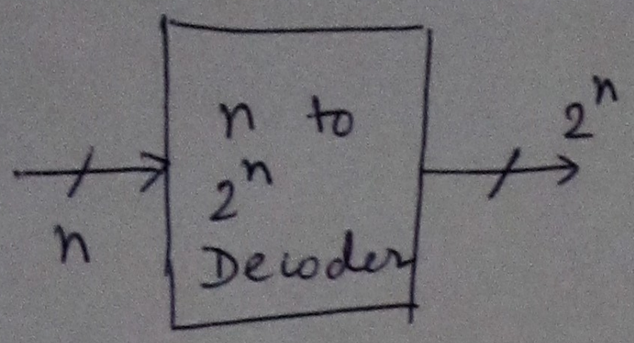
Slide-3



$Y_0 = \bar{a}\,\bar{b}\,\bar{c}$

$Y_1 = \bar{a}\,\bar{b}\,c$

$Y_2 = \bar{a}\,b\,\bar{c}$

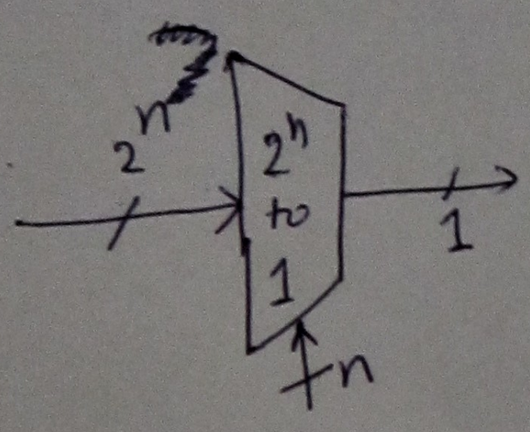
$\vdots$

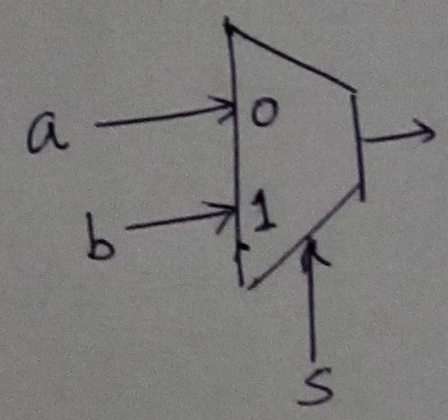
$Y_6 = ab\bar{c}$

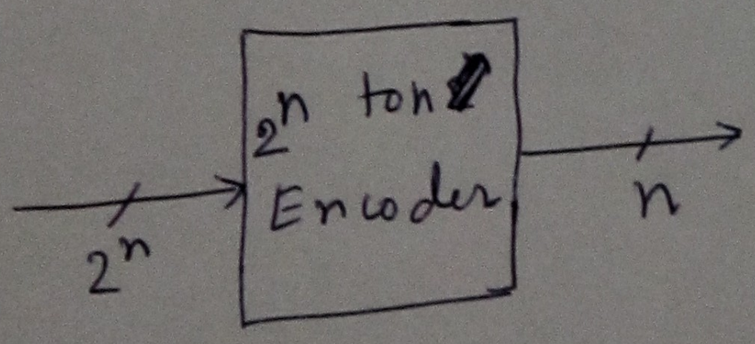
$Y_7 = abc$

3 to 8 decoder

a	b	c	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1





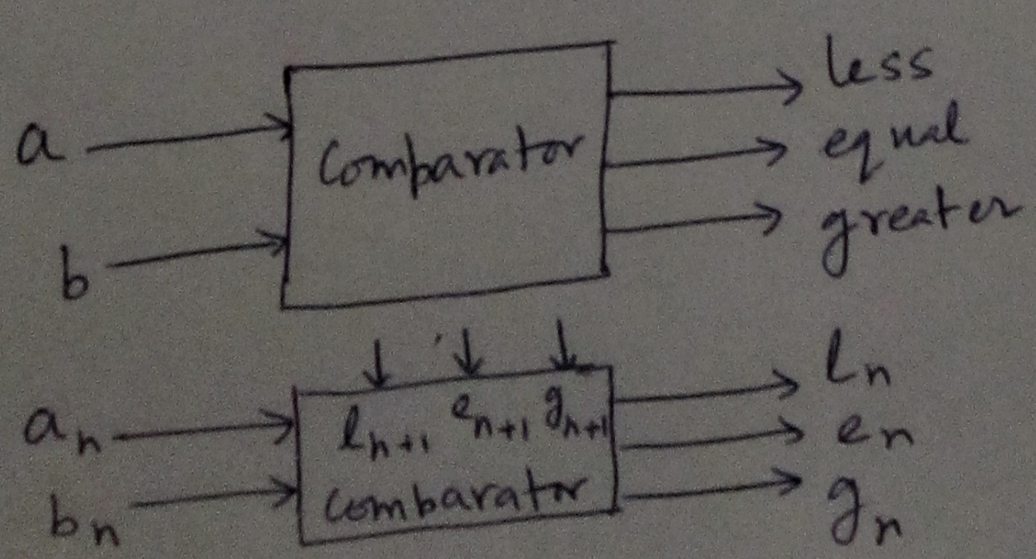


8 to 3 encoder

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
				⋮					⋮	
0	0	0	0	0	0	0	1	1	1	1

What if all inputs are zero?

What if multiple inputs are 1?

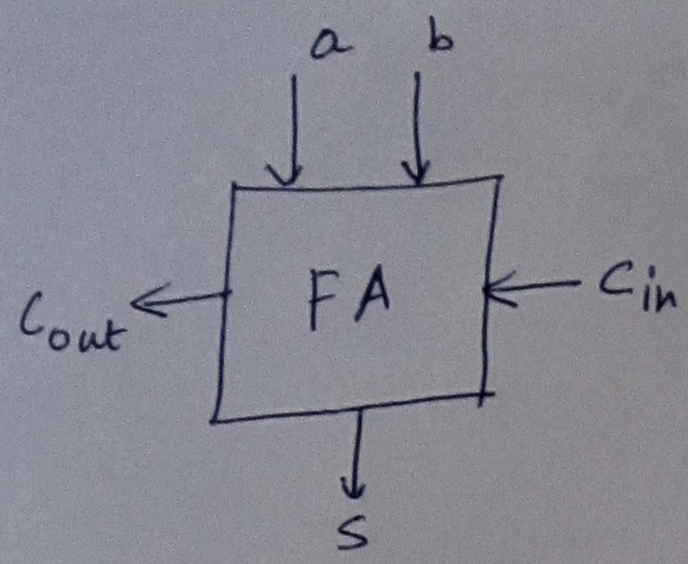


$l = \bar{a}\,b$

$e = \bar{a}\,\bar{b} + ab$

$g = a\,\bar{b}$

Slide-4



a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

SoP

$$S = \bar{a}\,\bar{b}\,C_{in} + \bar{a}\,b\,\bar{C}_{in} + a\,\bar{b}\,\bar{C}_{in} + a\,b\,C_{in}$$

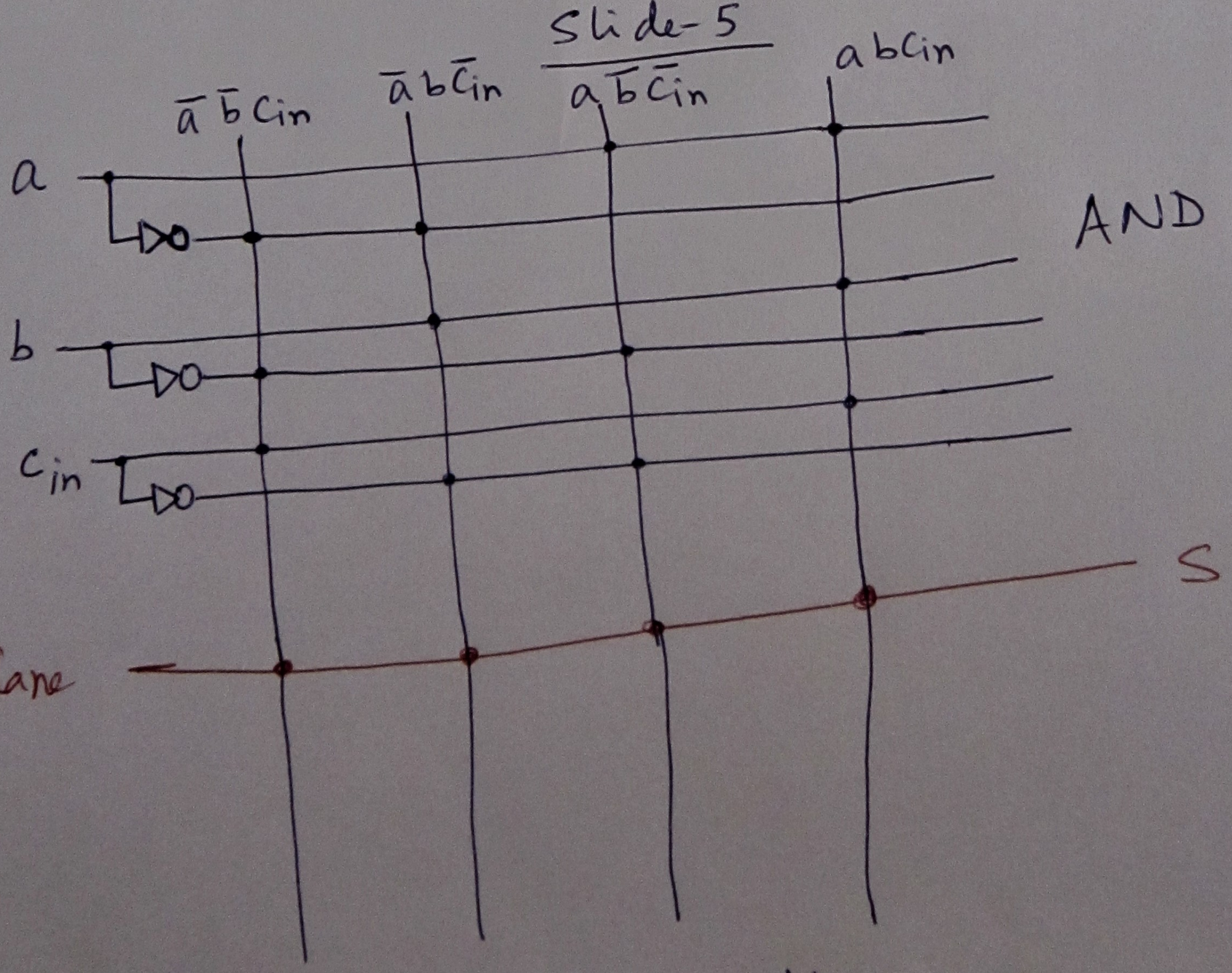
$$C_{out} = \bar{a}\,b\,C_{in} + a\,\bar{b}\,C_{in} + a\,b\,\bar{C}_{in} + a\,b\,C_{in}$$

PoS

$$S = (a+b+C_{in})(a+\bar{b}+\bar{C}_{in})(\bar{a}+b+\bar{C}_{in})(\bar{a}+\bar{b}+C_{in})$$

$$C_{out} = (a+b+C_{in})(a+b+\bar{C}_{in})(a+\bar{b}+C_{in})(\bar{a}+b+C_{in})$$

Slide-5



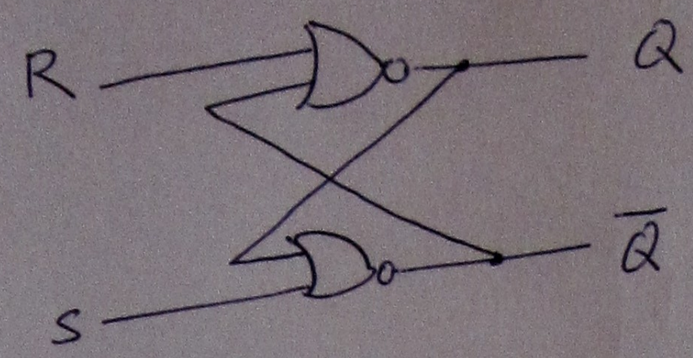
PLA for generating sum

Slide-5

ROM for full adder

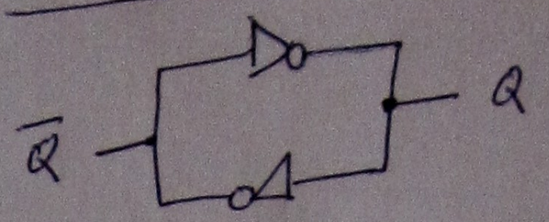
S	Cout
0	0
1	0
1	0
0	1
1	0
0	1
0	1
1	1

Slide-10

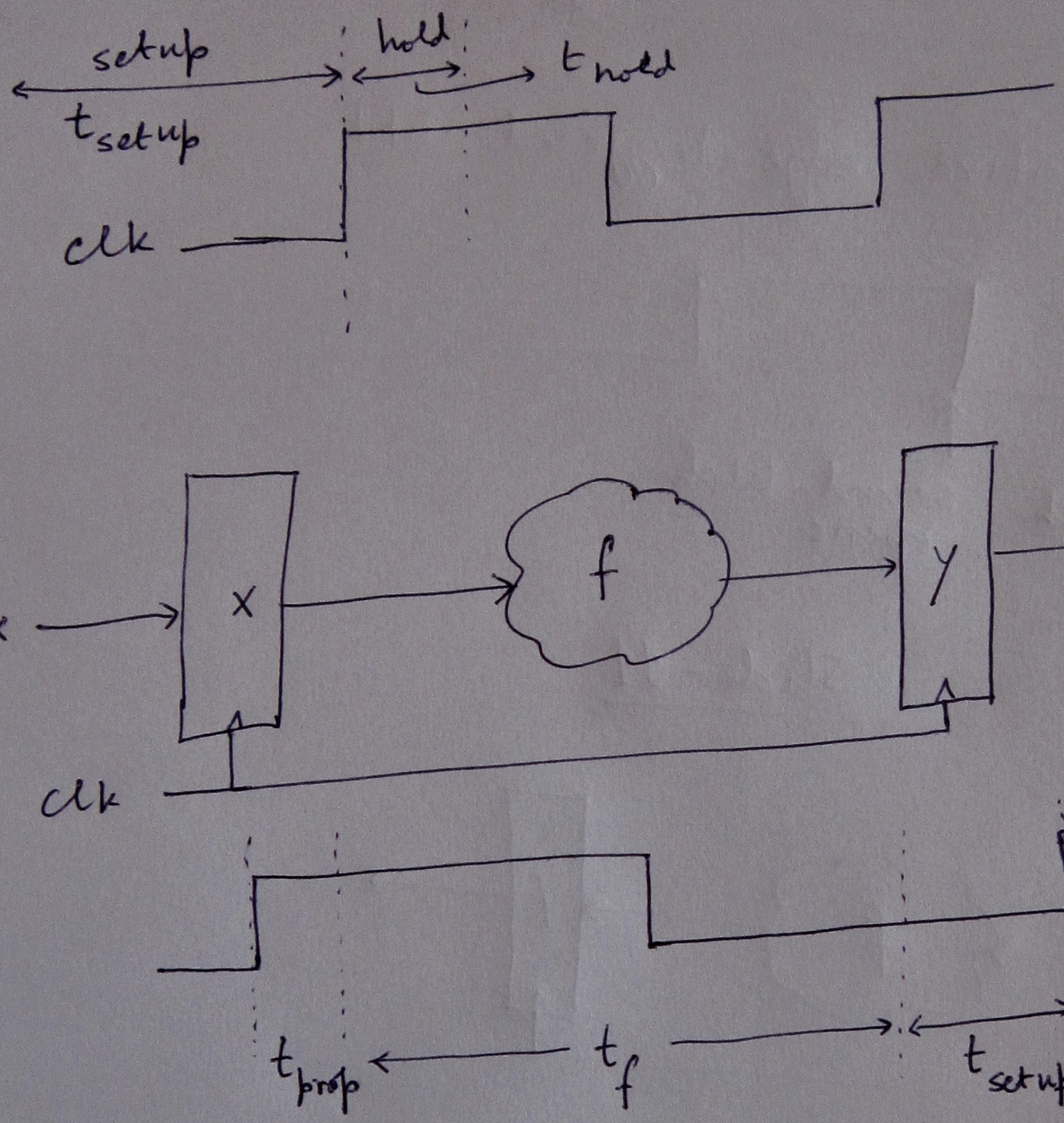


S-R Latch

When S=R=0



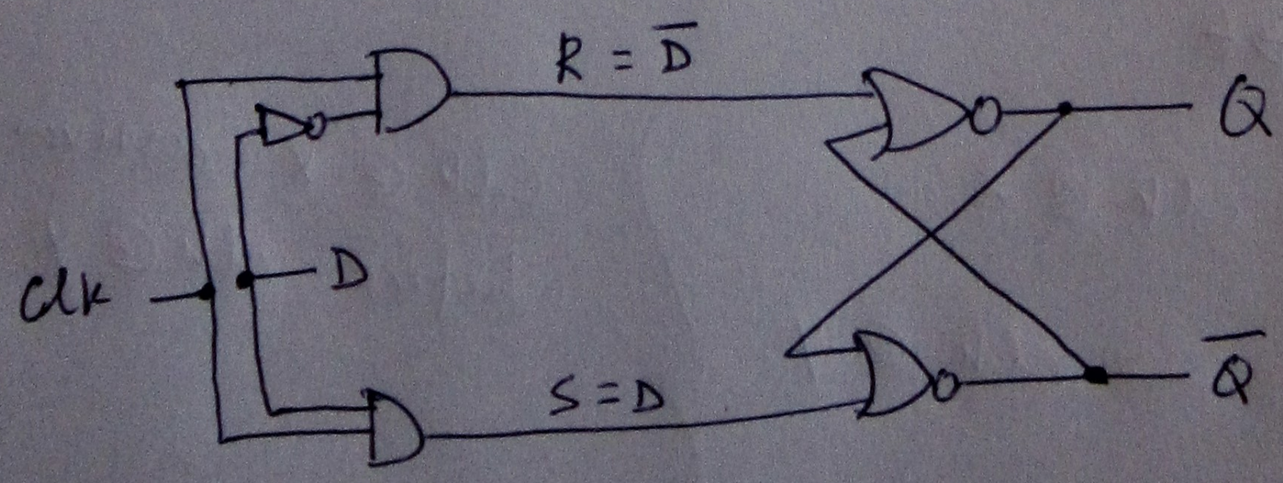
Cross-coupled inverter
Simplest storage element



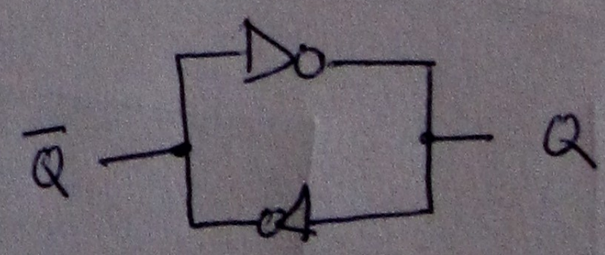
$$\tau \geq t_{prop} + t_f + t_{setup} + t_{skew}$$

$$t_f = \max_i L_f^i$$

D-latch

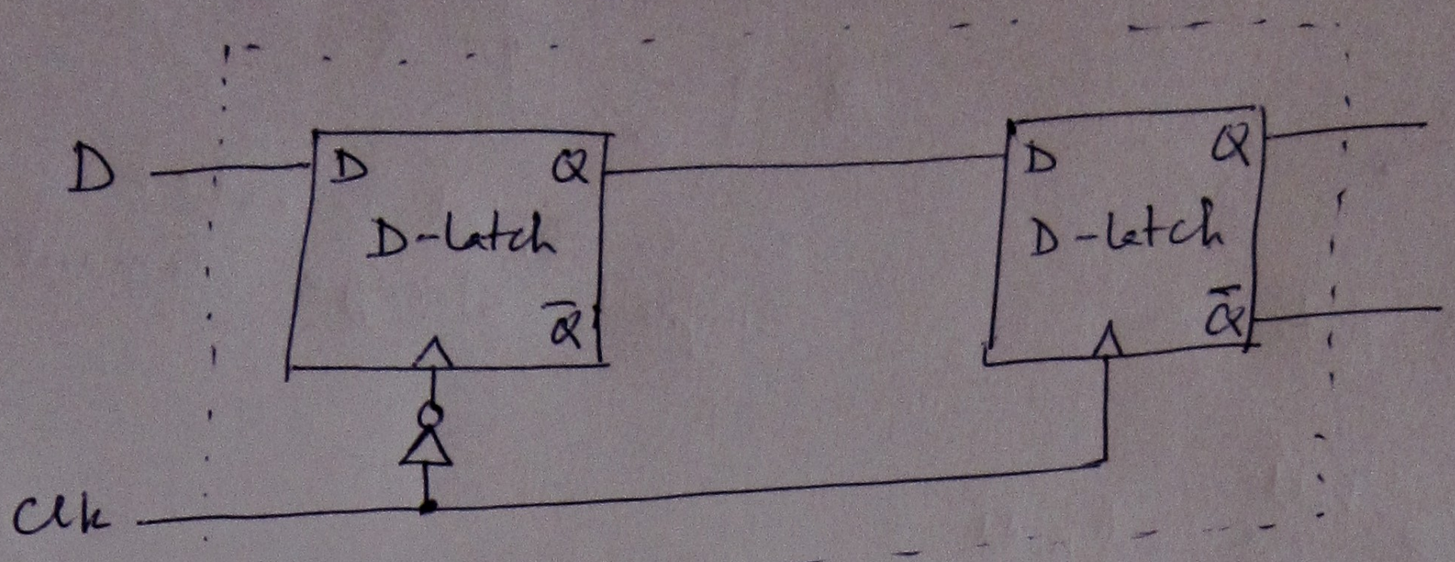


clk = 0

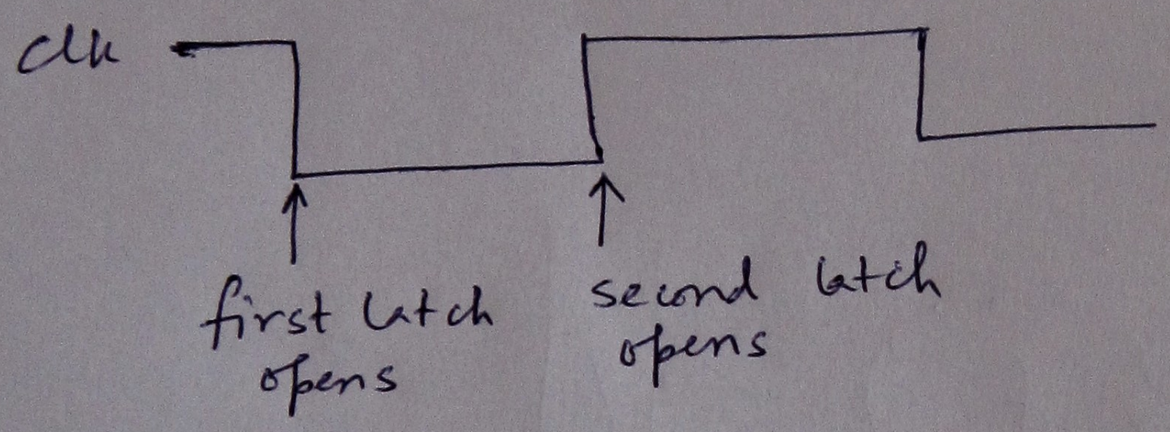


Slide-10

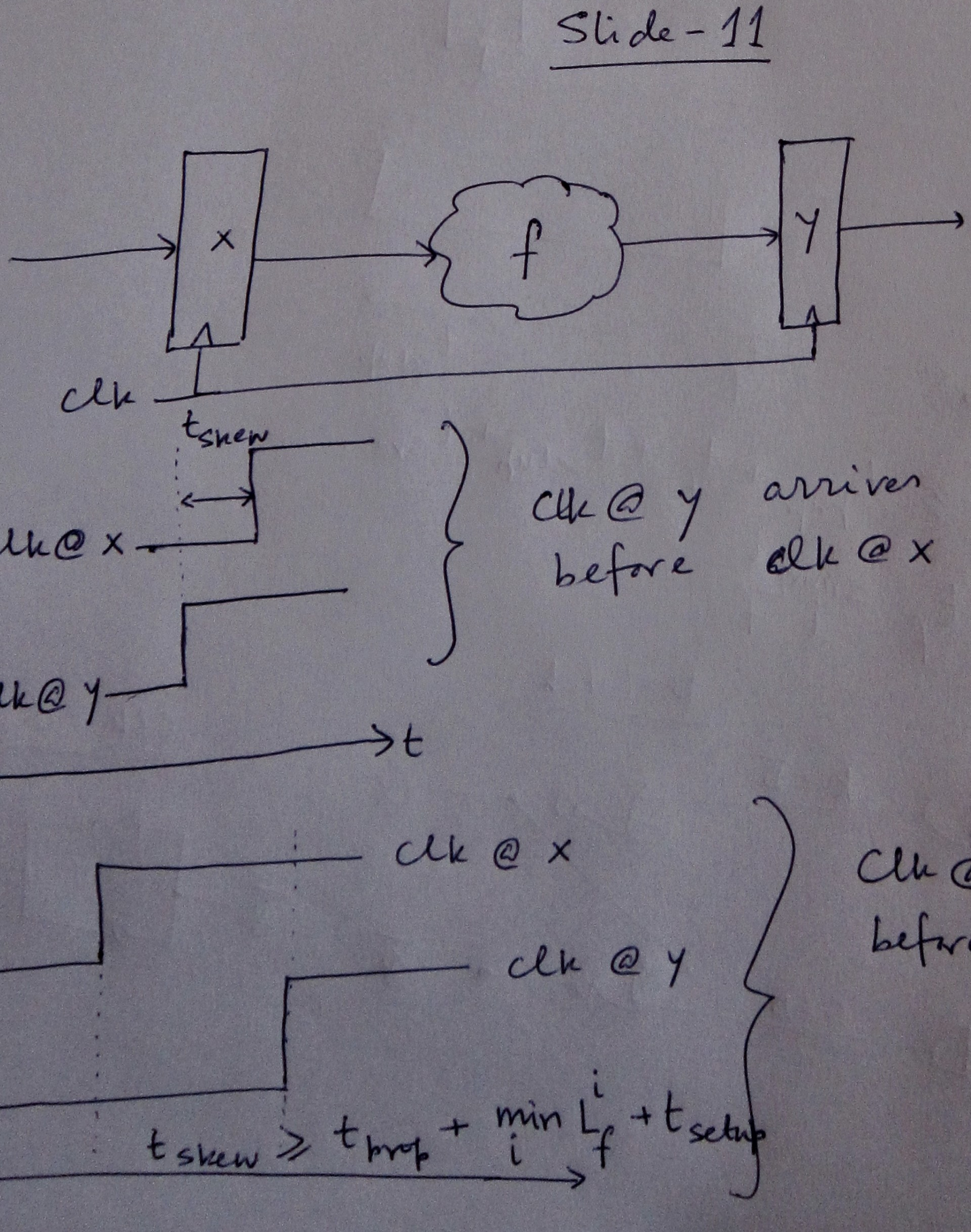
D-latch to D-flipflop



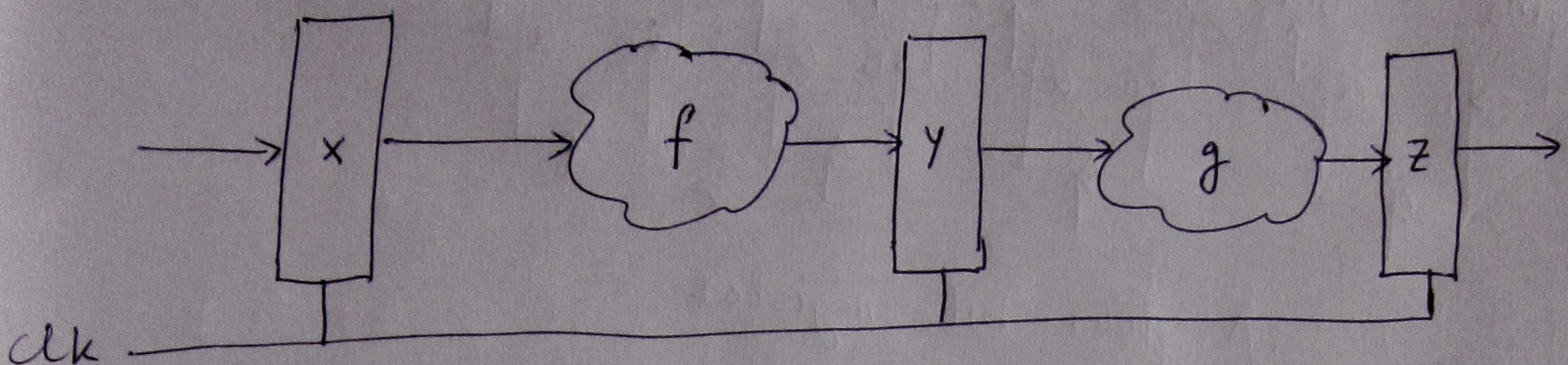
Positive edge-triggered DFF



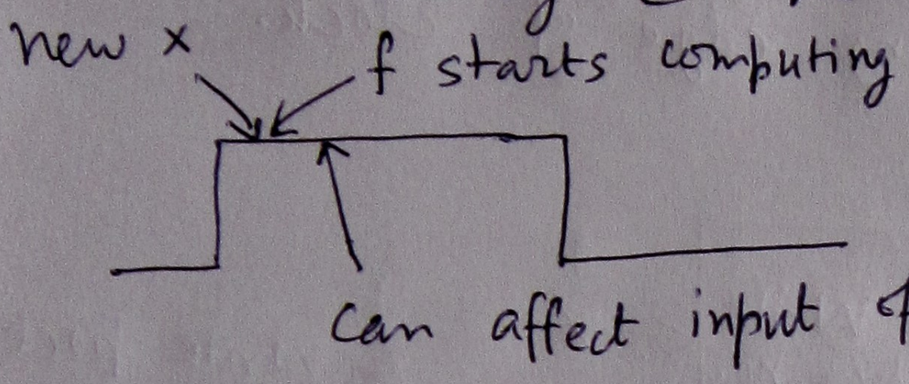
Slide-11



Slide-13

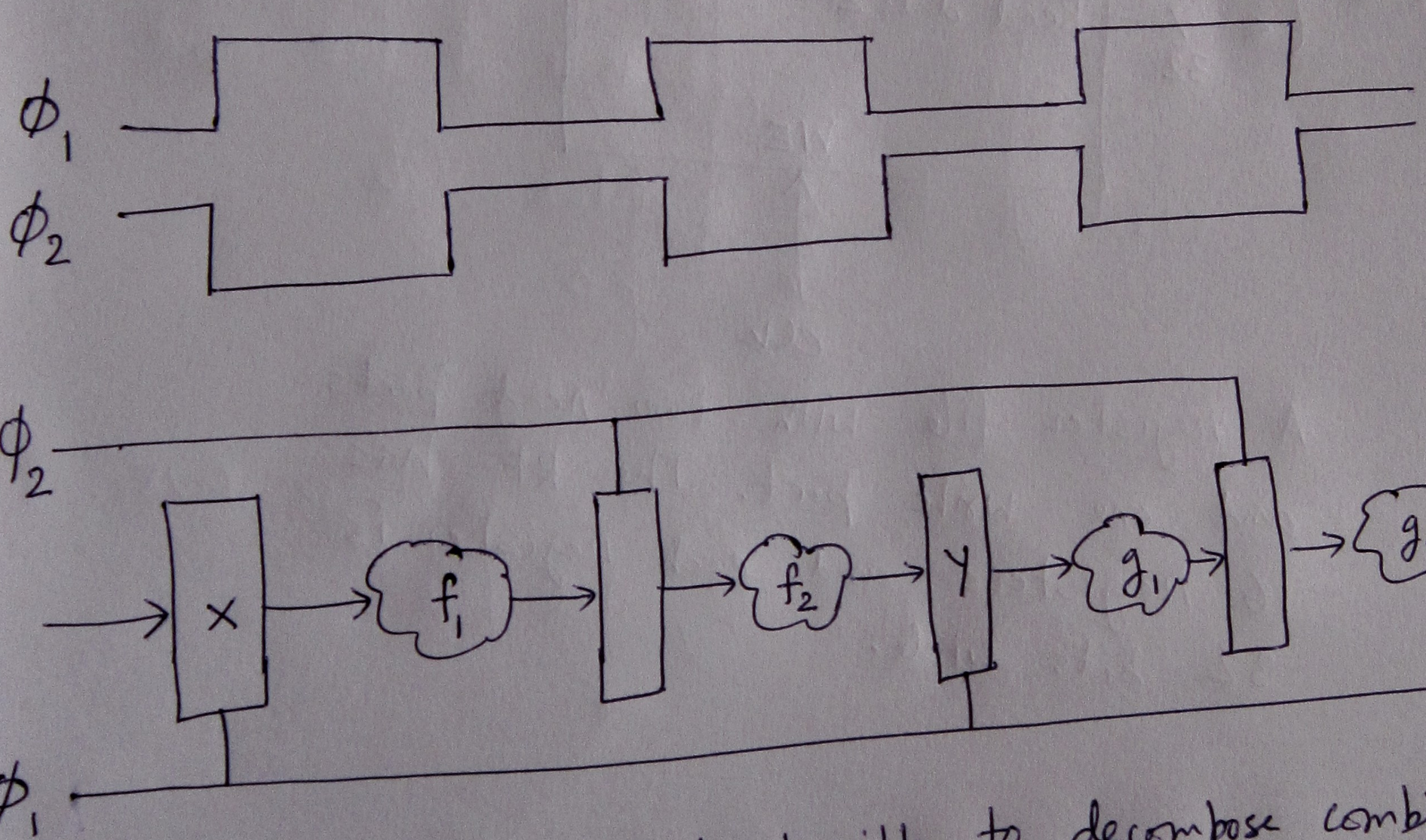


always @ clk ≡ Level-triggered latch



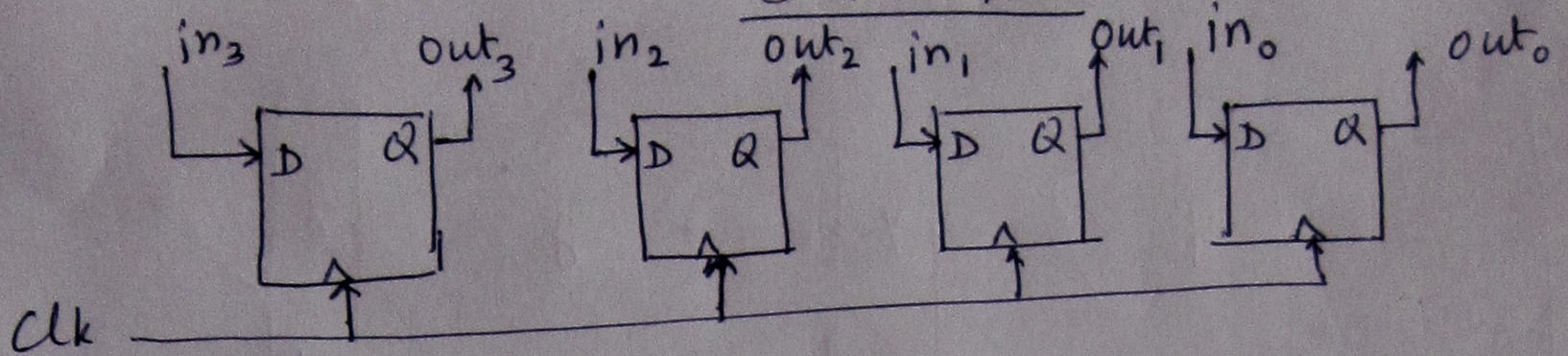
Decompose f and g

$f \equiv f_2 \circ f_1$, $g \equiv g_2 \circ g_1$



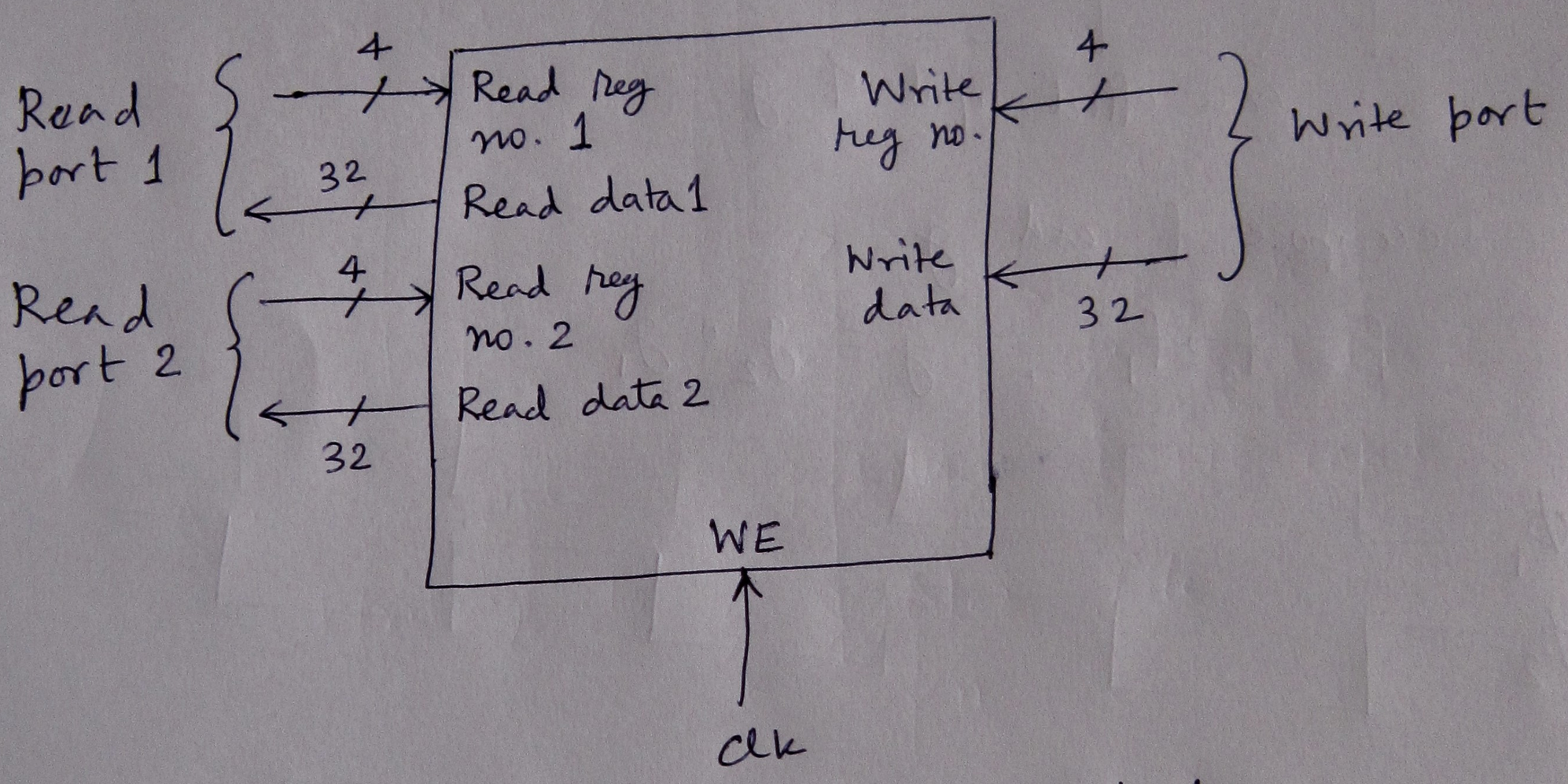
→ It may not always be possible to decompose combinational computation like this.

→ Needs to maintain two clock signals that are exactly out of phase.



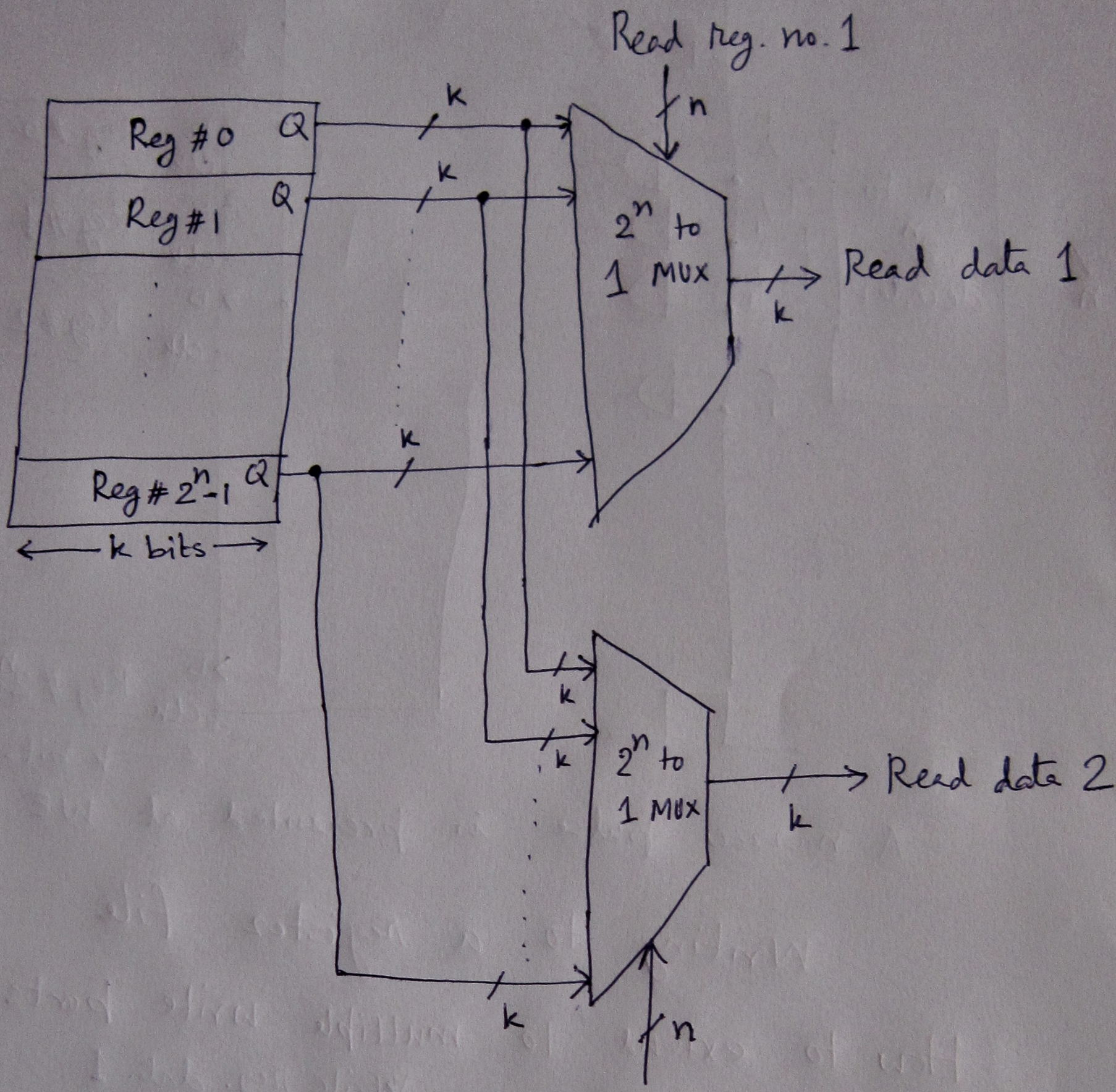
A four-bit register

→ Can read contents at any time
→ Can write only on rising edge of clock.



A register file with two read ports and one write port. The RF has 16 registers and each register is 32 bits wide.

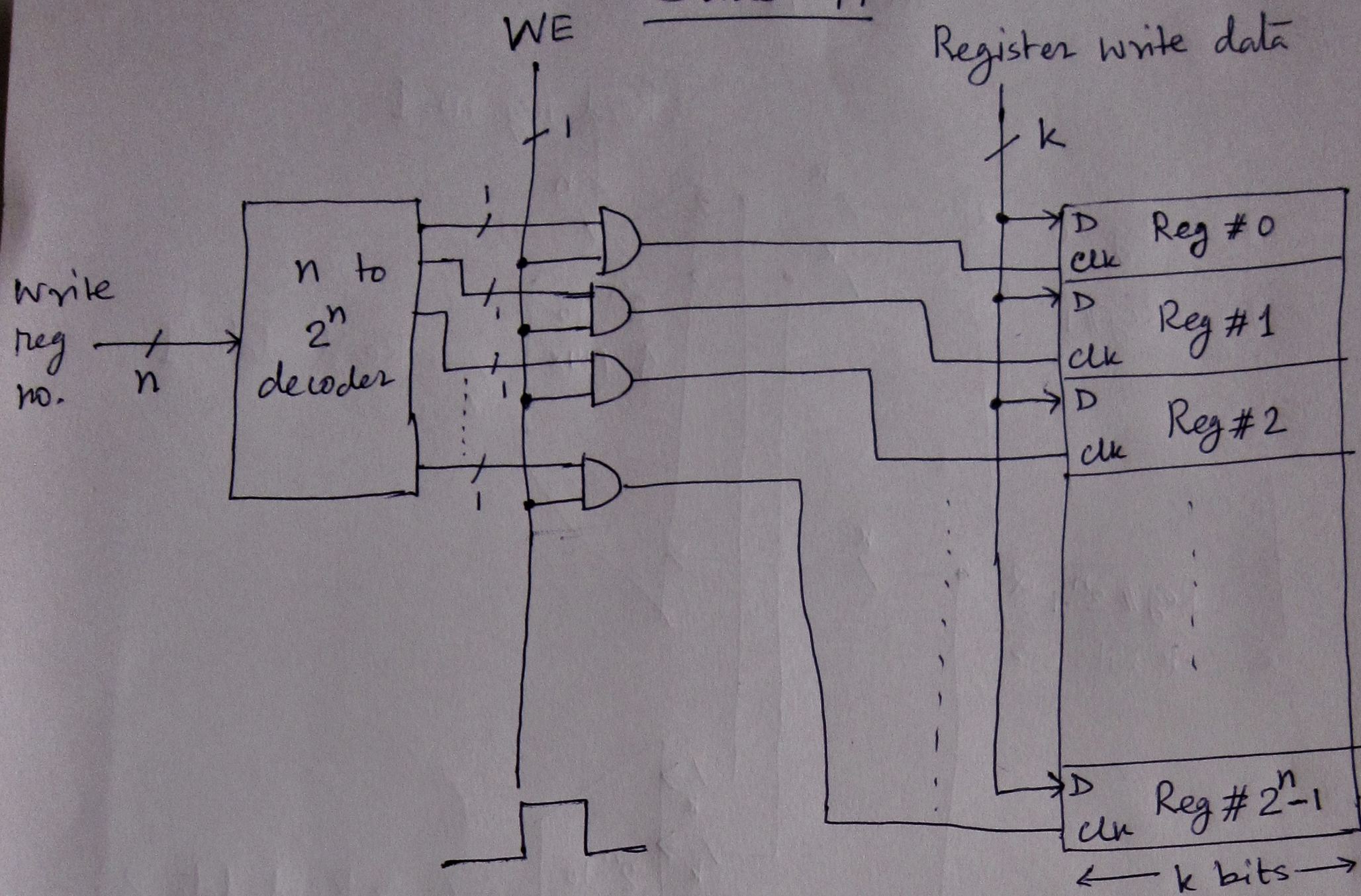
Slide-14



Read latency can be high due to large MUXes.

Reading from register file

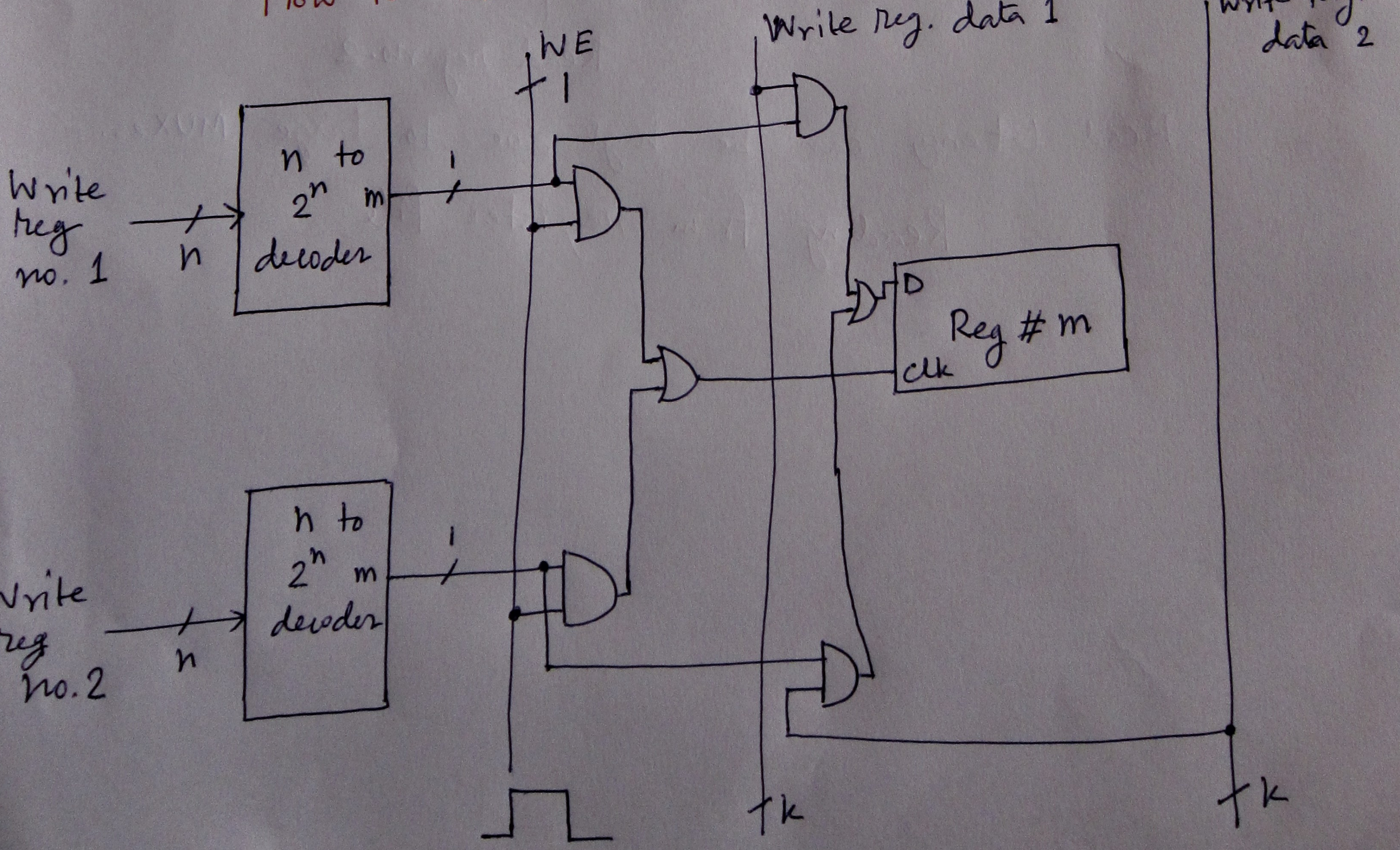
Slide - 14



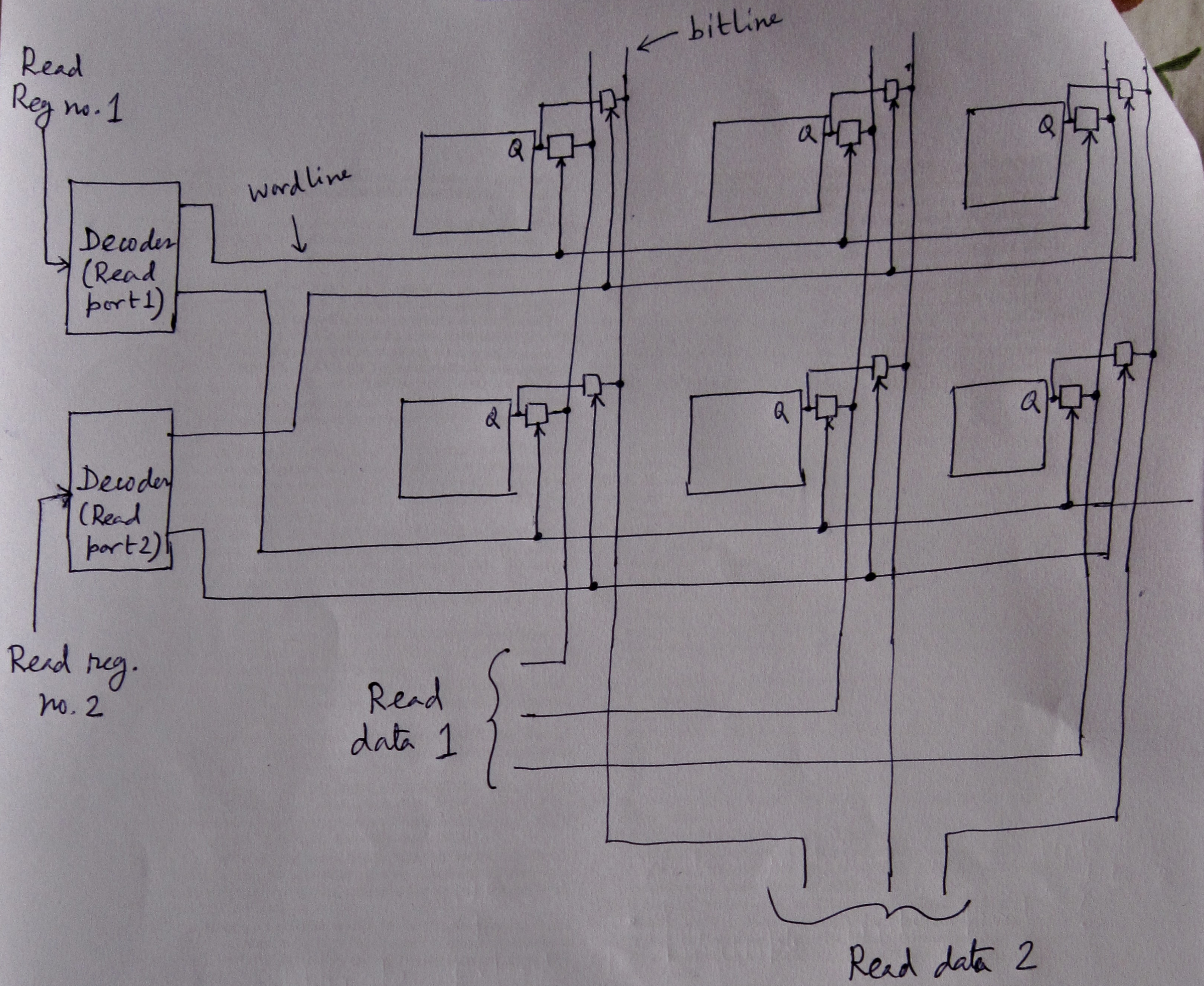
A narrow pulse is presented at WE input

Writing to a register file

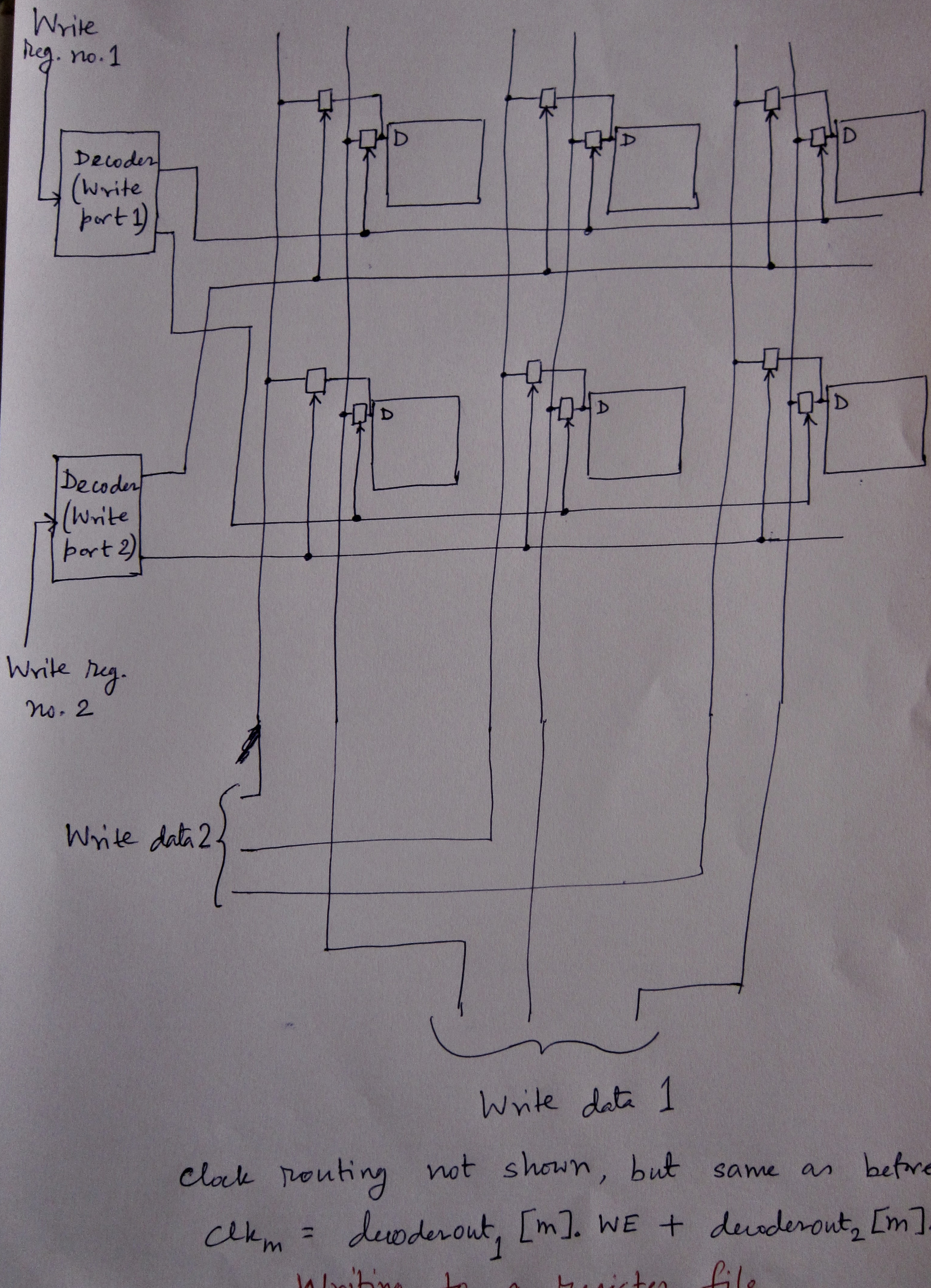
How to extend to multiple write ports?



Slide-14



Reading from register file (without MUX)

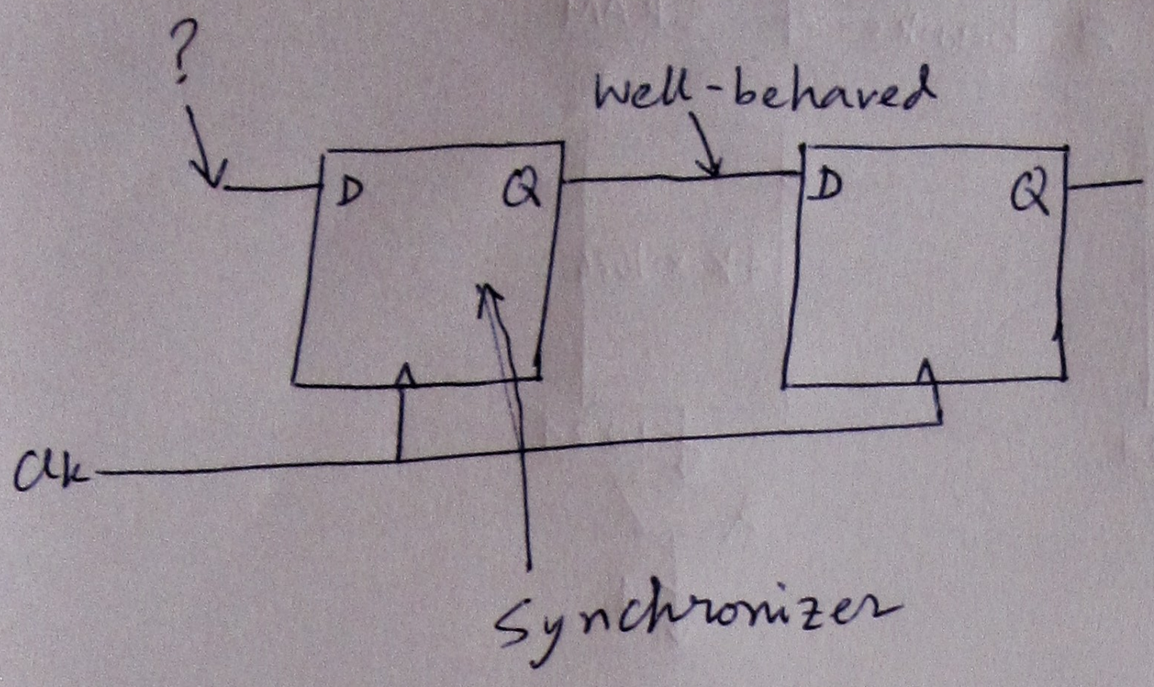


Clock routing not shown, but same as before.

$clk_m = decoderout_1[m].WE + decoderout_2[m].WE$

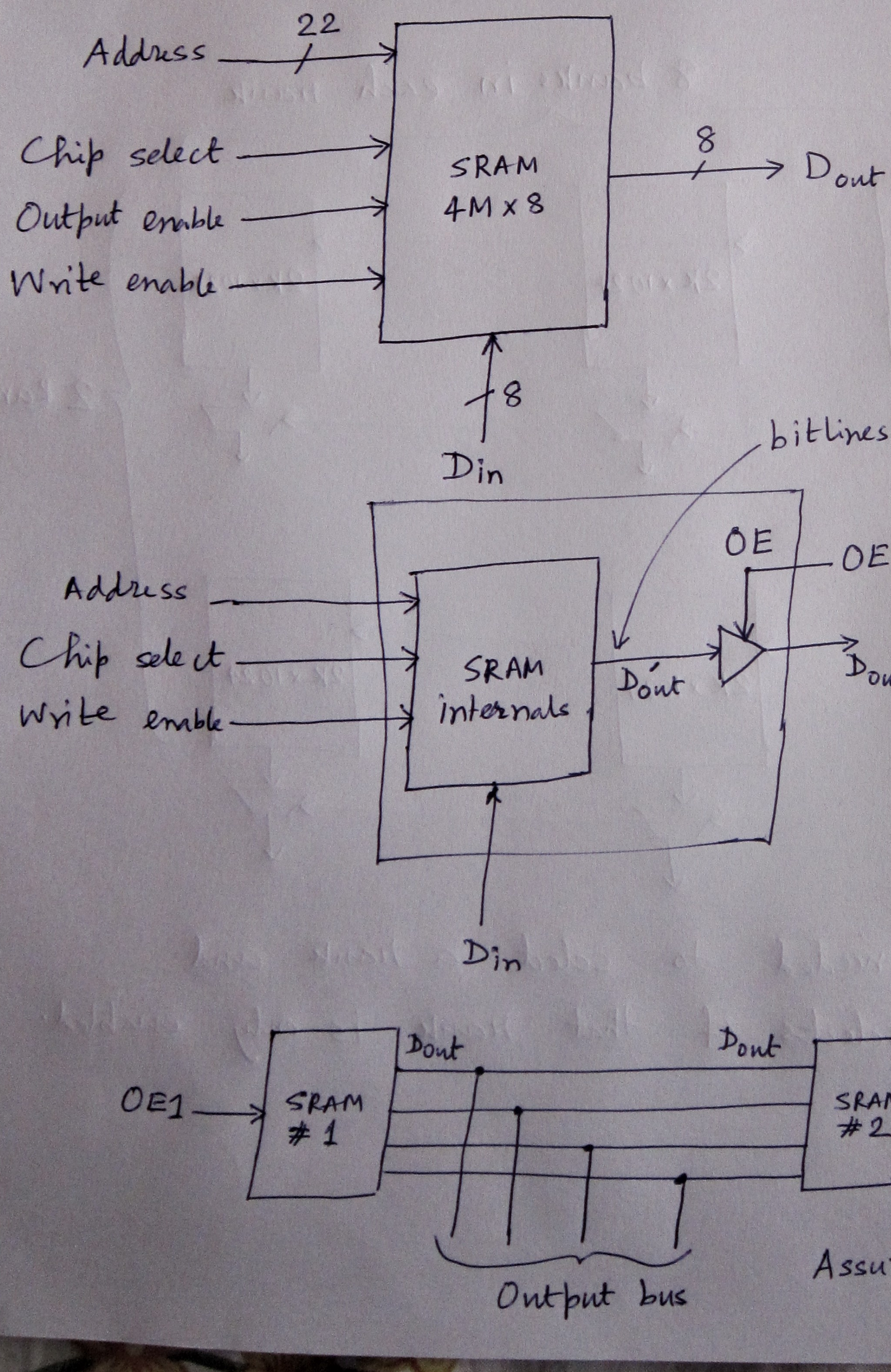
Writing to a register file

Slide - 25



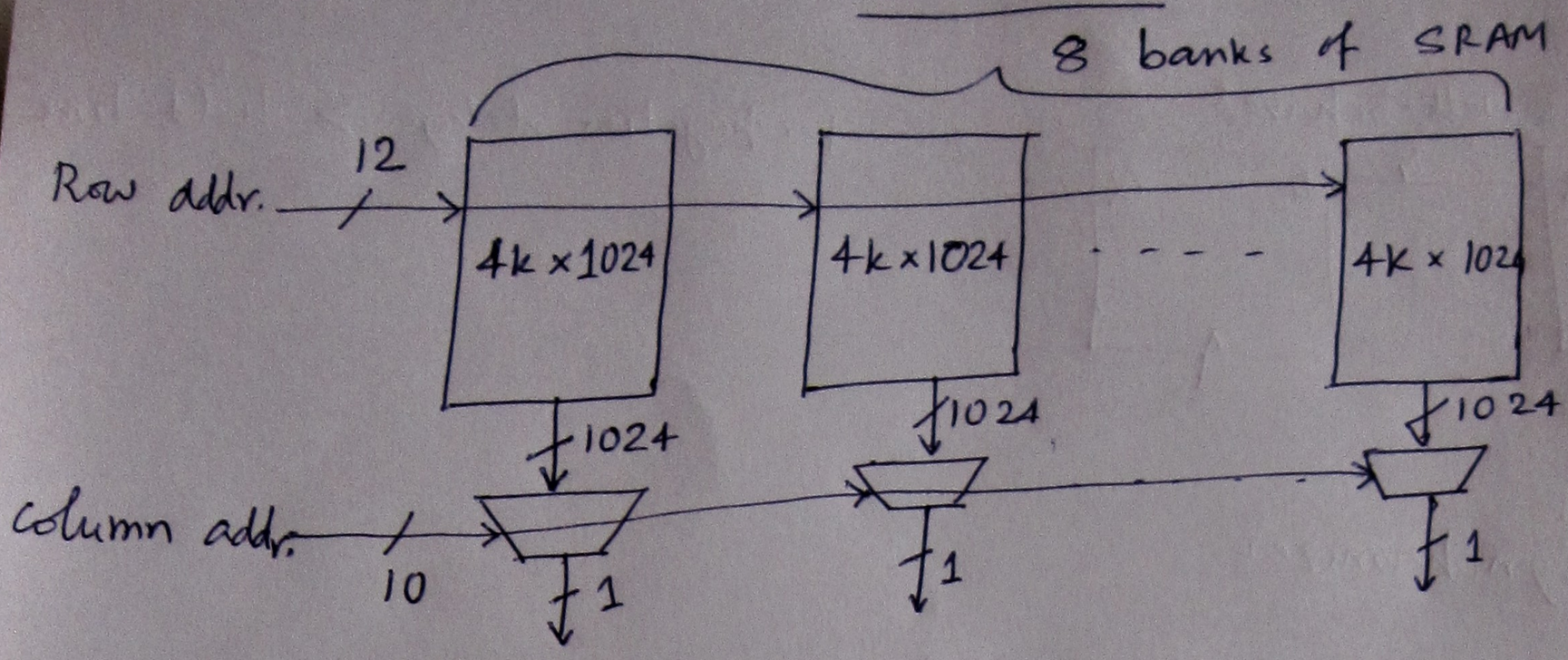
propagation delay > hold time

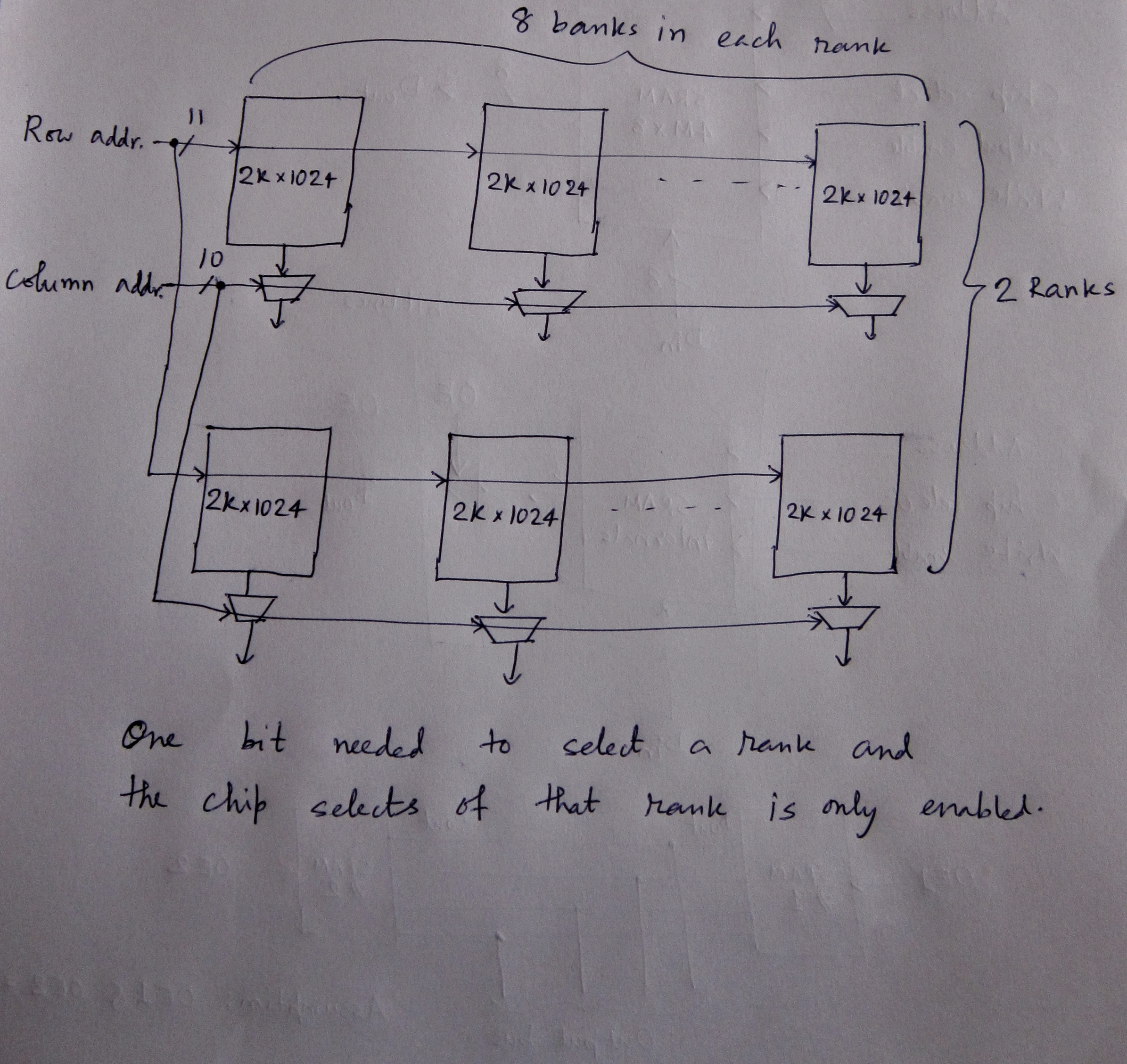
Slide - 27



Assumptin: OE1 & OE2 = 0

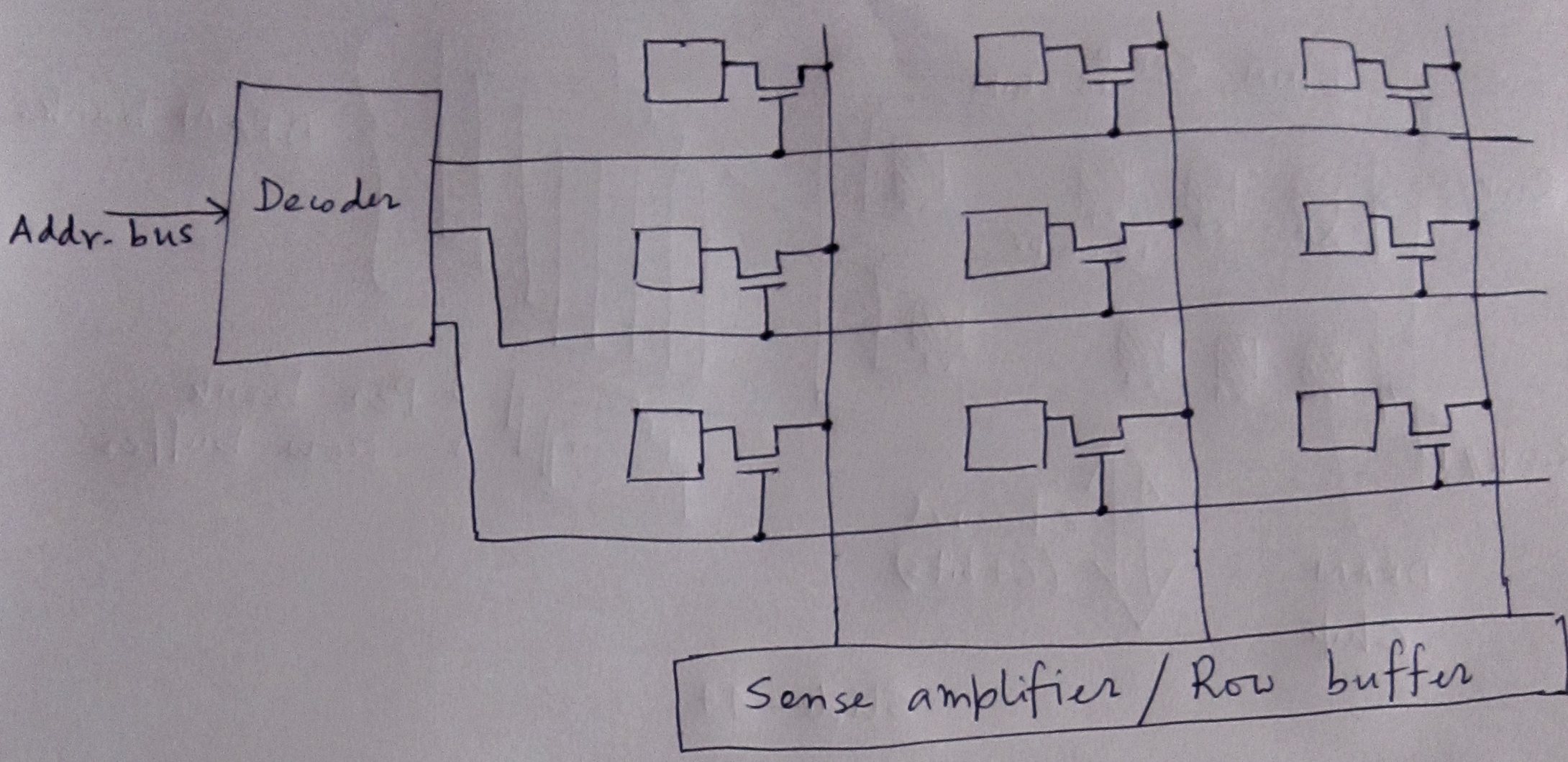
Slide-32





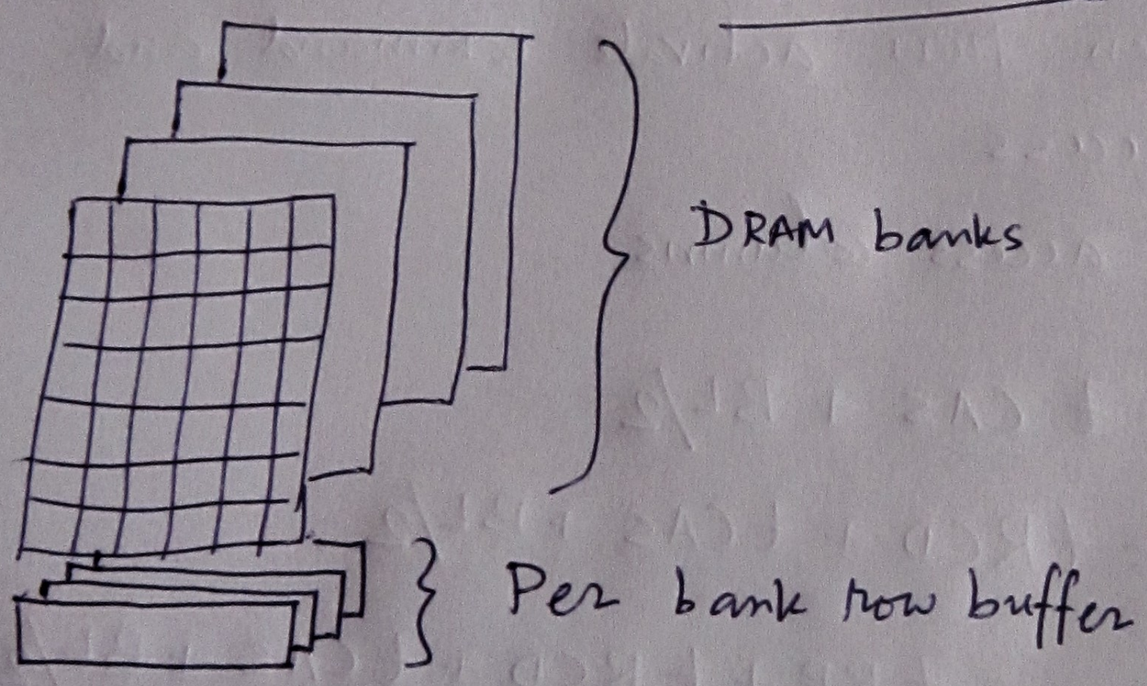
One bit needed to select a rank and the chip selects of that rank is only enabled.

Slide-34



Row x Column organization of a DRAM bank

Slide-36



Internals of a DRAM chip

Example: 512Mbit DRAM chip with 4 banks, 16K rows, 1K columns, x8

Chip capacity = 512 Mbit = 2^{29} bits

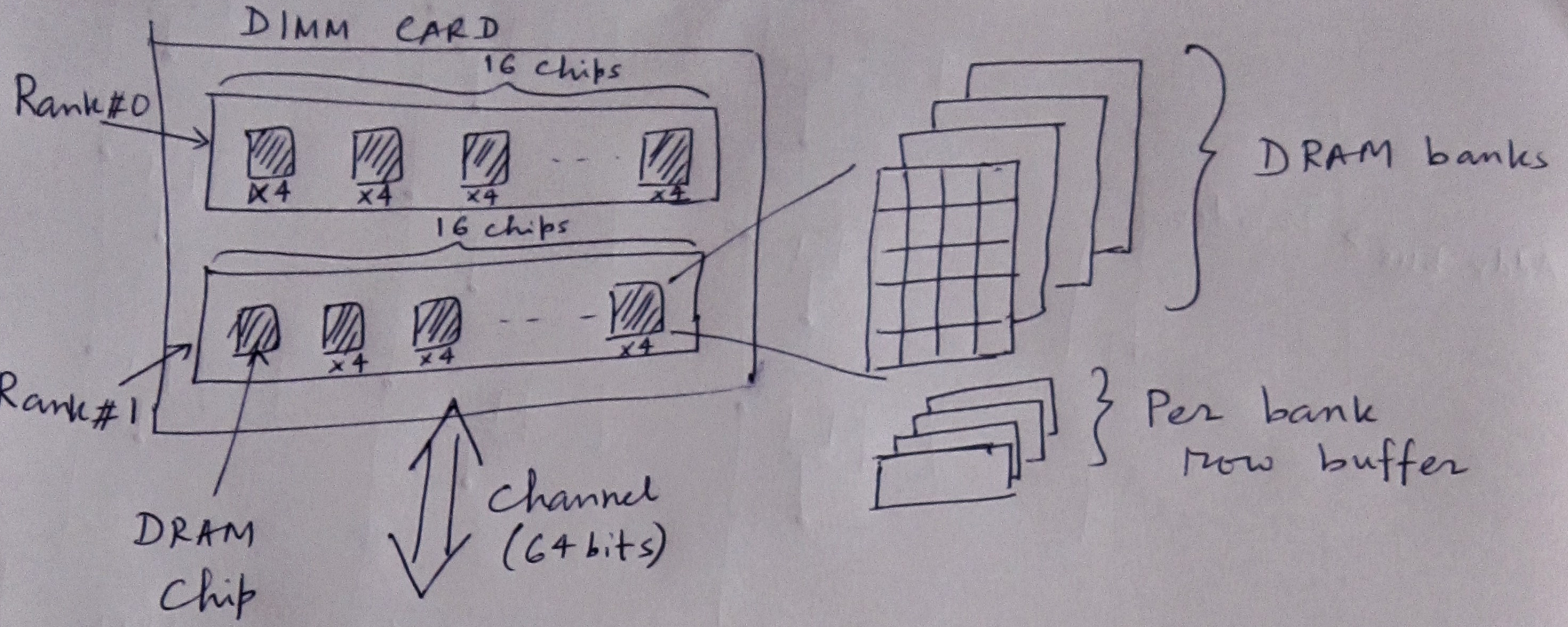
Capacity of one bank = $\frac{2^{29}}{4}$ bits = 2^{27} bits

Number of bits in a row = $\frac{2^{27}}{16K} = \frac{2^{27}}{2^{14}} = 2^{13}$ = no. of bitlines

Number of columns = $\frac{\text{no. of bitlines}}{\text{bits per column}} = \frac{2^{13}}{8} = 2^{10} = 1K$

x8 chip

Slide - 42



Slide - 44

Timing parameters:

tRP = time to precharge a bank

tRCD = time between row activate command and column access

tCAS = time to access columns

Row hit latency = tCAS + BL/2

Row miss latency = tRCD + tCAS + BL/2

Row conflict latency = tRP + tRCD + tCAS + BL/2

BL/2 cycles are the channel transfer time.

Usually, tRP = tRCD = tCAS

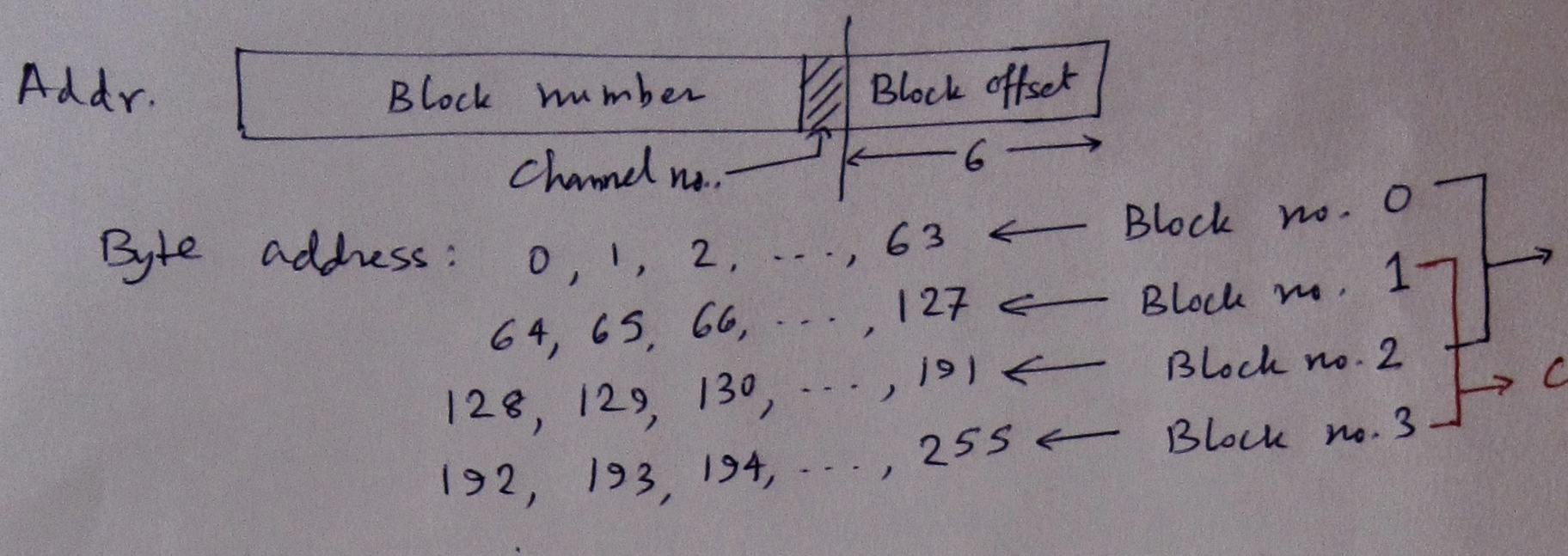
Typical DRAM module spec : DDRx - 2400 15-15-15

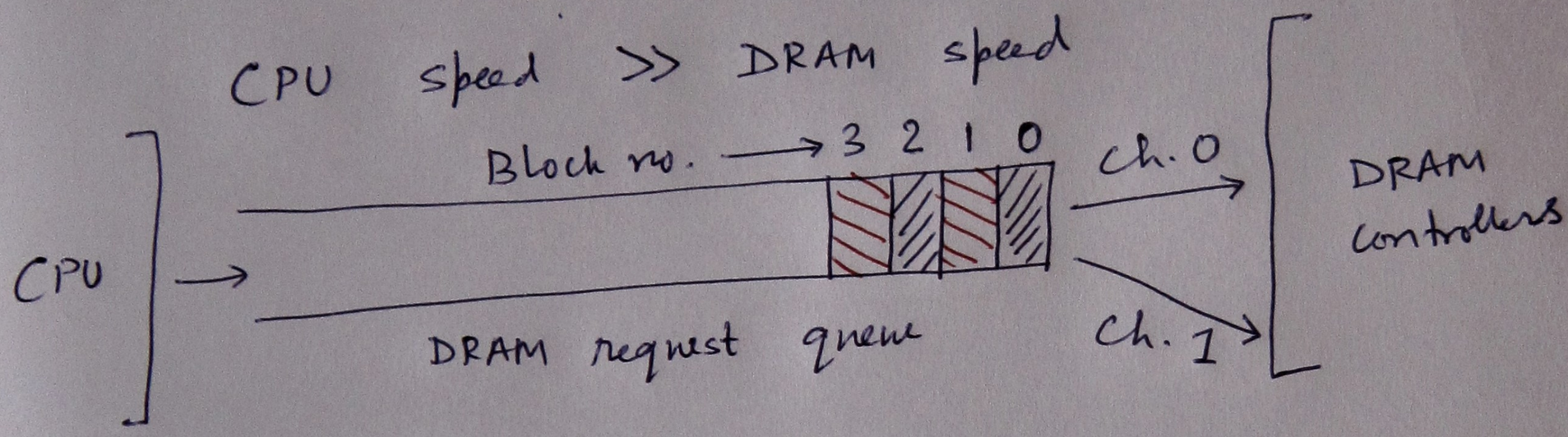
(2400: twice freq in MHz)

x ∈ {2, 3, 4}

15-15-15: tCAS, tRCD, tRP

Slide - 49

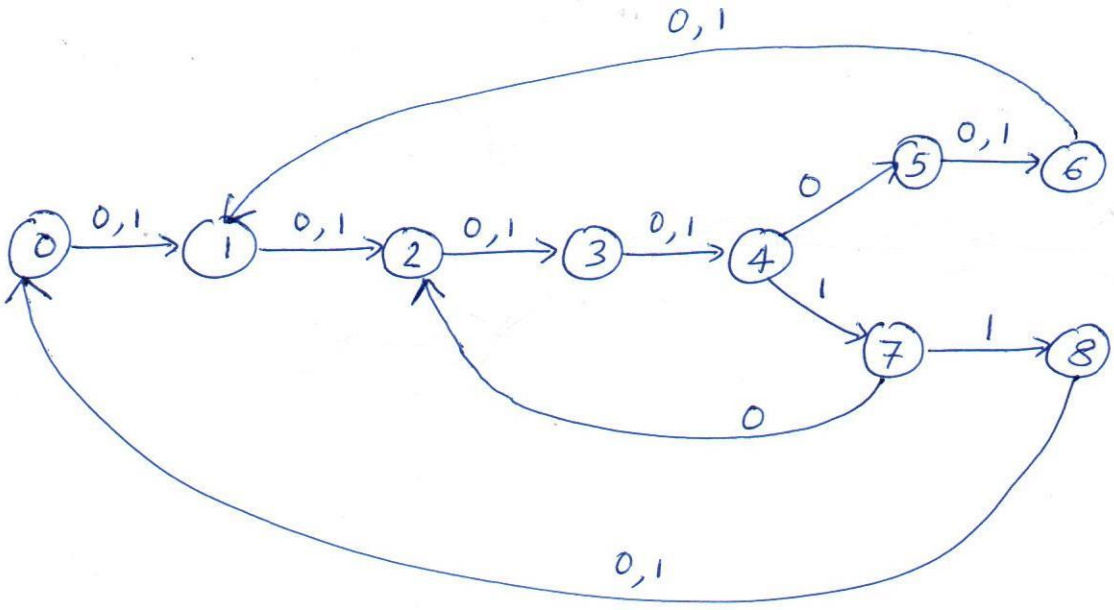




Slide - 50

Accessing a data structure sequentially: (assume 1 channel)

Row 0, Rank 0, Bank 0, columns
Row 0, Rank 0, Bank 1, columns
⋮
Row 0, Rank 0, Bank n, columns
Row 0, Rank 1, Bank 0, columns
⋮
Row 0, Rank 1, Bank n, columns
Row 0, Rank 2, Bank 0, columns
⋮
Row 0, Rank r, Bank n, columns
Row 1, Rank 0, Bank 0, columns
⋮



FSM with nine states and one-bit input. The input that triggers a state transition is shown on the arc for that state transition.

There are two branch points: one at state 4 and another at state 7. So, two dispatch ROMs would be needed.

There are two unconditional non-sequential jumps at states 6 and 8. These two will correspond to two constant inputs to the state selection multiplexor.