Professor Brendan Morris, SEB 3216, brendan.morris@unlv.edu

CPE100: Digital Logic Design I

Midterm02 Review

http://www.ee.unlv.edu/~b1morris/cpe100/

Logistics

- Thursday Nov. 15th
  - In normal lecture (13:00-14:15)
  - 1 hour and 15 minutes
- Chapters 2.7-3.4
  - Responsible for all material but emphasis on sections since Midterm01
- Closed book, closed notes
- No calculators
- Must show work and be legible for credit

Preparation

- Read the book (2nd Edition)
  - Then, read it again
- Do example problems
  - Use both Harris and Roth books
- Be sure you understand homework solutions

- Come visit during office hours for questions

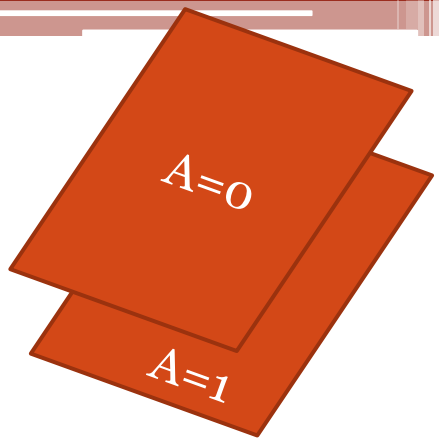
- Exam Advice: Be sure to attempt all problems.
  - Partial credit can only be given for something written on the page
  - Don't spend too much time thinking

Chapter 2.7 K-Maps

- Logic minimization in graphical form
  - Generally easier than using Theorems/Axioms
  - Expected to know up to 5 variables

- Use K-map to encode truth table
  - Adjacent rows/columns only differ by a single bit to exploit combining
  - Implement both SOP (“1”) and POS (“0”) forms
    - Draw largest circle possible to cover each 1
  - Take advantage of Don’t Cares (“X”) to have more simple logic

Chapter 2.7 Kmap Example



- 5-input function (A,B,C,D,E)
  - Create two 4-input K-maps and "stack"

A = 0

DE \ BC	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

A = 1

DE \ BC	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

  - Draw bubbles within 4x4 and in between stack (above or below)
    - E.g. cell 5 and 21 → B'CD'E

Example 8

- $Y = \sum m(0,1,2,3,8,9,16,17,20,21,24,25,28,29,30,31)$

A = 0

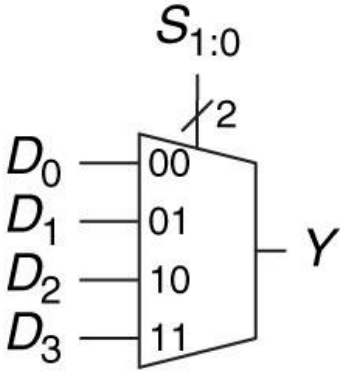
DE \ BC	00	01	11	10
00	1	4	12	1
01	1	5	13	1
11	1	7	15	11
10	1	6	14	10

A = 1

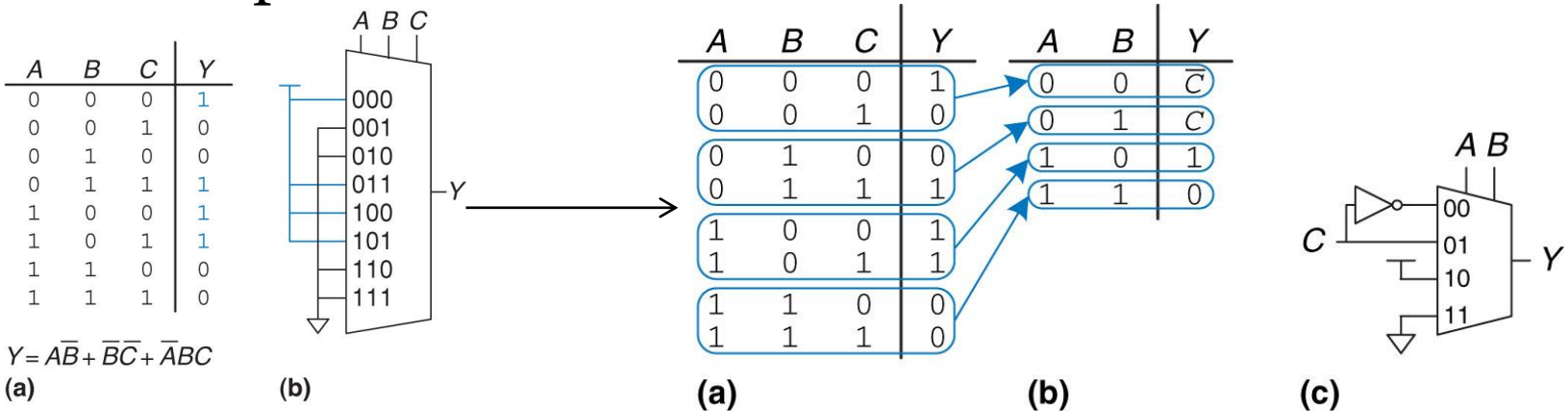
DE \ BC	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	19	23	1	27
10	18	22	1	26

  - Be sure to check “above/below”
- Y=AD’+ A’B’C’ + ABC + C’D’

Chapter 2.8.1 – Mux

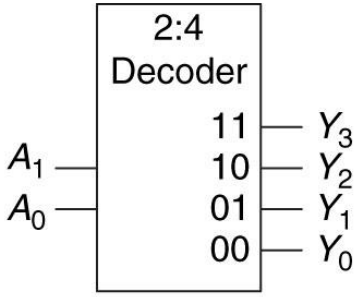


- Select one of N inputs for output
  - Select → $\log_2 N$-bits
- Mux logic:
  - Use as a lookup table with zero outputs tied to ground and one output to V_{DD}
  - Use simplification technique for smaller mux size
    - Combine rows and move far right input variable into the output column

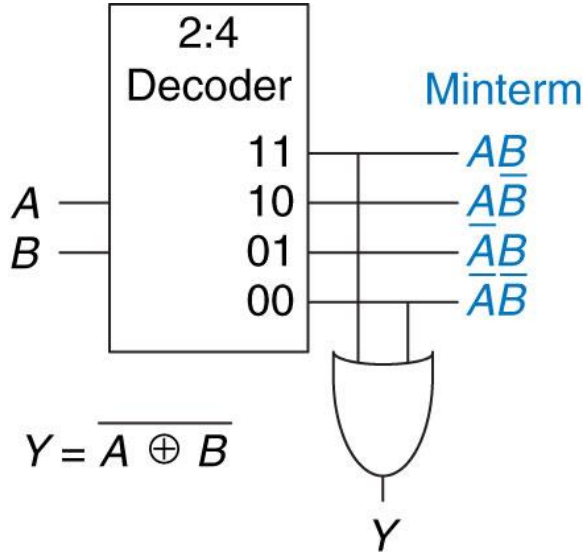


Chapter 2.8.2 Decoder

- Given N inputs → 2^N (one-hot) outputs
  - Each output is a row of truth table
- Decoder logic:
  - Build SOP logic by OR-ing output

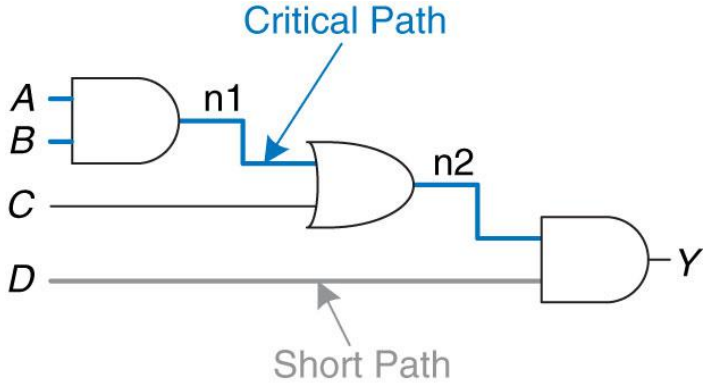


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

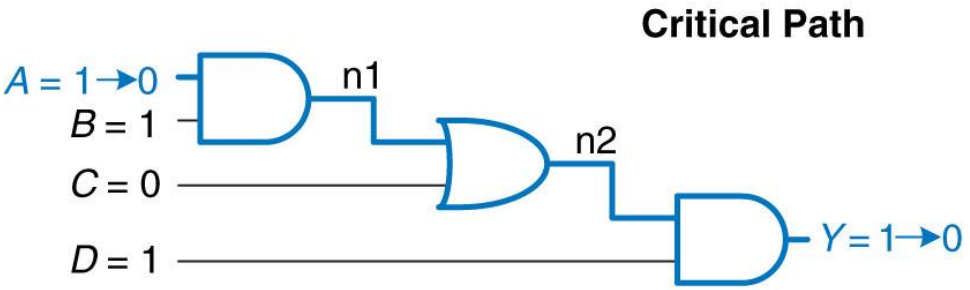


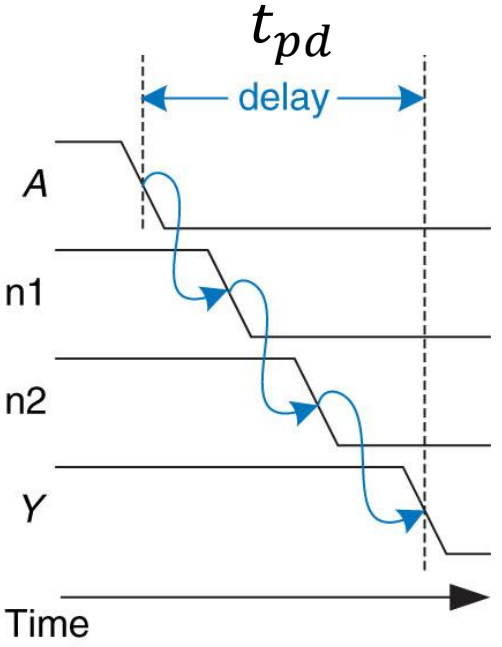
Chapter 2.9 Timing

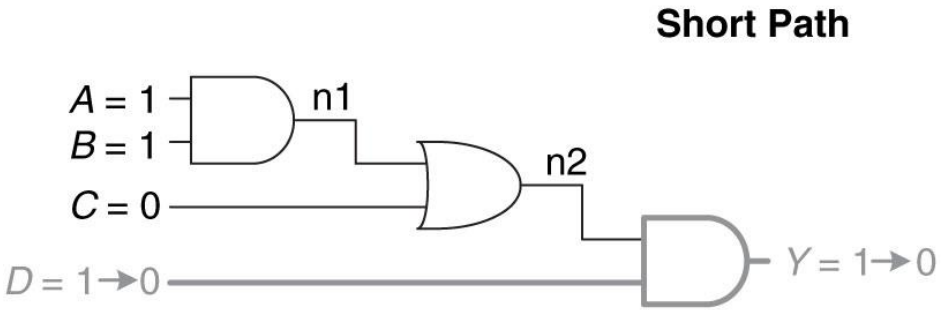
- Takes time (delay) for input change to cause output change
  - Signal must travel through logic gates
- Two important delay components
  - Propagation - t_{pd} is max time from input to final stable output (longest path)
  - Contamination - t_{cd} is minimum time from input to first change in output (shortest path)

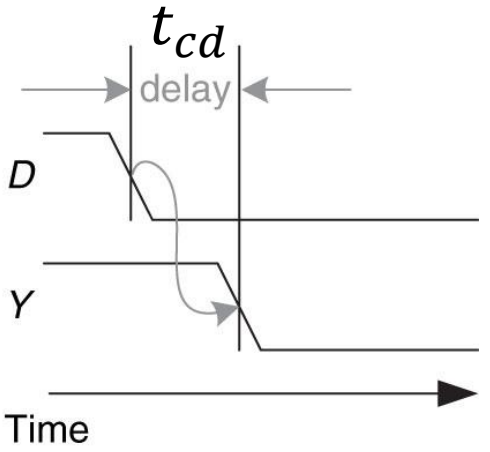


Chapter 2.9 Timing







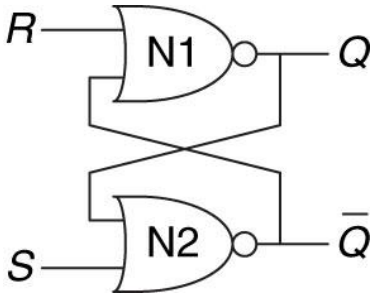


Chapter 3 Sequential Logic Design

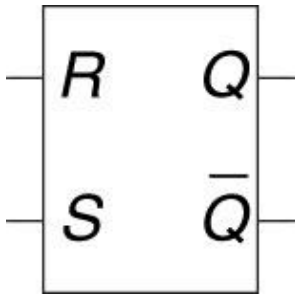
- Logic that depends on both current input as well as past input values (memory)
- State – all information about a circuit necessary to explain its future behavior
- Latches and flip-flops – state elements that store a single bit of state (memory element)
- Synchronous sequential circuits – combinational logic followed by a register

Chapter 3.2.1 SR Latch

- Bistable circuit to store state Q and $\bar{Q}$
  - S – set $Q = 1$
  - R – reset $Q = 0$
  - $S = R = 0$ – hold Q state



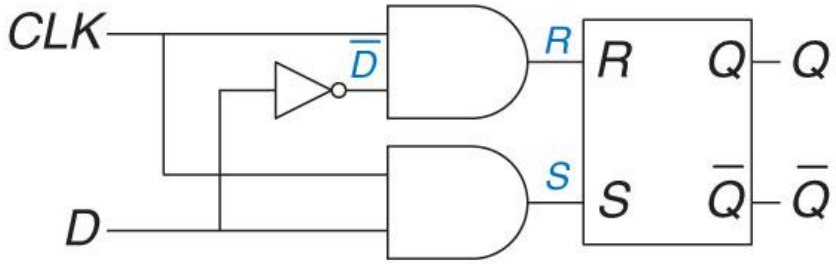
- Circuit symbol and operation
  - Note: logic does not hold for $S = R = 1$



Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Chapter 3.2.2 D Latch

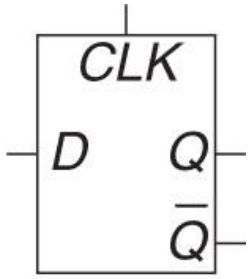
- Simplify SR Latch logic
  - D – single input
  - CLK – pass D on high cycle
  - Avoids previous $Q \neq \bar{Q}$ case



(a)

CLK	D	$\bar{D}$	S	R	Q	$\bar{Q}$
0	X	$\bar{X}$	0	0	Q_{prev}	$\bar{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

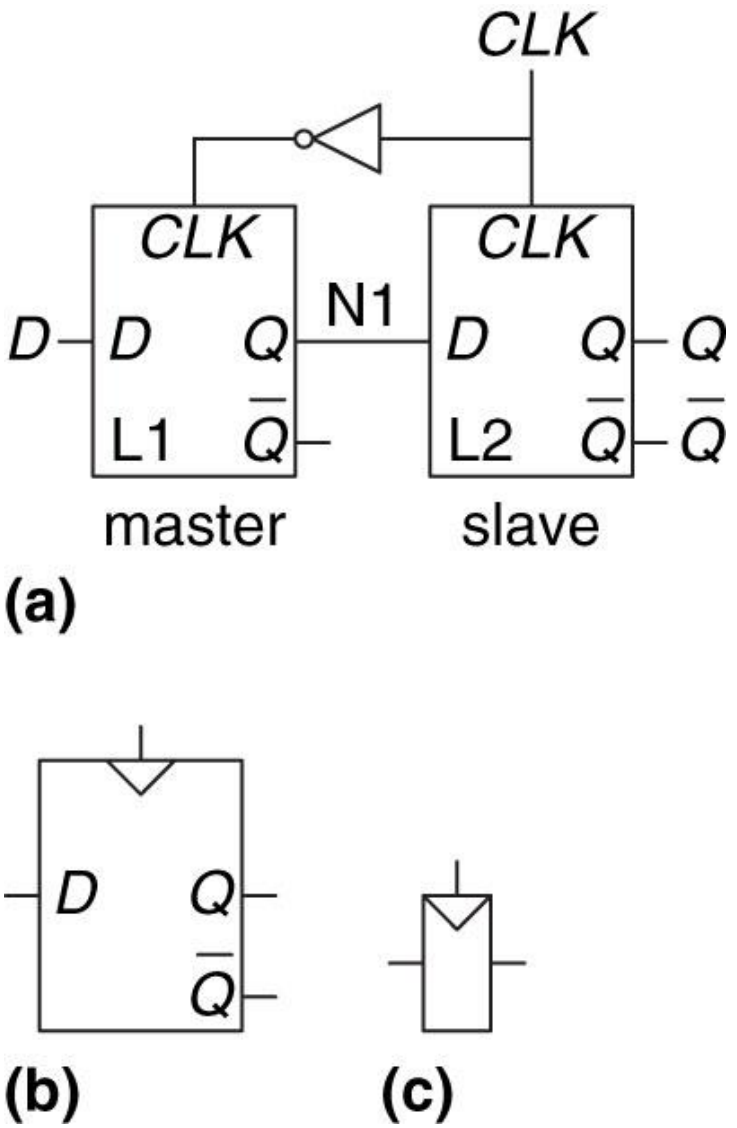
(b)



(c)

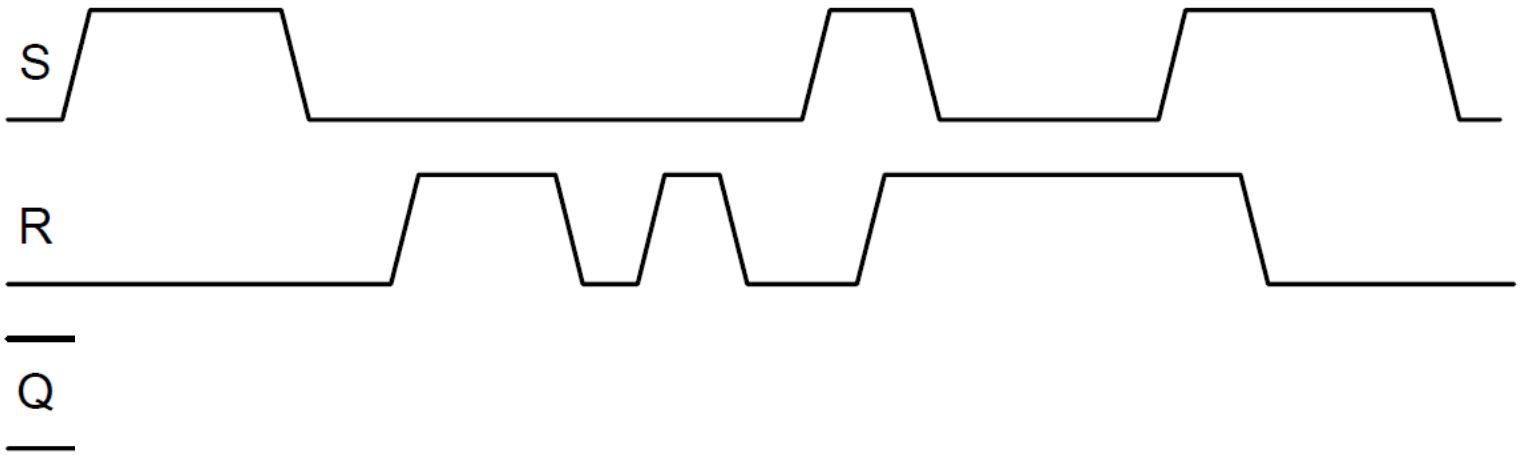
Chapter 3.2.3 D Flip-Flop

- More tightly controlled timing than D latch
  - Only passes D value on rising edge of CLK
- Edge-triggered device
  - Only activated on CLK transition from 0→1
  - Samples value of D at time of rising edge to pass through to Q



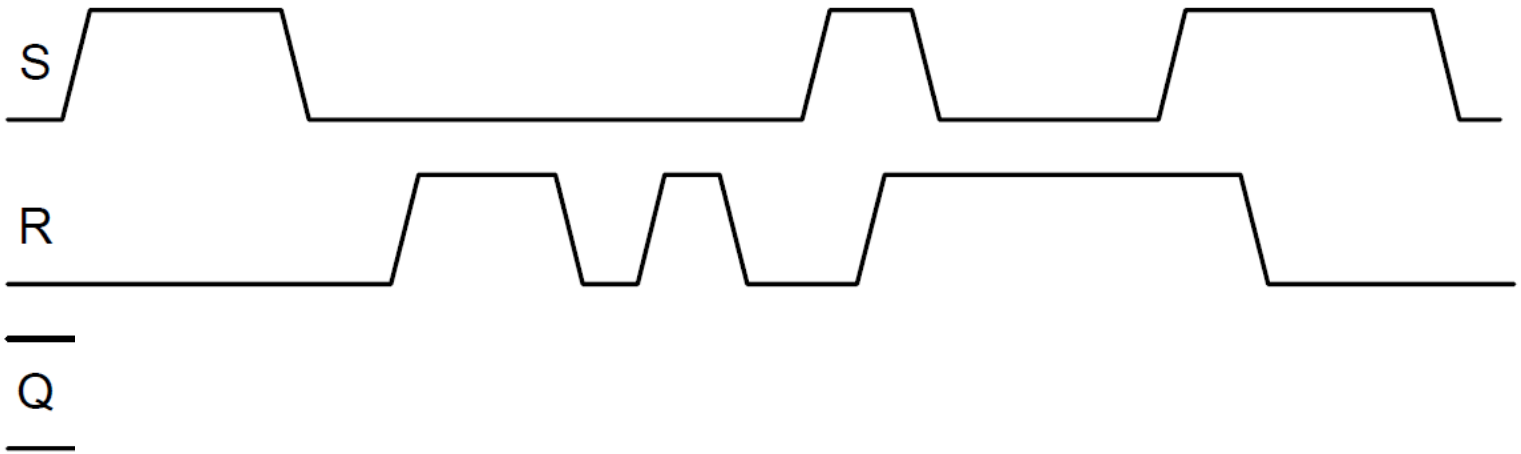
Chapter 3.2 Examples

- Given SR Latch provide output Q



Chapter 3.2 Examples

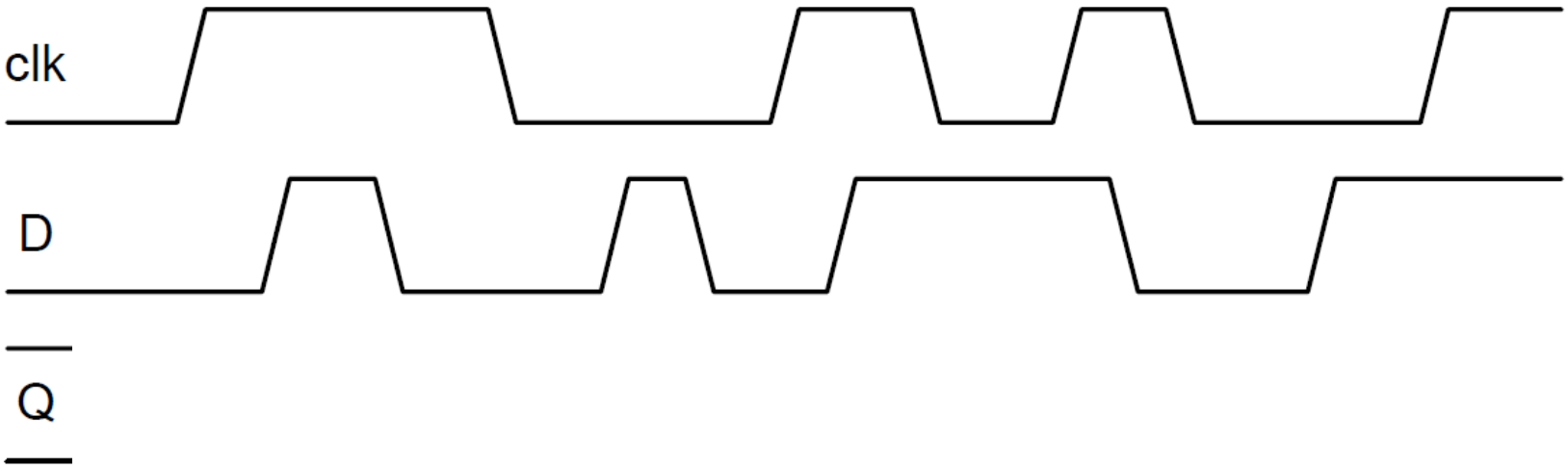
- Given SR Latch provide output Q
  - Note: $S = R = 1$ sets $Q = 0$



S	1	0	0	0	0	0	1	1	0	1	1
R	0	0	1	0	1	0	0	1	1	1	0
Q	**1**	**Qp =1**	**0**	**Qp =0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
Q'	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**

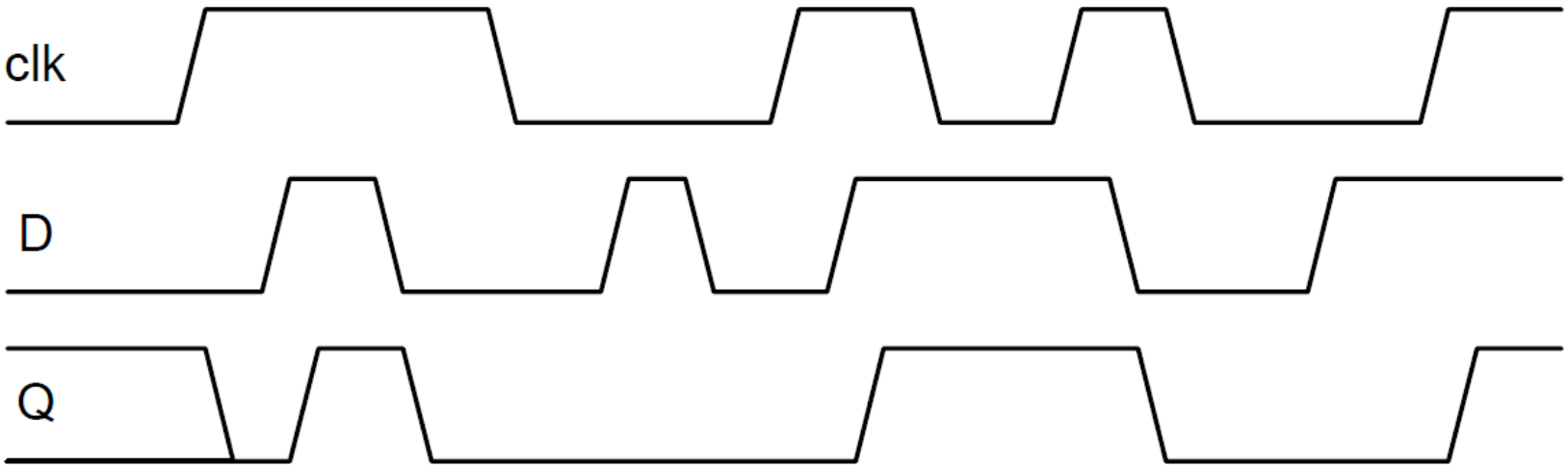
Chapter 3.2 Examples

- Given D Latch provide output Q
  - Note: Q "follows" D during CLK high period



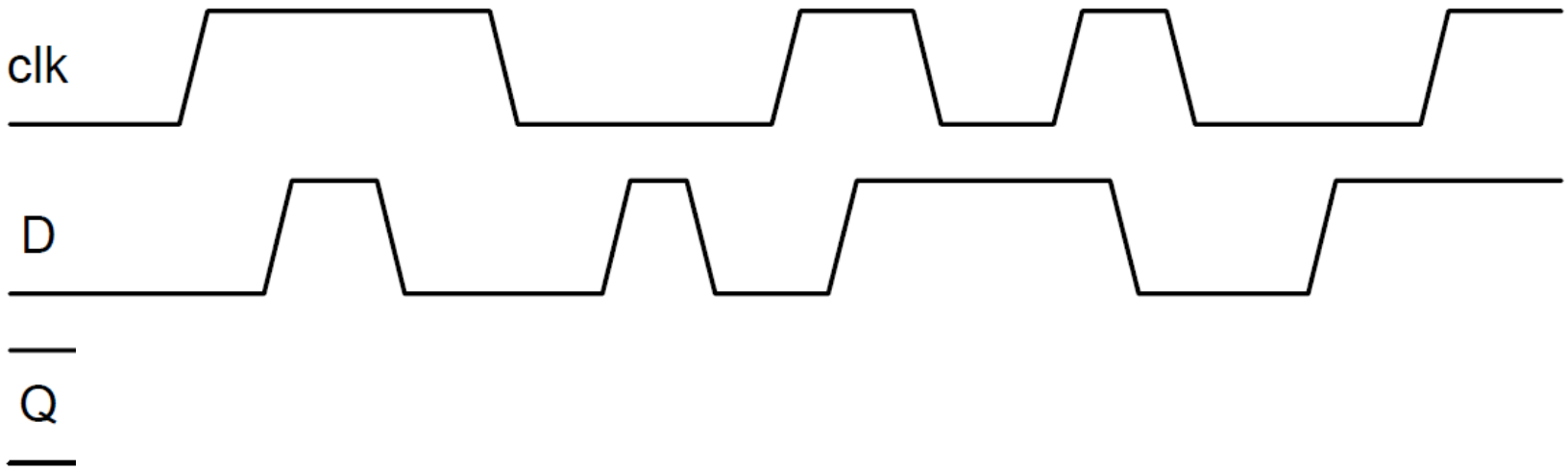
Chapter 3.2 Examples

- Given D Latch provide output Q
  - Note: Q "follows" D during CLK high period



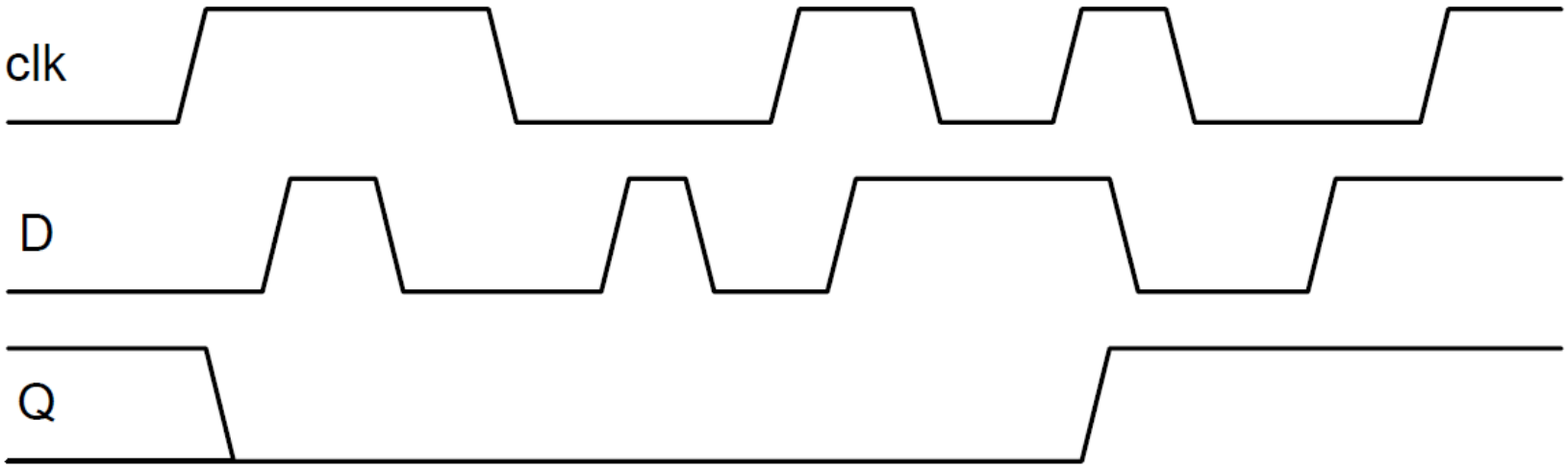
Chapter 3.2 Examples

- Given D flip-flop provide output Q



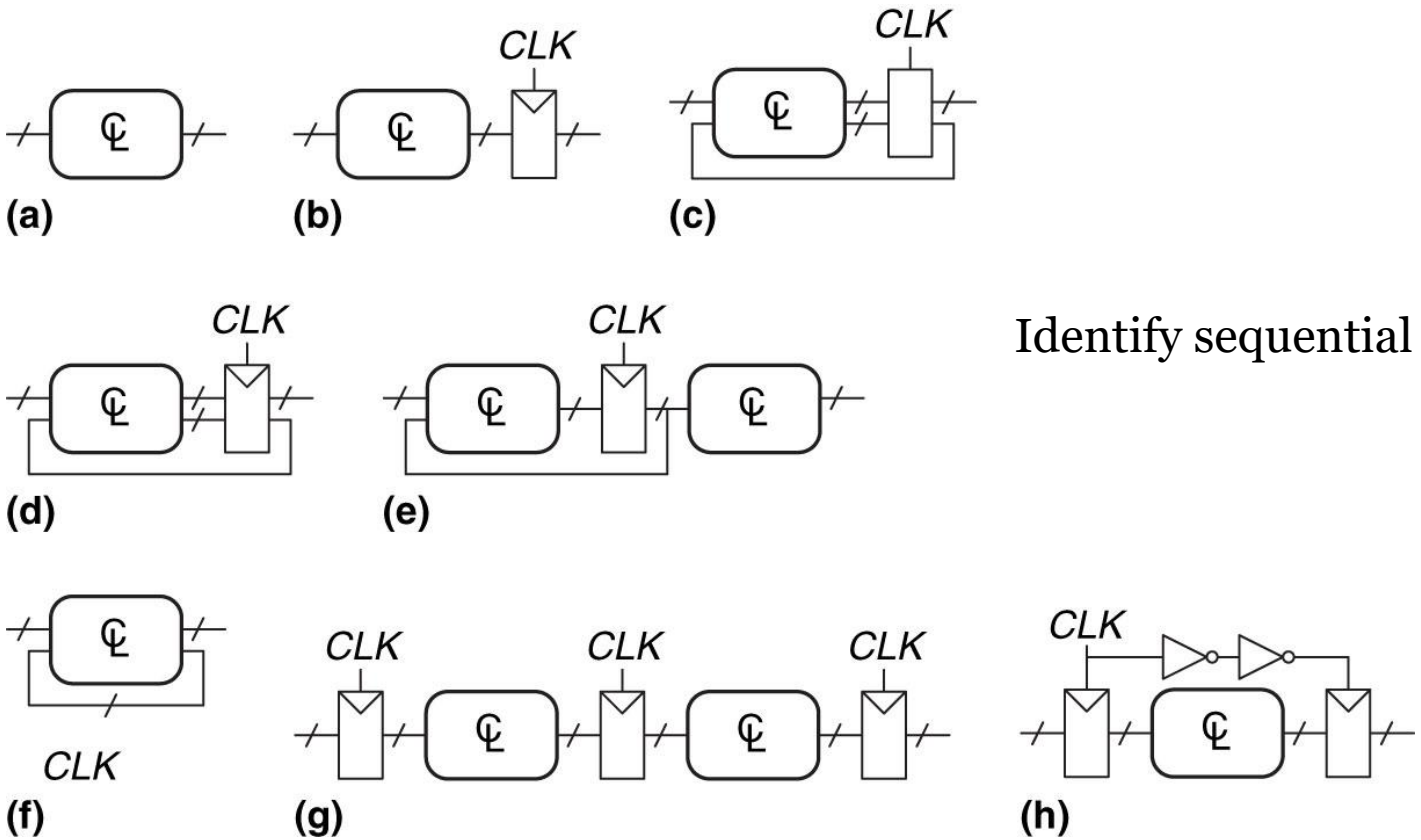
Chapter 3.2 Examples

- Given D flip-flop provide output Q
  - Note: Q "samples" D during CLK 0→1 transition



Chapter 3.3 Sequential Circuit Design

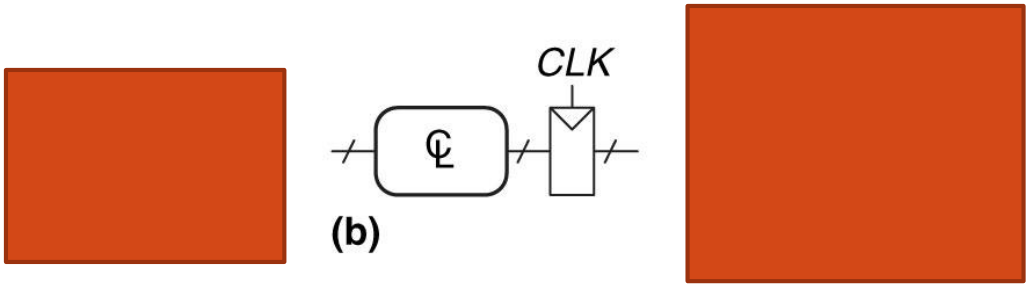
- Synchronous Design
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit element is a register
  - All registers receive the same clock signal
  - Every cyclic path contains at least one register

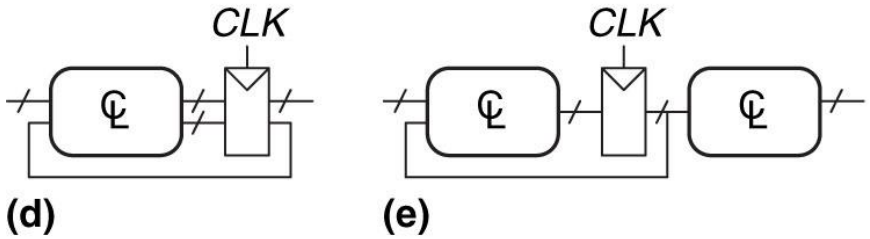


Identify sequential designs

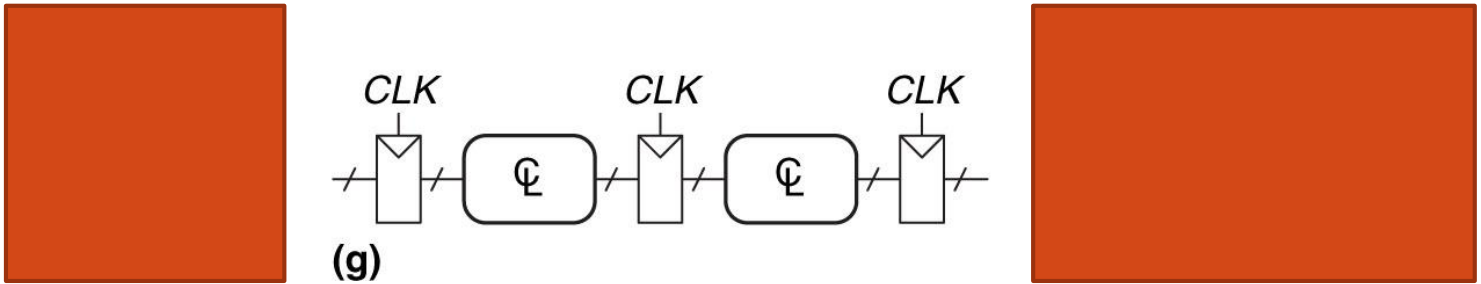
Chapter 3.3 Sequential Circuit Design

- Synchronous Design
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit element is a register
  - All registers receive the same clock signal
  - Every cyclic path contains at least one register



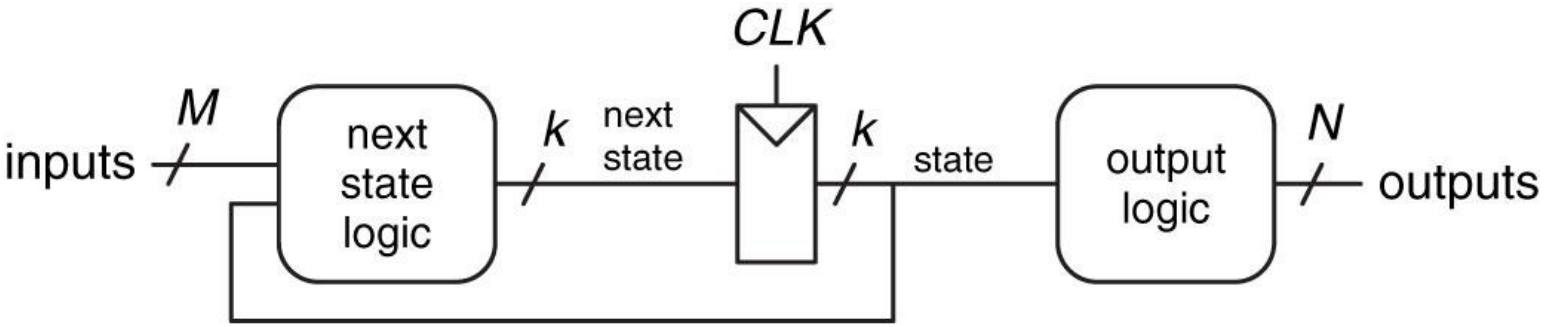


Identify sequential designs



Chapter 3.4 Finite State Machine

- Technique for representing synchronous sequential circuit
  - Consists of combinational logic and state register
  - Moore machine – output only dependent on state (not inputs)



Chapter 3.4 FSM Design Steps

1. **Identify inputs and outputs**
2. **Sketch state transition diagram**
3. Write state transition table
4. Select state encodings
5. Rewrite state transition table with state encodings
6. Write output table
7. Write Boolean equations for next state and output logic
8. Sketch the circuit schematic

Chapter 3.4 FSM Examples

- Given problem description, give state transition diagram
- Given state transition diagram, encode state and provide next state/output equations
- Given FSM circuit, describe what system does and give state transition/output tables

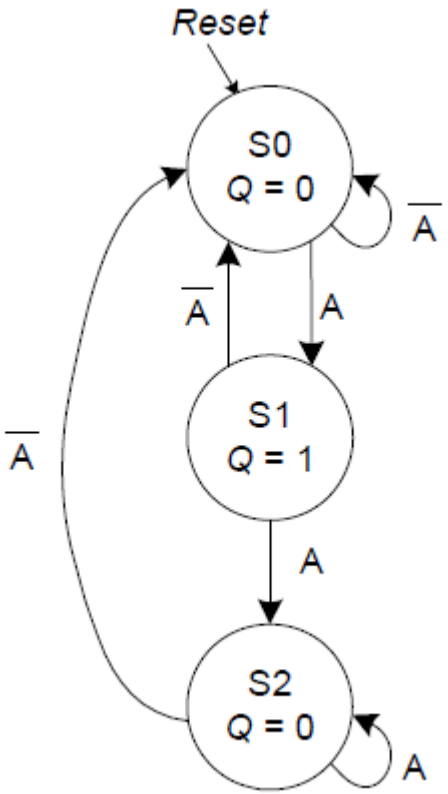
Chapter 3.4 FSM Examples

- Design and edge detector circuit. The output should go HIGH for one cycle after the input makes a 0 → 1 transition.

- Single input: A

FSM Example

- State transition diagram



- State/Output Tables
  - Use binary state encoding

S_1S_0	A	$S_1'S_0'$	Q
00	0	00	0
00	1	01	0
01	0	00	1
01	1	10	1
10	0	00	0
10	1	10	0

FSM Example

- Equations
- $S_1' = AS_1 + AS_0$
- $S_0' = A\overline{S_1}\,\overline{S_0}$
- $Q = S_1$

- Circuit diagram