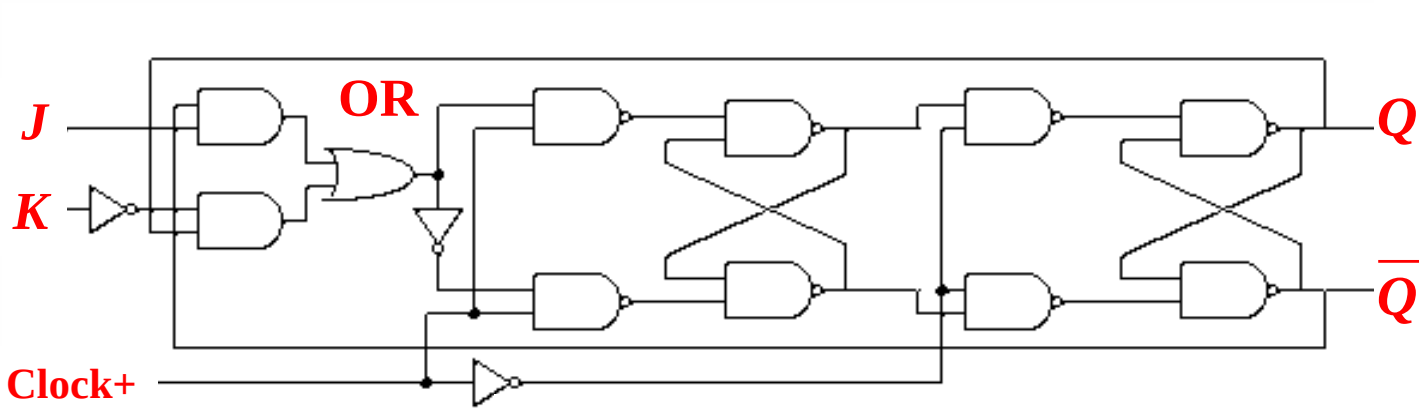
D FF With Asynchronous S/R

The J-K Master-Slave Flip-Flop



- **It is often useful to have a FF that will not have indeterminate outputs when S and R inputs are both 1 simultaneously.**
- **The J-K FF, shown above, fits those requirements. The J input corresponds to "Set," while K corresponds to "Reset."**
- **The J-K FF is designed so that the condition of J = K = 1 does not result in an indeterminate output when clocked.**
- **There may still be asynchronous RS inputs with the usual cautions. However, the J-K inputs are not restricted.**

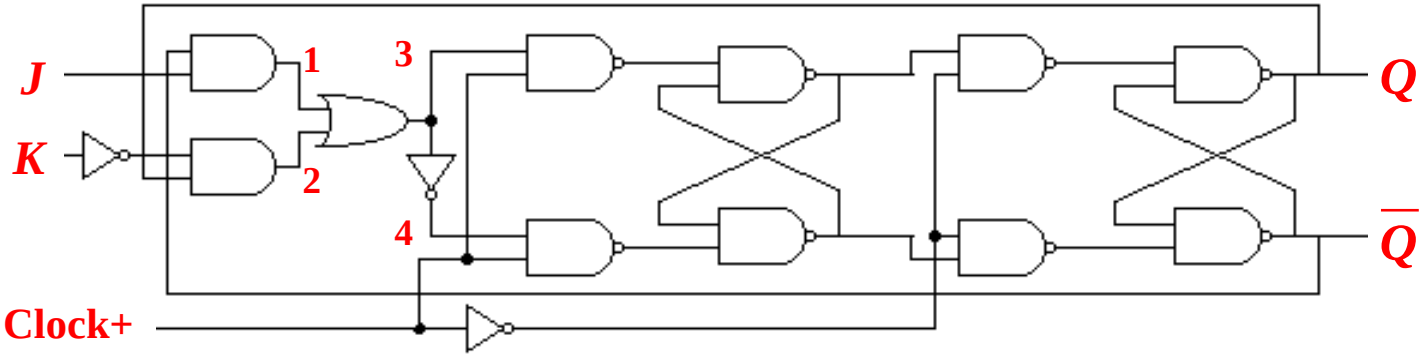
Internals of the J-K Flip-Flop



J-K Master-Slave Flip Flop

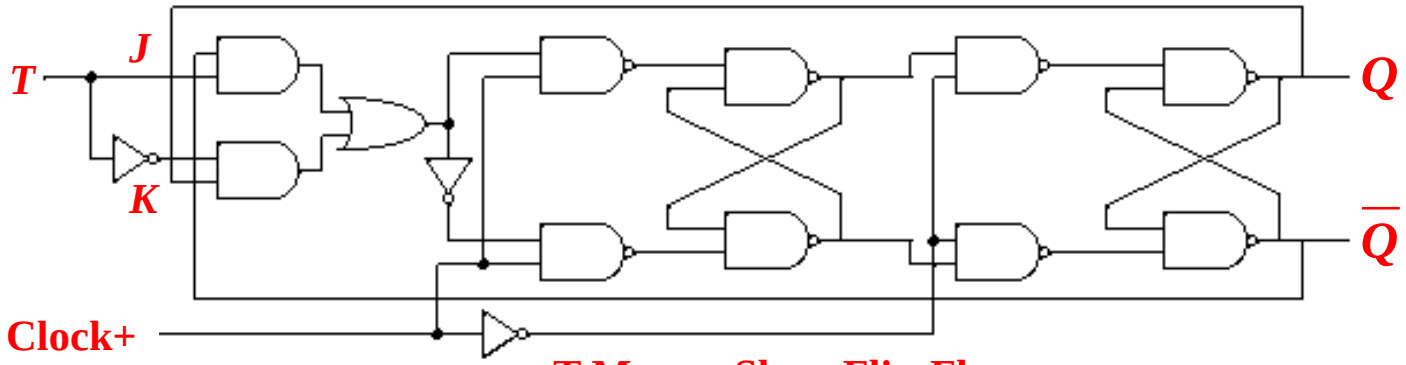
- A master-slave J-K FF can be designed as shown above.
- The key states are J=K=1, for either output state (set or reset).
- If Q = 1 ("Set") and J = K = 1, output of the OR = 0, so the ff will reset.
- Likewise, if Q = 0 ("Reset") and J = K = 1, OR = 1, and the ff will be set.
- (For J=1 and K=0, or J=0 and K=1) normal set or reset occurs.)
- Then for J = K = 1, when the clock ticks $Q \rightarrow$ the opposite state.

JK Flip Flop Truth Table



Inputs		Outputs		AND Outputs		OR Outputs		New Outputs	
J	K	Q	Q	1	2	3	4	Q	Q
0	0	0	1	0	0	0	1	Same	Same
0	0	1	0	0	1	1	0	Same	Same
0	1	0	1	0	0	0	1	Same	Same
0	1	1	0	0	0	0	1	0	1
1	0	0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0	Same	Same
1	1	0	1	1	0	1	0	1	0
1	1	1	0	0	0	0	1	0	1

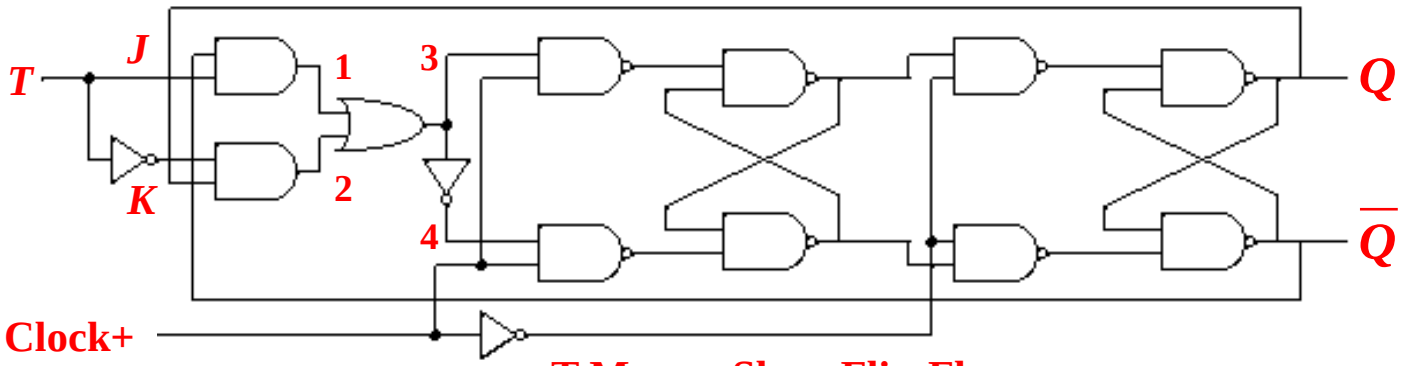
The Toggle Flip-Flop



T Master-Slave Flip-Flop

- The *T* FF is like a JK FF with *J* and *K* tied together (K input inverted).
- Then if T = 1, and clock = 1, the ff "toggles" to the opposite state.
- If *T* = 0, the ff does not change state on the clock "tick."
- The *T* FF is a master-slave ff; output changes on the back edge of the clock.
- Set T = 1 permanently, and the T FF toggles on every clock pulse.
- Note Q tied to the K input and Q-not tied to the J input. This "feedback," along with the connected J and K inputs, enables the T FF to work properly.

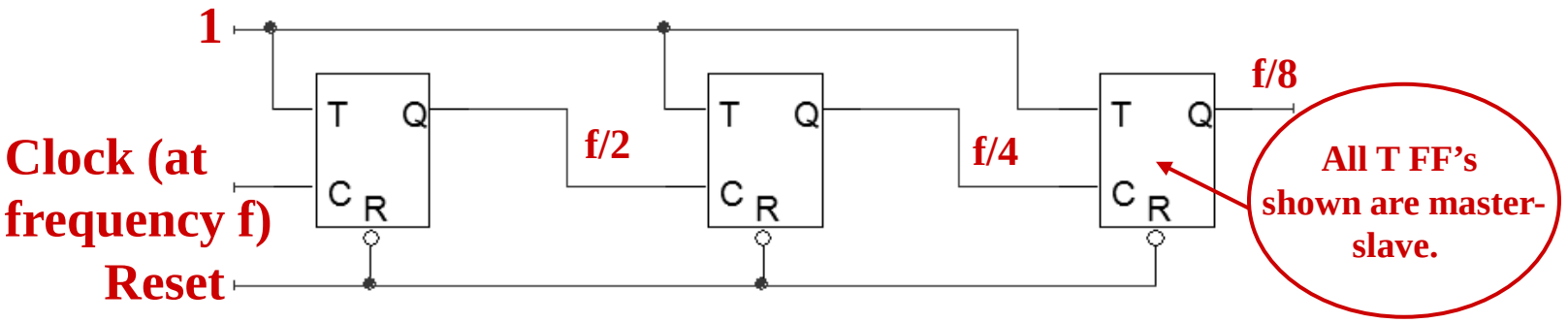
Toggle Flip Flop Truth Table

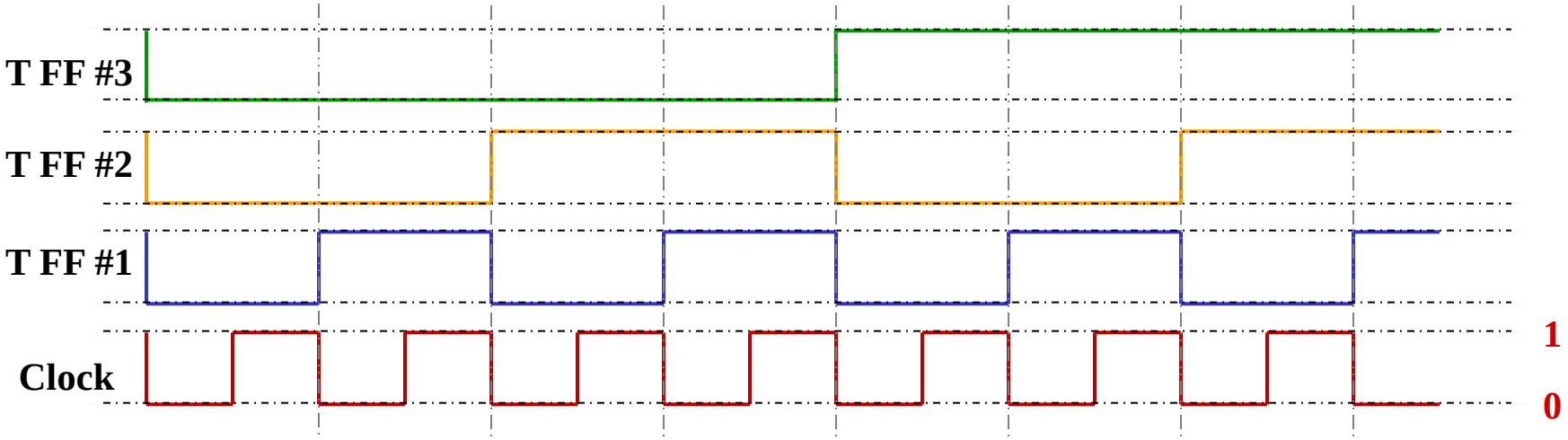


T Master-Slave Flip-Flop

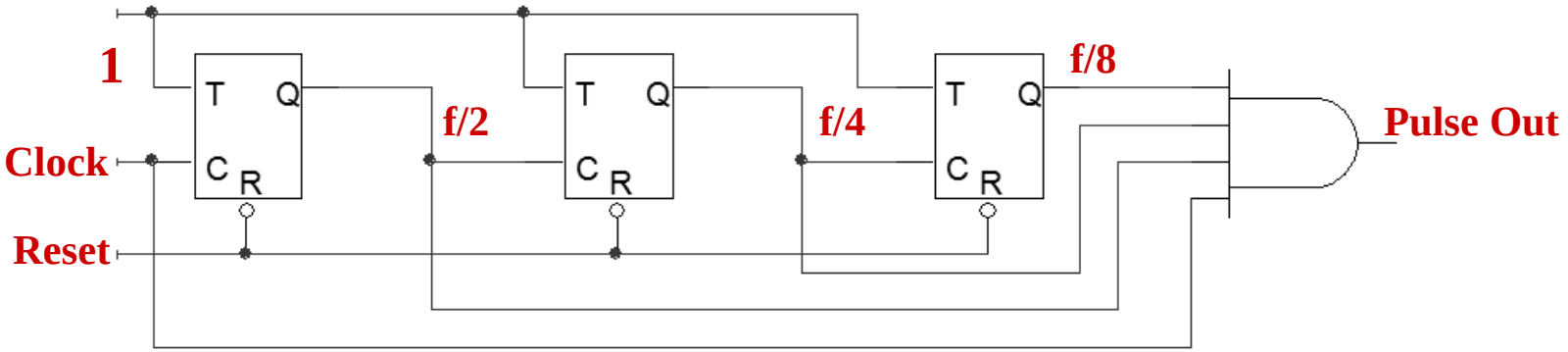
Input T	Outputs		AND Outputs		OR Outputs		New Outputs	
	Q	$\overline{Q}$	1	2	3	4	Q	$\overline{Q}$
0	0	1	0	0	0	1	Same	Same
0	1	0	0	1	1	0	Same	Same
1	0	1	1	0	1	0	1	0
1	1	0	0	0	0	1	0	1

Master-Slave T FF as a Frequency Divider



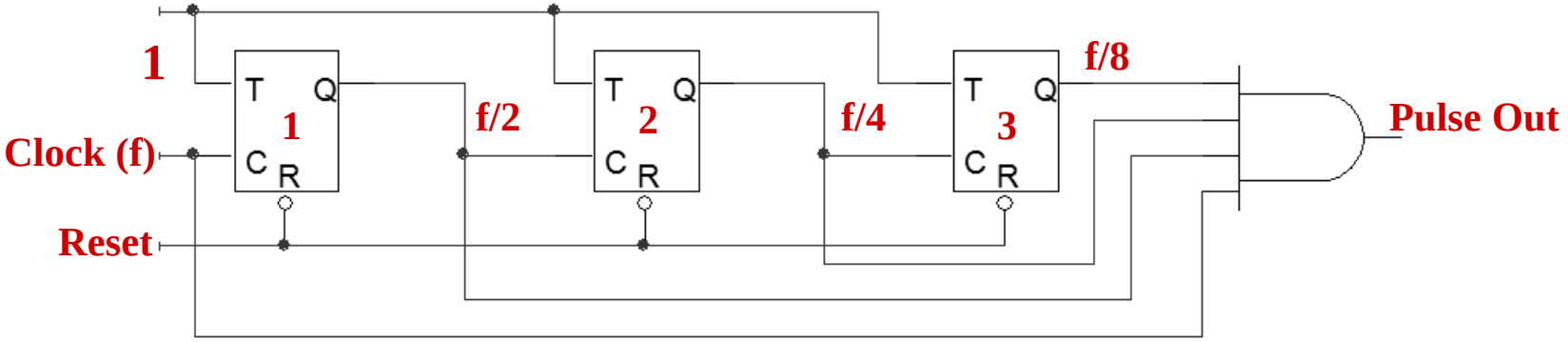


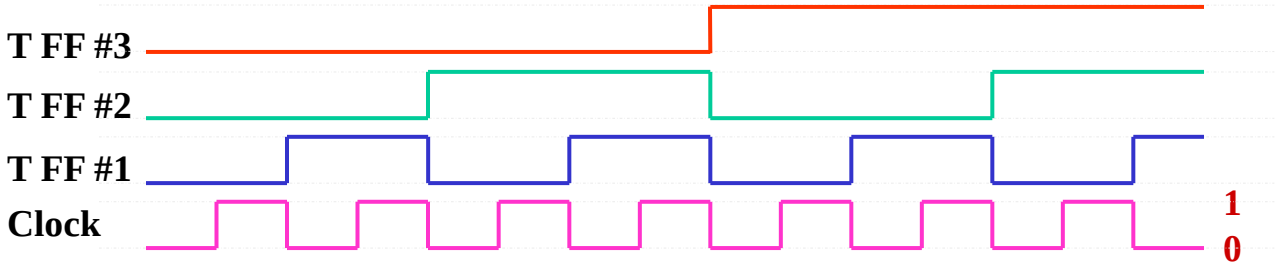
Timing (Continued)



- Suppose that we want to decode a state of the frequency divider circuit seen on the last slide.
- When the "/8" and "/4" and "/2" outputs are high, we want to AND those signals with clock to get a decoded pulse to perform some operation (see diagram above).
- How do we show the timing on this sequential decoder?

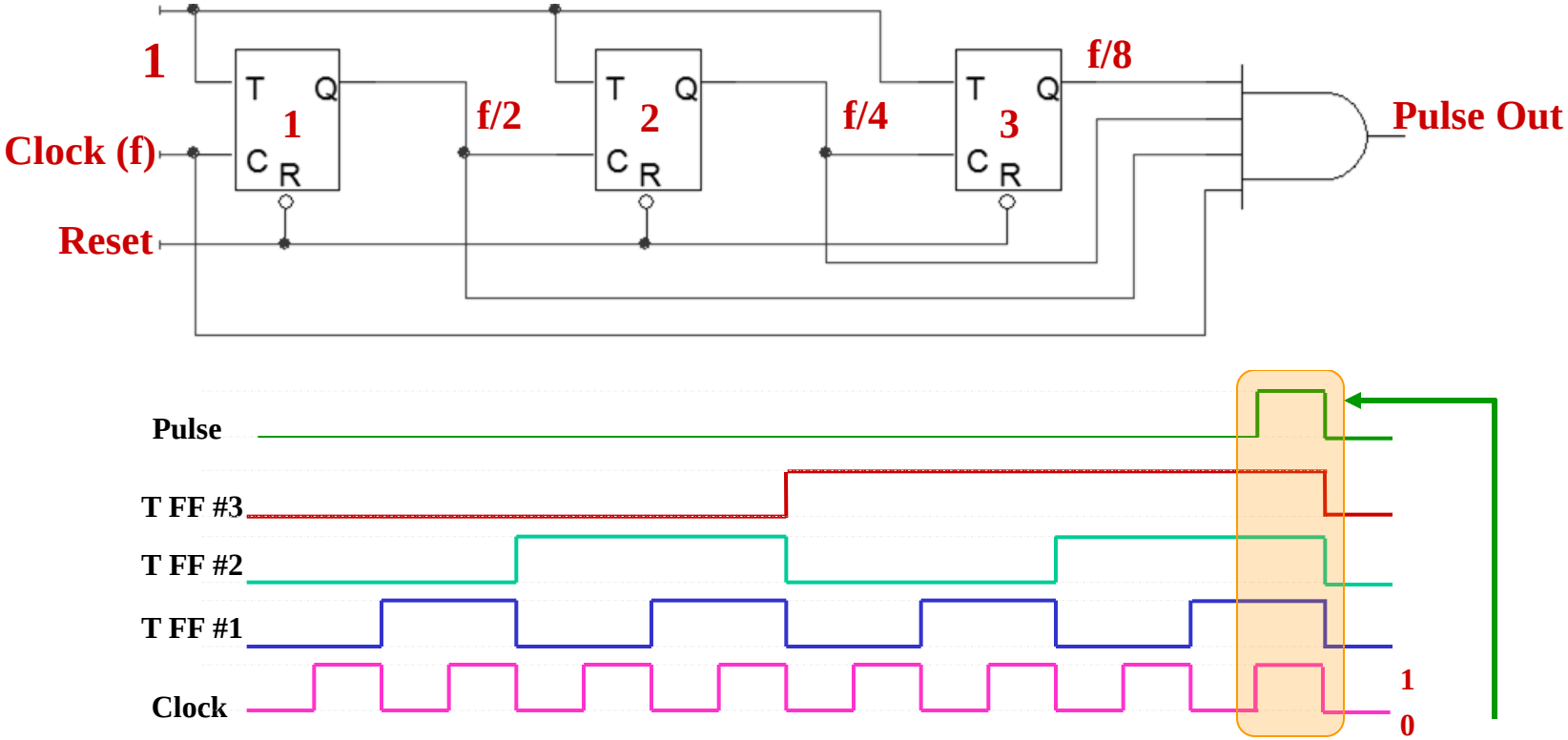
Timing (Continued)





- We know that the output of FF #1 clocks FF#2, and #2 clocks #3.
- On the first falling edge of the clock, FF #1 toggles. On the falling edge of the "Q" output of FF #1, FF #2 toggles, etc.

Timing (Concluded)



- On the timing diagram, we look for the time when the three Q's are high.
- We then diagram "Pulse Out" based on the AND of the 3 signals plus clock.

Flip – Flop Summary

- **We have devoted a good deal of time to the study of the latch, or flip-flop, because it is important in modern computer circuitry.**
- **The primary uses for latches are:**
  - **D FF: ALU registers, input/output buffers, shift registers, fast memory.**
  - **J-K FF: Control functions and status indicators.**
  - **T FF: Frequency division and counter circuits.**
- **We will study more complex ff circuits next.**