Binary MSB most significant bit

LSB least ~

Byte 8 bits

Word Natural unit of data for a processor (32/64 bit)

2's complement MSB negative E. $\underset{-8\,4\,2}{1\,1\,1\,0} = -2$

0 1~7 8~15

-8~-1 0 1~7

Negation flip bits → +1 Pf. $x+\overline{x}=2^n-1$

E. 0101 → 1010 +1 = 1011

5 -8+2+1 = -5

Extension E. 0110 → 0000 0110

1110 → 1111 1110

Commutative: $X+Y=Y+X$, $X\cdot Y=Y\cdot X$

Associative: $X+(Y+Z)=(X+Y)+Z$, $X\cdot(Y\cdot Z)=(X\cdot Y)\cdot Z$

Distributive: $X\cdot(Y+Z)=(X\cdot Y)+(X\cdot Z)$, $X+(Y\cdot Z)=(X+Y)\cdot(X+Z)$

Absorption: $X+(X\cdot Y)=X$, $X\cdot(X+Y)=X$

Identities: $A+0=A$, $A+1=1$, $A+A=A$, $A+\overline{A}=1$, $0\cdot A=0$, $1\cdot A=A$, $A\cdot A=A$, $A\cdot\overline{A}=0$

DeMorgan's Law: $\overline{X\cdot Y}=\overline{X}+\overline{Y}$, $\overline{X+Y}=\overline{X}\cdot\overline{Y}$

Boolean algebra comm, asso, dist (a·(b+c) = (a·b)+(a·c) and vv.)

Delay - propagation t_{pd} max delay from any i → any o

- contamination t_{cd} min delay ~

critical path: longest (Σt_{pd})

short path: shortest (Σt_{cd})

→ ✓ after crit. ✓, but before, may have transient X → glitch

Timing diagram.

unique

Canonical forms 1. truth table

rare to be min. { 2. sum of minterms

3. product of maxterms

2-level logic

A Cascade of Vocabulary

complement: inverse of a variable (from set theory, the set of elements not in A)
$\overline{X}$, $\overline{A}$ X, A

literal: a variable or the complement of a variable
X, $\overline{X}$, $\overline{A}$, Y XY, A$\overline{B}$

product term: a conjunction (logical and) of literals can't be literal!
XY, C$\overline{D}$ X+Y, A$\overline{B}$+C, I(J+K)

minterm: product term that includes all input variables, complemented or not
ABC, AB$\overline{C}$ AC, $\overline{B}$

sum term: a disjunction (logical or) of literals
X+Y, $\overline{A}$+B+C XY, A+$\overline{B}$C

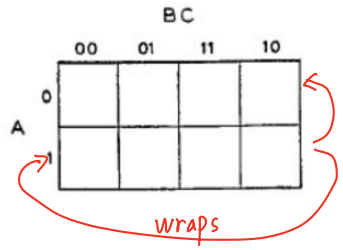
maxterm: sum term that includes all input variables, complemented or not
A+B+C, A+$\overline{B}$+C A+C, A+BC

sum of products (SoP) expression: sum (logical or) of product (logical and) terms
AB+A$\overline{C}$+ABC AB+A$\overline{C}$+B

product of sums (PoS) expression: product (logical and) of sum (logical or) terms
(A+B)($\overline{C}$+$\overline{B}$)(A+$\overline{B}$+C (A+B)($\overline{C}$+$\overline{B}$)(A)
)

Simp methods 1. Algebraic
2. CAD
3. Manual — Karnaugh maps, bubble pushing
↳ truth table, redrawn w/ adjacency

Adj. prop. For any adj. move, 1 bit flips
Look for 2 adj. 1s



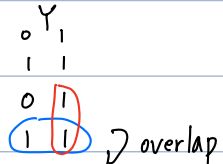
E. $F = \bar{X}Y + X\bar{Y} + XY$

$= \bar{X}Y + X$ (dist.)

Also $= Y + X\bar{Y}$

$F = (X + \bar{X}Y) + (Y + X\bar{Y})$

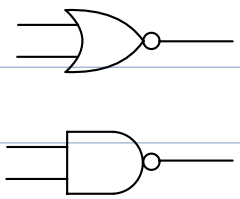
$F = X + Y$



Group - can overlap
- has all 1s
- has no 0s
- dim must be 2^n !

>4 var. 3D-ish

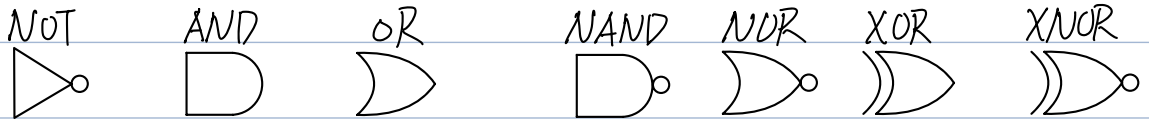
Design 1. Populate truth table
2. Populate K-map
3. if several outs, find common groups
4. sum the terms



Don't-care: X

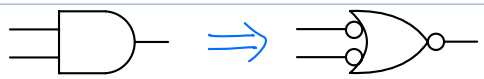
E. priority encoder: identifies the digit w/ the MS 1.
valid bit: 1 V, 0 X (if no 1 present). (simplifies T-table)

Bubble pushing



	NOT	AND	OR	NAND	NOR	XOR	XNOR
2-input CMOS #	2	6	6	4	4	8	8

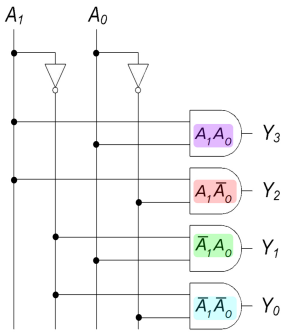
1. introduce bubbles
2. push → flip
3. add additional bubbs

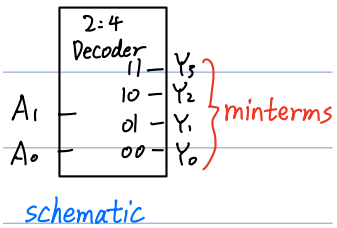


Fundamental building block ana. functions

Decoder k input, 2^k output (one 1, others 0)

one-hot



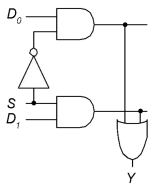


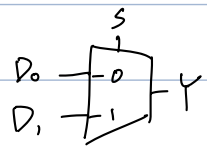
schematic

Multiplexer (mux)

$\log_2 N$ switches to select N inputs.

$Y = D_0\bar{S} + D_1 S$

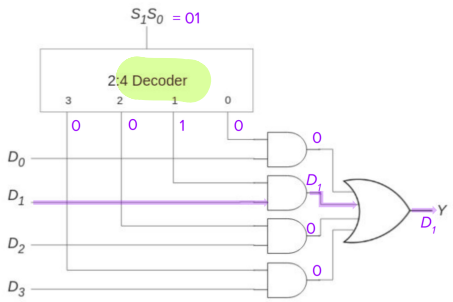




4-bit, use a decoder

implementation: connect inputs to +Vcc & G,

8→4 Choose some as select bits, write Y as a fucn. of the other bits.



Shifter 1. Logical fill extra by 0

E. 11001 >> 2 = 00110, 11001 << 2 = 00100

2. Arithmetic fill by value of MSB on right shift (preserves sign)

E. 11001 >>> 2 = 11110, 11001 <<< 2 = 00100

3. Rotator rotate by circle

E. 11001 RoR 2 = 01110, 11001 RoL 2 = 00111

$A << N = A \times 2^N$

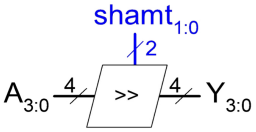
- **Example:** 00001 << 2 = 00100 $(1 \times 2^2 = 4)$
- **Example:** 11101 << 2 = 10100 $(-3 \times 2^2 = -12)$

$A >>> N = A \div 2^N$

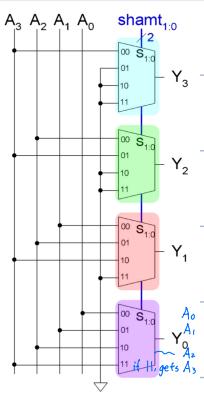
- **Example:** 01000 >>> 2 = 00010 $(8 \div 2^2 = 2)$
- **Example:** 10000 >>> 2 = 11100 $(-16 \div 2^2 = -4)$

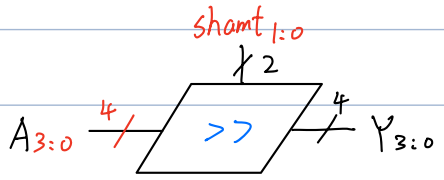
Watch overflow!

Implementation



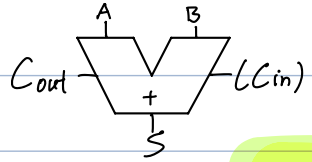
shamt	Y_3	Y_2	Y_1	Y_0
0 0	A_3	A_2	A_1	A_0
0 1	0	A_3	A_2	A_1
1 0	0	0	A_3	A_2
1 1	0	0	0	A_3





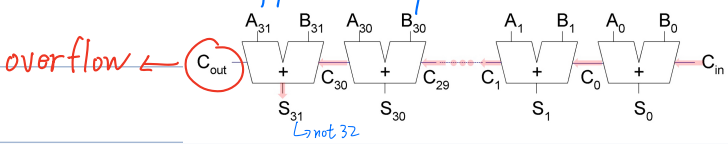
Adder Full: w/ carry, Half: w/o carry

carry ripples thru (need wait)



Multibit 1. Ripple carry

n x full, O(n) width ! slow :(



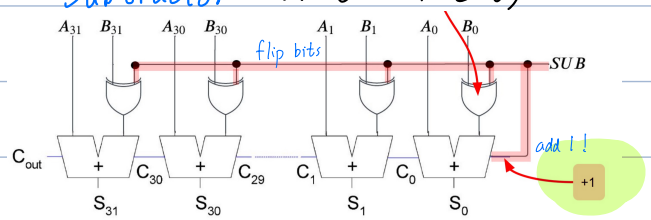
C_{30}	A_{31}	B_{31}	C_{31}	S_{31}		overflow = $C_{30} \oplus C_{31}$
0	0	0	0	0	pos + pos = pos	0
0	0	1	0	1	pos + neg cannot overflow	0
0	1	0	0	1	pos + neg cannot overflow	0
0	1	1	1	0	neg + neg = pos	1
1	0	0	0	1	pos + pos = neg	1
1	0	1	1	0	pos + neg cannot overflow	0
1	1	0	1	0	pos + neg cannot overflow	0
1	1	1	1	1	neg + neg = neg	0

E. -1 + -1

$C_{30} \neq C_{31}$, so $A_{31} = B_{31}$

$C_{31} = 1$ if two of $C_{30}, A_{31}, B_{31} = 1$

Subtractor $A - B = A + (-B)$



2. Carry lookahead (reduces carry chain)

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

$$= A_i B_i + C_i (A_i + B_i)$$

$$= G_i + C_i P_i \quad (G_i \equiv A_i B_i,\ P_i \equiv A_i + B_i)$$

generate, or, propogate and there's C already

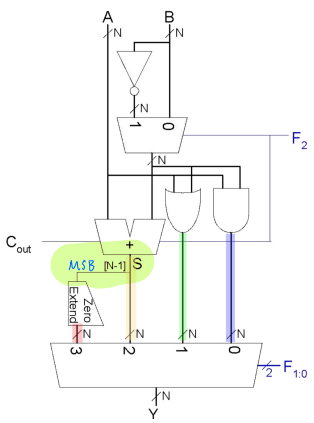
E. $C_2 = G_1 + C_1 P_1 = G_1 + (G_0 + C_0 P_0) P_1 = G_1 + G_0 P_1 + C_0 P_0 P_1$

inputs only

C_i has i terms, max $i+1$ literals

tot. delay $\propto$ Gate delay $\propto \log_2(\text{input})$

ALU Arithmetic logic unit



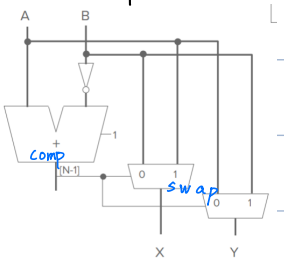
$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	A < B

SLT (set less than) (only Y_0 used)

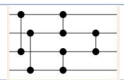
E. Sorter in: 4 4-bit int, out: 4 4-bit int, sorted

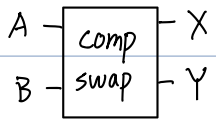
1. Design algorithm E. bubble: compare → swap → move largest → end

2. Use blocks to implement



3. Implement cir.





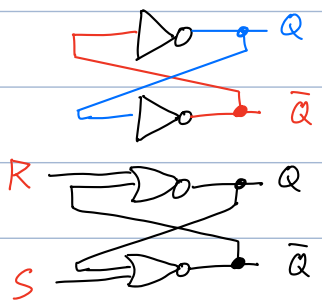
Depth = 3, size = 5

Sequential (has memory)

Store 1 bit bi-stable element (cross-coupled)

+control 1. SR latch

	S	R	Q(t+1)
set	1	0	1
res	0	1	0
hold	0	0	Q(t)
invalid	1	1	??



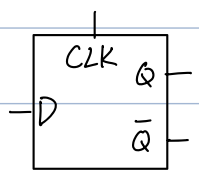
NOR

avoids

2. D-latch CLK = 0: Q = Q opaque

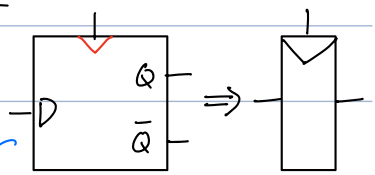
CLK = 1, Q = D transparent

imple. w/ SR + (o)D



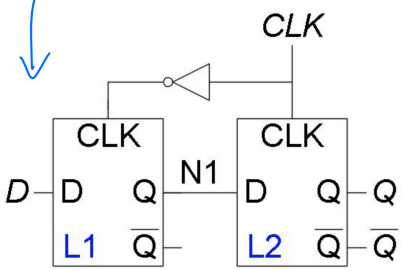
CLK ⊓⊔ 1 cycle: 1 rising edge, 1 falling ---

FF rising edge triggered C 0→1 ⇒ Q=D



DFF imp. only one latch is transparent

When C 0→1, Q = D (edge sensitive)



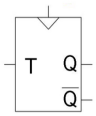
Options -enabled (if EN=0, no update) (MUX, 0→D)

-reset PRE = 1 forces Q=1, CLR = 1 forces Q=0

immediately, regardless CLK

Other types

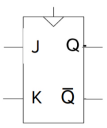
T ("toggle") flip-flop:



T	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

rising edge

JK flip-flop:



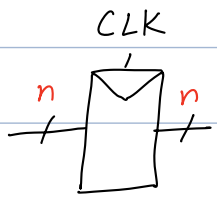
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

~X: from 0 ~=0: to 0

X~: from 1 =1 ~~to 1~~

Register

FF for each bit, same CLK



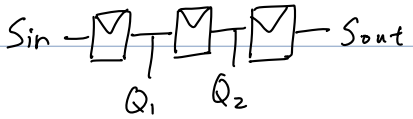
Counter

Add 1 per CLK cycle

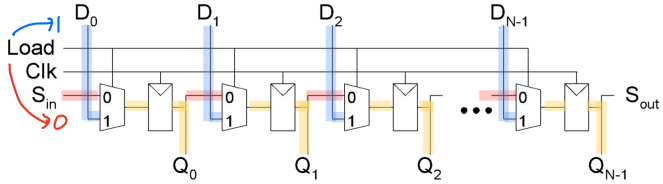
Shift register

serial FF, not parallel

Every cycle, FF shift a bit (adds delay)



Shift + ll



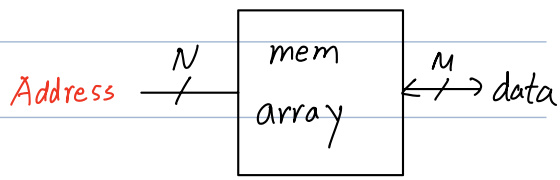
(E. parallel in → serial out)

Memory array

large amount of data (typically volatile)

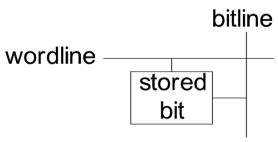
register → mem → disk

1. Dynamic random access mem. DRAM
2. Static RAM SRAM
3. Read only mem ROM



2D. n-bit add. → $2^N \times M$ (depth, width)

Read

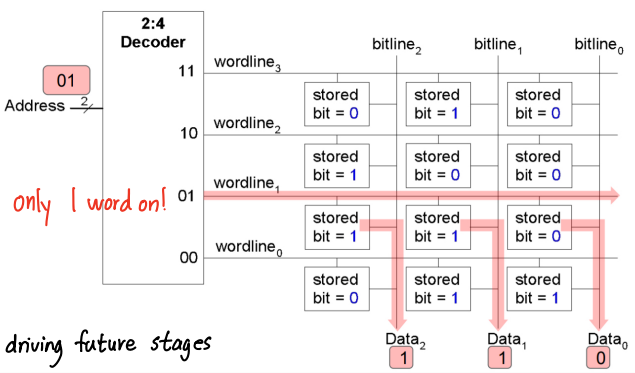


wordline	stored bit	bitline
0	X	Z
1	0	0
1	1	1

Z means "high impedance" (hi-Z, tri-stated, or floating) means signal is neither driven to a logical high nor low level and does not drive connected circuit. It is said to be "floating"

Only selected worldline drives bitlines, all others float.

↳ disconnected, not driving future stages



read M bits at a time (entire row)

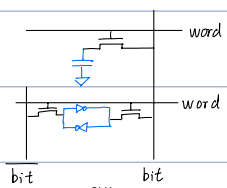
RAM

volatile, quick R/W

periodic maintain data

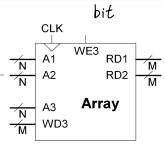
DRAM store data w/ capacitor: small, but leak, read destoyed

w/ bistable inverter: large, but fast read



ROM

not volatile, but hard to write

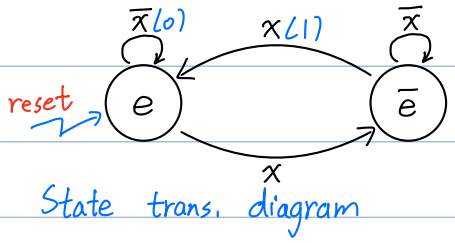


E. lookup table (LUT), multi-ported mem (1+ words at a time)

Finite state machine (use FF to hold state Q, combinatorial logic to compute Q_{n+1})

E. parity checker

DFA no accept state



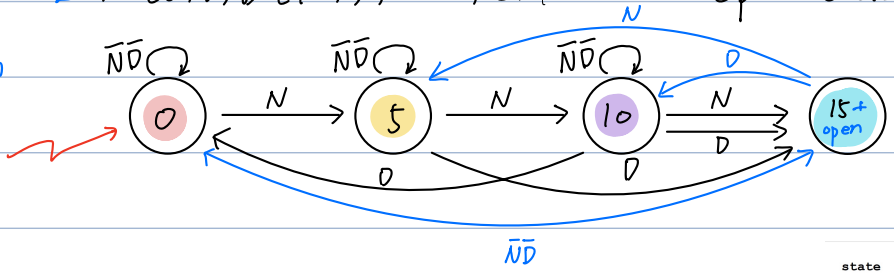
State trans. diagram

Design E. vending machine selling 15¢ item, accept 5¢, 10¢, no change

ND not allowed — std. components

1. identify I/O — I: N (5¢), D (10¢), RES, CLK — O: open (release)

2. Transition diagram



3. Trans. table

For each state and input, find next state and output

state	inputs N	inputs D	next state	output open
0¢	0	0	0¢	0
	0	1	10¢	
	1	0	5¢	
	1	1	x	
5¢	0	0	5¢	0
	0	1	15+¢	
	1	0	10¢	
	1	1	x	
10¢	0	0	10¢	0
	0	1	15+¢	
	1	0	15+¢	
	1	1	x	
15¢	0	0	0¢	1
	0	1	10¢	
	1	0	5¢	
	1	1	x	

4. Encode the states Q_1Q_0: 0 → 00, 5 → 01, 10 → 10, 15 → 11

5. Truth table w/ state, output encoding

DFF / TFF / JKFF

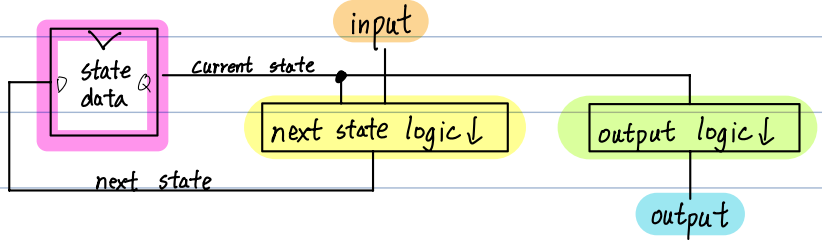
exploit don't cares

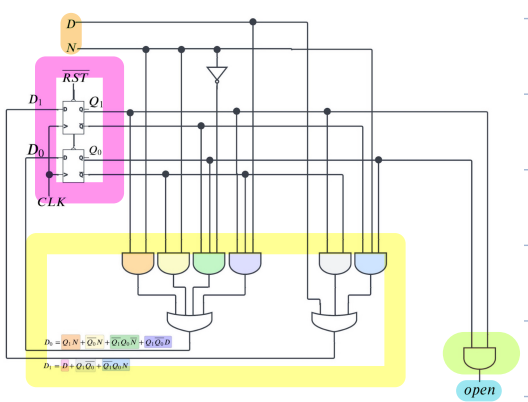
state Q_1Q_0	inputs N	inputs D	next state D_1D_0	output open
0¢ 00	0	0	0¢ 00	0
0¢ 00	0	1	10¢ 10	
0¢ 00	1	0	5¢ 01	
0¢ 00	1	1	x xx	
5¢ 01	0	0	5¢ 01	0
5¢ 01	0	1	15+¢ 11	
5¢ 01	1	0	10¢ 10	
5¢ 01	1	1	x xx	
10¢ 10	0	0	10¢ 10	0
10¢ 10	0	1	15+¢ 11	
10¢ 10	1	0	15+¢ 11	
10¢ 10	1	1	x xx	
15¢ 11	0	0	0¢ 00	1
15¢ 11	0	1	10¢ 10	
15¢ 11	1	0	5¢ 01	
15¢ 11	1	1	x xx	

6. Boolean for Q_{n+1} and output

open = Q_1Q_0 , K-map

7. Build circuit





Q_{n+1} always depends on Q_n and input, but output diff.

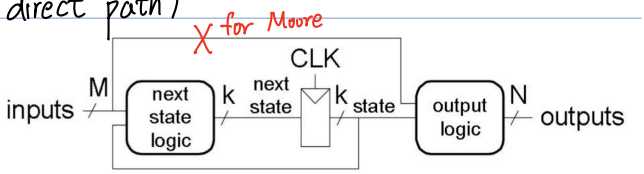
Moore FSM outputs depends only on Q (no I → O direct path)

Mealy FSM outputs can depend on I.

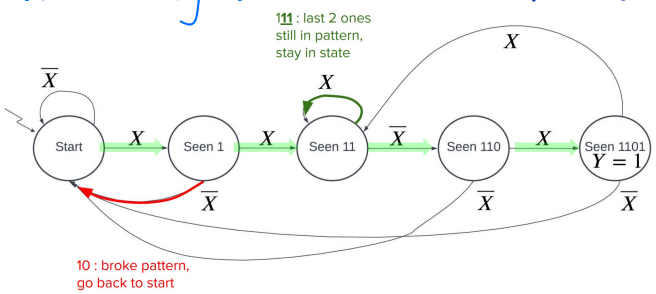
output may change w/o CLK (w/ input)

output on edges (state & input)

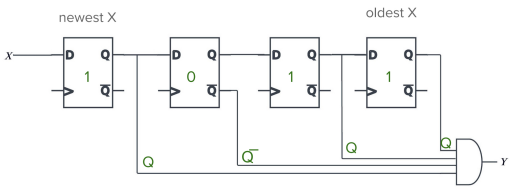
Can use one-hot encoding and TFF



Pattern recognition E. last 4 bits are 1101



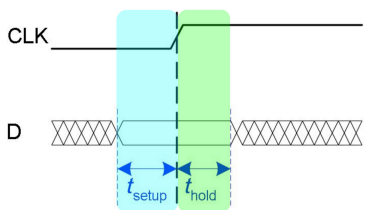
w/ shift register



Timing FF inputs must be stable at CLK ↘

t_{setup} : before ↘ that D must be stable (typ. longer)

t_{hold} after



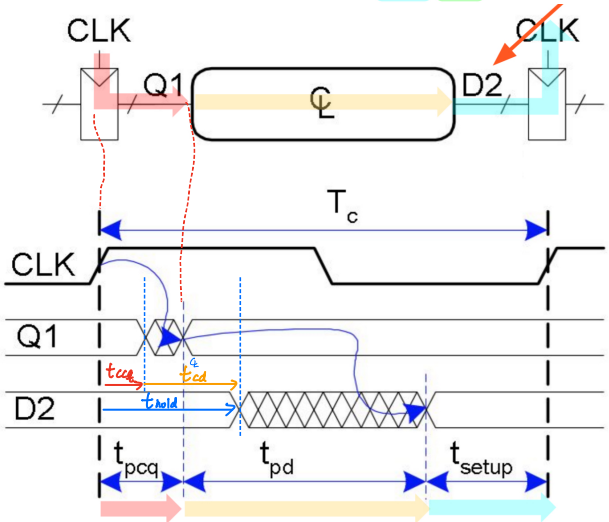
depends on ① Time for Q to stable

② Time for G

min time → unstable max

-c contamination delay -q propagation

$-cq$: CLK → Q, $-d$: Q → D



$t_{hold} < t_{ccq} + t_{cd}$ avoid too fast

$T_c \geq t_{pcq} + t_{pd} + t_{setup}$ slow

CLK skew = diff. between 2 CLK ↘s, consider worst

t_{setup} CLK 2 arrives early

$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$

t_{hold} CLK 1 ~ early

$t_{skew} + t_{hold} < t_{ccq} + t_{cd}$