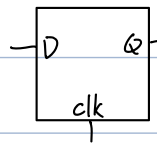
Latch sync.: clock ; async: local handshake

:(waste on clk wait if finish early (dead time)

CSE clocked storage element 1. latch (level sensitive)

2. FF (edge triggered)

3. register (sys. level latch/FF)

Behavioral 

Transparent high latch

if clk = 1, Q = D

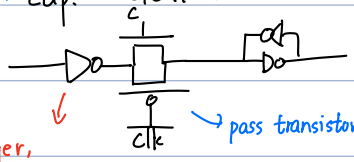
Timing t_{DQ} t_{CQ} (rising)

t_{setup} t_{hold} window before/after falling edge that D must be stable

Build 1. static cross-coupled gates (pos. feedback)

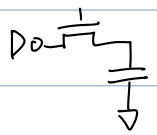
3 op. points (2 stable)

2. dynamic cap. (leak)

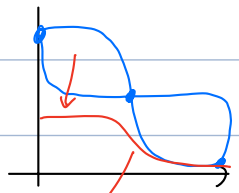
Latch 

pass transistor gated by clk

stronger, need fight stored transistor

static 

dynami

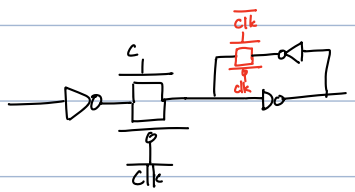


S Voltage transfer compressed ↓ → only one op

if too weak, not writable. Also fight → I_{crow}

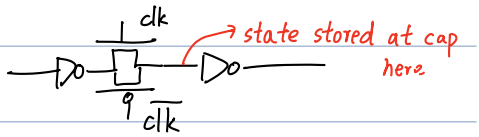
Want to shut feedback while writing. Gate w/ $\overline{clk}$

↳ gated feedback (not in SRAM, since want density)



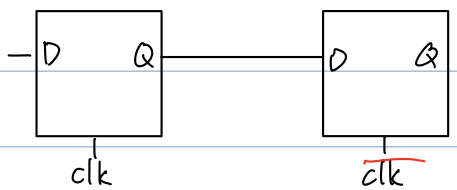
D issue: leak (same w/ dyn. logic)

Not used any more

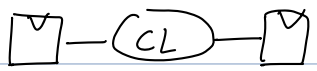


FF (edge)

param L1 L2 t_{cq}, t_{setup}, t_{hold}

master-slave 

@negedge, L1 closes, L2 opens

Late mode (long path) 

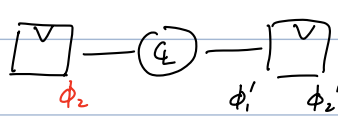
$t_{cq} + t_{setup}$ = latch overhead

(± t_{skew} +) $t_{cq} + t_d + t_{setup} \leq T_{cycle}$

Early mode (short path) $t_{cq} + t_d \geq t_{hold} (+ t_{skew})$ (E. shift register)

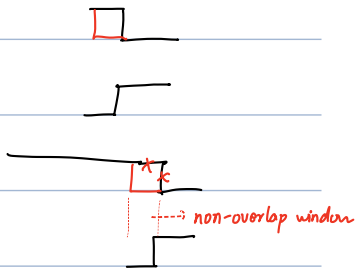
Short path padding w/ ▷o—

Need indep. control of clk phase in ϕ / ϕ'

sln. 

$t_{setup, hold}$: master

t_{cq} : slave

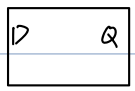


rising edge of slave → captured by falling ~ of master

non-overlap window for all → master receive earlier

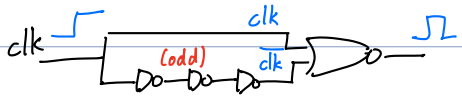
con: dead time at window → less time for long path → f_{max} ↓

can be tuned in modern processor



Pulse mode clking (D-latch ⇝ ff)

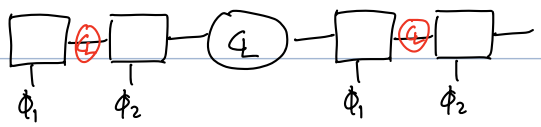
worse early m. issue (+ t_{pulse}) → two-sided constraints

clk chopper (one shot) 

pulse @ negedge

if global chop → distribute → running.

if local $\overline{clk}$ chop → more gates :(

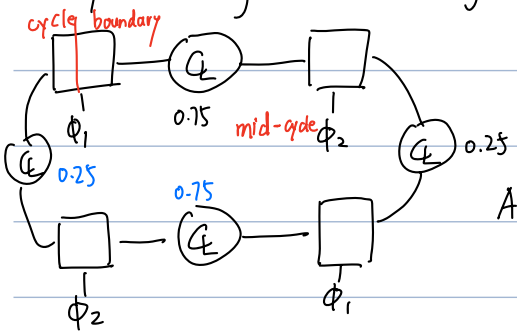
(separated)
Transparent latch design 

Make logic between M-S

Long path 1. D must arrive before closing edge of any latch (assume $t_{setup} = 0$) } global constraints

2. Loop latency ≤ # loop cycles

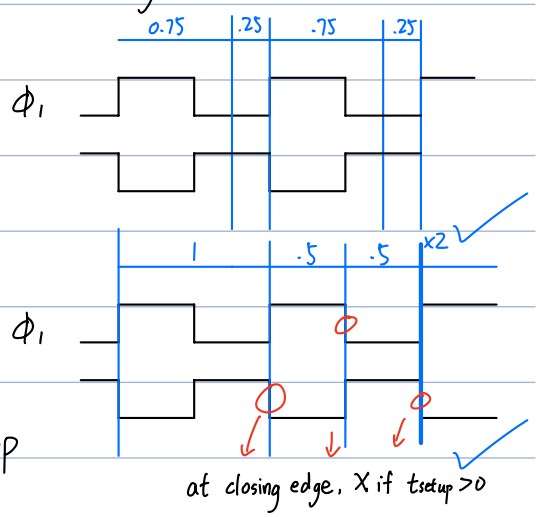
Cycle stealing: if one ℓ group too big, OK



Arbitrarily choose a starting point

Now 1, 0.5, 0.5, 0 (skewed)

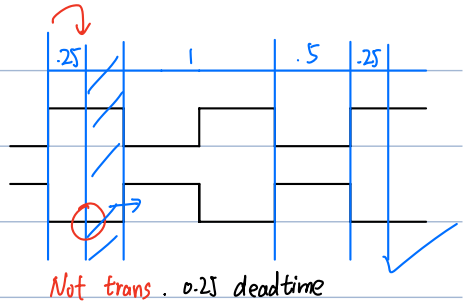
barely works, no deadtime, can slop



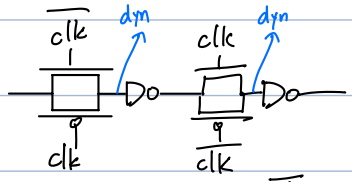
Now 0.25, 1, 0.5, 0.25 2.25 cycles >2, loop violation?

Data does not have to launch at CLK ↑. Can push 1st 0.25 back so cycle is still 2.25 → false violation ✓

Timing violation may not be local



Dyn. latch / FF (rare)

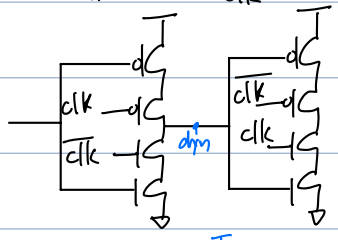


(master-slave) ↓↑ fast!

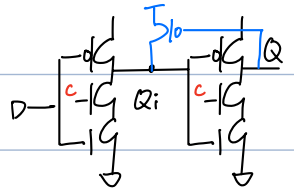
if clk, $\overline{clk}$ both 1, race, may flush

NORA (no race, C^2mos)

break connection w/ pt



TSPC (true single phase clking)



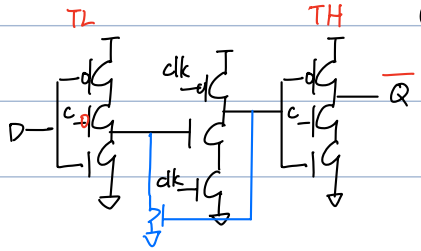
if c=1, $Q = \overline{Q_i} = D$

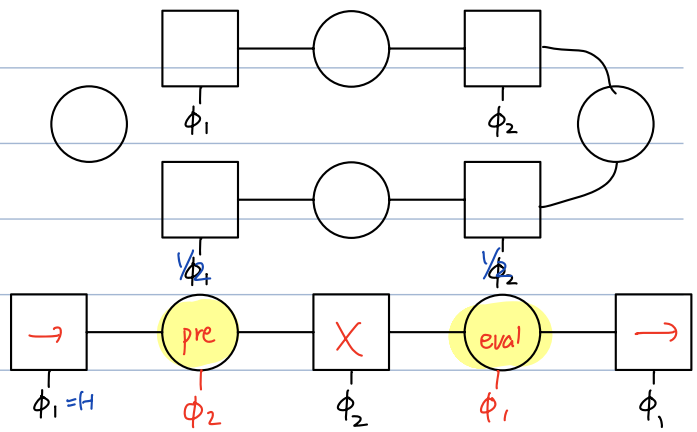
if c=0, opaque (dynamic if D or Q_i = 1)

either Q / Q_i dyn., but not both

FF (↓↑)

inverting





Dyn. flaw waste half of the cycle on pre

pipeline 2-phase, hide pre

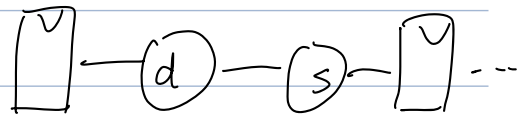
when ϕ_1 H, ϕ_2 pre, closed

Partition issue. Need = :(

If eliminate ϕ_2 latch. | ϕ_2 prevents next to be corrupted by pre. Need value before pre.

Race ϕ_2 eval vs ϕ_1 pre, so mostly ϕ_2 latch still kept

Sta. mix dyn.



1. Static at end of any cycle, before pp

X: power. If FF outputs same Q → static ★; if diff Q → dyn. ★ (only switch once, hazard-free)

static after dyn. is worst ——— (flutter when driven by dyn.)

Microprocessor (E. GPU, ...)

1. Data path : logic blocks handing / op. data (E. adder, shifter, register)
2. Control (random logic, not structured) tend buggy :(
3. Arrays : memory (very structured) (E. RF, SRAM, embedded DRAM)

Num ★ fixed point (int), signed (2's c), unsigned

fp (mantissa + exp), fpu

2's c overflow = $c_n \oplus c_{n-1}$

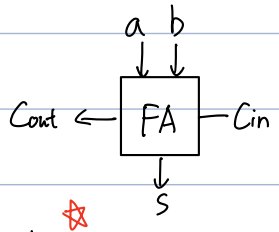
endian in virtuoso, make bus $a\langle 0:31\rangle$ (MSB ↑ LSB ↑)

lil $a\langle 31:0\rangle$

big $a\langle 0:31\rangle$

Adder (FA) $s = a \oplus b \oplus c_{in}$

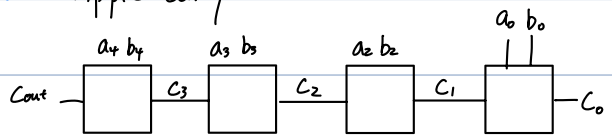
$c_{out} = ab + ac_{in} + bc_{in}$ (majority)



self dual $\bar{s}(a, b, c_{in}) = s(\bar{a}, \bar{b}, \bar{c_{in}})$ (pdn, pun) } if building push-pull, identical pun, pdn

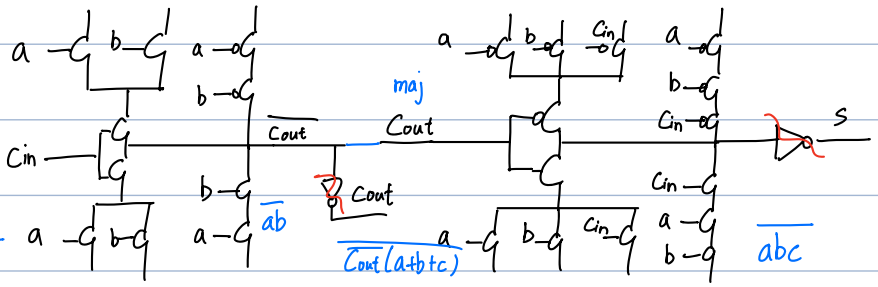
$\bar{c}$ ~ = c(~, —) } great for layout

E. 8-bit ripple carry ~

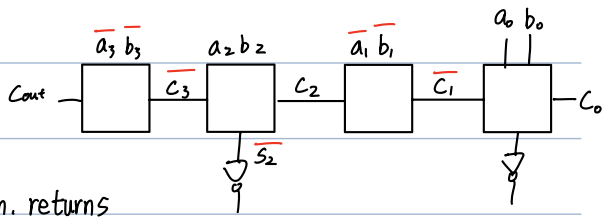


for ripple, s_3 slowest

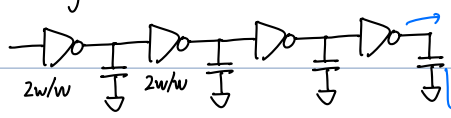
layout : just FA, and tile it well



Can make generate $\overline{Cout}$ and $\overline{sum}$, alternate inverting inputs



Sizing



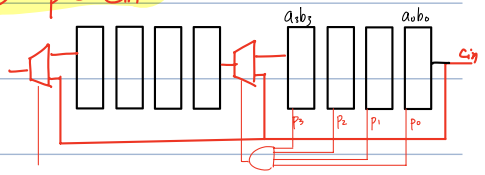
want $w \to \infty$, dim. returns

S not in crit. path, make S↓ → size C chain (same size)

sub invert B, $c_{in} = 1$

Parallel $C_{out} = ab + aC_{in} + bC_{in}$

$= g + pC_{in}$ $\quad g \equiv ab,\ p \equiv a+b = a \oplus b \rightarrow S = p \oplus C_{in}$



1. Carry bypass E. break 16-bit into E. 4x4

MUX, sel bit ANDs all $p_i = a_i \oplus b_i$

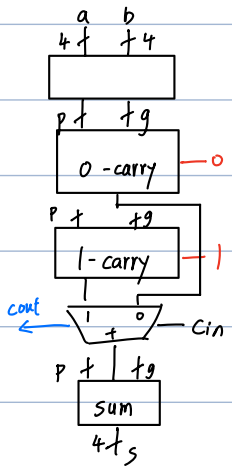
E. n=16 $t_{crit} \approx 3t_{max} + 2.4\, t_{ripple}$

2. Carry select E. 4x4

Calculate all carries if C_{in} is 0; if C_{in} = 1

use MUX to select

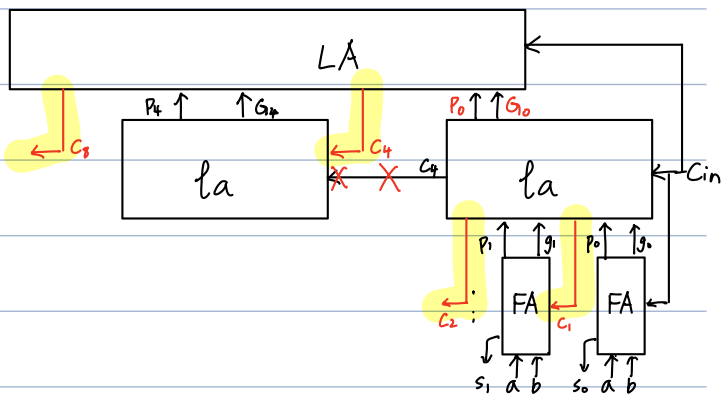
$t_{crit} = t_{pg} + t_{carry0} + t_{carry1} + 4t_{mux} + t_{sum}$



(x4)

3. CLA $C_{i+1} = g_i + p_i C_i$

$= \overbrace{g_i + p_i g_{i-1} + p_i \cdots p_1 g_0}^{G} + \overbrace{p_i \cdots p_0 C_0}^{P}$



E. 4 : radix-4 lookahead

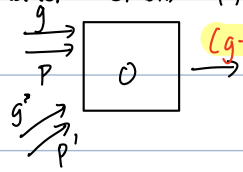
E. 16 can take 4x radix 4 & ripple :(

Hierarchial $P_0 \equiv p_3 p_2 p_1 p_0 C_{in}$

$G_0 \equiv g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$

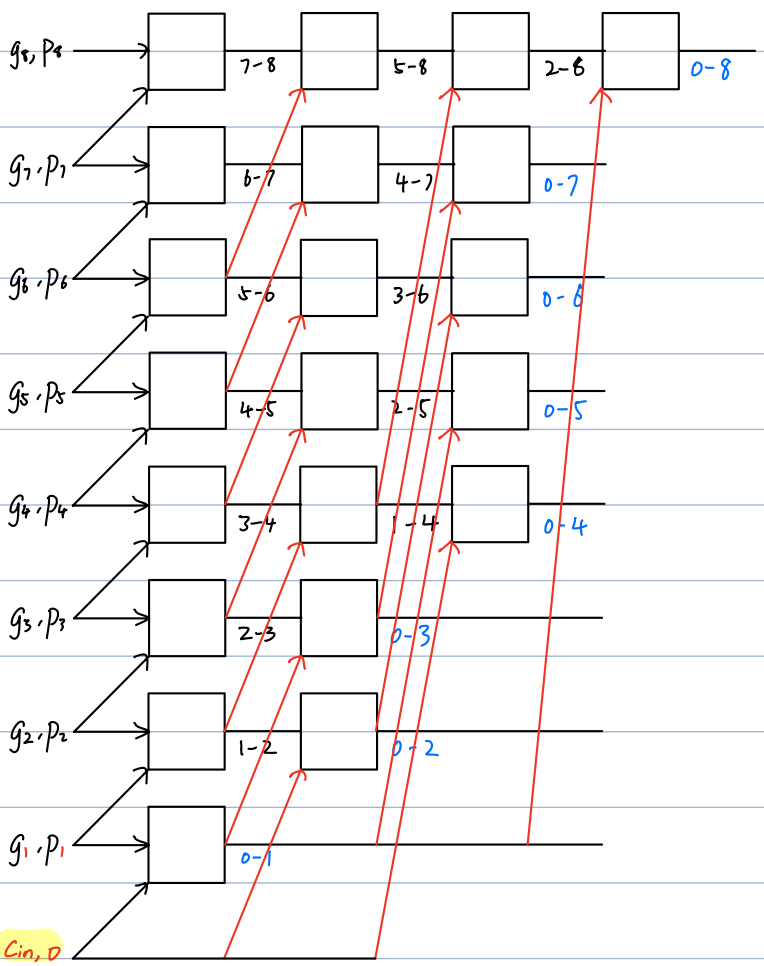
tree adders (radix-4). Radix 2 is simpler.

Radix-2



KS 0-1-2-4

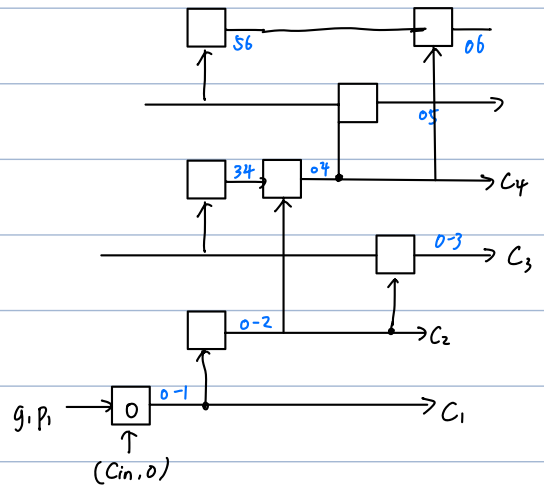
fo of 1 or 2!



78 58 08

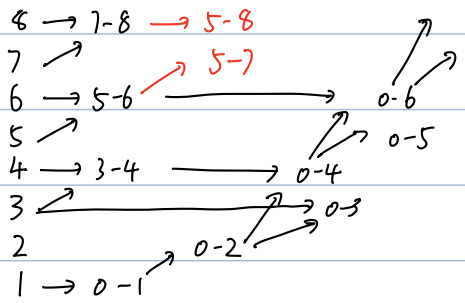
57 07

BK



area efficient, but not same fanout

harder to layout



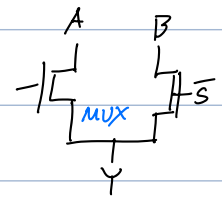
even bits grabbed

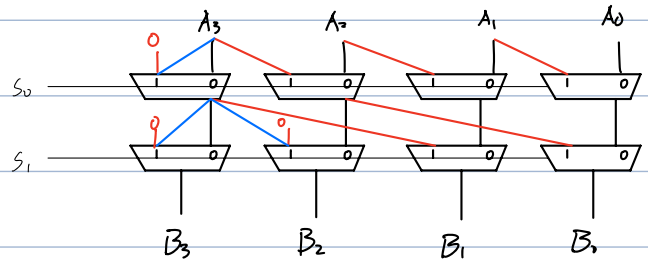
Shifter 1. logical always pad 0 (shamt)

2. arithmetic pad MSB (sext) when shift right

logarithmic shifter

s = shamt





if s_2, need shift by 4.

if arithmetic, shift in msb

stack = 2+1 (input inverter) :(

can buf w/ D0

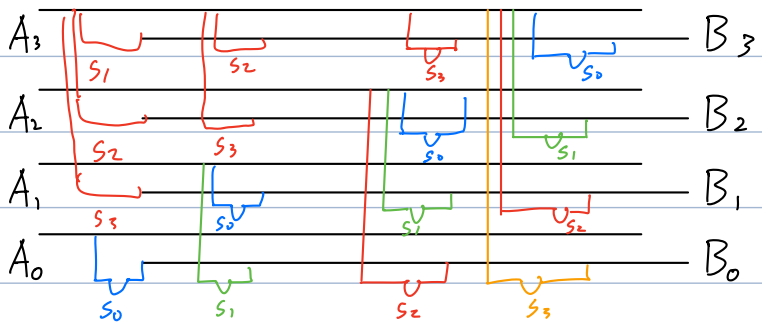
Barrel shifter (pt) [4-bit]

shamt: 1-hot $s_{3:0}$

arith >>

2 stack height ☆

need ⊥

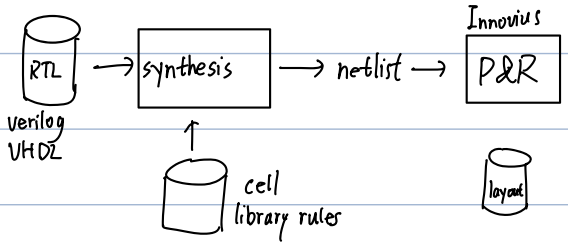


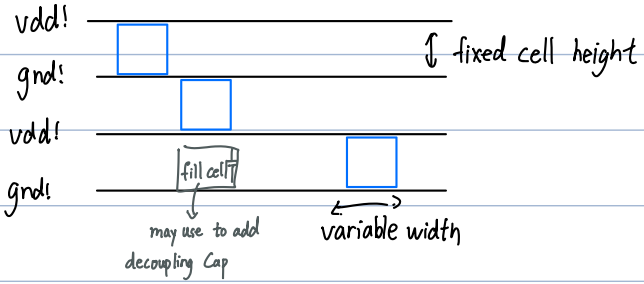
Rotator out → in

Control

1. random logic → multi-level → semi-custom
2. structured logic (PLA) programmable logic array, area ineff.
1. RTL (register transfer level) (structured, careful) → logic synthesis tool → layout (place route) → P&R

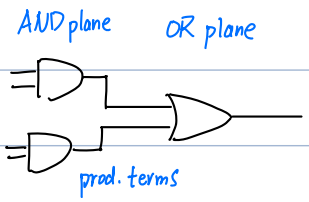
cells (lib) → static timing analysis





P&R uses feedback to optimize

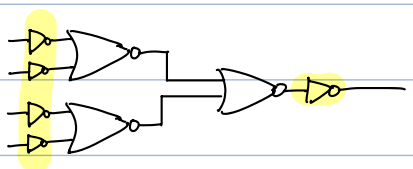
PLA (programmable logic array) structured (E. instr. decode)



Two-level logic : SoP, PoS

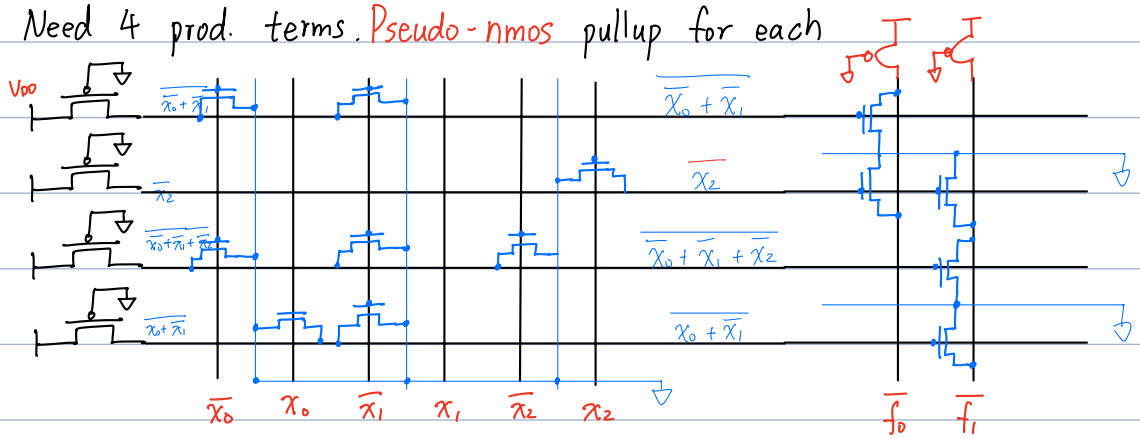
Espresso : minimize SoP terms (heuristic)

bubble pushing AND-OR ⇒ NOR-NOR



Static PLA E. $f_0 = x_0x_1 + \bar{x}_2$, $f_1 = x_0x_1x_2 + \bar{x}_2 + \bar{x}_0x_1$

Need 4 prod. terms. Pseudo-nmos pullup for each



Espresso format

(noninverted) AND plane: x_0 x_1 x_2 | OR plane: f_0 f_1 → Espresso

1 1 -	1 0
- - 0	1 1
1 1 1	0 1
0 1 -	0 1

0: use $\bar{x}$

1: use x

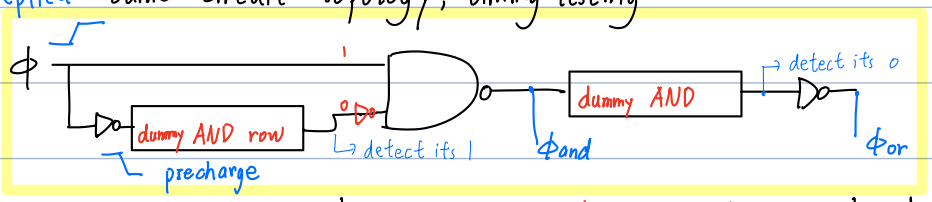
-: d.c.

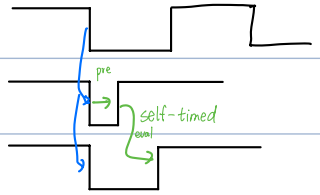
Dynamic PLA dyn. nor — dyn. nor (turn pseudo-n into clked)

but no Do available for domino → clk w/ ϕ_{AND}, ϕ_{OR}. Don't start ϕ_{OR} before AND evaled.

self-timed → need delay elements

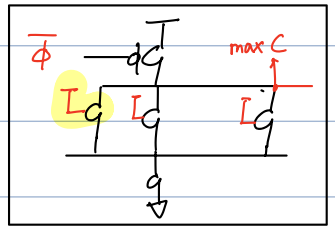
Replica : same circuit topology, timing testing





Layout dummy and row exact same. Load w/ slowest precharge time

(large C on precharge node E. 3 nfets, all inputs high)



slow eval put 1 input high, rest 0

wait for output to be evaled to 0 → ϕ_{or} ↓

Testing

$I_{DD,Q}$ I_{DD} at quietsant (some leak, but can't be huge)

Shut any static current (E. analog, half latch)

Memory (array) S/DRAM (volatile)

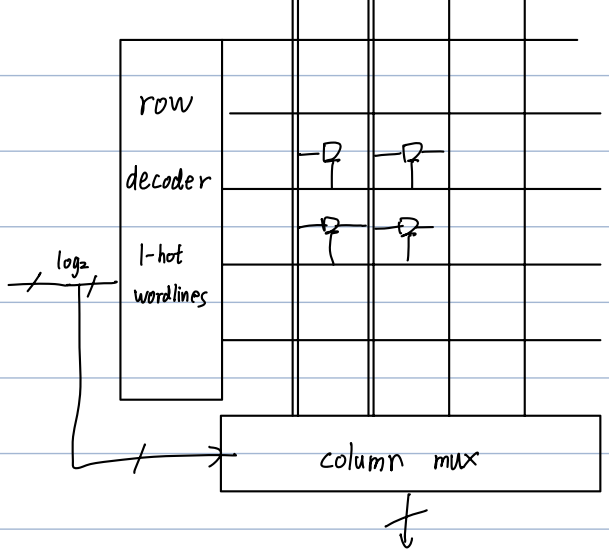
D trench into Si (deep reactive ion etching) → Cap

S row decoder : word lines (demux 1-hot)

col mux : bitlines

Logical organization E. 1024 × 64
10b 8b 8b

Physical E. 256 × 256 ↕ take 2b addr. to col. mux, 8b to row dec.



Cell (6T)

R 1. precharge bit, $\overline{bit}$ (row dec)

2. select 1 word line (access on for word)

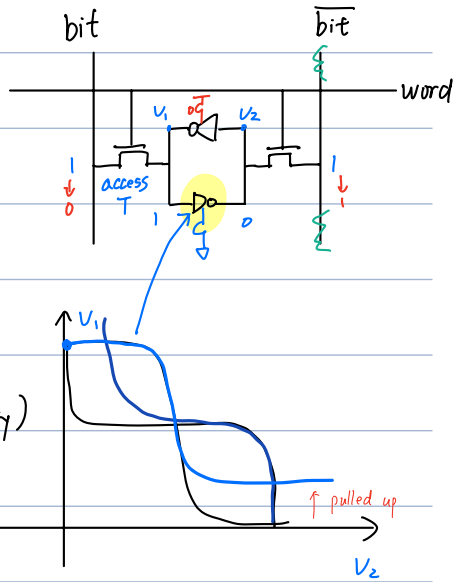
3. state pulls low bit or $\overline{bit}$

4. detect (don't want to wait all the way → 0, slow, power)

issue R may upset cell (read stability)

at precharge, both sides V_{DD}, Do can't go all → 0. (wing stability)

Need access transistor to 1.5~2x smaller than Don.



W 1. precharge

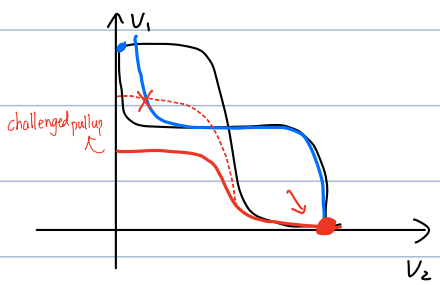
2. sel. wordline

3. Pull bit/$\overline{bit}$ down → flip SRAM

issue W writability (want diagram to collapse)

access T 3x stronger than Dop

Sizing by changing L and W. → dense

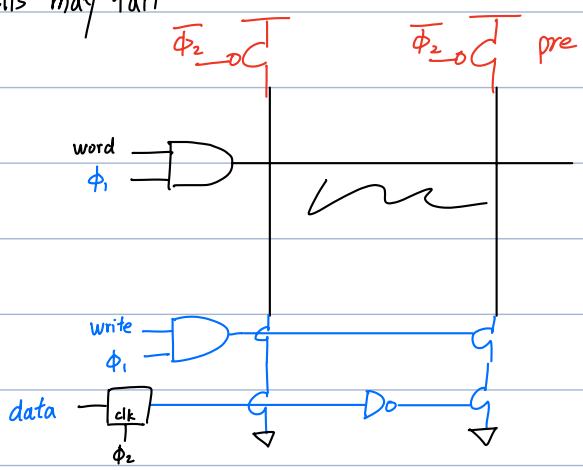


Layout w/ push rules (denser DRC)

small → dopant variation → V_T ↑ → strength ~→ → cells may fail

sln. 1. when not accessing, run at < V_{DD}. Only V_{DD} @ reading

2. redundancy (E columns), take out if bad



Peripheral self timed ; or ϕ_1, ϕ_2

R 1. pre @ $\phi_2 = 1$

2. access @ $\phi_1 = 1$ (clk qualification)

detect : skewed D$\circ$ (single-ended)

W some pre, access clk

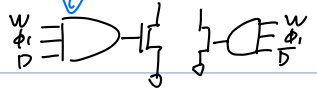
data needs to be ϕ_1-stable (E. from ϕ_2 latch)

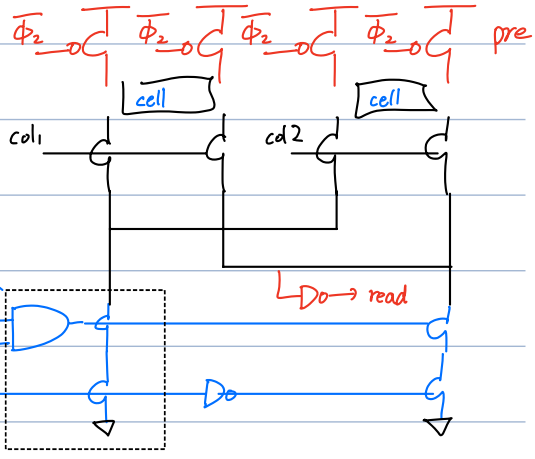
3-high stack

+Mux column mux gives more space for W.

4-high stack

sln. combine 2-stack AND logic

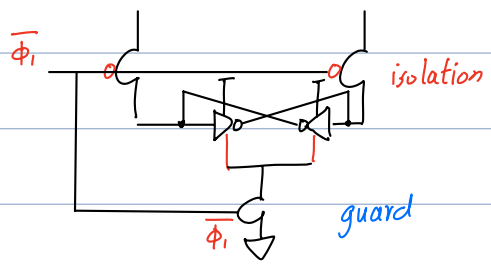




SA (differential read)

limited by GBW → regenerative circuits

need know when to amplify → clk



access iso on, guard off (D$\circ$ off), one low

pre : iso off, guard on, D$\circ$ amplify w/ regenerative feedback

issue if inv. not exact same, DC offset → flip to wrong state

Inverters w/ regenerative feedback not limited by GBW issue → fast, but need clk.

Decoder (glorified AND) E. 3→8 : 8 ANDs

issue: input ↑, high → multilevel

Final output match SRAM roles → special DR for dec.

Typically dynamic nor:

race-based nor

Make ϕ_1 delay ϕ_2 to avoid race cond.

D$\circ$ pfet critical → need wide fingers