Introduction to Computer Engineering – EECS 203

http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/

Instructor:	Robert Dick
Office:	L477 Tech
Email:	dickrp@northwestern.edu
Phone:	847–467–2298

TA:	Neal Oza
Office:	Tech. Inst. L375
Phone:	847-467-0033
Email:	nealoza@u.northwestern.edu

TT:	David Bild
Office:	Tech. Inst. L470
Phone:	847-491-2083
Email:	d-bild@northwestern.edu

NORTHWESTERN
UNIVERSITY

Outline

Word description to state diagram

- Design a vending machine controller that will release (output signal r) an apple as soon as 30¢ have been inserted
- The machine's sensors will clock your controller when an event occurs. The machine accepts only dimes (input signal d) and quarters (input signal q) and does not give change
- When an apple is removed from the open machine, it indicates this by clocking the controller with an input of d
- The sensors use only a single bit to communicate with the controller

Word description to state diagram

- We can enumerate the inputs on which an apple should be released

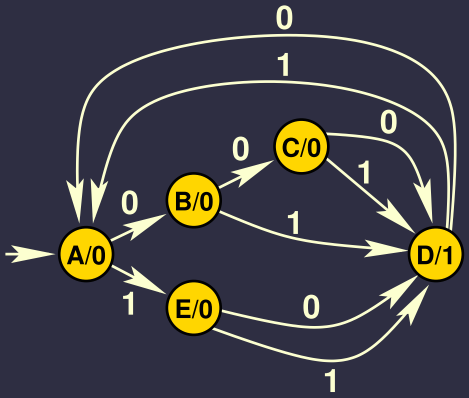
$$ddd + ddq + dq + qd + qq$$
$$d(dd + dq + q) + q(d + q)$$
$$d(d(d + q) + q) + q(d + q)$$

For d, $i = 0$, for q, $i = 1$

$$0(0(0 + 1) + 1) + 1(0 + 1)$$

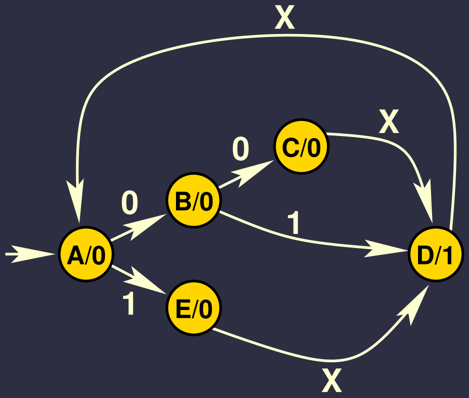
Word description to state diagram

0(0(0 + 1) + 1) + 1(0 + 1)



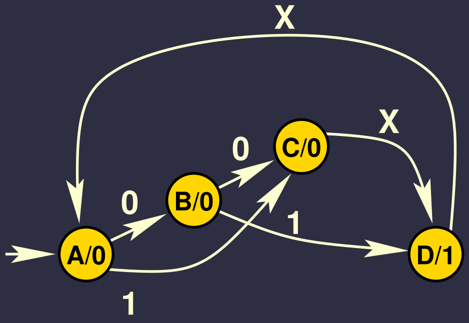
Word description to state diagram

0(0(0 + 1) + 1) + 1(0 + 1)



Word description to state diagram

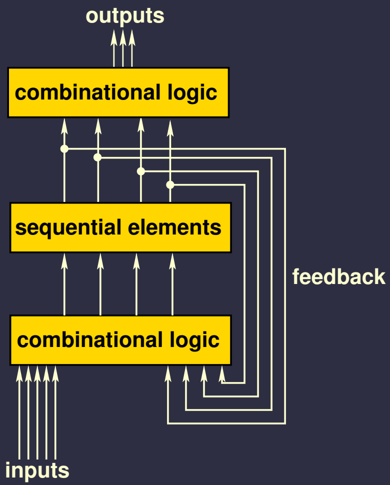
0(0(0 + 1) + 1) + 1(0 + 1)



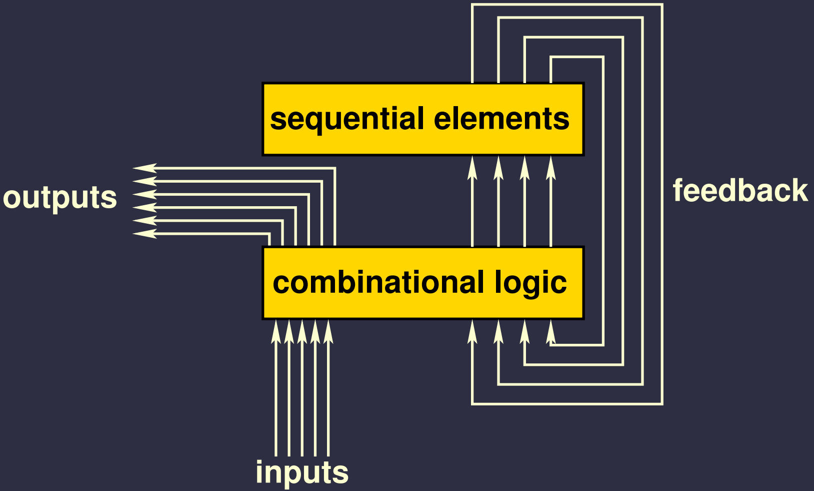
State diagram to state table

Current state	next state i=0	next state i=1	output (r)
A	B	E	0
B	C	D	0
C	D	D	0
D	A	A	1
E	D	D	0

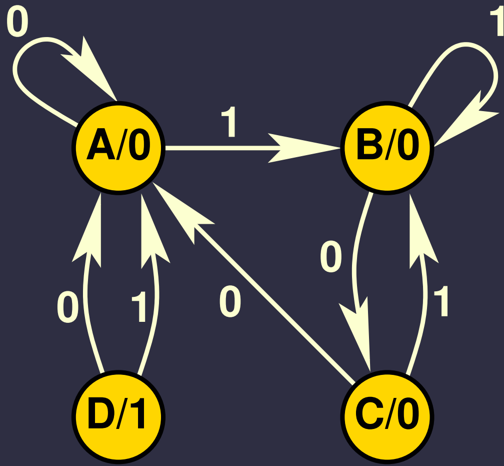
Moore block diagram



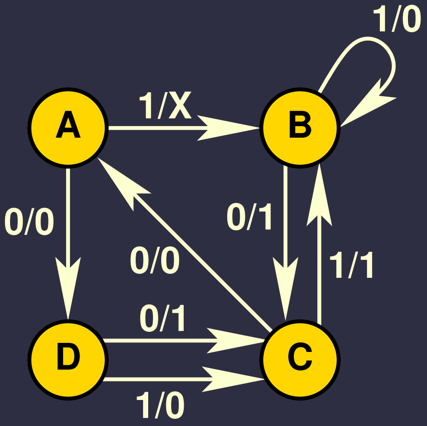
Mealy block diagram



Moore FSMs



Mealy FSMs



Mealy tabular form

	s^+/q	
s	0	1
A	D/0	B/X
B	C/1	B/0
C	A/0	B/1
D	C/1	C/0

FSM design summary

- Specify requirements in natural form
- Manually derive state diagram
  - Automatic way to go from English to FSM, however more theory required
  - Can minimize state count, however, more theory also required
  - See me if you want more information on this, or take a compilers course and a graduate-level switching theory course, or take my ECE 303
- Assign values to states to minimize logic complexity
- Optimize implementation of state and output functions

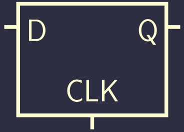
Outline

Back to latches

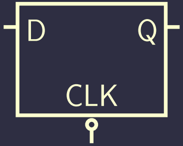
- Latches: Level sensitive
- Flip-flops: Edge-triggered

Review: Clocking conventions

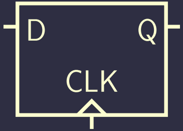
Active-high transparent



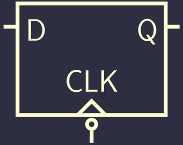
Active-low transparent



Positive (rising) edge



Negative (falling) edge



Latch and flip-flop equations

RS

$$Q^+ = S + \overline{R}\ Q$$

D

$$Q^+ = D$$

Latch and flip-flop equations

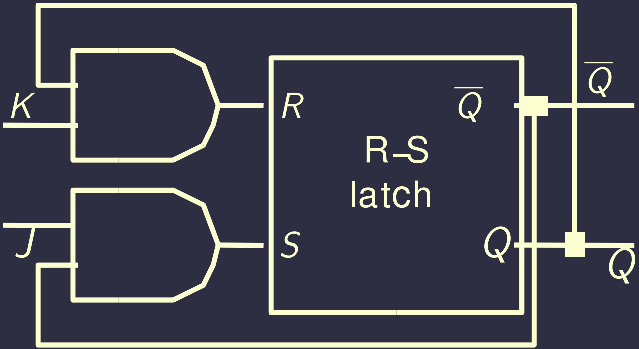
JK

$$Q^+ = J\,\overline{Q} + \overline{K}\;Q$$

T

$$Q^+ = T \oplus Q$$

JK latch



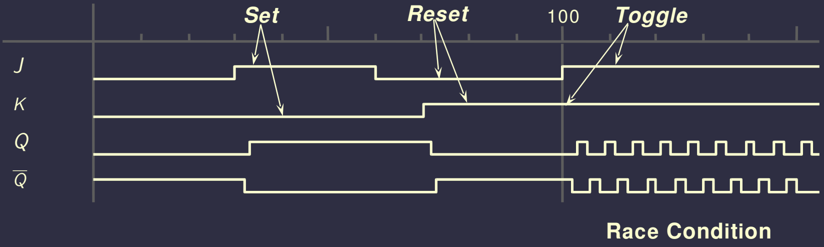
Use output feedback to ensure that $RS \neq 11$

$Q^+ = Q\,\overline{K} + \overline{Q}\;J$

JK latch

J	K	Q	Q^+	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

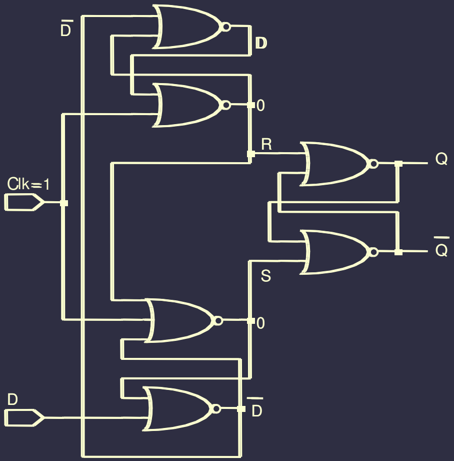
JK race



Falling edge-triggered D flip-flop

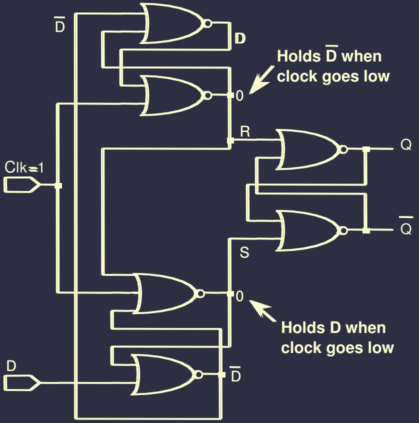
- Use two stages of latches
- When clock is high
  - First stage samples input w.o. changing second stage
  - Second stage holds value
- When clock goes low
  - First stage holds value and sets or resets second stage
  - Second stage transmits first stage
- $Q^+ = D$
- One of the most commonly used flip-flops

Falling edge-triggered D flip-flop



Clock high

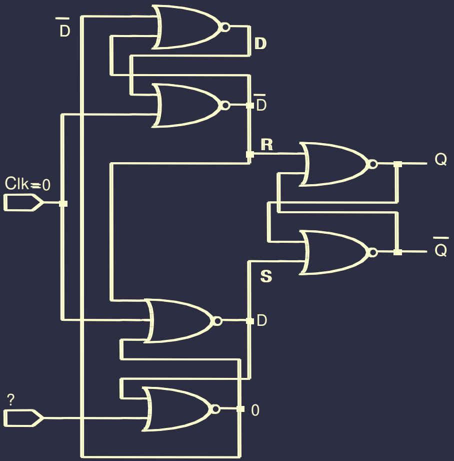
Falling edge-triggered D flip-flop


$\overline{D}$
D
Holds $\overline{D}$ when clock goes low
0
R
Q
$\overline{Q}$
Clk=1
S
0
Holds D when clock goes low
D
$\overline{D}$

Clock switching

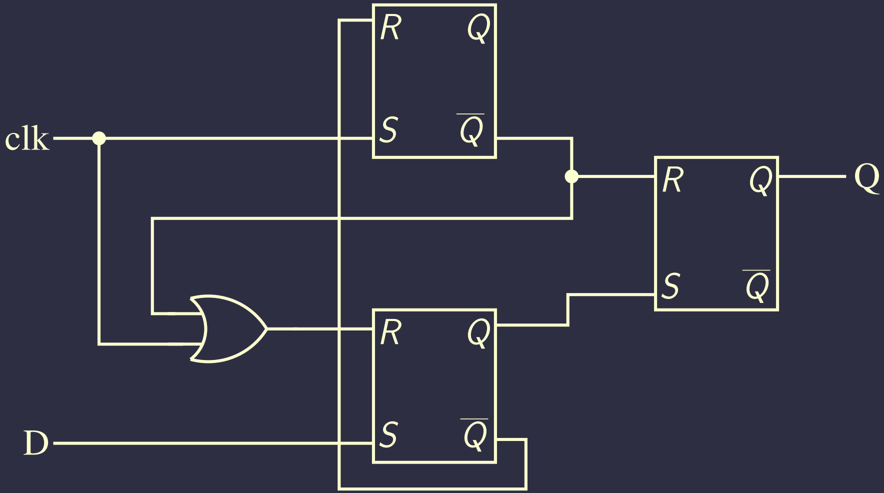
Inputs sampled on falling edge, outputs change after falling edge

Falling edge-triggered D flip-flop

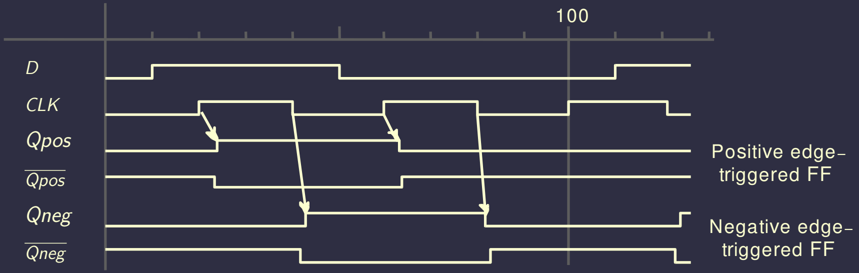


Clock low

Another view of an edge-triggered DFF



Edge triggered timing



RS clocked latch

- Storage element in narrow width clocked systems
- Dangerous
- Fundamental building block of many flip-flop types

JK flip-flop

- Versatile building block
- Building block for D and T flip-flops
- Has two inputs resulting in increased wiring complexity
- Edge-triggered varieties exist

D flip-flop

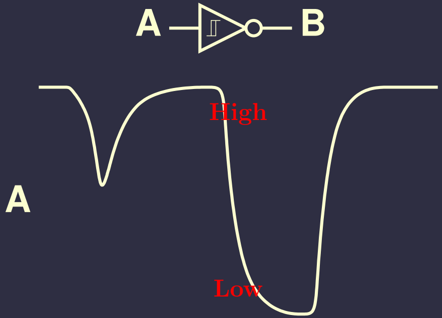
- Minimizes input wiring
- Simple to use
- Common choice for basic memory elements in sequential circuits

Outline

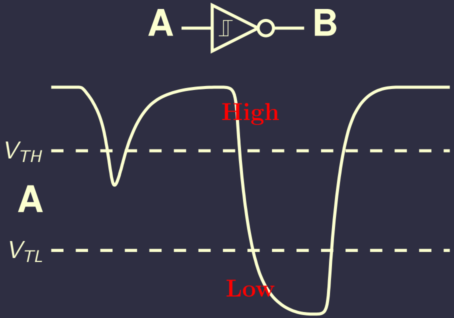
Debouncing

- Mechanical switches bounce!
- What happens if multiple pulses?
  - Mutliple state transitions
- Need to clean up signal

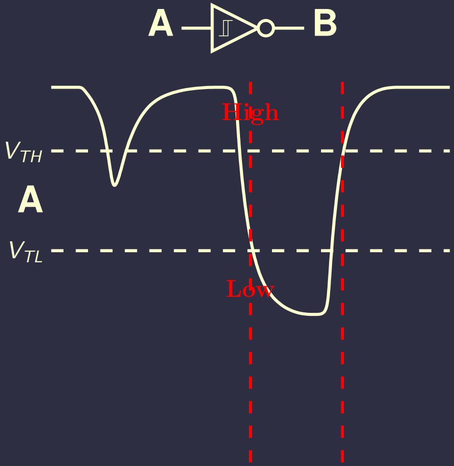
Schmitt triggers



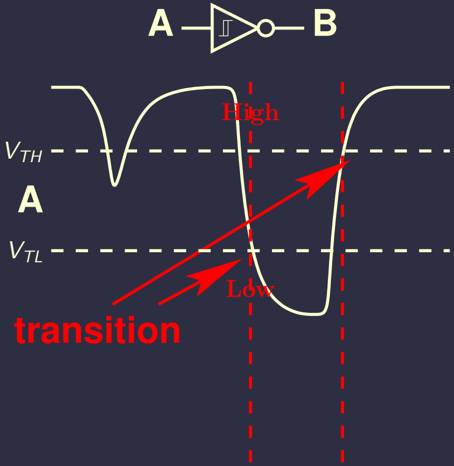
Schmitt triggers



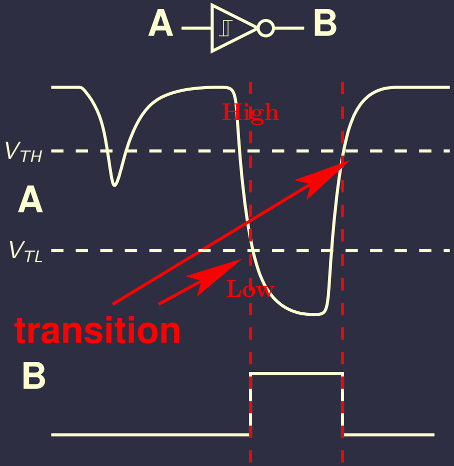
Schmitt triggers



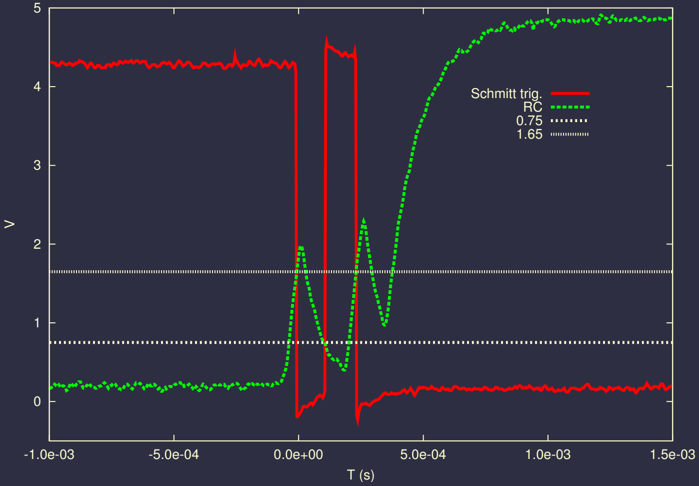
Schmitt triggers



Schmitt triggers



Debouncing


5
4
3
2
1
0
V
-1.0e-03
-5.0e-04
0.0e+00
5.0e-04
1.0e-03
1.5e-03
T (s)
Schmitt trig.
RC
0.75
1.65

Outline

Assigned reading

- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Review Sections 5.1–5.7
  - If FSMs don't make sense now, please ask questions, or see me
  - FSMs are tricky at first – Almost everybody has this moment of epiphany at which they suddenly make sense
- Section 9.1–9.6

Computer geek culture references

- Parsers and lexical analyzers
- Writing problem-specific languages
- A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers principles, techniques, and tools*. Addison-Wesley, MA, 1986
- Lex and yacc
- Flex and bison