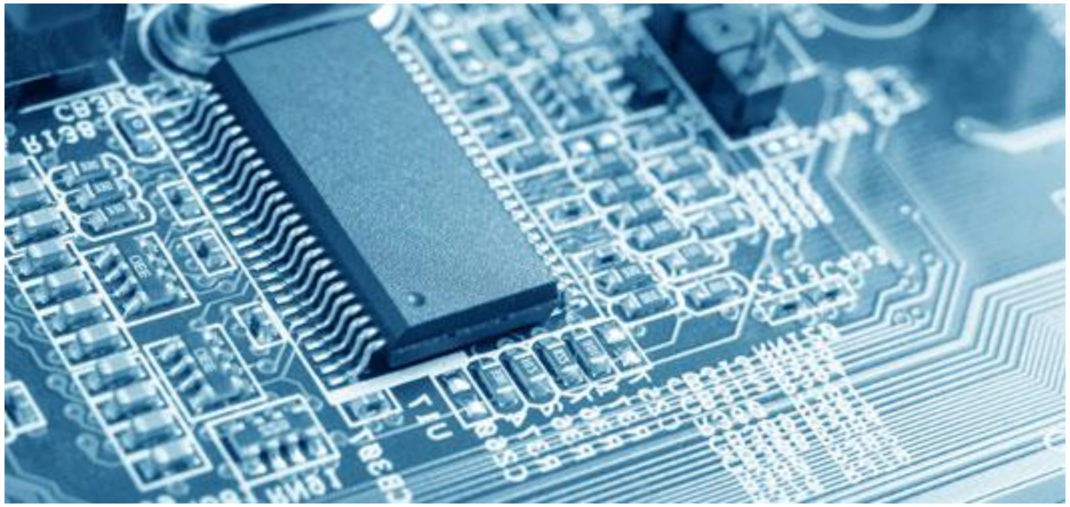
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



CSCI502 – Hardware/Software Co-Design

Self-Study Lecture 2 – Introduction to Digital Circuits

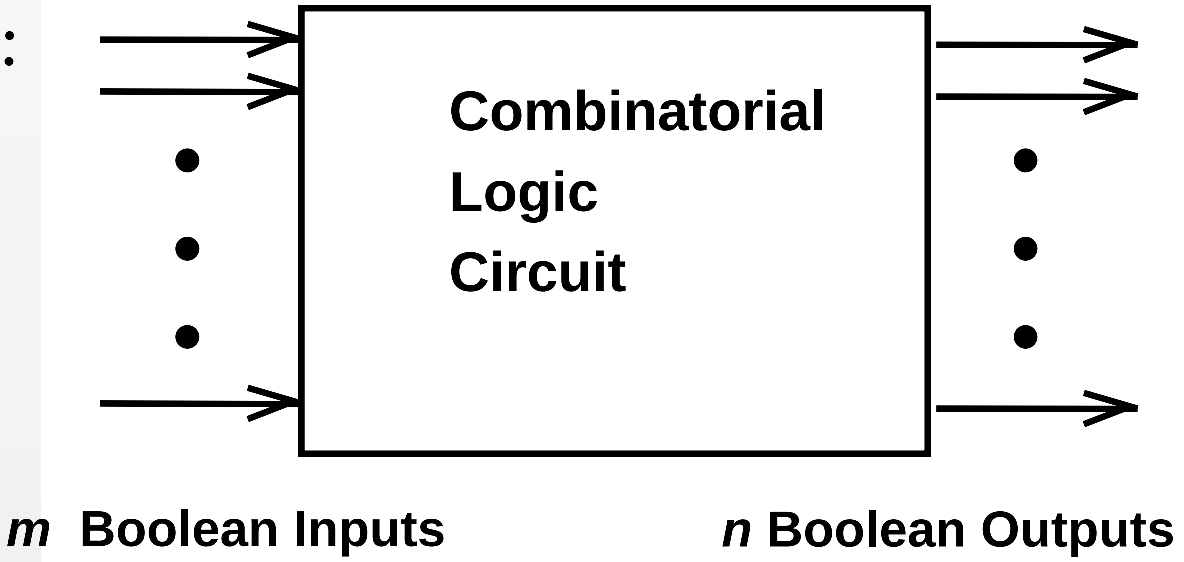
16-22 January, 2018

Course Logistics

Reference Reading (available in Moodle and library):
Logic and Computer Design Fundamentals. 5th edition: **Chapters 3 – 4 (slide relevant material)**

Combinational Circuits

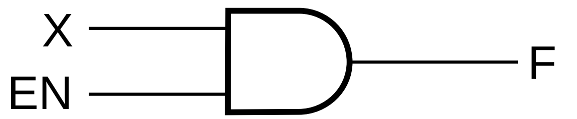
- A combinational logic circuit has:
  - A set of *m* Boolean inputs,
  - A set of *n* Boolean outputs, and
  - *n* switching functions, each mapping the 2^m input combinations to an output such that the current output depends only on the current input values
- A block diagram:



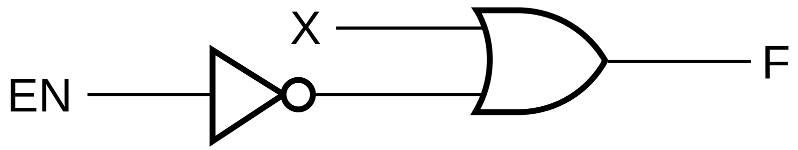
Enabling Function

- *Enabling* permits an input signal to pass through to an output
- *Disabling* blocks an input signal from passing through to an output, replacing it with a fixed value
- The value on the output when it is disable can be Hi-Z (as for three-state buffers and transmission gates), 0 , or 1

- When disabled, 0 output
- When disabled, 1 output



(a)



(b)

Decoding

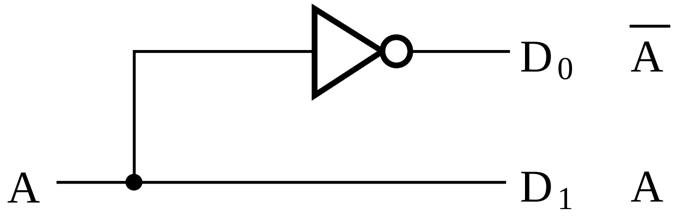
- Decoding – the conversion of an *n*-bit input code to an *m*-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform decoding are called ***decoders***
- Here, functional blocks for decoding are
  - called *n*-to-*m* line decoders, where $m \leq 2^n$, and
  - generate 2^n (or fewer) minterms (canonical combinations) for the *n* input variables

Decoder Examples

- 1-to-2-Line Decoder

A	D_0	D_1
0	1	0
1	0	1

(a)

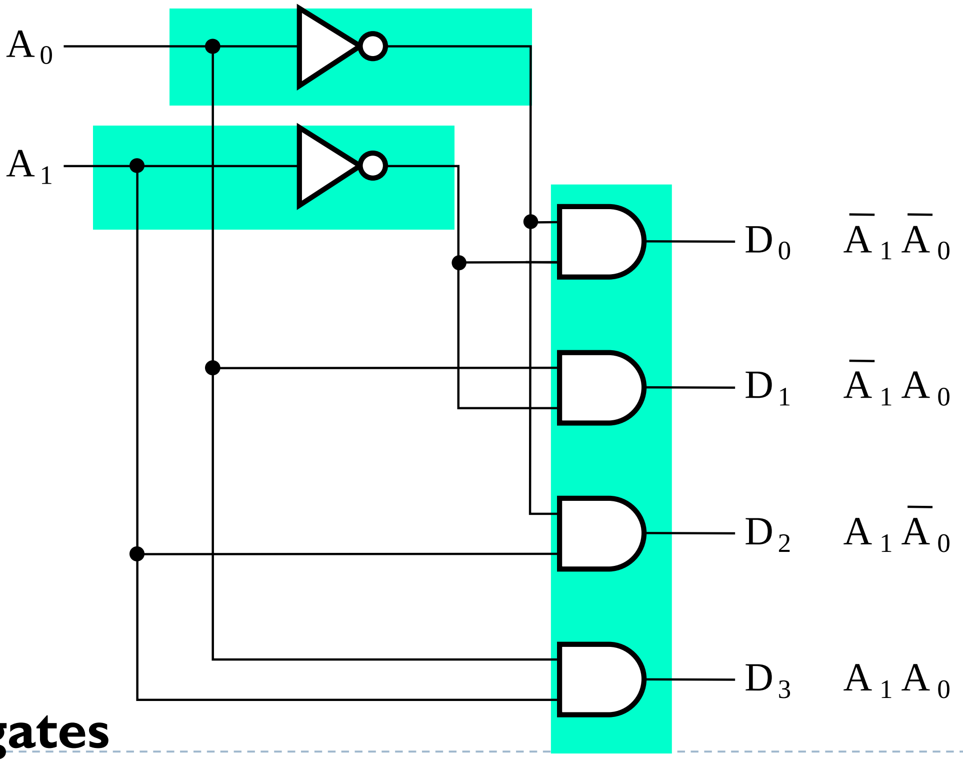


(b)

- 2-to-4-Line Decoder

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

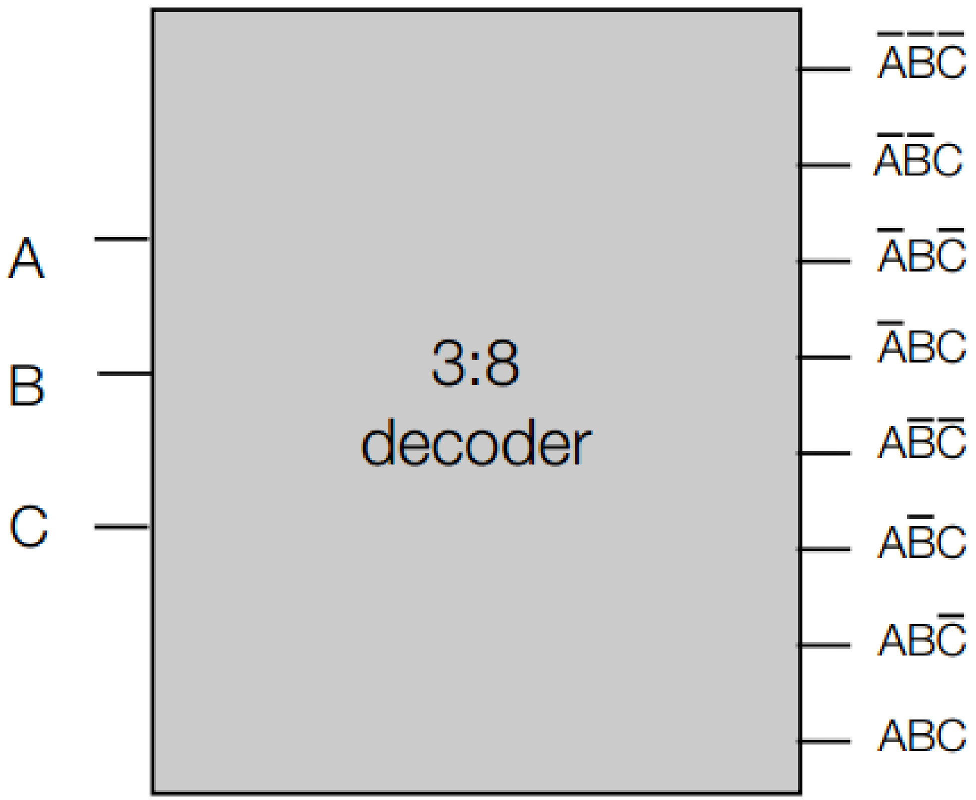


(b)

- **Note that the 2-4-line made up of 2 1-to-2-line decoders and 4 AND gates**

Example: 3-to-8 Decoder

Converts n-bit input to m-bit output, where n <= m <= 2^n



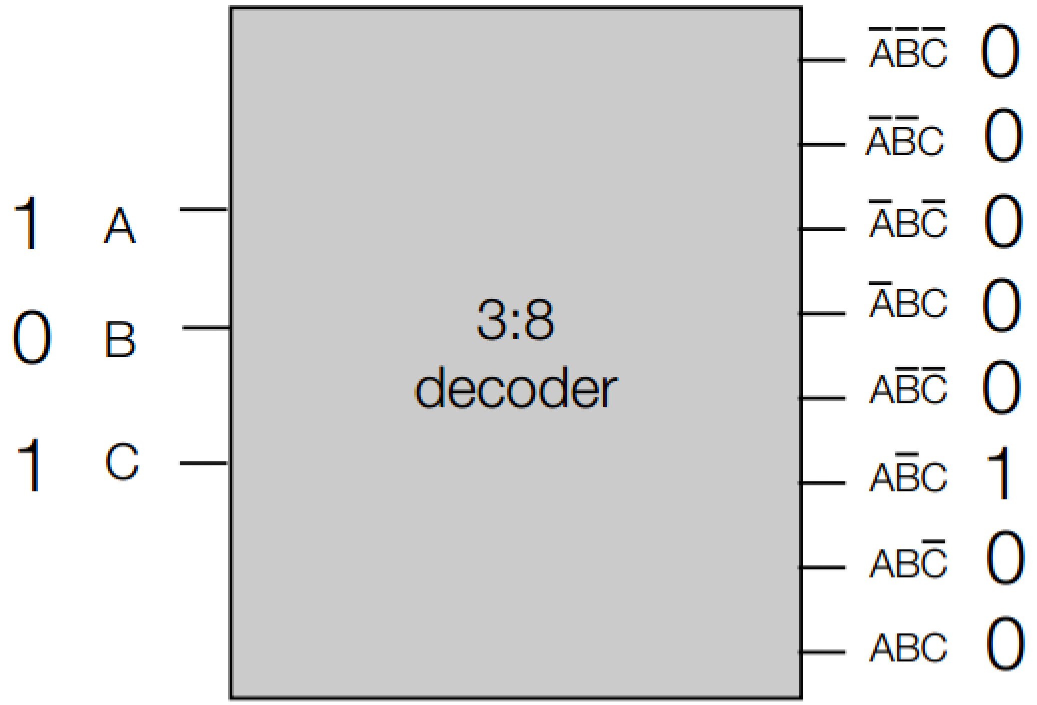
"Standard" Decoder: ith output = 1, all others = 0,

where i is the binary representation of the input (ABC)

Example: 3-to-8 Decoder

Converts n-bit input to m-bit output, where n <= m <= 2^n

e.g., ABC = 101 (i=5)

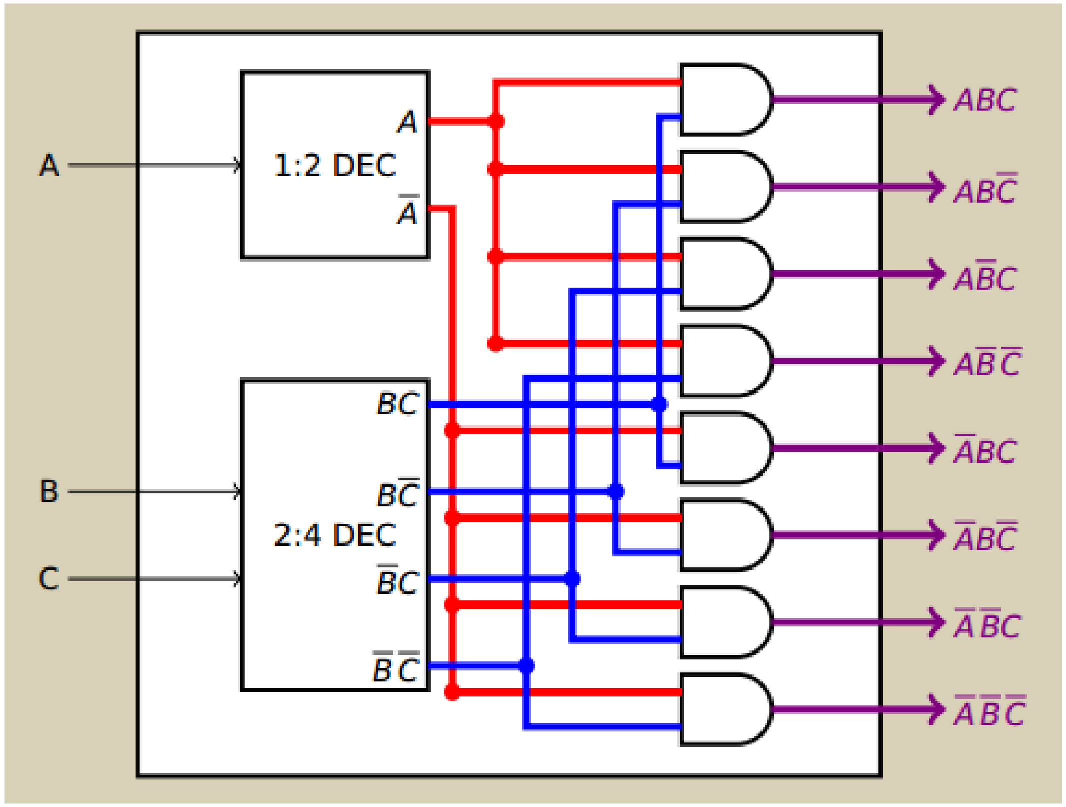


"Standard" Decoder: ith output = 1, all others = 0,

where i is the binary representation of the input (ABC)

Decoder Expansion Example

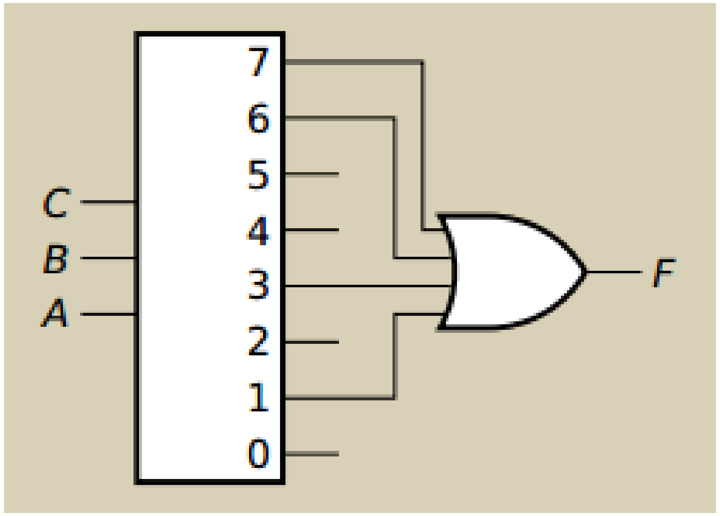
Applying *hierarchical design* again, the 2:4 DEC helps construct a 3:8 DEC.


A
B
C
1:2 DEC
A
$\overline{A}$
2:4 DEC
BC
$B\overline{C}$
$\overline{B}C$
$\overline{B}\,\overline{C}$
ABC
$AB\overline{C}$
$A\overline{B}C$
$A\overline{B}\,\overline{C}$
$\overline{A}BC$
$\overline{A}B\overline{C}$
$\overline{A}\,\overline{B}C$
$\overline{A}\,\overline{B}\,\overline{C}$

Implementing a Function Using Decoder Example

E.g., $F = A\overline{C} + BC$

C	B	A	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

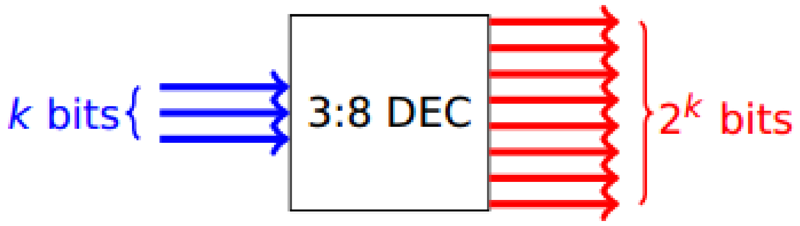


Warning: Easy, but not a minimal circuit.

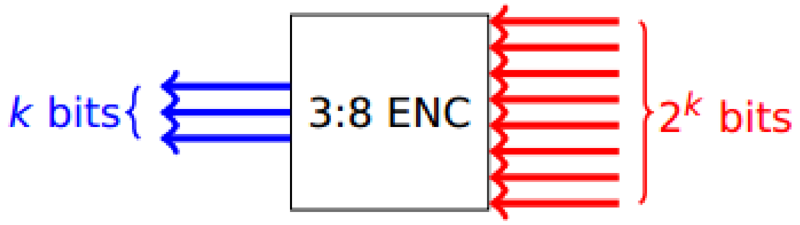
Encoding

- Encoding - the opposite of decoding: the conversion of an *m*-bit input code to a *n*-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform encoding are called *encoders*
- An encoder has 2^n (or fewer) input lines and *n* output lines which generate the binary code corresponding to the input values
- Typically, an encoder converts a code containing exactly one bit that is 1 to a binary code corresponding to the position in which the 1 appears.

Encoders and Decoders



BCD			One-Hot							
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

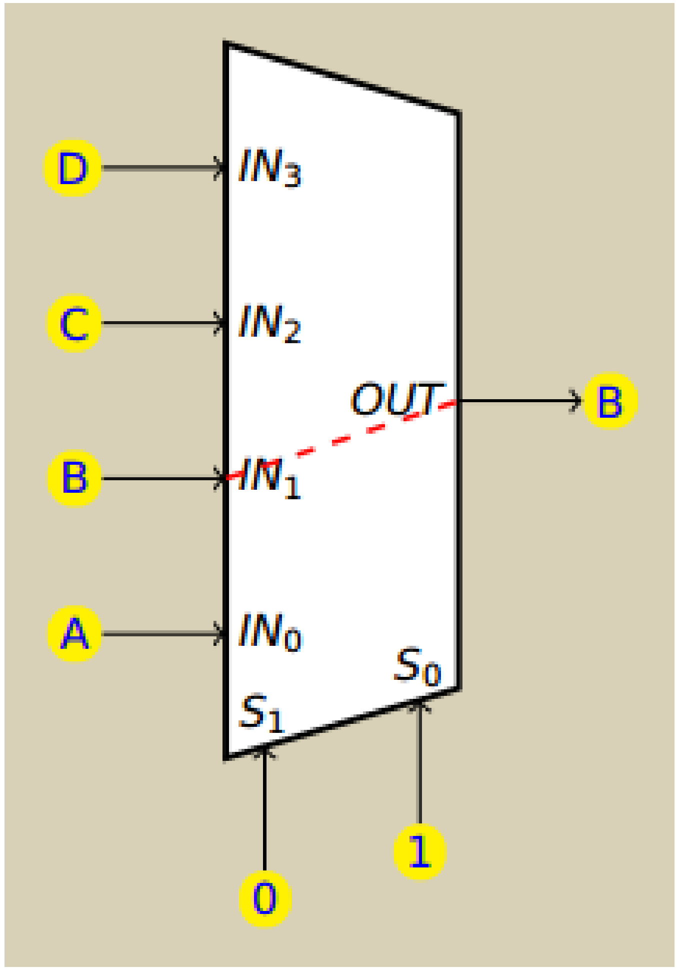


Selecting

- Selecting of data or information is a critical function in digital systems and computers
- Circuits that perform selecting have:
  - A set of information inputs from which the selection is made
  - A single output
  - A set of control lines for making the selection
- Logic circuits that perform selecting are called ***multiplexers***

Multiplexers

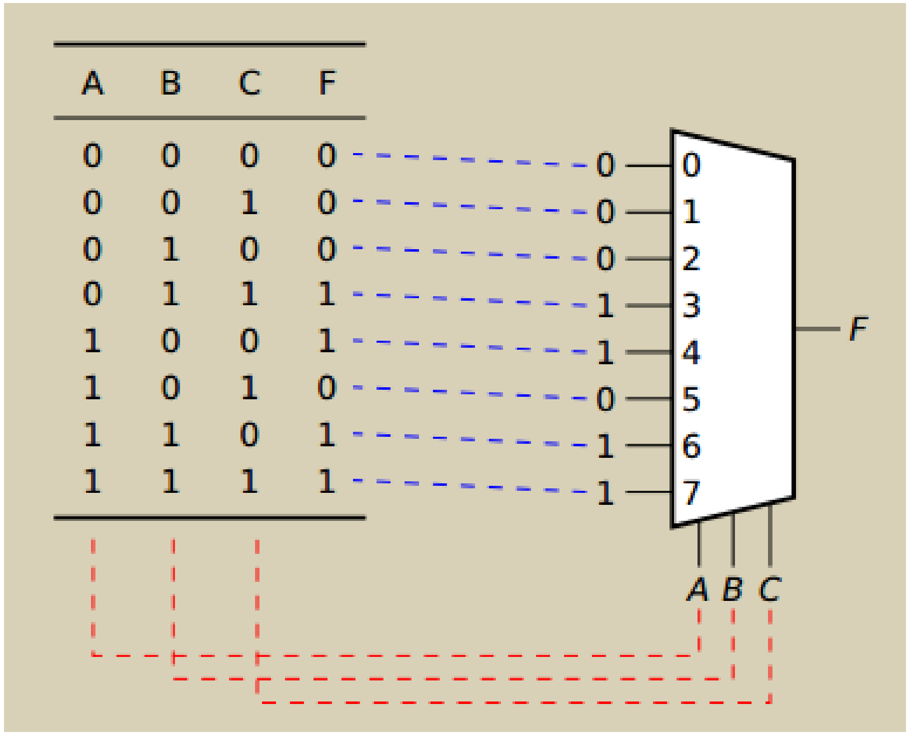
- A multiplexer selects information from an input line and directs the information to an output line
- A typical multiplexer has *n* control inputs ($S_{n-1}, \ldots S_0$) called *selection inputs*, 2^n information inputs ($I_{2^n-1}, \ldots I_0$), and one output Y
- A multiplexer can be designed to have *m* information inputs with $m < 2^n$ as well as *n* selection inputs



Combinational Logic Implementation

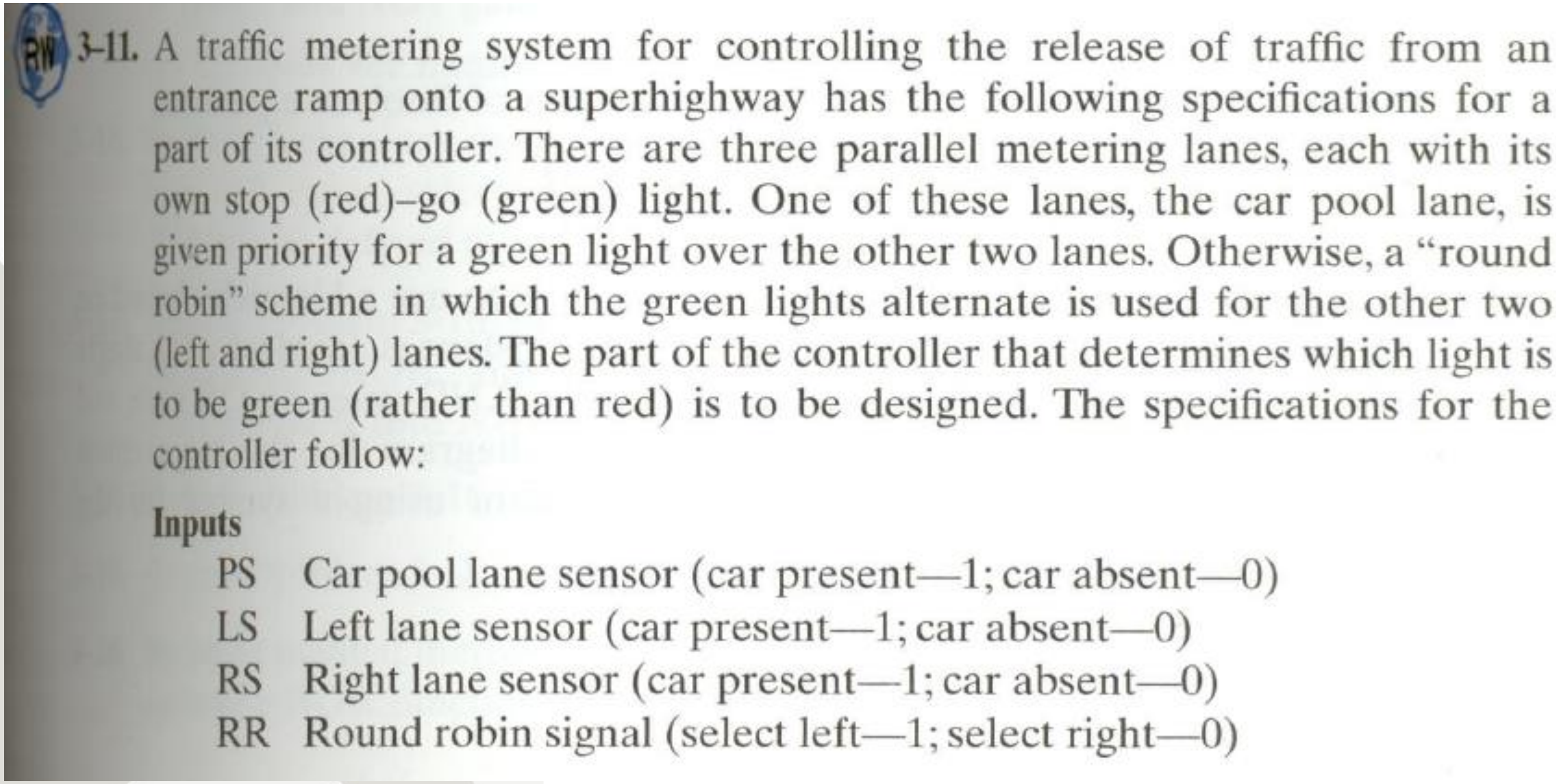
Think of a function as using k input bits to choose from 2^k outputs.

E.g., $F = BC + A\overline{C}$



A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Design Example

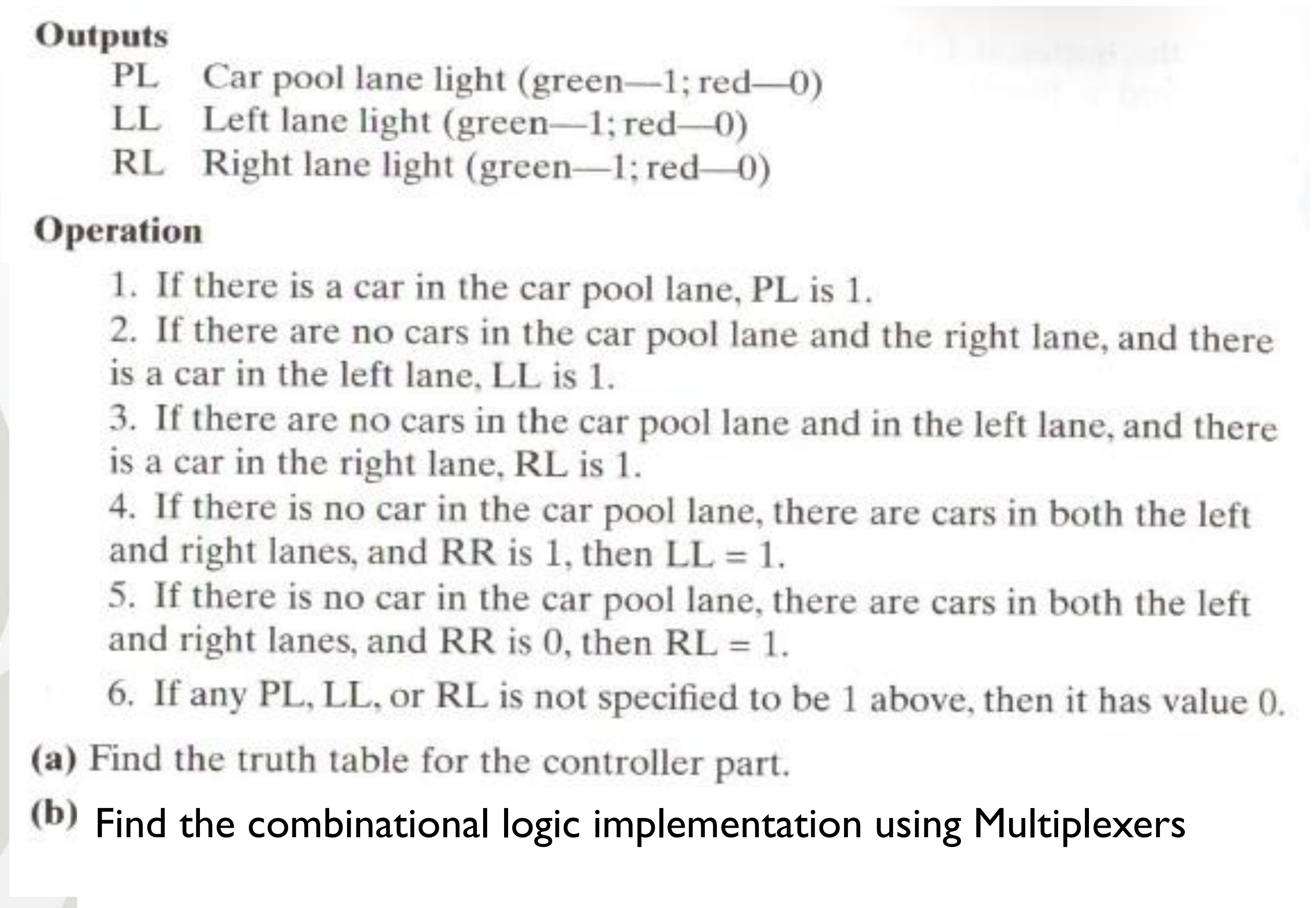


3-11. A traffic metering system for controlling the release of traffic from an entrance ramp onto a superhighway has the following specifications for a part of its controller. There are three parallel metering lanes, each with its own stop (red)–go (green) light. One of these lanes, the car pool lane, is given priority for a green light over the other two lanes. Otherwise, a "round robin" scheme in which the green lights alternate is used for the other two (left and right) lanes. The part of the controller that determines which light is to be green (rather than red) is to be designed. The specifications for the controller follow:

Inputs

PS	Car pool lane sensor (car present—1; car absent—0)
LS	Left lane sensor (car present—1; car absent—0)
RS	Right lane sensor (car present—1; car absent—0)
RR	Round robin signal (select left—1; select right—0)

Design Example Cont.



Outputs

PL	Car pool lane light (green—1; red—0)
LL	Left lane light (green—1; red—0)
RL	Right lane light (green—1; red—0)

Operation

1. If there is a car in the car pool lane, PL is 1.
2. If there are no cars in the car pool lane and the right lane, and there is a car in the left lane, LL is 1.
3. If there are no cars in the car pool lane and in the left lane, and there is a car in the right lane, RL is 1.
4. If there is no car in the car pool lane, there are cars in both the left and right lanes, and RR is 1, then LL = 1.
5. If there is no car in the car pool lane, there are cars in both the left and right lanes, and RR is 0, then RL = 1.
6. If any PL, LL, or RL is not specified to be 1 above, then it has value 0.

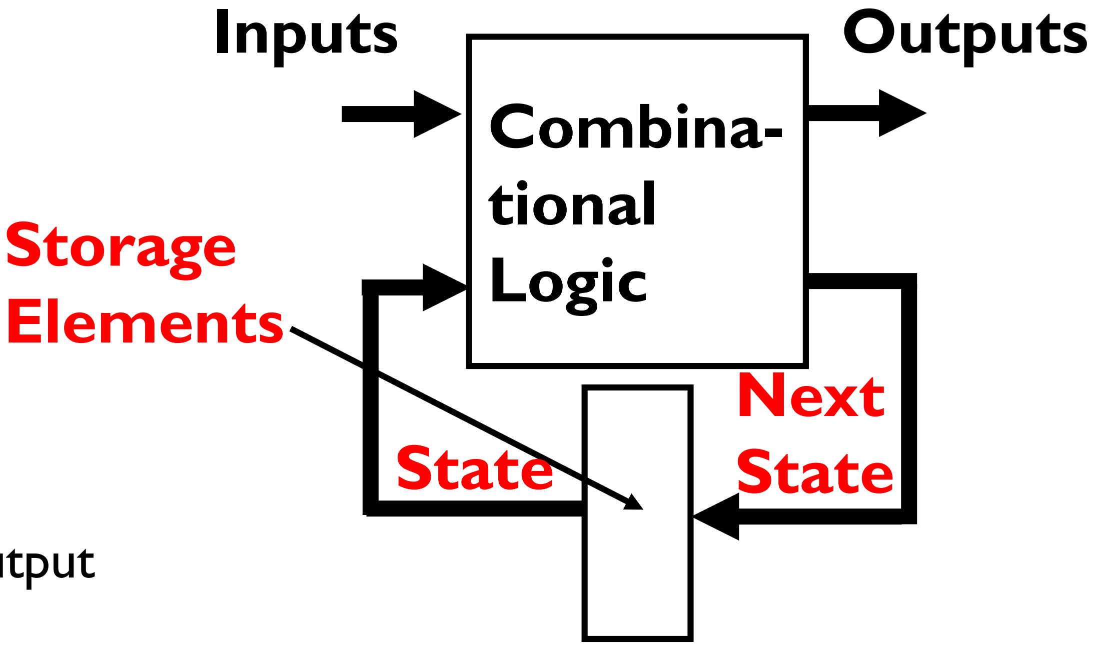
(a) Find the truth table for the controller part.

(b) Find the combinational logic implementation using Multiplexers

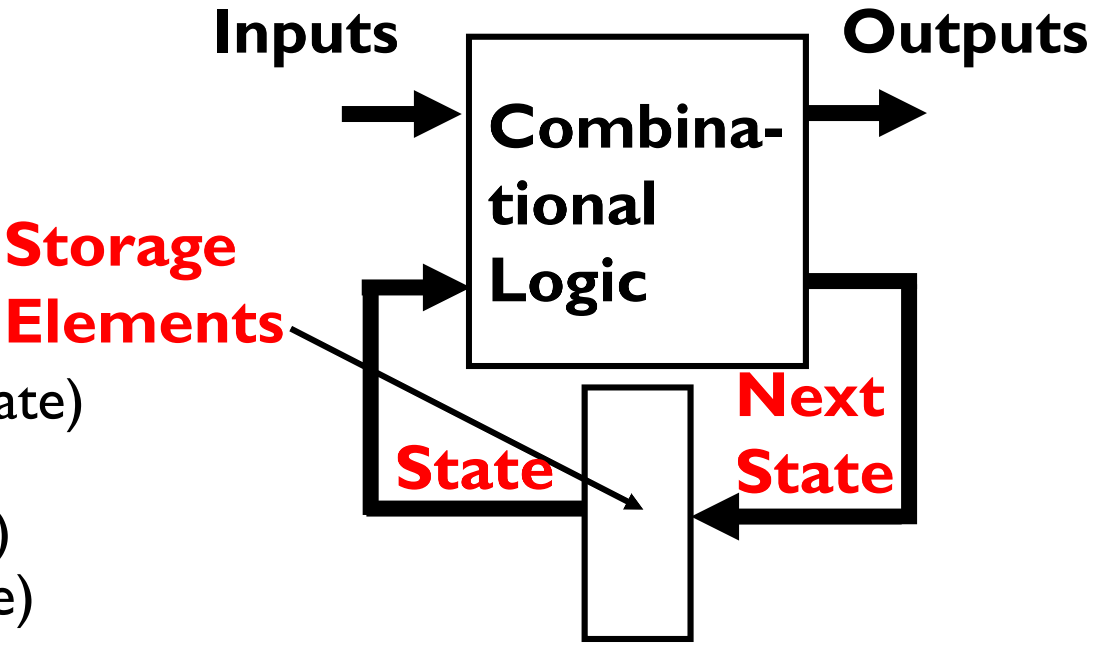
Sequential Circuits

- A Sequential

  circuit contains:
  - Storage elements: Latches or Flip-Flops
  - Combinational Logic:
    - Implements a multiple-output switching function
    - **Inputs** are signals from the outside.
    - **Outputs** are signals to the outside.
    - Other inputs, **State** or **Present State**, are signals from storage elements.
    - The remaining outputs, **Next State** are inputs to storage elements.



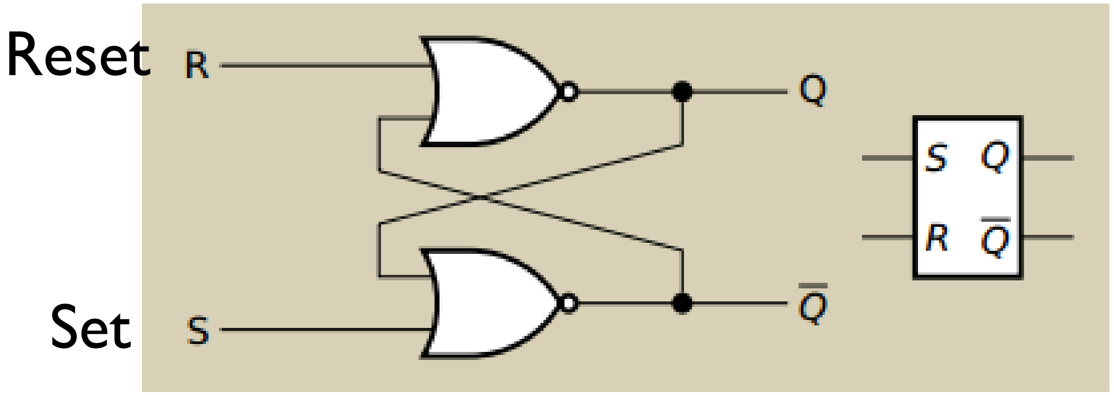
Sequential Circuits



- Combinational Logic
  - *Next state function*
    Next State = f(Inputs, State)
  - *Output function* (**Mealy**)
    Outputs = g(Inputs, State)
  - *Output function* (**Moore**)
    Outputs = h(State)
- Output function type depends on specification and affects the design significantly

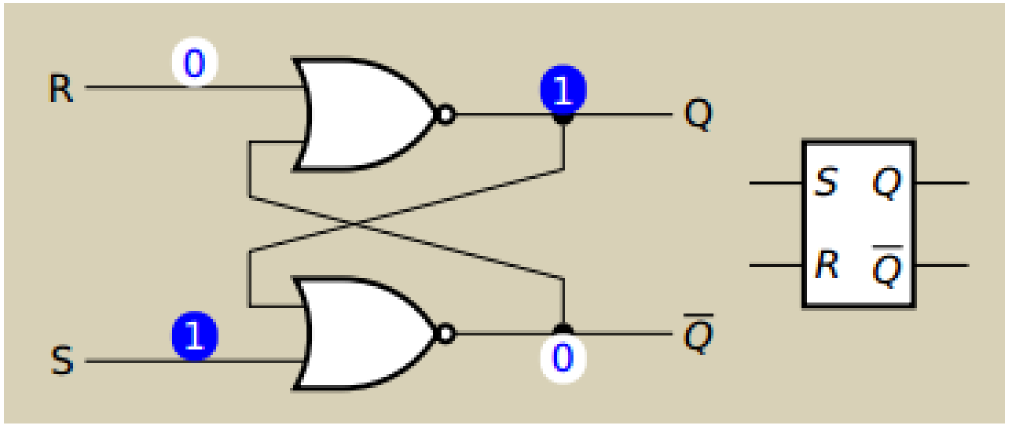
Basic (NOR) S – R Latch

Cross-coupling two NOR gates form the S – R Latch



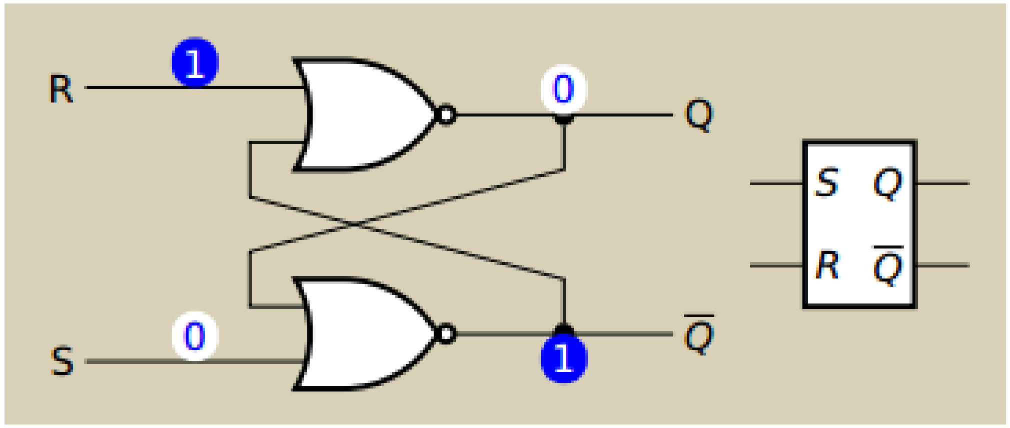
R	S	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		

Basic (NOR) S – R Latch



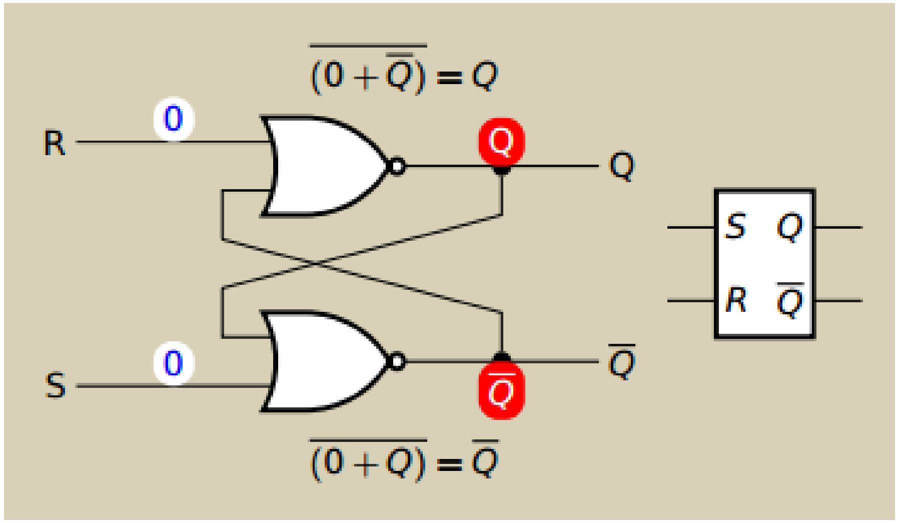
R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0			
1	1			

Basic (NOR) S – R Latch



R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1			

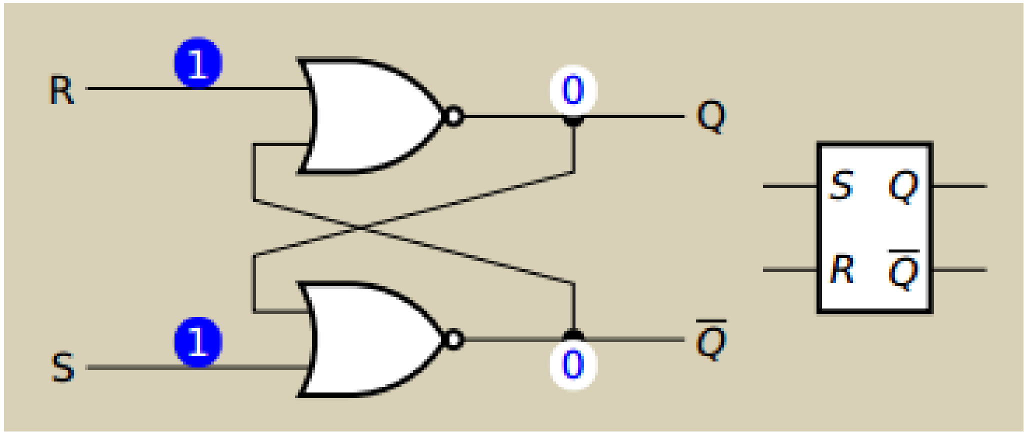
Basic (NOR) S – R Latch



$\overline{(0+\overline{Q})}=Q$
R
0
Q
Q
S
0
$\overline{Q}$
$\overline{Q}$
$\overline{(0+Q)}=\overline{Q}$
S Q
R $\overline{Q}$

R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q=1$)
1	0	0	1	Reset ($Q=0$)
1	1			

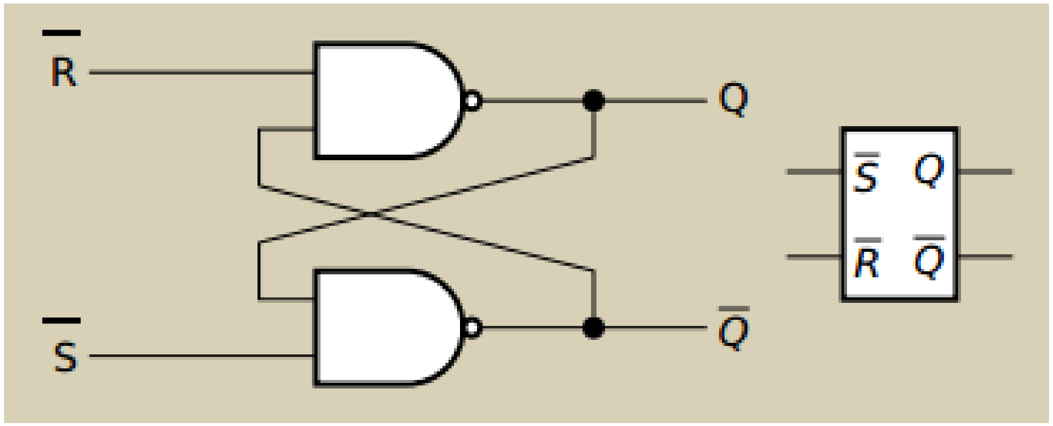
Basic (NOR) S – R Latch



R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1	0	0	Bad. Do not use.

Basic (NAND) $\bar{S} - \bar{R}$ Latch

- "Cross-Coupling"
  two NAND gates gives $\bar{S}$ - $\bar{R}$ latch:
- Which has the time sequence behavior:

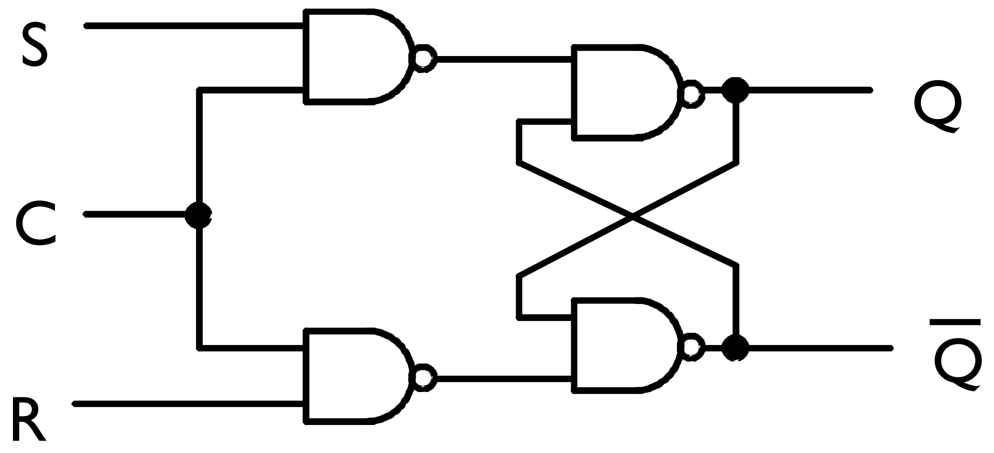


- S = 0, R = 0 is <u>forbidden</u> as input pattern

$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$	
0	0	1	1	Bad. Do not use.
0	1	0	1	Reset ($Q = 0$)
1	0	1	0	Set ($Q = 1$)
1	1	Q	$\bar{Q}$	Hold previous value

Clocked S - R Latch

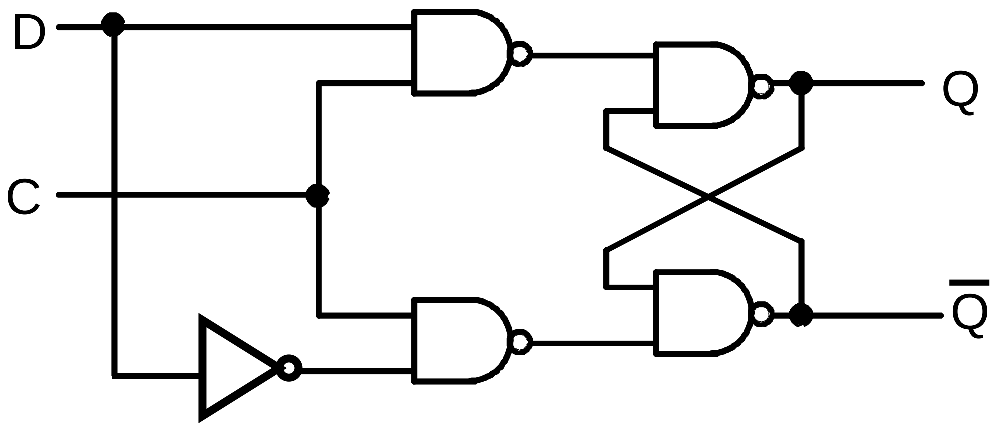
- Adding two NAND gates to the basic $\overline{S}$ - $\overline{R}$ NAND latch gives the clocked S – R latch:



- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high.
- C means "control" or "clock".

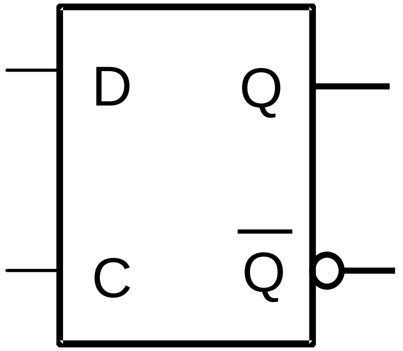
D Latch

- Adding an inverter to the S-R Latch, gives the D Latch:
- Note that there are no "indeterminate" states!

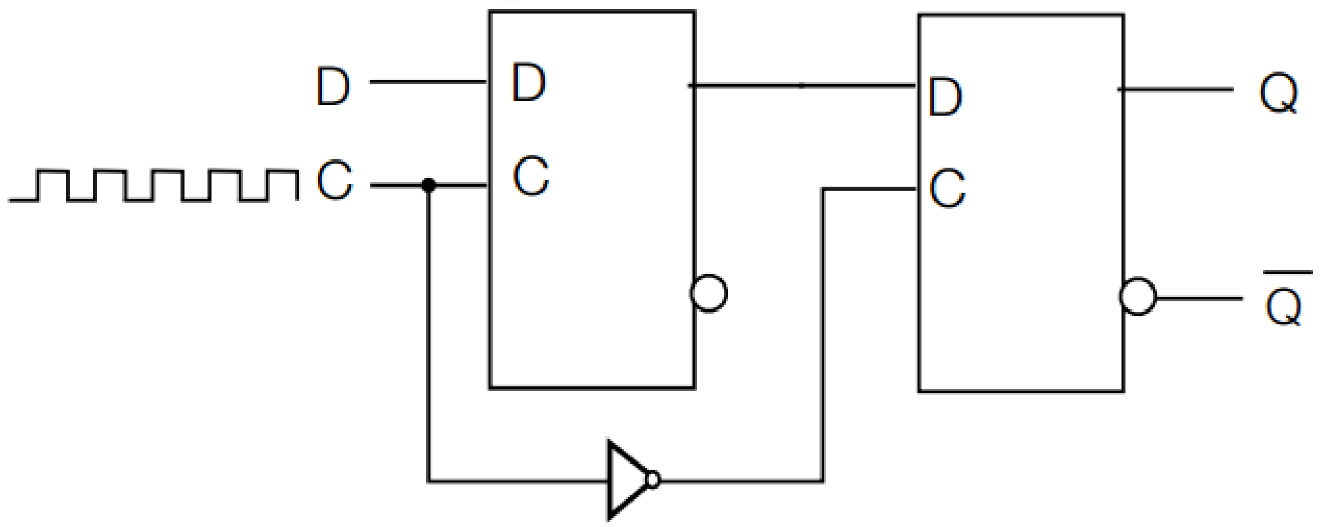


C	D	Q	$\overline{Q}$	
0	X	Q	$\overline{Q}$	No change
1	0	0	1	Reset state
1	1	1	0	Set state

The graphic symbol for a D Latch is:



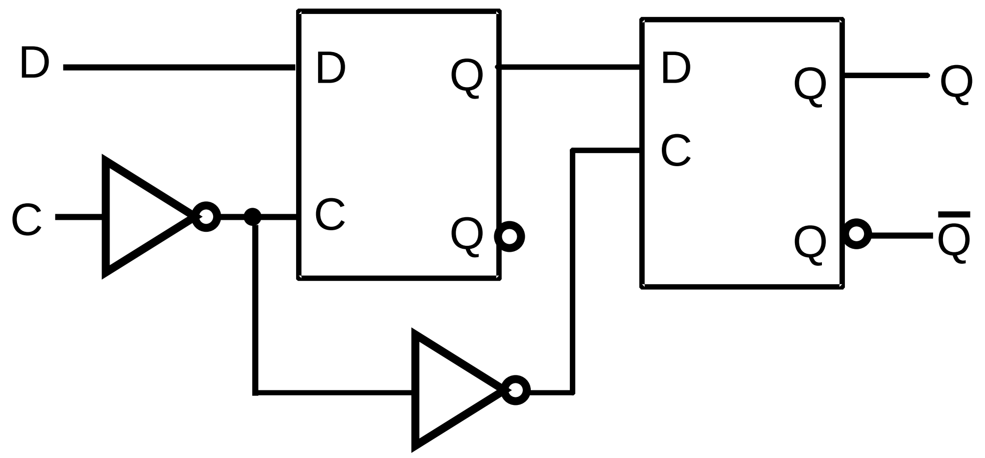
Edge-Triggered D Flip-Flop



- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Master latch “on”, Slave latch “off”
    - New D input read into master
    - Previous Q values still emitted (not affected by new D inputs)
  - C=0: Master latch “off”, Slave latch “on”
    - Changing D inputs has no effect on Master (or Slave) latch
    - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

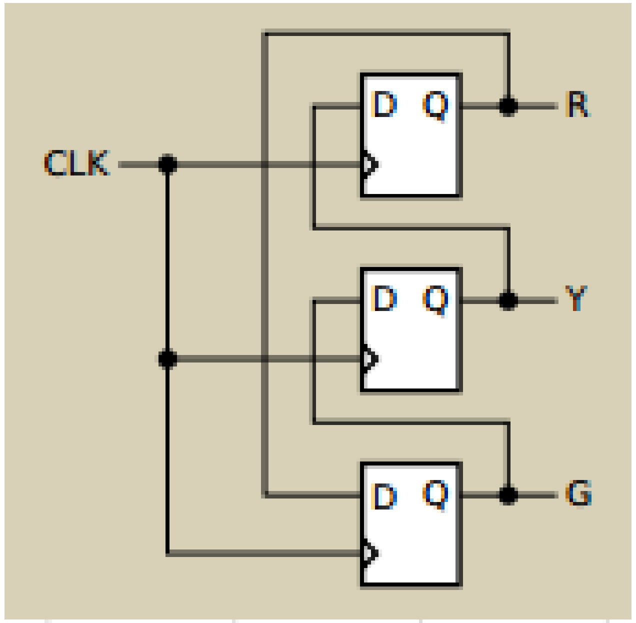
Positive-Edge Triggered D Flip-Flop

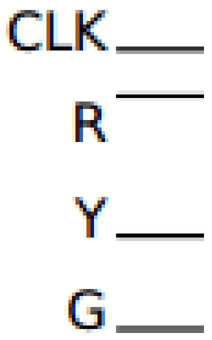
- Formed by adding inverter to clock input



- Q changes to the value on D applied at the positive clock edge within timing constraints to be specified
- Our choice as the **standard flip-flop** for most sequential circuits

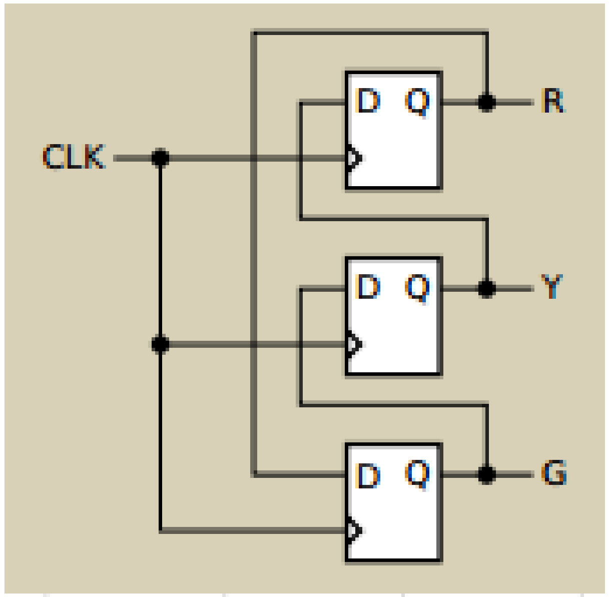
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller

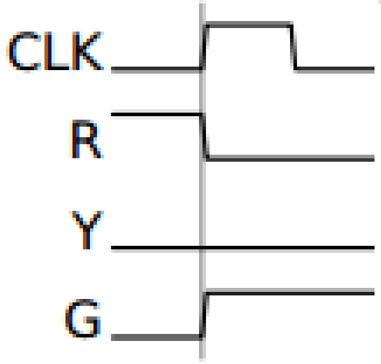




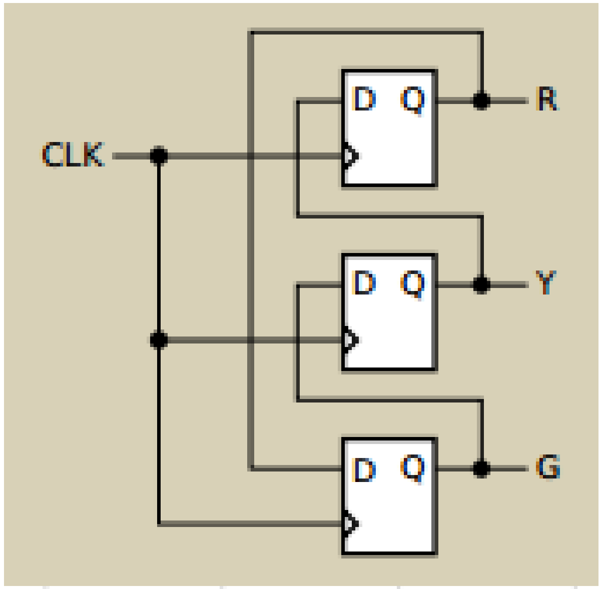
Initial state: Red light

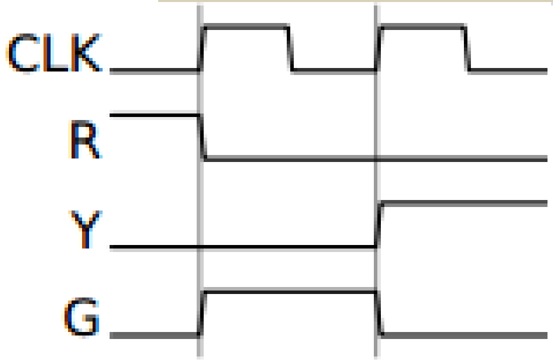
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller



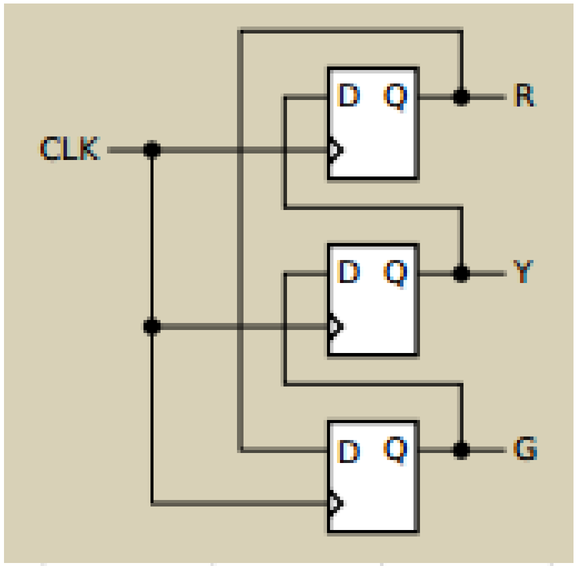


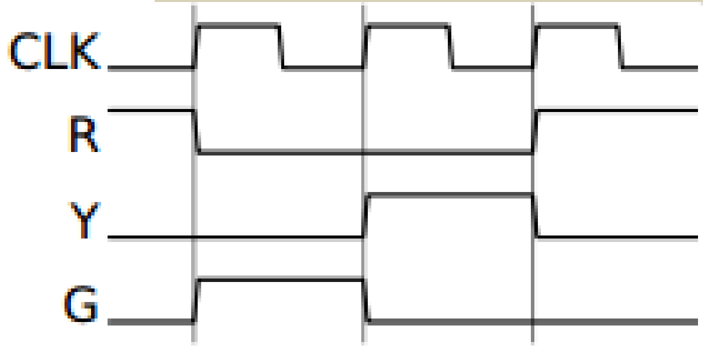
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller



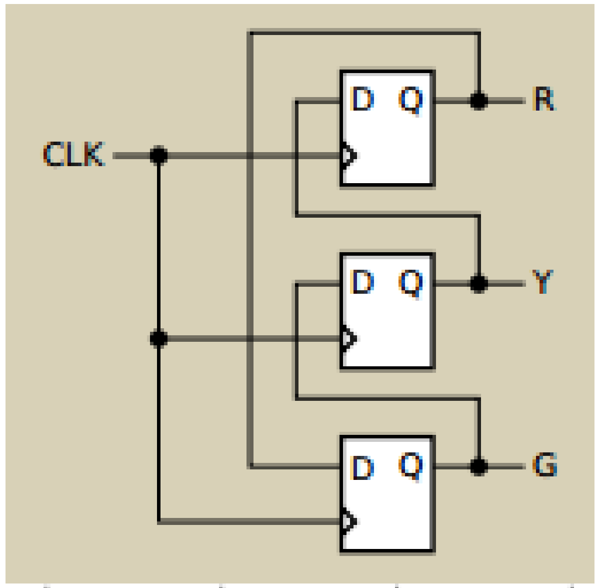


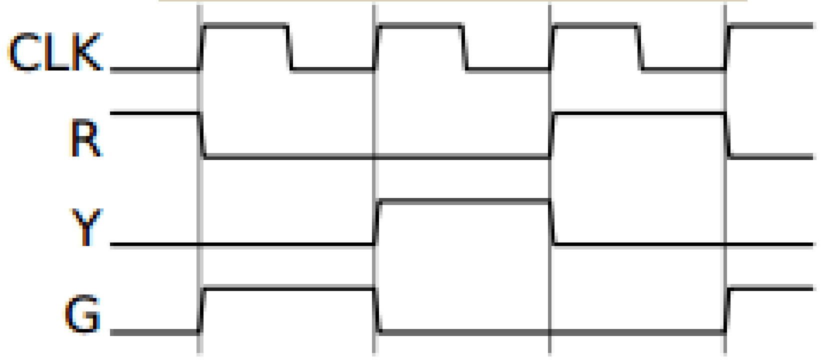
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller





Positive-Edge Triggered D Flip-Flop: Traffic Light Controller





Any Questions?