

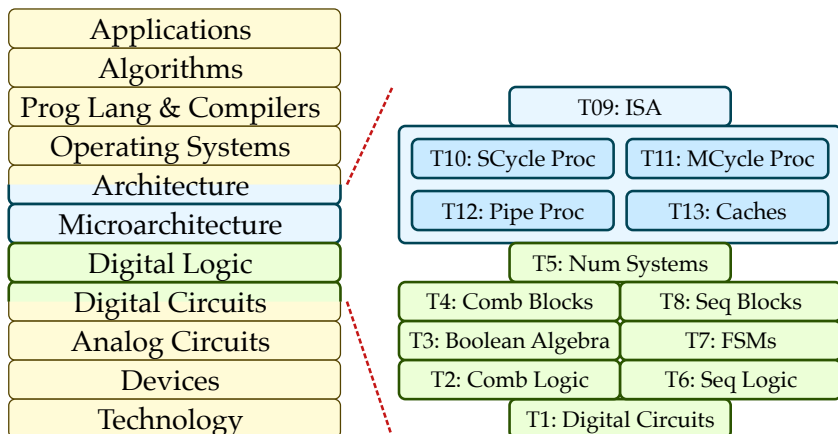
ECE 2300 Digital Logic and Computer Organization Fall 2026

Topic 6: Sequential Logic

School of Electrical and Computer Engineering
Cornell University

revision: 2026-09-29-10-15

1	Latches, Flip-Flops, and Registers	3
1.1.	SR Latch	3
1.2.	D Latch	5
1.3.	D Flip-Flop	6
1.4.	D Flip-Flop with Synchronous Reset	7
1.5.	D Flip-Flop with Synchronous Reset and Enable	7
1.6.	Multi-Bit Register	8
2	Sequential Gate-Level Networks	9
3	Sequential Gate-Level Timing	15
3.1.	Setup Time	18
3.2.	Hold Time	23
3.3.	Clock Skew	28
3.4.	Timing Constraints in Practice	31
3.5.	Summary of Sequential Gate-Level Timing	33
4	Verilog Modeling	34

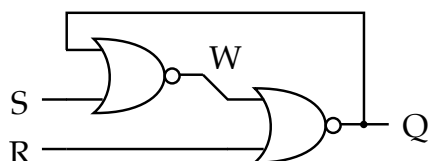


Copyright © 2026 Christopher Batten. All rights reserved. This handout was prepared by Prof. Christopher Batten at Cornell University for ECE 2300 / ENGRD 2300 Digital Logic and Computer Organization. Download and use of this handout is permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial or non-commercial means requires written permission.

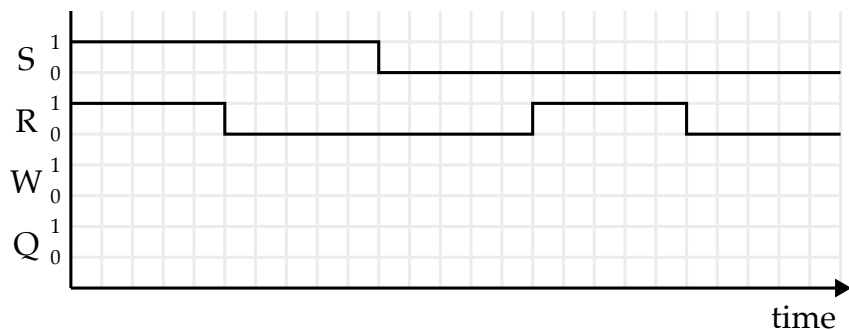
1. Latches, Flip-Flops, and Registers

- *Combinational Logic*: outputs only depend on current inputs
- *Sequential Logic*: outputs depend on current inputs *and previous inputs*

1.1. SR Latch



S	R	W	Q
0	0		
0	1		
1	0		
1	1		



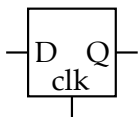
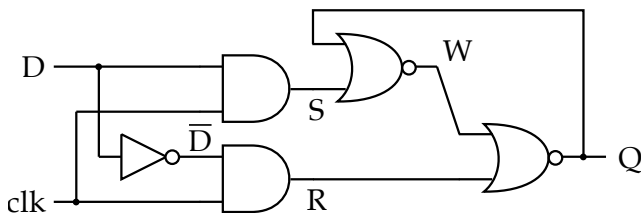
- *SR Latch*: bi-stable state element with set and reset inputs

- Let's create a new kind of truth table
- Treat Q_{prev} as an input and Q_{next} as an output

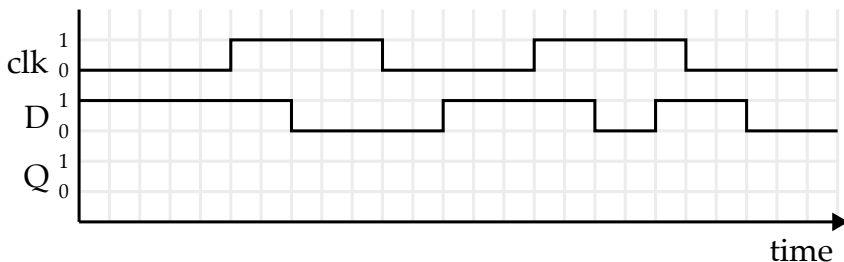
S	R	Q_{prev}	W	Q_{next}	$Q_{next,next}$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

1.2. D Latch

- *SR Latch*: Asserting one input determines not only *what* the new state should be but also *when* it should change
- *D Latch*: One input controls *what* the next state should be, while second input controls *when* the state should change

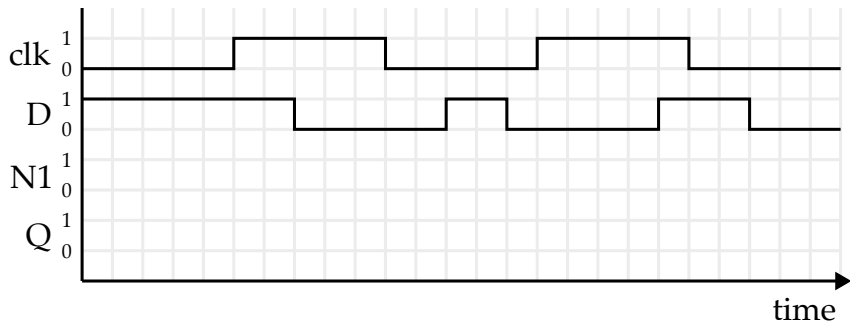
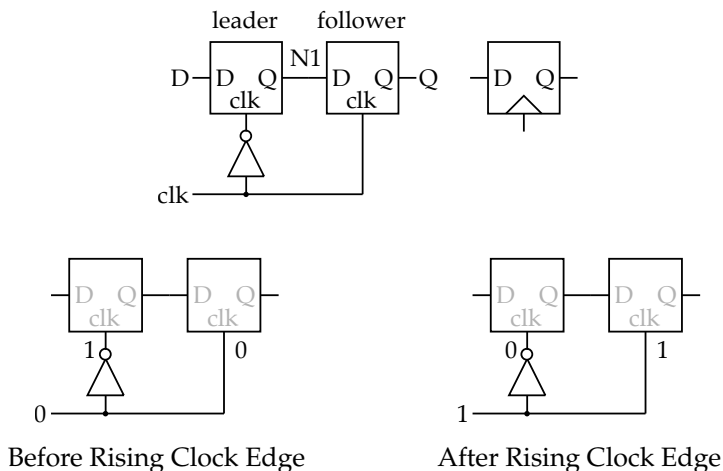


clk	D	$\bar{D}$	S	R	W	Q
0	0					
0	1					
1	0					
1	1					

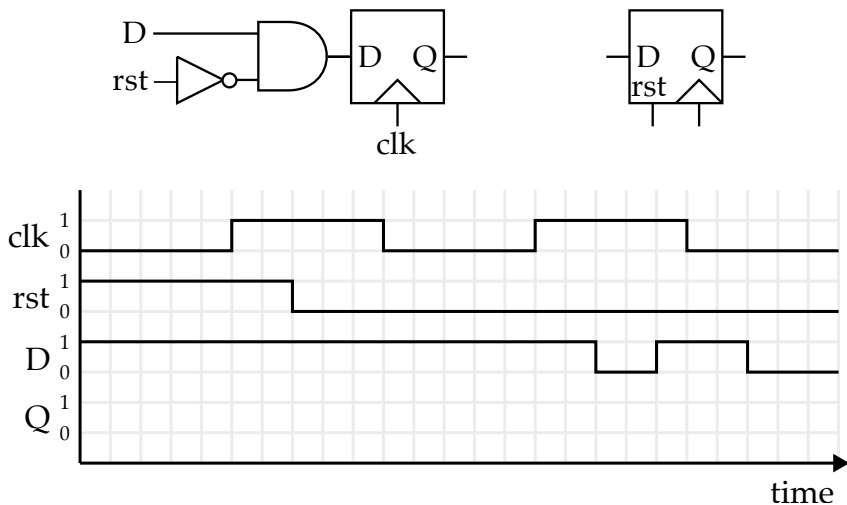


1.3. D Flip-Flop

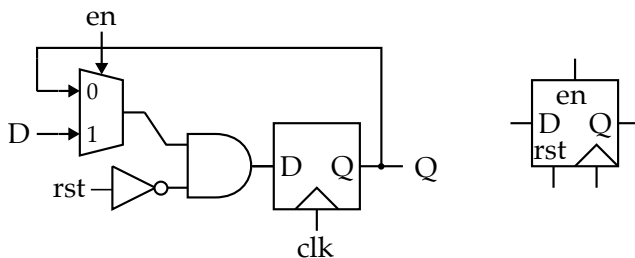
- *D Latch*: Input is *sampled* continuously when clock is high
- *D Flip-Flop*: Input is only *sampled* at a specific instance in time (on the rising edge of the clock)



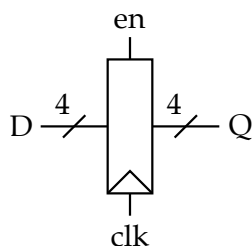
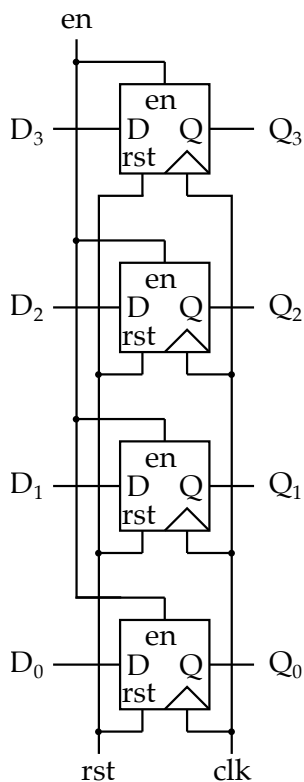
1.4. D Flip-Flop with Synchronous Reset



1.5. D Flip-Flop with Synchronous Reset and Enable

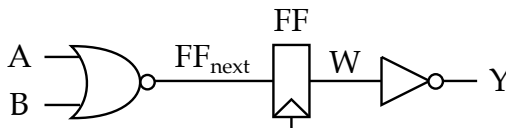


1.6. Multi-Bit Register



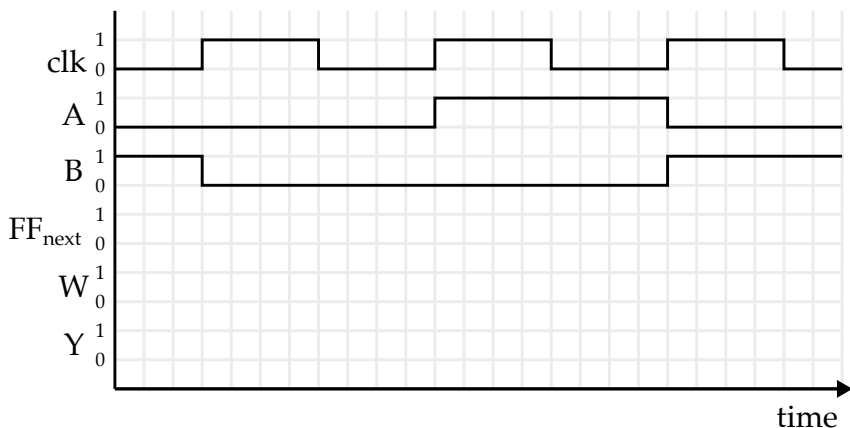
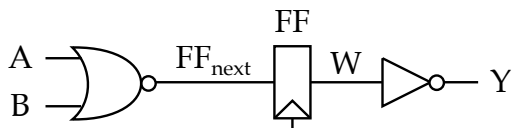
2. Sequential Gate-Level Networks

- *Combinational truth tables* have one column for every input, intermediate signal, and output; one row for all possible input values
- *Sequential truth tables* have one column for every input, *flip-flop*, intermediate signal, and output; one row for all possible input values *and possible flip-flop values*

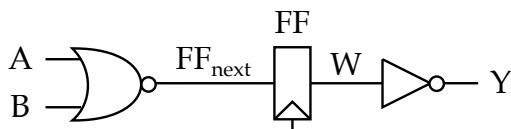


Truth Table

<i>A</i>	<i>B</i>	<i>FF</i>	<i>FF_{next}</i>	<i>W</i>	<i>Y</i>
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



- Drawing waveforms for sequential logic over many cycles can become quite tedious
- *Simulation tables* are a more compact way to illustrate waveforms for sequential logic assuming a zero delay model



Explicit-Clock Simulation Table

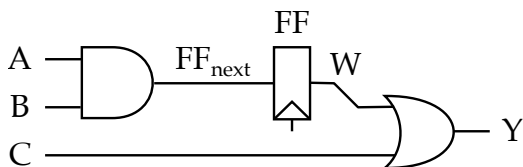
- Clock signal is shown as a column
- One row per *clock phase*, row shows values during that clock phase

clk	A	B	FF	FF _{next}	W	Y
1	0	1				
0	0	1				
1	0	0				
0	0	0				
1	1	0				
0	1	0				
1	0	1				
1	0	1				

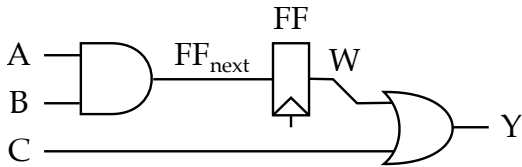
Implicit-Clock Simulation Table

- Clock signal is *not* shown as a column
- One row per *clock cycle*, row shows values when clock is high phase

A	B	FF	FF _{next}	W	Y
0	1				
0	0				
1	0				
0	1				

**Truth Table**

A	B	C	FF	FF_{next}	W	Y
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

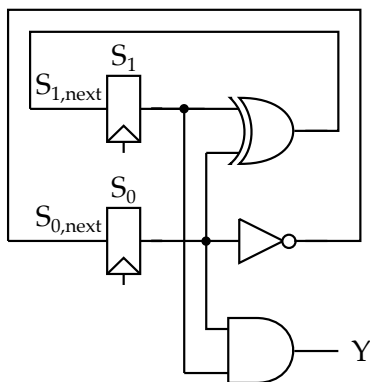


Implicit-Clock Simulation Table

cycle	A	B	C	FF	FF _{next}	W	Y
0	1	1	0				
1	1	1	0				
2	0	1	0				
3	0	1	1				
4	0	1	1				

- Use arrows to show dependencies between values
 - *Horizontal dependency arrows* correspond to combinational logic
 - *Vertical dependency arrows* correspond to sequential logic

Complete the truth table and implicit-clock simulation table for the given sequential gate-level network.



Truth Table

S_1	S_0	$S_{1,next}$	$S_{0,next}$	Y
0	0			
0	1			
1	0			
1	1			

Implicit-Clock Simulation Table

Assume all flip-flops are reset to zero.

cycle	S_1	S_0	$S_{1,next}$	$S_{0,next}$	Y
0					
1					
2					
3					
4					
5					
6					
7					