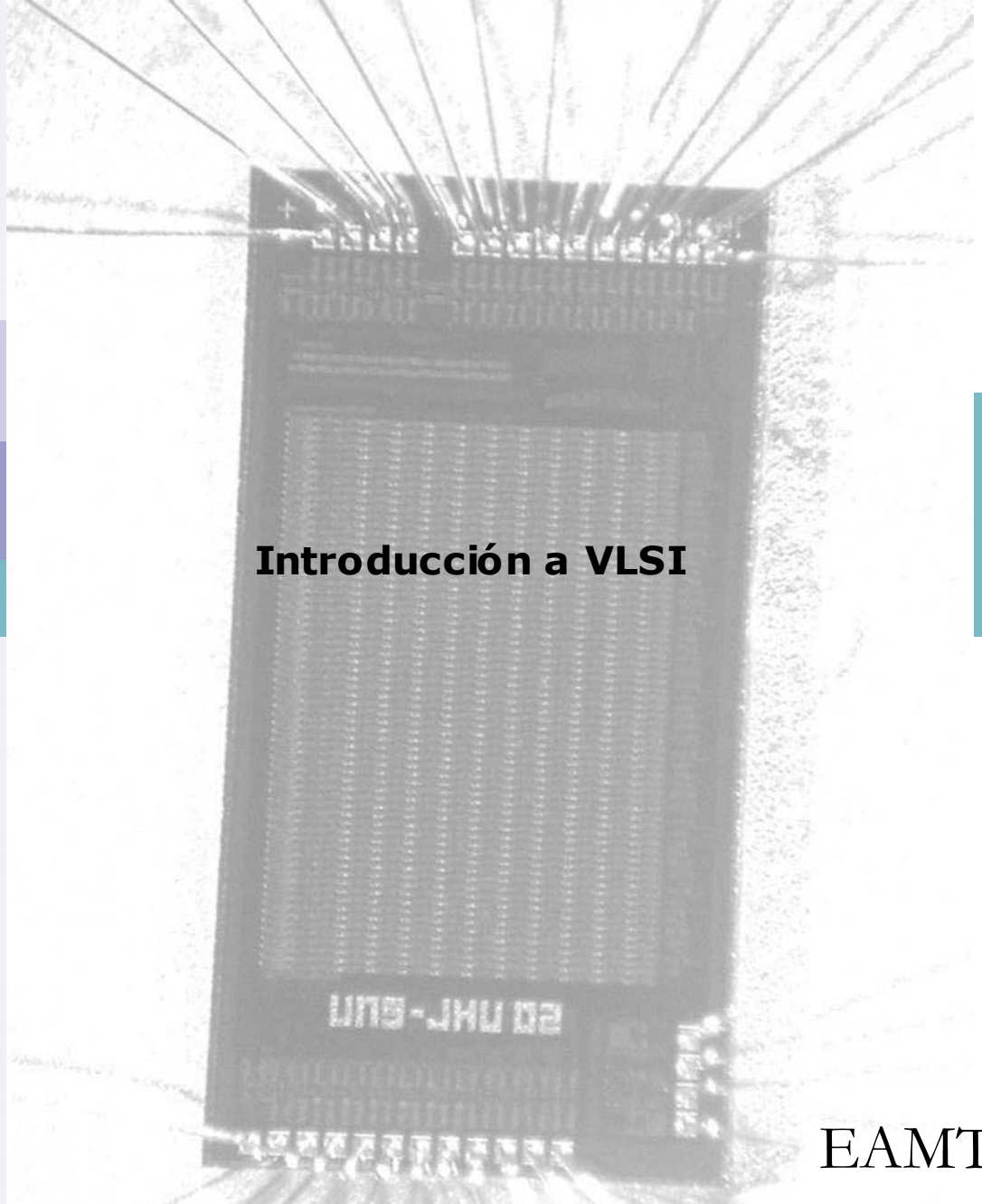
Introducción a VLSI

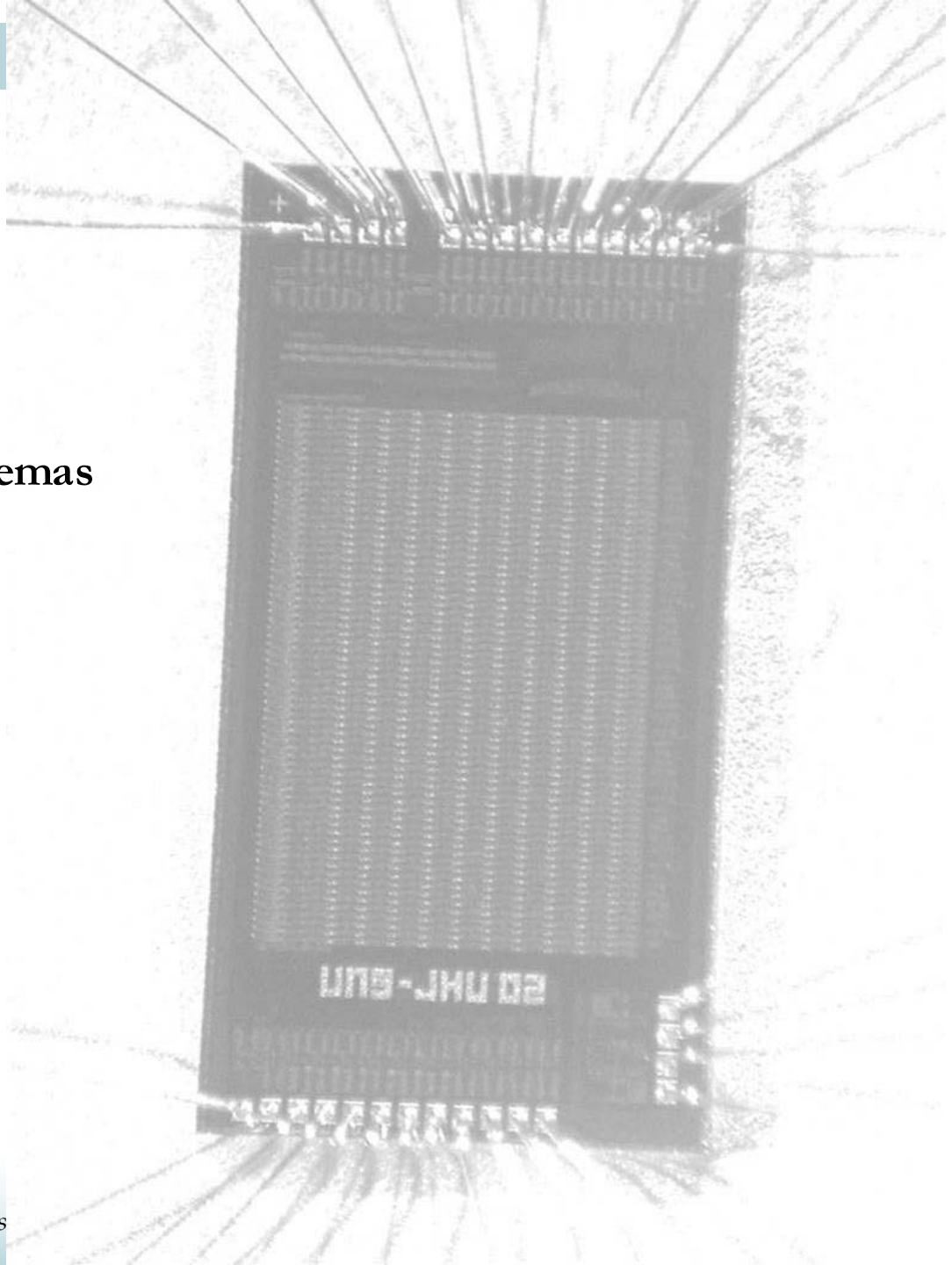


EAMTA 2006

Clase 4: Lógica Secuencial y Subsistemas

Programa

- El Transistor MOS
- Layers y Layout
- Lógica Combinacional
- **Lógica Secuencial y Subsistemas**

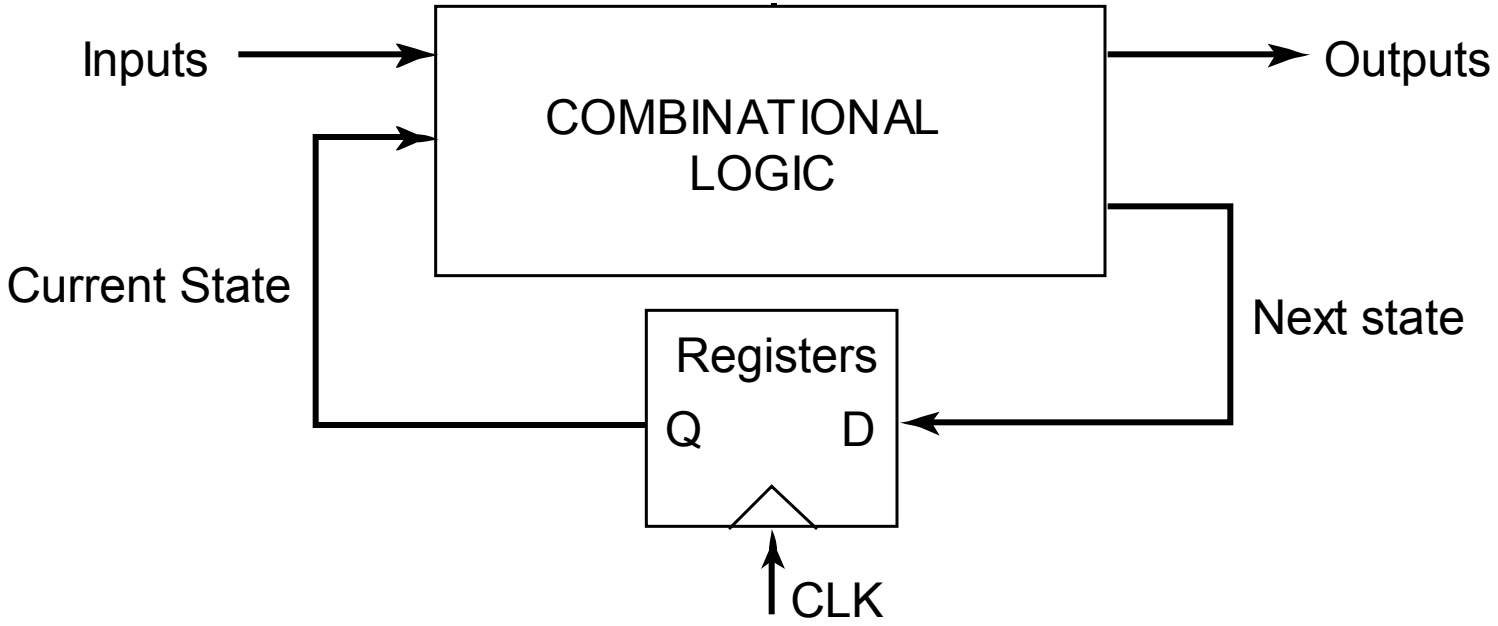


Organización

- Lógica secuencial
- Sumadores
- Otros subsistemas
- Memorias

Sequential Logic Circuits

Sequential Logic



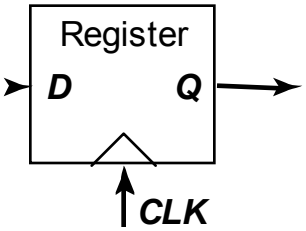
2 storage mechanisms

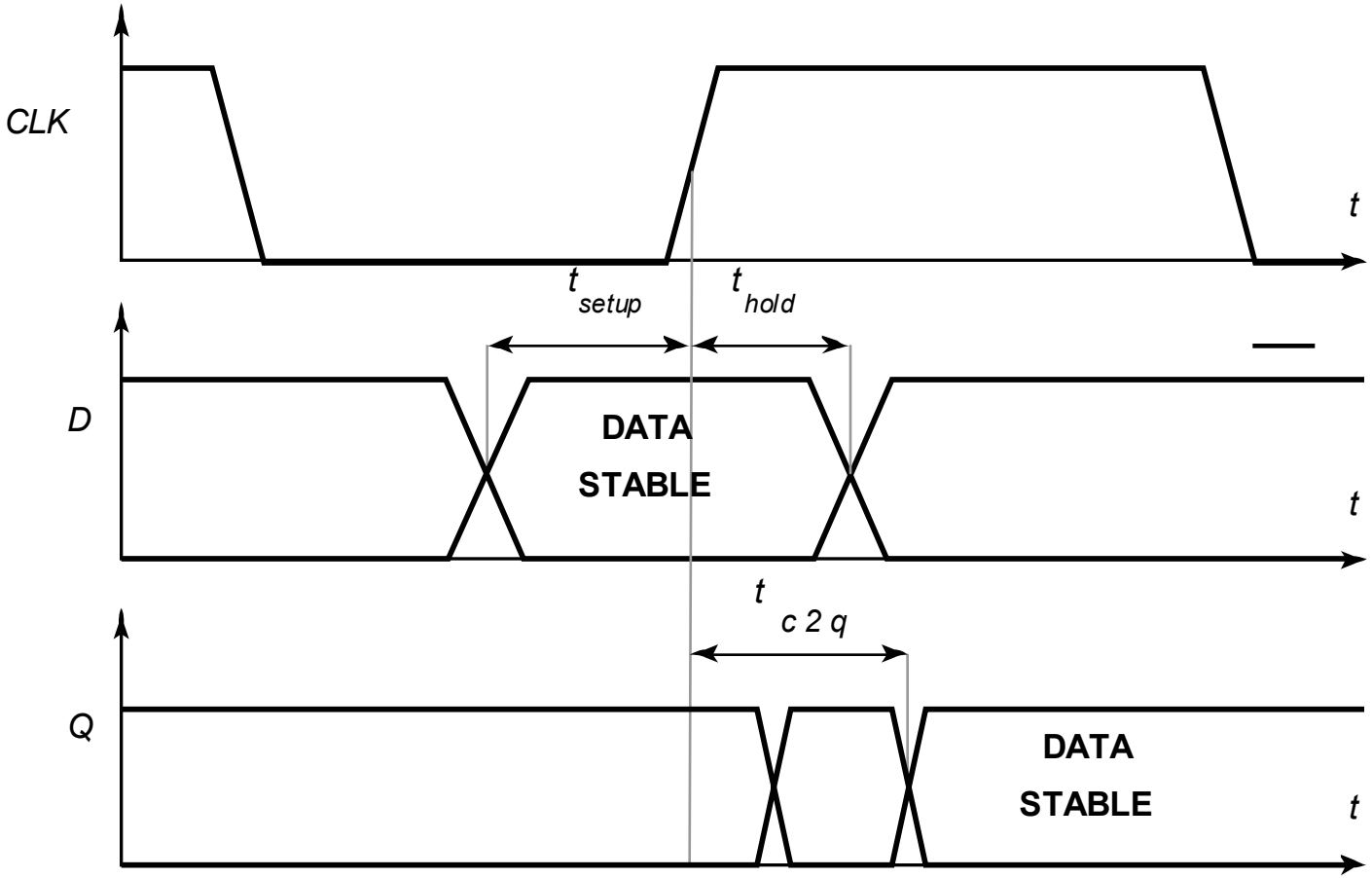
- **positive feedback**
- **charge-based**

Naming Conventions

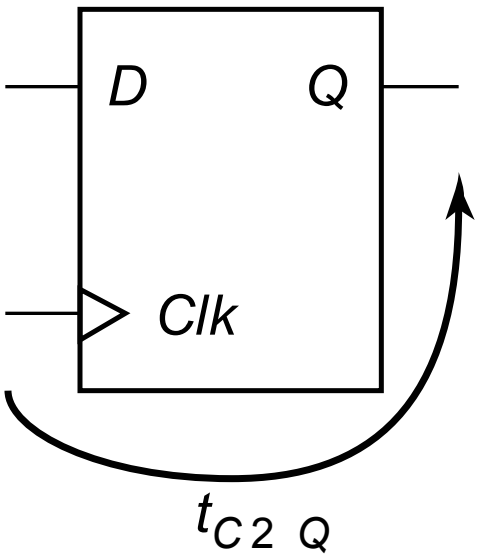
- In our text:
  - a latch is level sensitive
  - a register is edge-triggered
- There are many different naming conventions
  - For instance, many books call edge-triggered elements flip-flops
  - This leads to confusion however

Timing Definitions

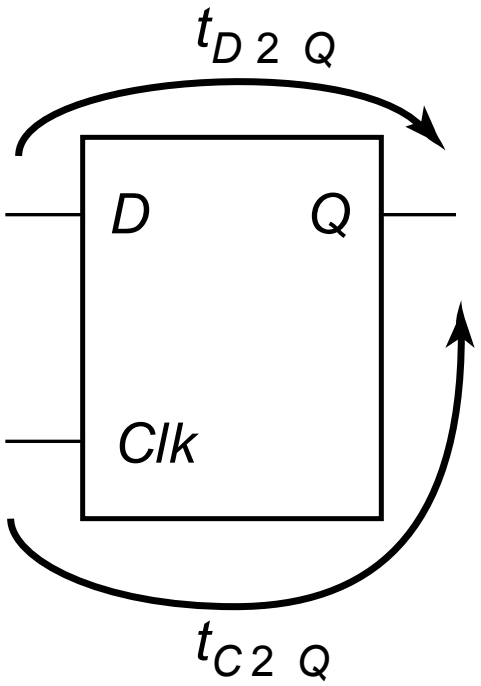




Characterizing Timing

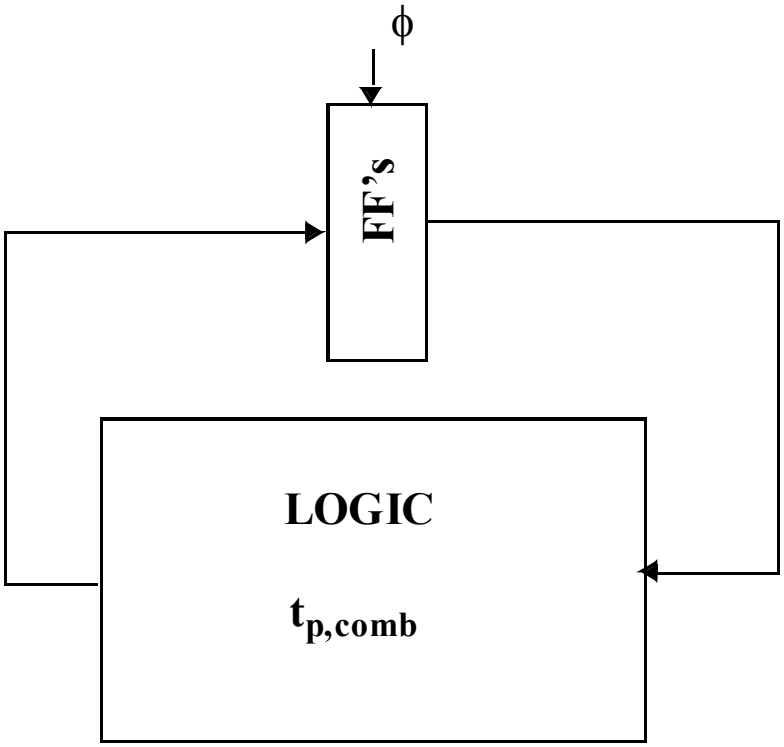


Register



Latch

Maximum Clock Frequency



$$t_{clk\text{-}Q} + t_{p,comb} + t_{setup} = T$$

Also:
$t_{cdreg} + t_{cdlogic} > t_{hold}$

t_{cd}: contamination delay = minimum delay

Data is hold long enough so it can be output

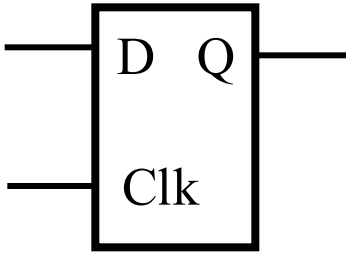
Latch versus Register

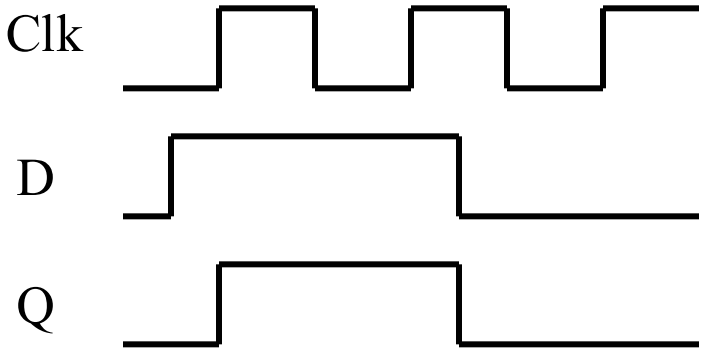
- Latch

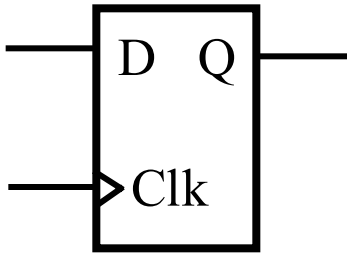
  stores data when clock is low

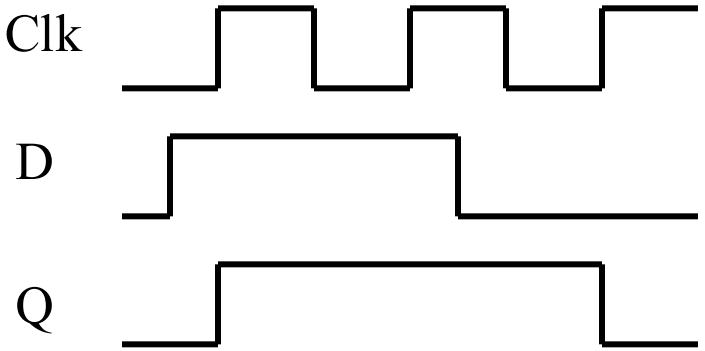
- Register

  stores data when clock rises

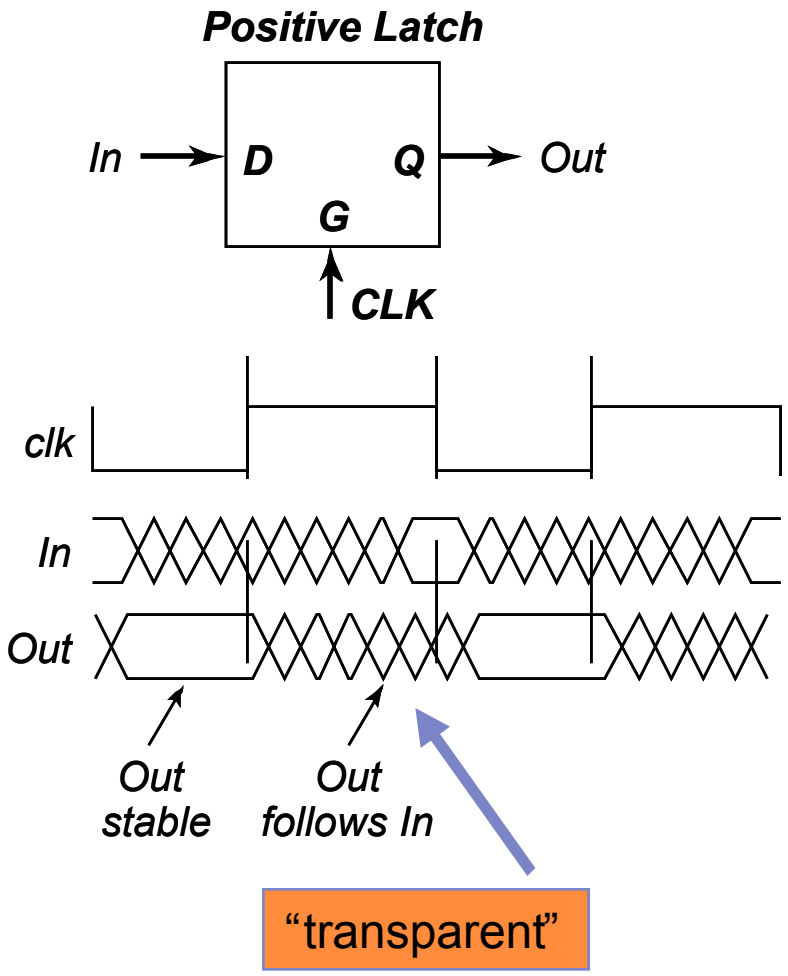


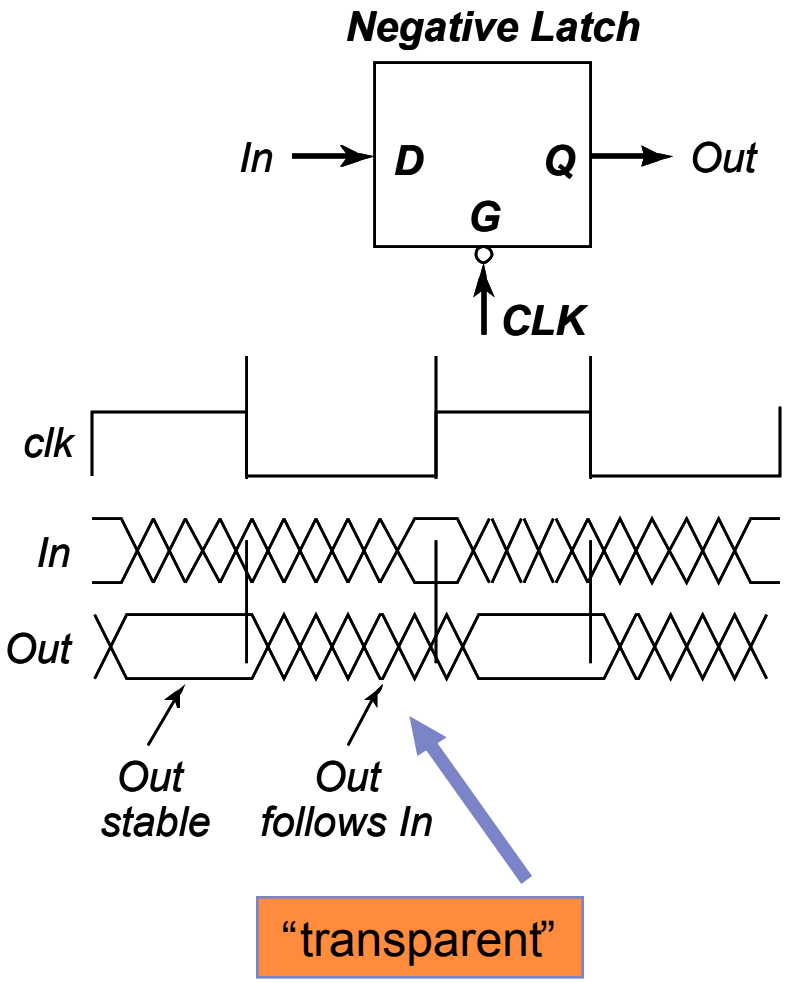






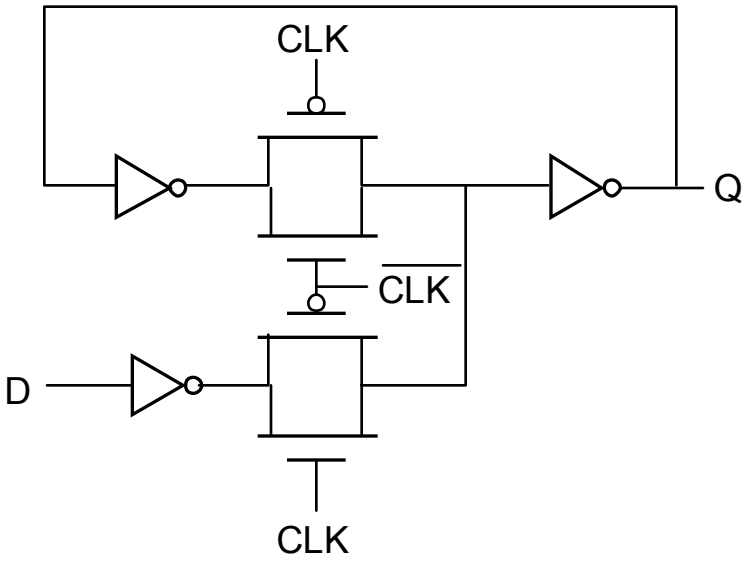
Latches


Positive Latch
In
D
Q
Out
G
CLK
clk
In
Out
Out stable
Out follows In
"transparent"

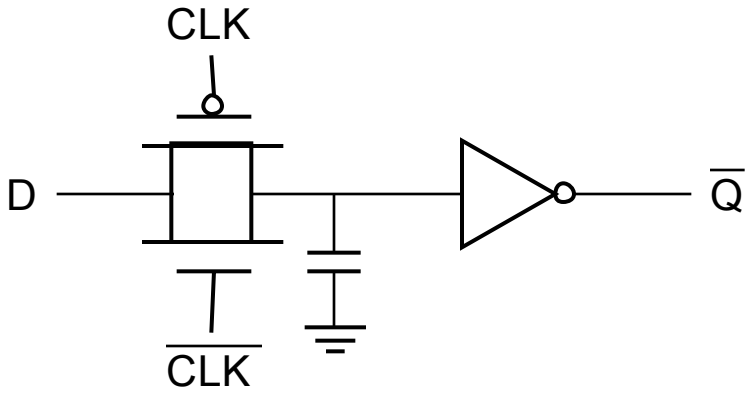

Negative Latch
In
D
Q
Out
G
CLK
clk
In
Out
Out stable
Out follows In
"transparent"

Storage Mechanisms

Static



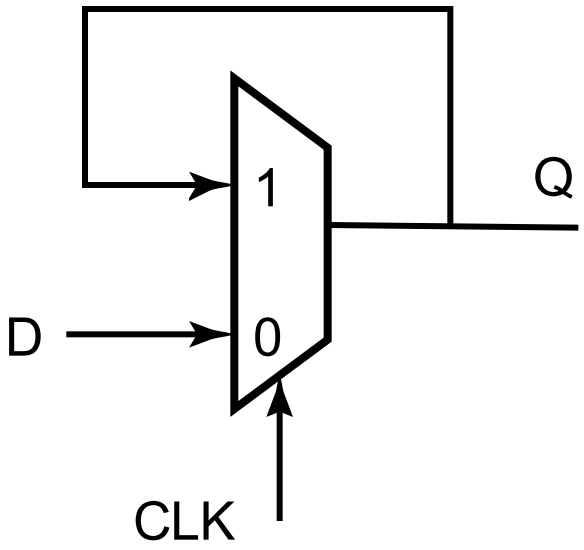
Dynamic (charge-based)



Race might occur with a master-slave

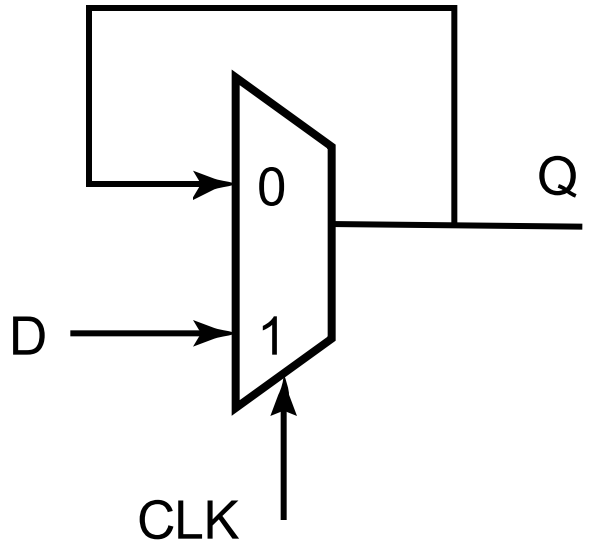
Mux-Based Latches

Negative latch (transparent when CLK= 0)



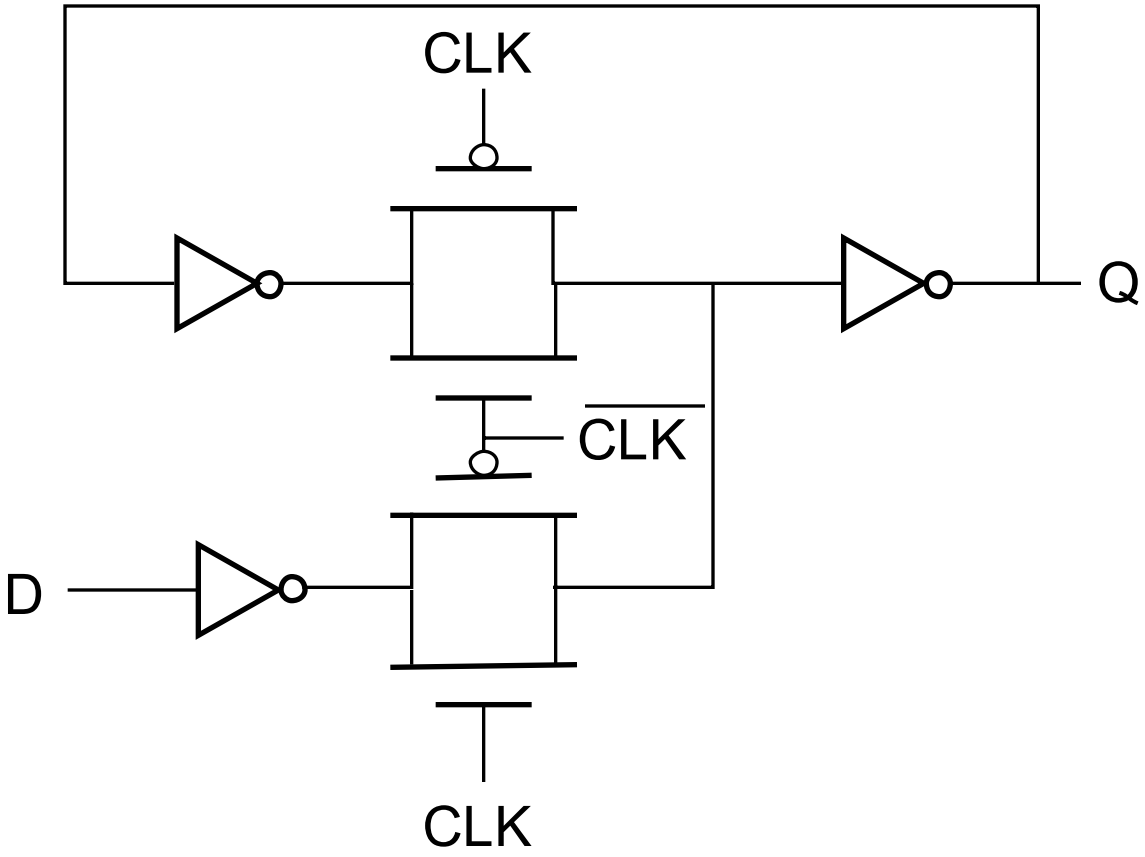
$$Q = \overline{Clk} \cdot Q + Clk \cdot In$$

Positive latch (transparent when CLK= 1)



$$Q = Clk \cdot Q + \overline{Clk} \cdot In$$

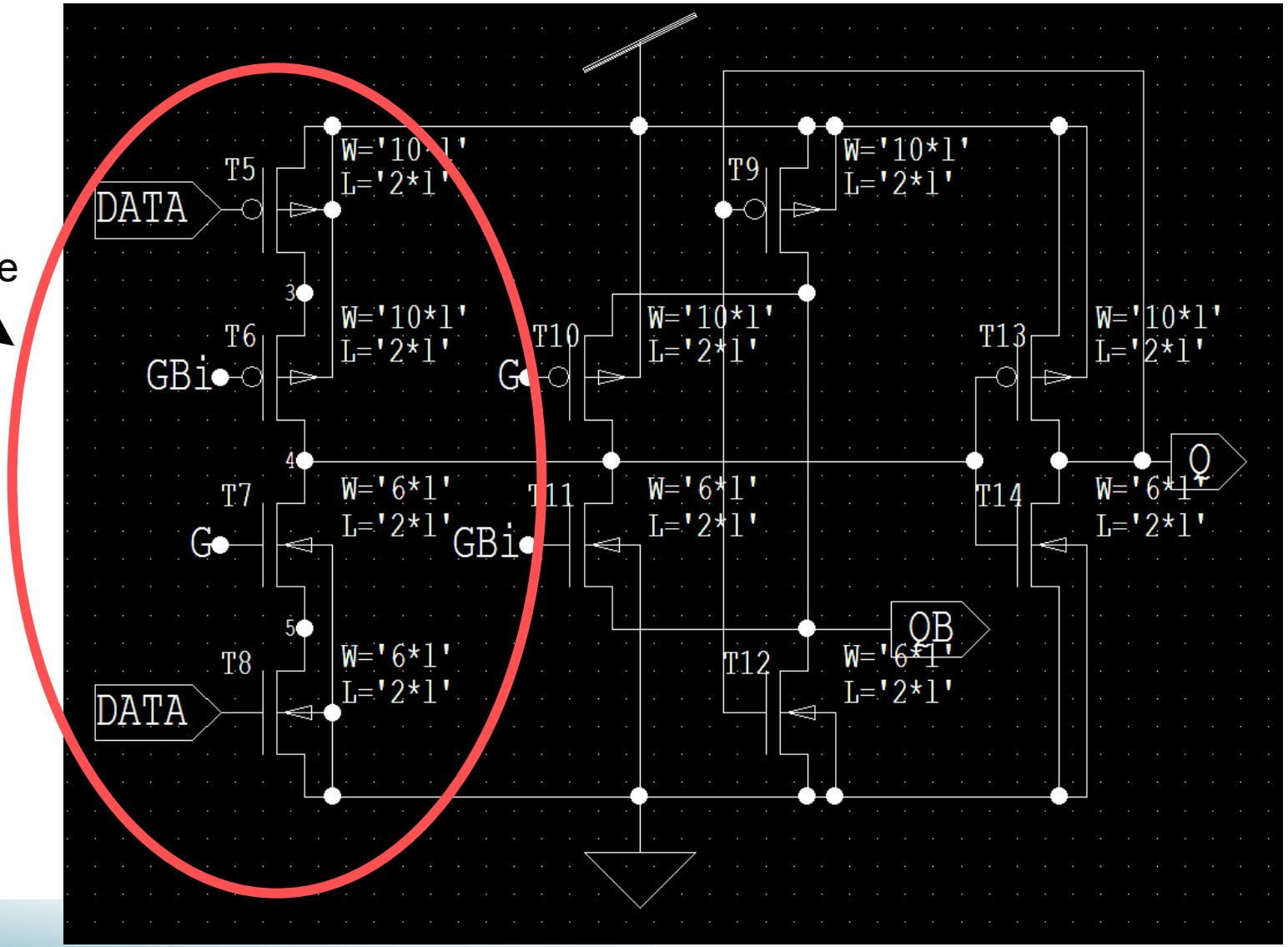
Mux-Based Latch



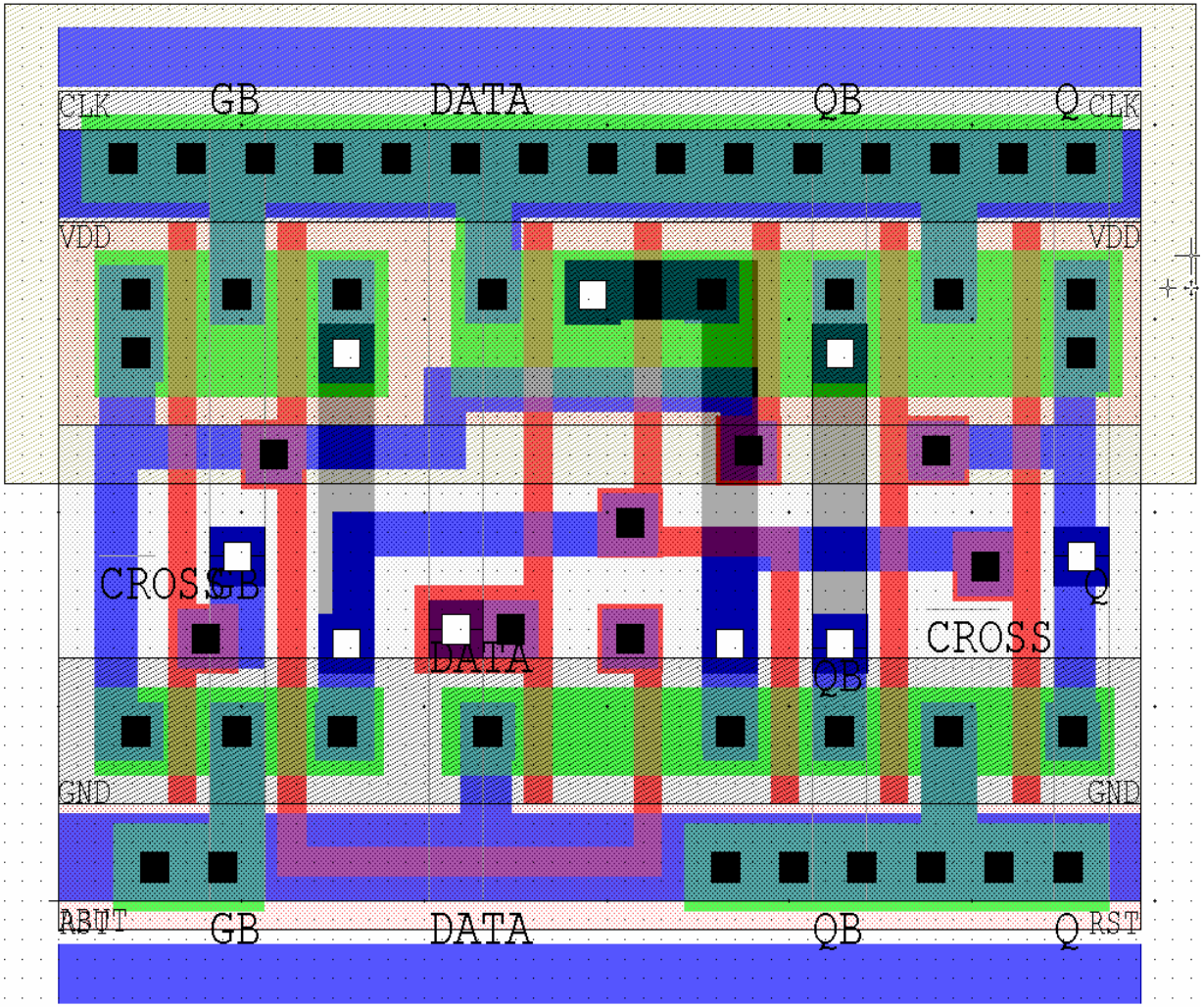
Number of transistor that the clock has to drive is an important metric.

In this case, there are 4 (four) transistors in the CLK line

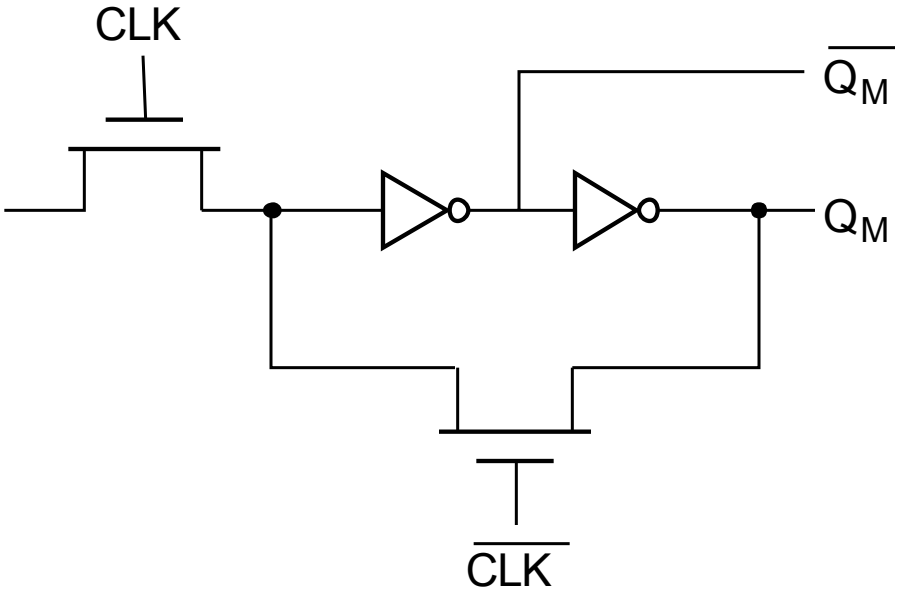
Mux-based latch



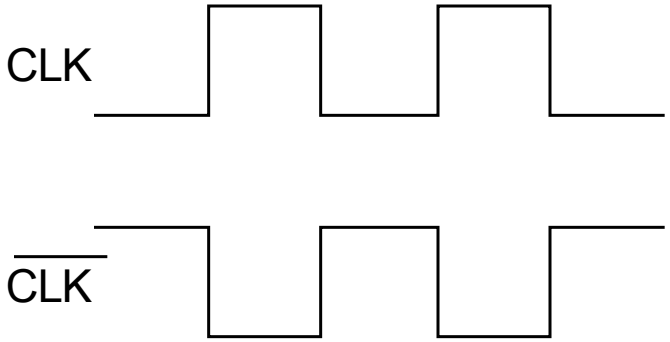
Mux-based latch



Mux-Based Latch



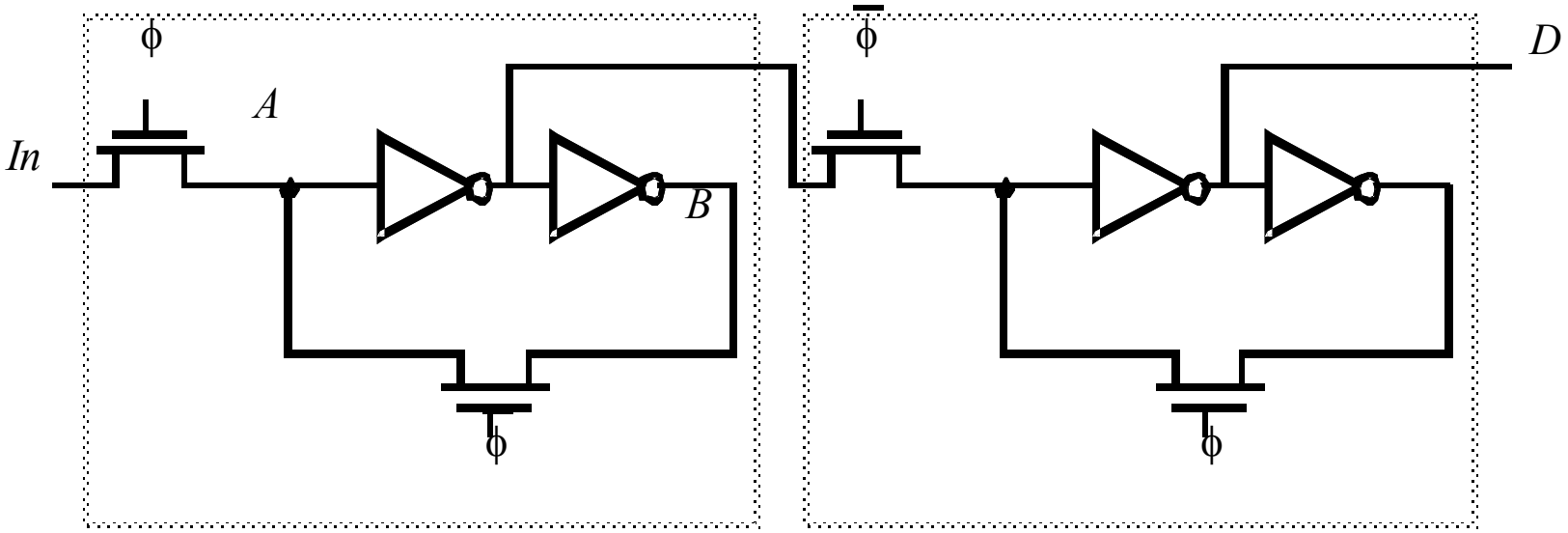
NMOS only



Non-overlapping clocks

Simple but degrades a High input to Vdd-Vtn

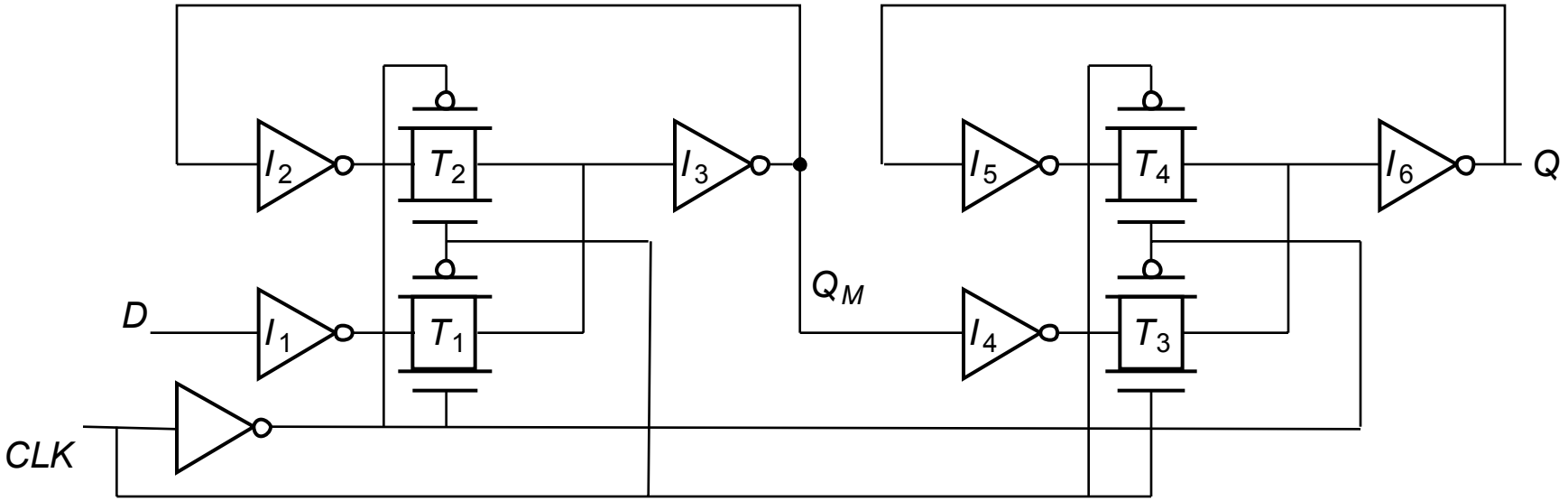
Master-Slave (Edge-Triggered) Register



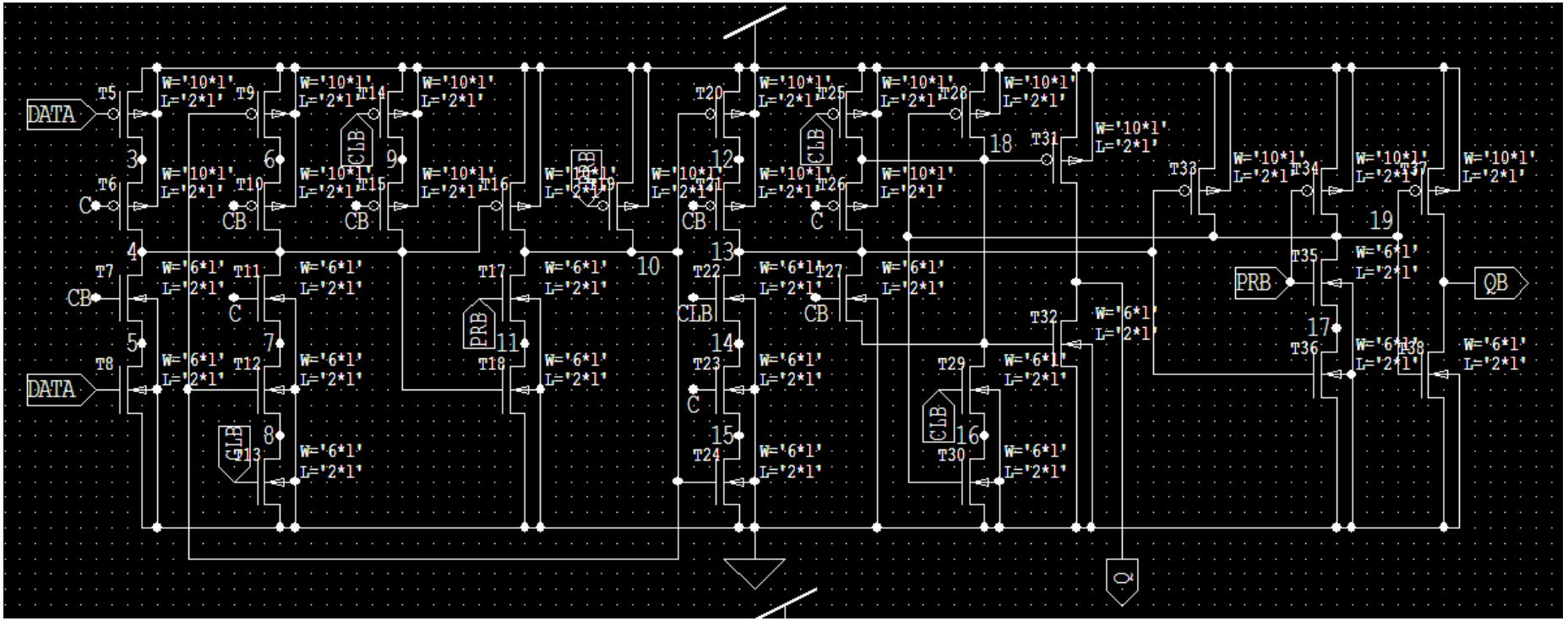
Two opposite latches trigger on edge. Also called master-slave latch pair
This is a positive edge-triggered Reg.
A negative edge-triggered can be realized by switching registers.

Master-Slave Register

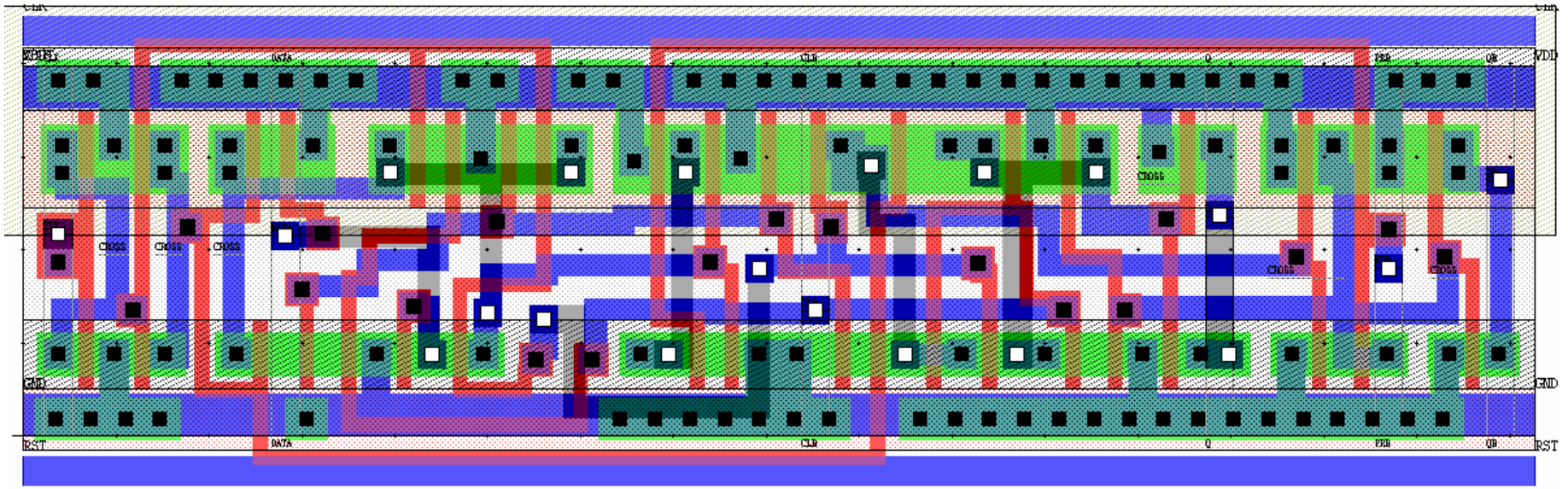
Multiplexer-based latch pair



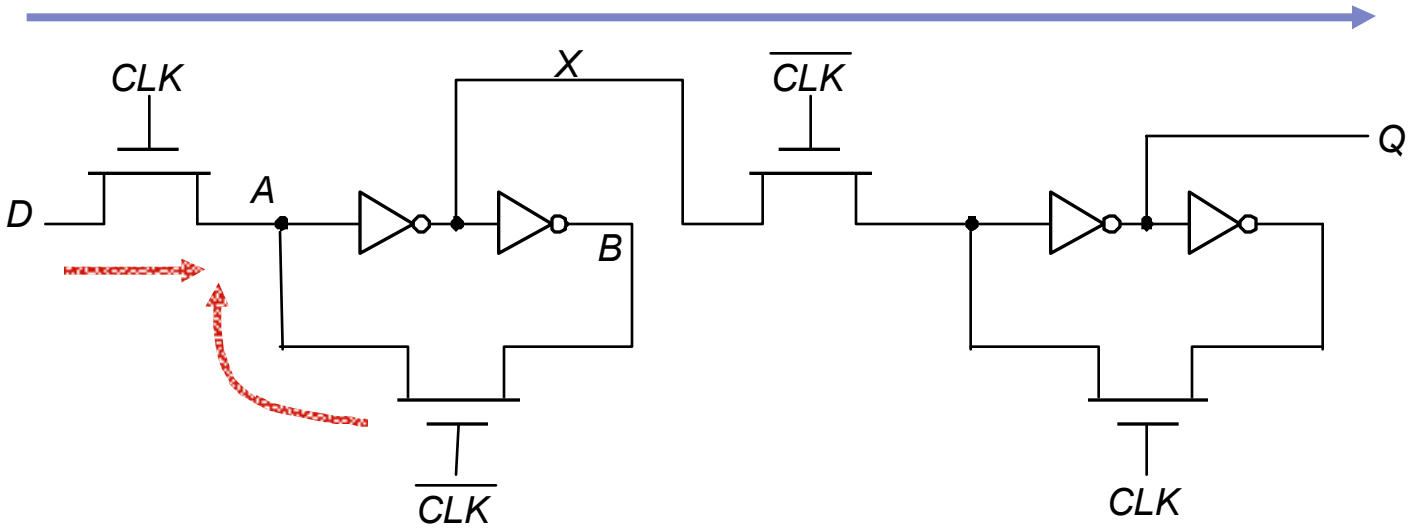
DFF PC


DATA
C
CB
DATA
CB
C
CB
PRB
CLB
GLB
CLB
CB
C
CLB
CB
C
CLB
PRB
QB
Q
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17
T18
T19
T20
T21
T22
T23
T24
T25
T26
T27
T28
T29
T30
T31
T32
T33
T34
T35
T36
T37
T38
W='10*1' L='2*1'
W='6*1' L='2*1'

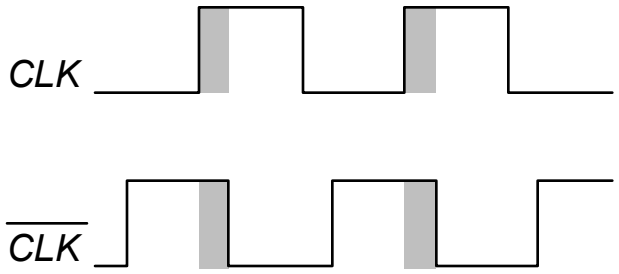
DFF PC



Avoiding Clock Overlap



(a) Schematic diagram

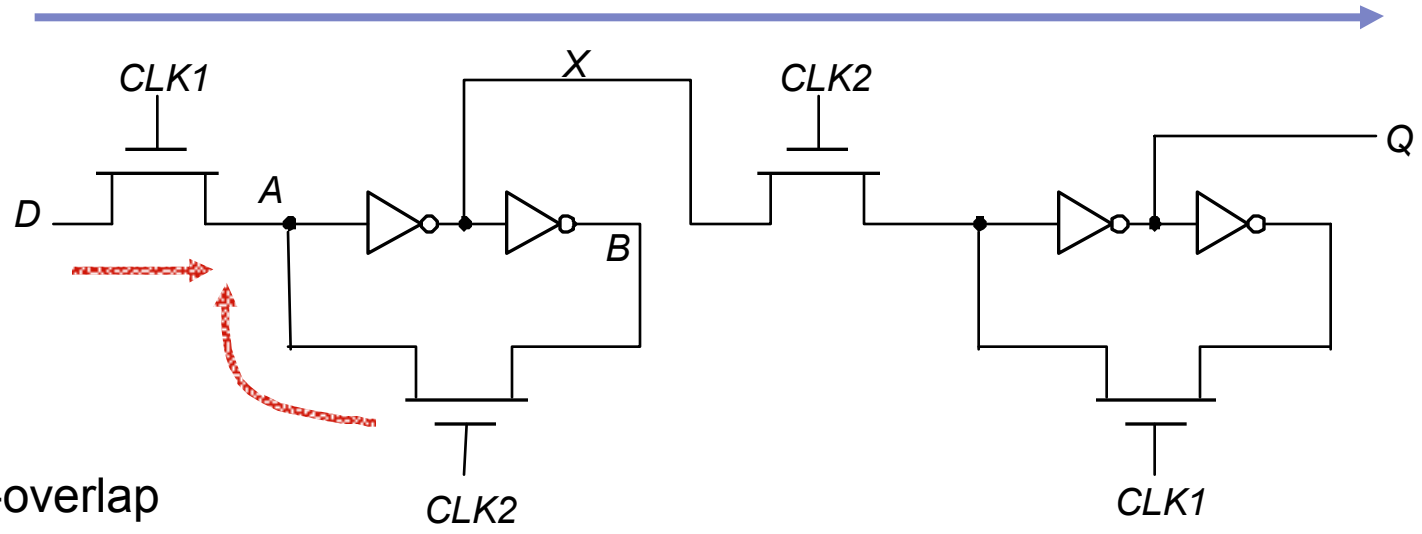


(b) Overlapping clock pairs

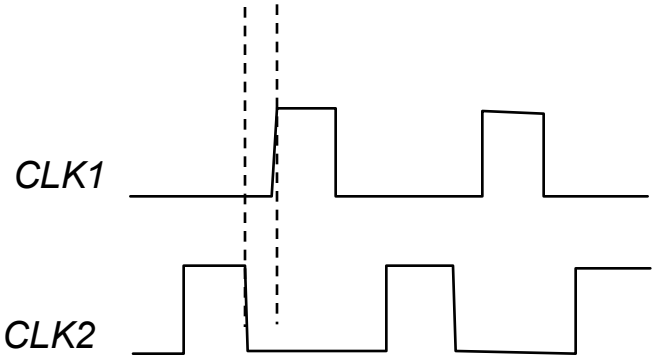
CLK and CLKB are 1 at the same time, then D passes directly to output (race)

Output can change in the rising edge of the CLK for a negative edge-triggered FF

Non-overlapping Clocks



(a) Schematic diagram



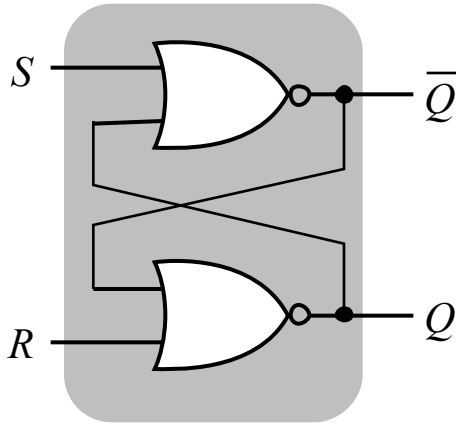
(b) Non-Overlapping clock pairs

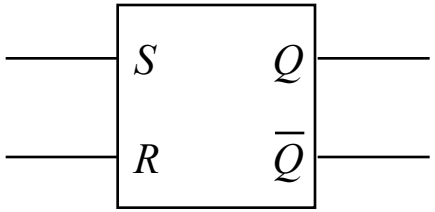
CLK1 and CLK2 are non-overlapping

T non-overlap has to be small enough so that the charge in the intermmediate nodes is preserved (gates are in tri-state)

Overpowering the Feedback Loop — Cross-Coupled Pairs

NOR-based set-reset

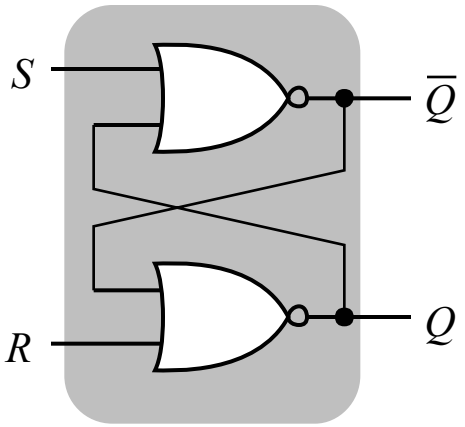




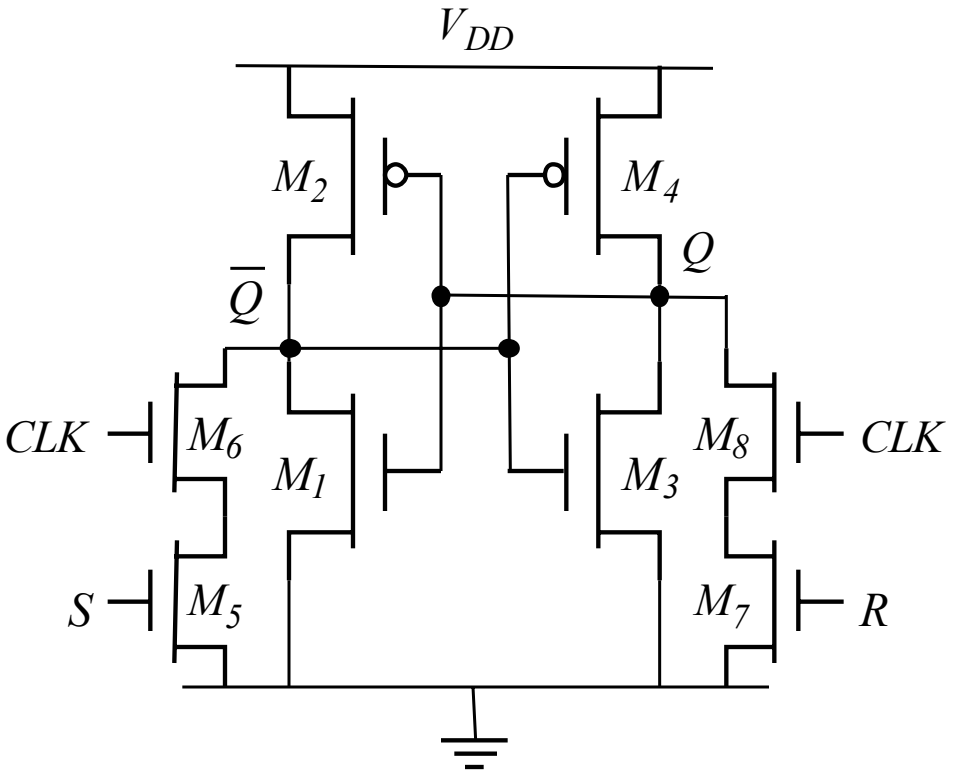
S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

Forbidden State

Clocked SR



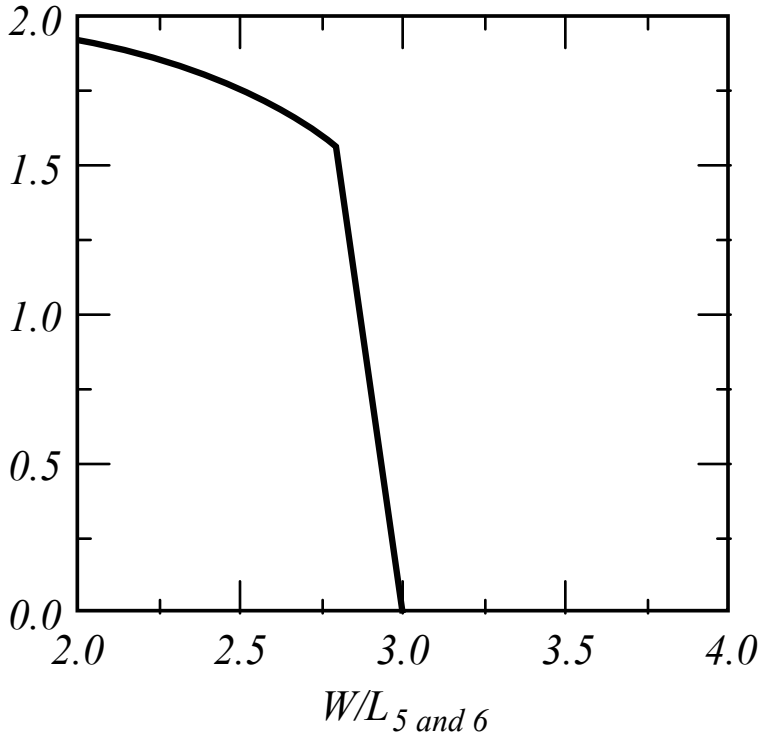
Added clock

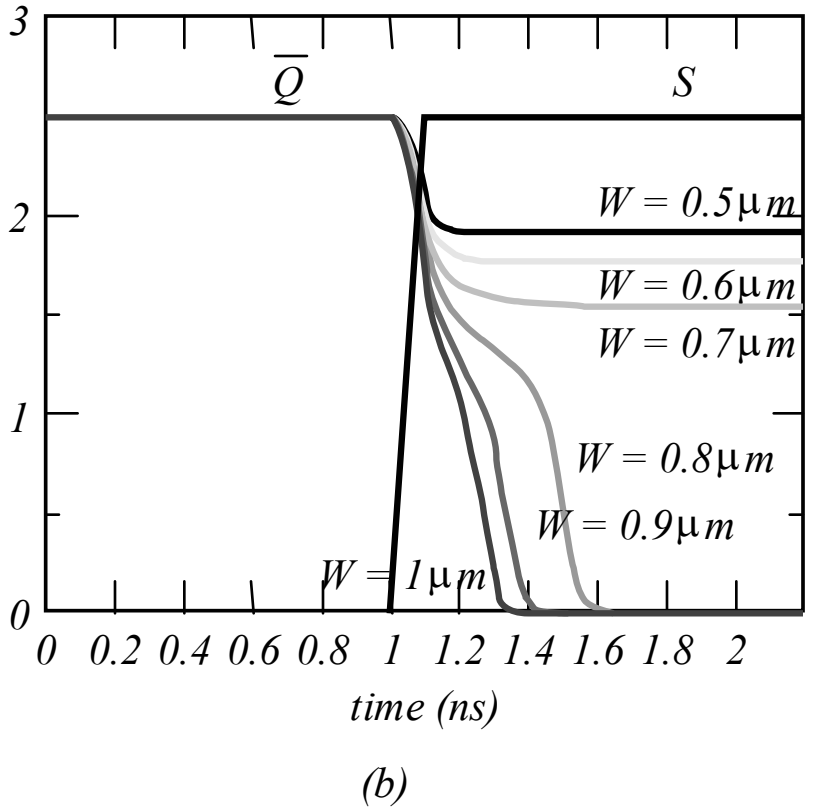


This is not used in datapaths any more, but is a basic building memory cell

Transistors M5-M6 (M7-M8) has to be sized to overpower M2 (M4) and pull that node down

Sizing Issues

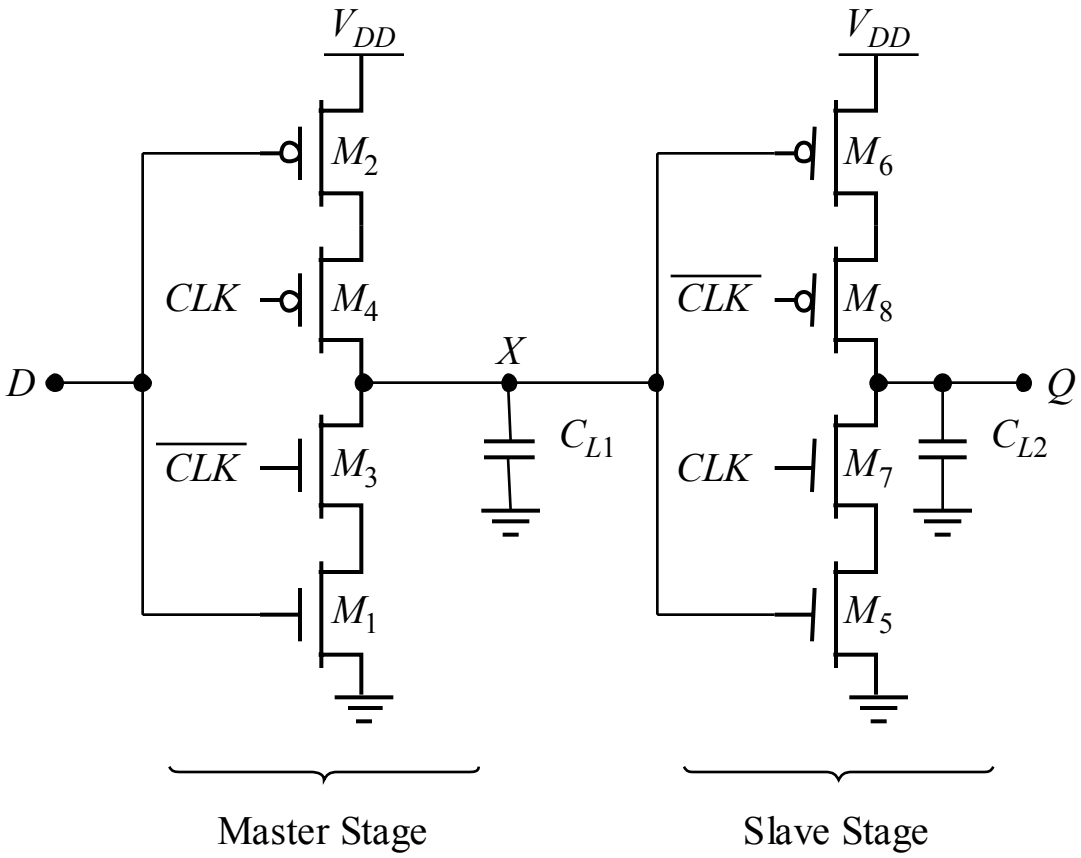




Output voltage dependence on transistor width

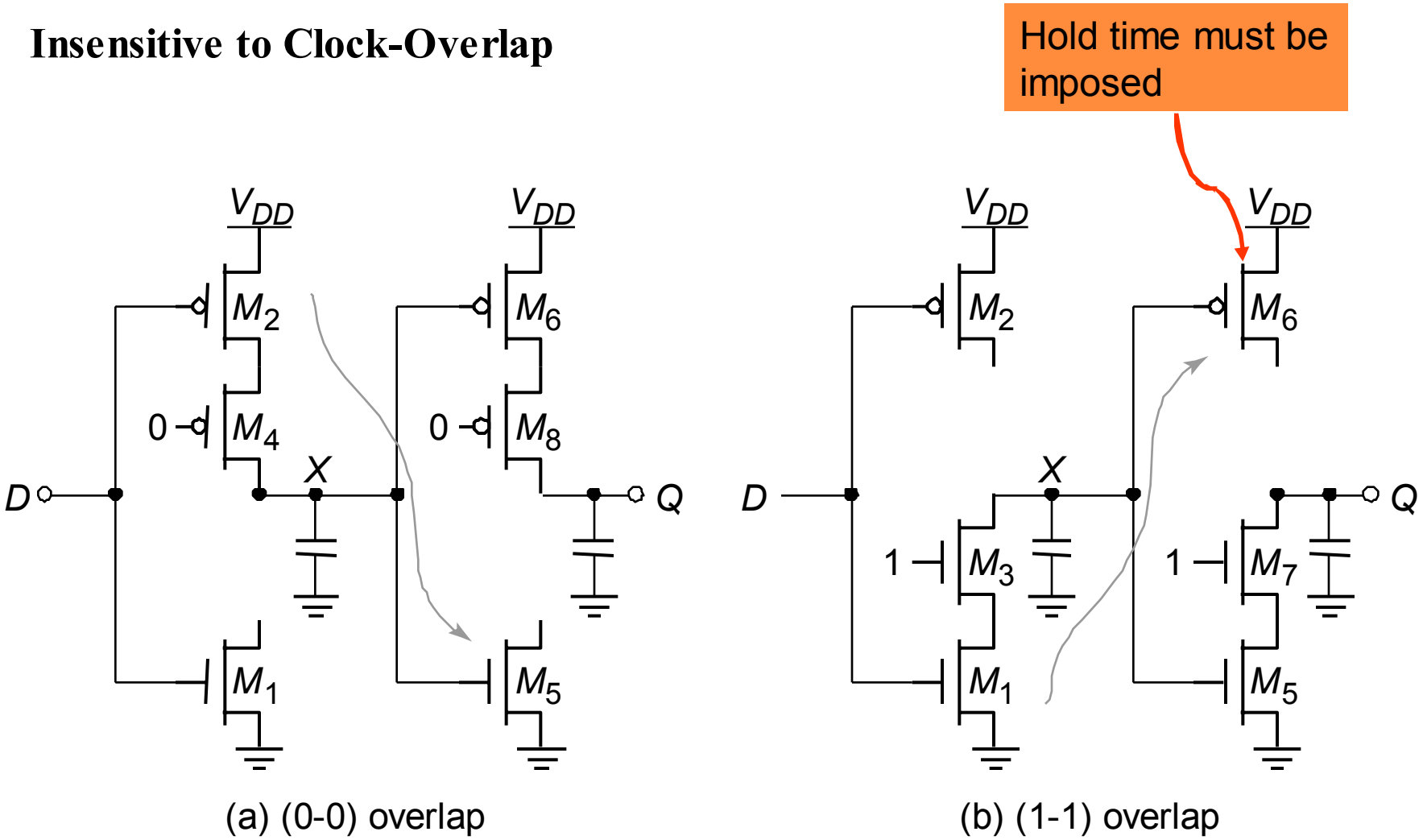
Transient response

Other Latches/Registers: C²MOS



"Keepers" can be added to make circuit pseudo-static

Insensitive to Clock-Overlap



Insensitive as long as the rise and fall times are sufficiently small

Adders

Adders

- Important subsystem in digital design, so we care about performance
- Good example of ways to think about building systems larger than 1-2 gates
- Basic problem: Bit *N* of the result depends on 0 - (N-1) input bits.
  - Could build separate circuits to compute partial sums
    - Fast but very Big
  - Use bit adders that compute the value of each bit and connect them together to form N-bit adders

Half adders

For a one-bit add, the sum is just the XOR of the inputs, and the carry out is just the AND

$$S = A \oplus B$$

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Cells that compute this function are called *half-adders*

Full adders

To get a cell that we can compose into a multi-bit adder, we need to have a carry in input as well.

- These are called full adders

C_{in}	A	B	AB(G)	A+B(P)	A⊕B(P)	S	C_{out}
0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0
0	1	0	0	1	1	1	0
0	1	1	1	1	0	0	1
1	0	0	0	0	0	1	0
1	0	1	0	1	1	0	1
1	1	0	0	1	1	0	1
1	1	1	1	1	0	1	1

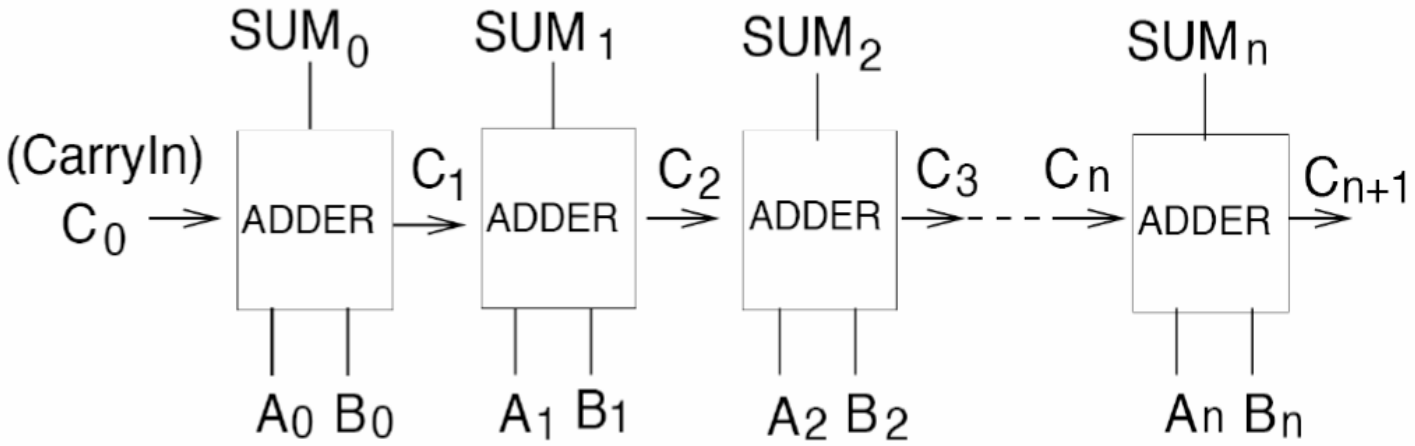
$$S = ABC_{in} + (A + B + C_{in})\overline{C_{out}}$$

$$C_{out} = AB + (A + B)C_{in}$$

Note that the carry-out can be *generated* when A and B are true, or *propagated* if A or B are true and C_{in} is true

Multi-Bit adders

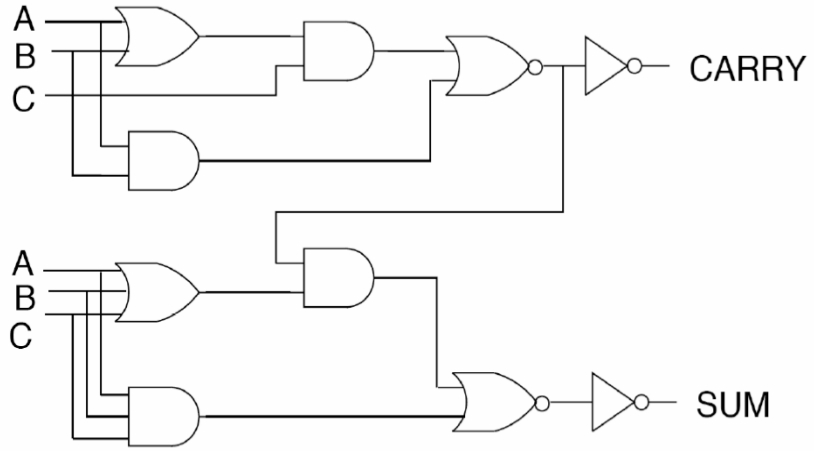
- By simply cascading full-adder blocks, we obtain an n-bit adder:

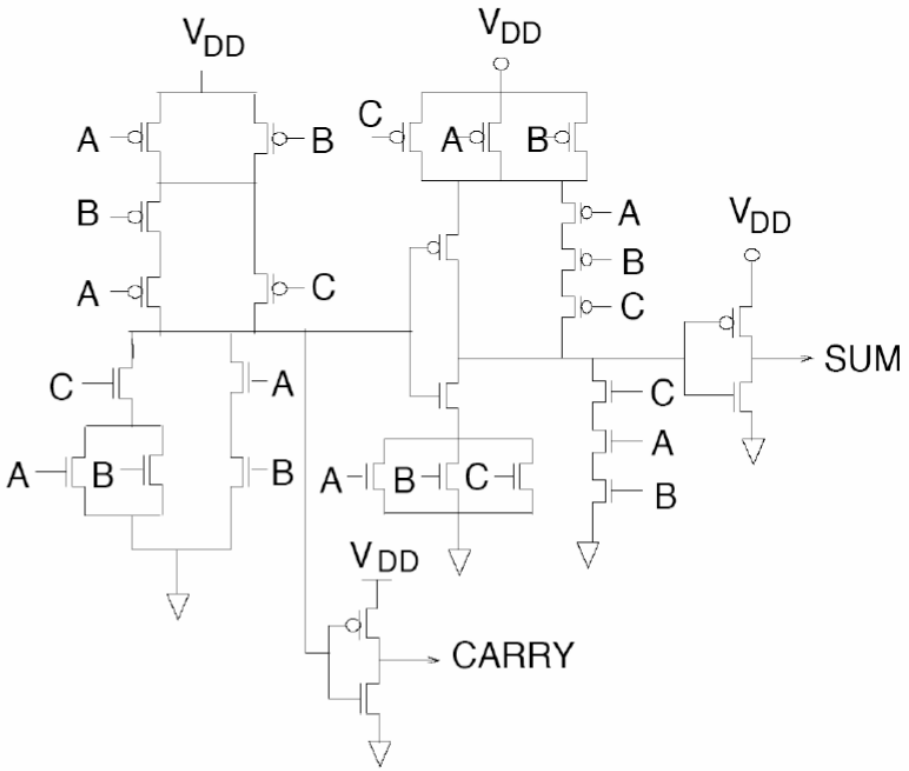


- The time required to complete an addition is limited by the carry bit *rippling* through the structure.
- Worst case, a carry will ripple all the way through from least significant bit to most significant bit.

Gate level implementation

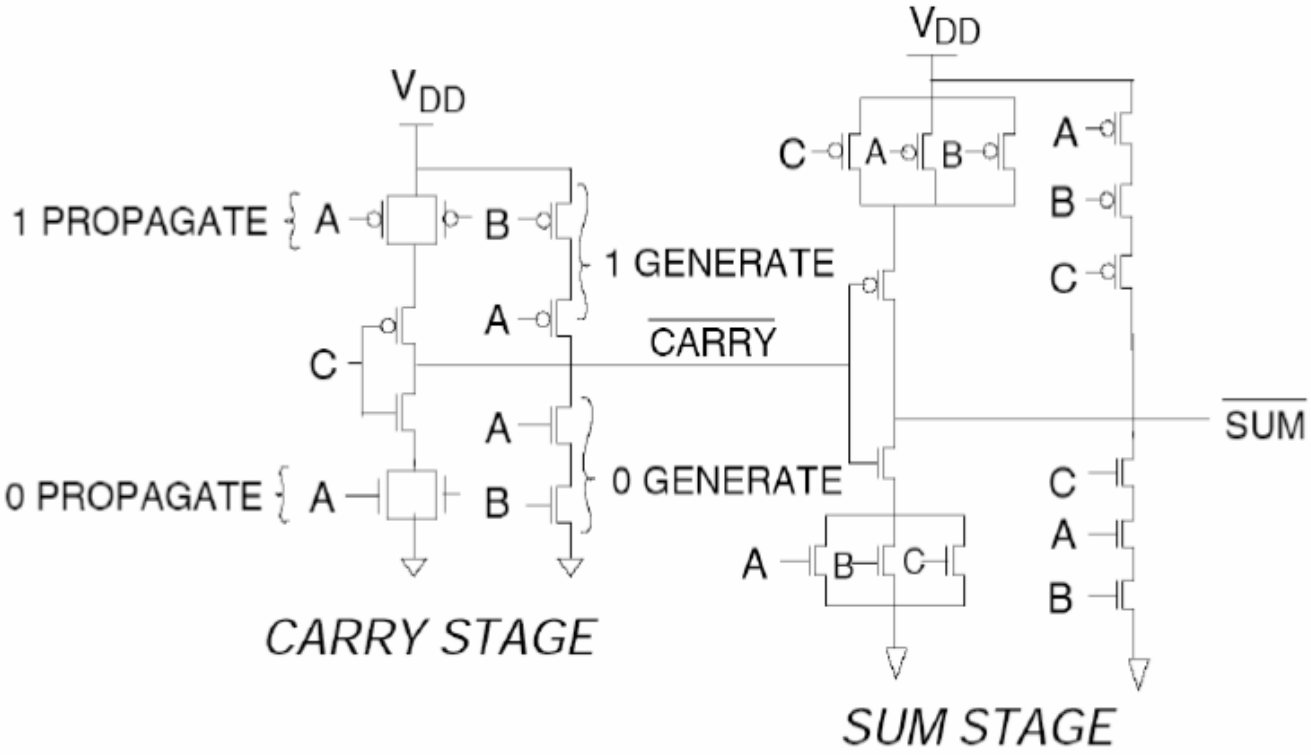
Much more worried about the delay from A,B, C_{in} --> C_{out} than A, B, C_{in} --> Sum





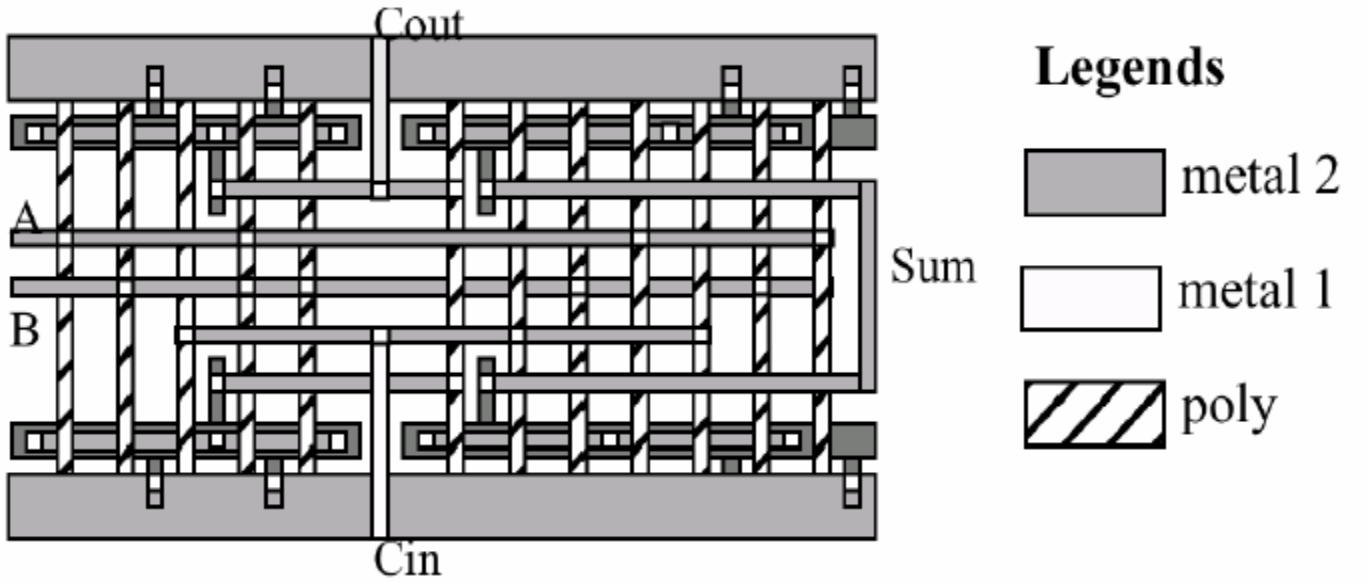
Better full adder (Mirror adder)

- We can improve the delay by using the following implementation:

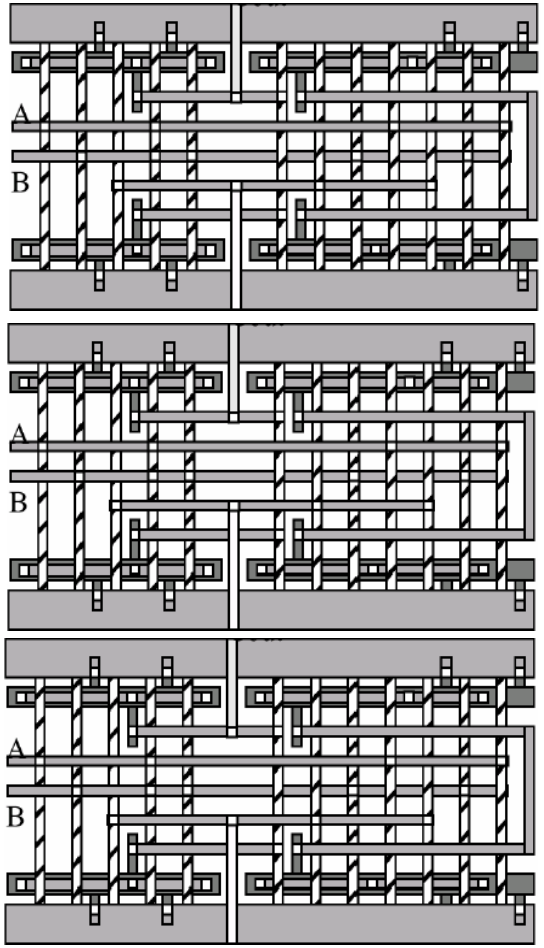


- Notice that the n- and p-networks are not duals!

Bit slice layout

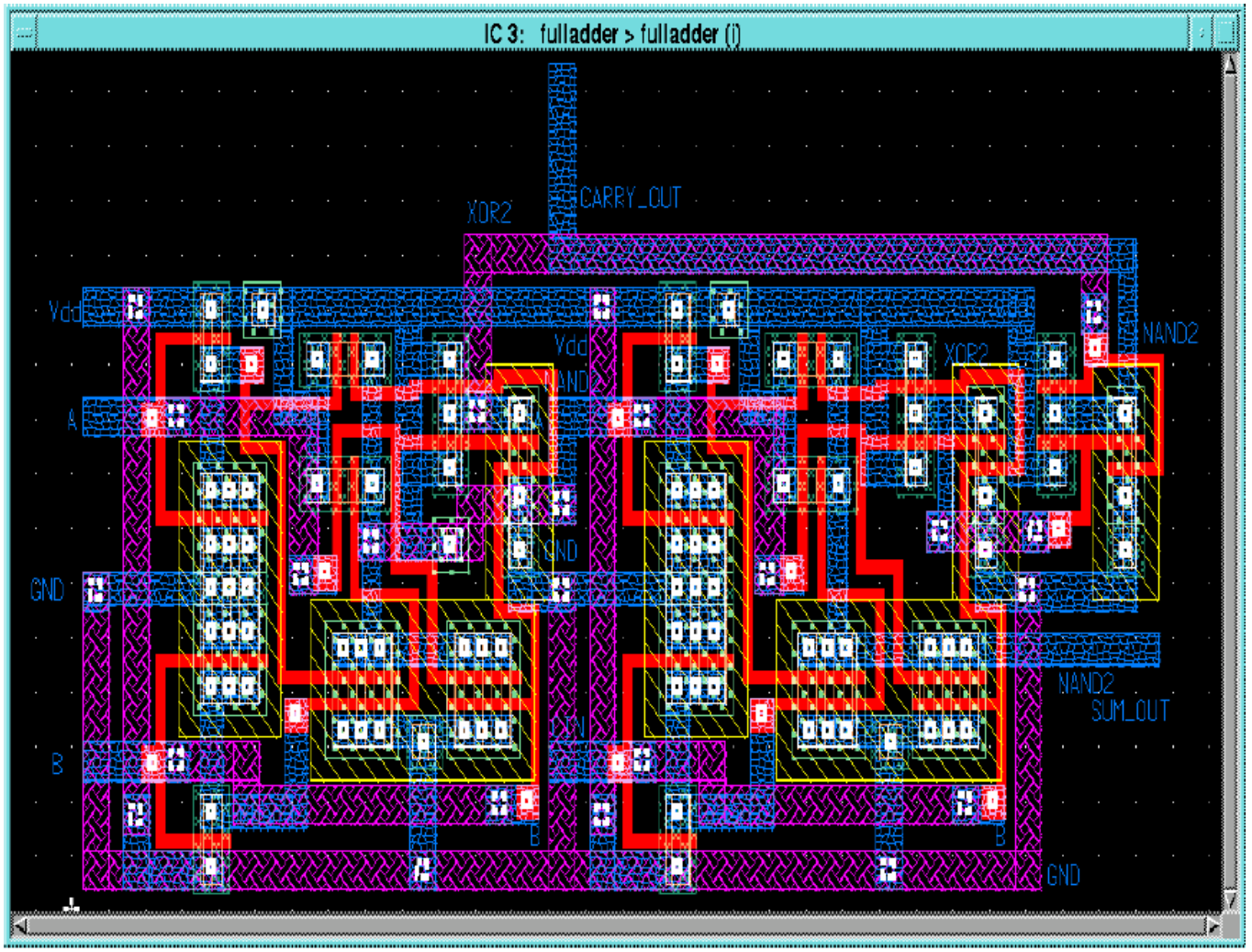


Bit slice layout



Multi-bit adders can be obtained by repeating automatically the cell

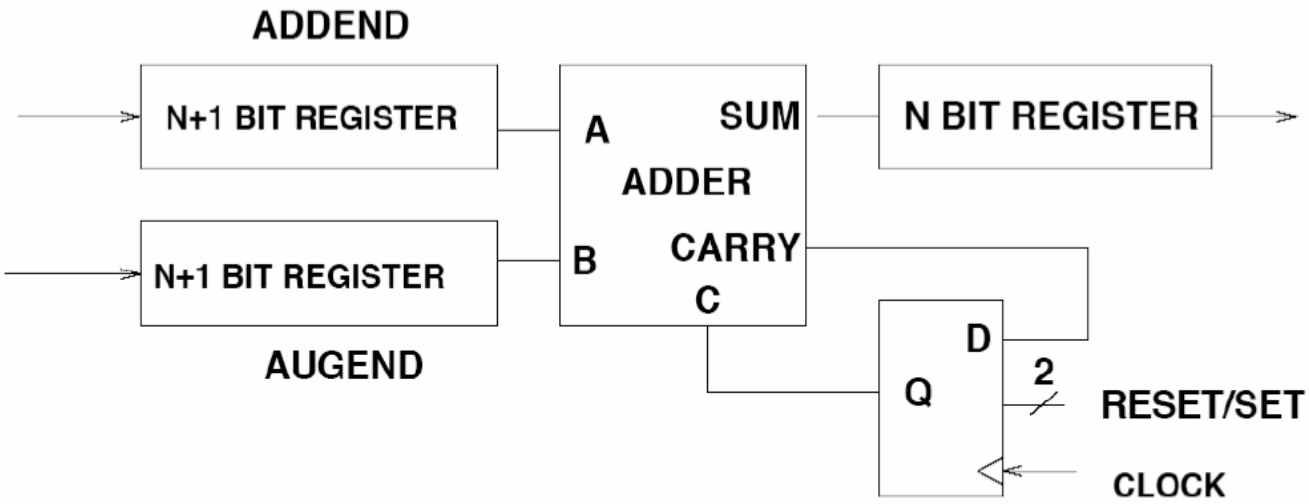
Bit-slice Full Adder



Eight-bit Adder

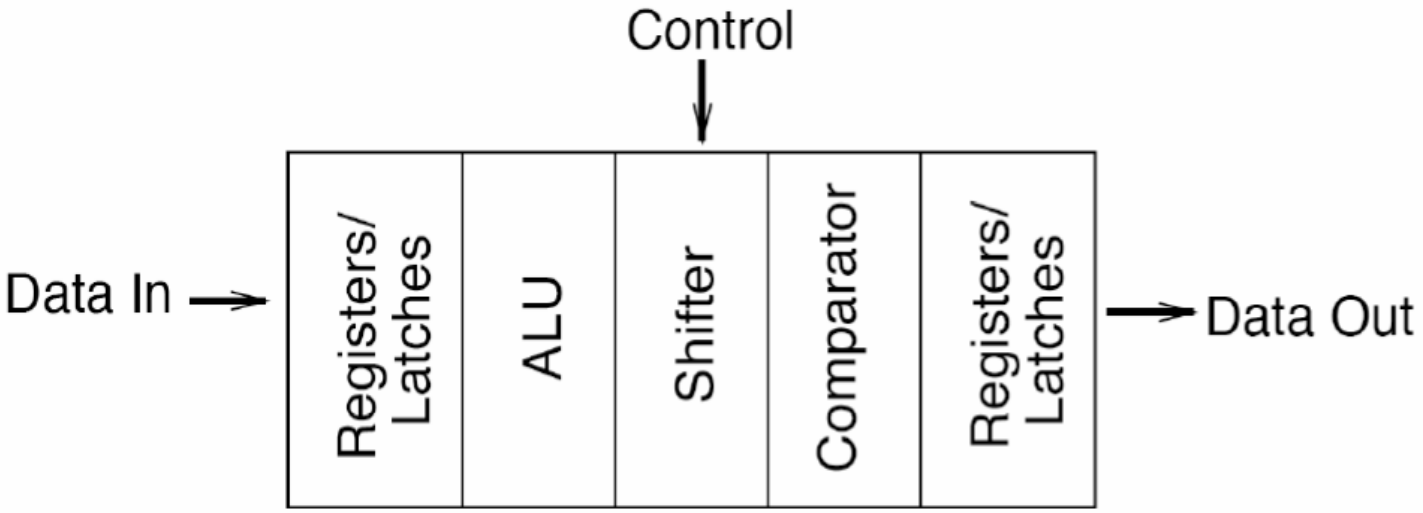
Serial Adder

- Ripple-carry adder has latency proportional to number of bits -- why not just re-use the same full adder for each bit?
  - In this case, want to equalize delay of sum and carry bits



Datapath organization

- Microprocessors (and many other systems) are made up of a regular datapath and irregular control logic.
- In a microprocessor, the datapath is a functional block so flexible that it becomes programmable



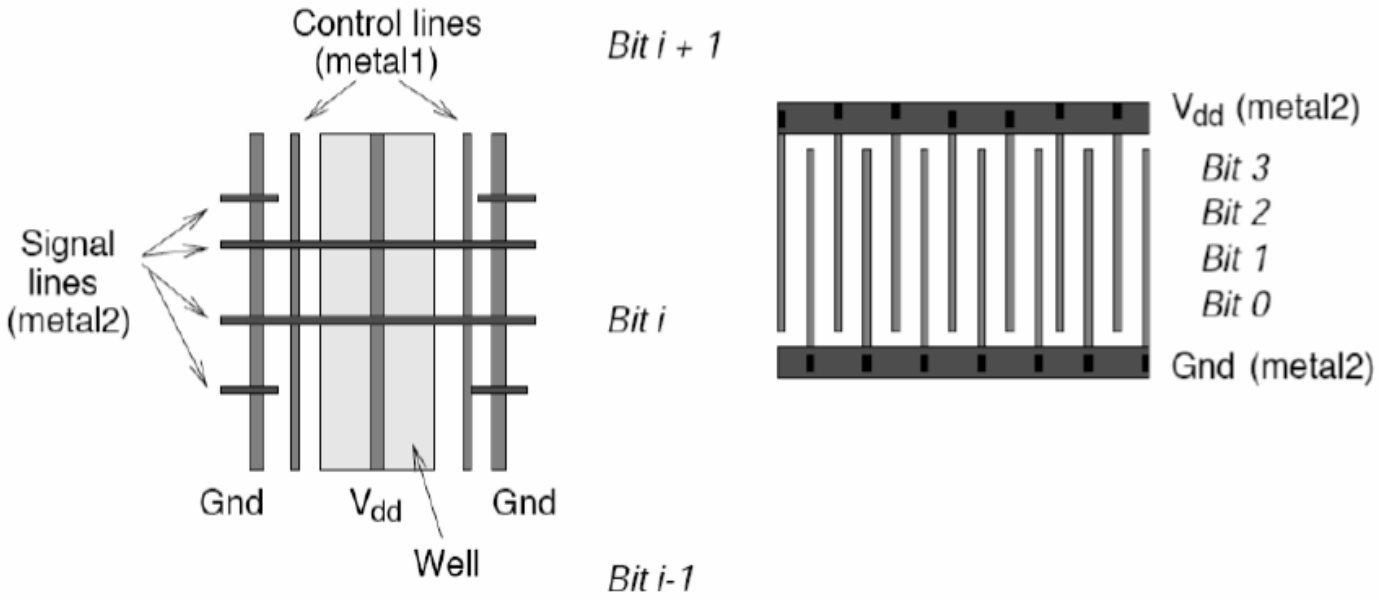
Datapath organization

Drive data horizontally through the datapath

Control signals arrive vertically

Allows designer to stack different function units together

Power, ground, clock typically run parallel to control



Substraction

Exploit 2's complement notation

- -X is generated by inverting each bit in X and adding one to the result
- Used because X + -X = 0 with this representation for -X

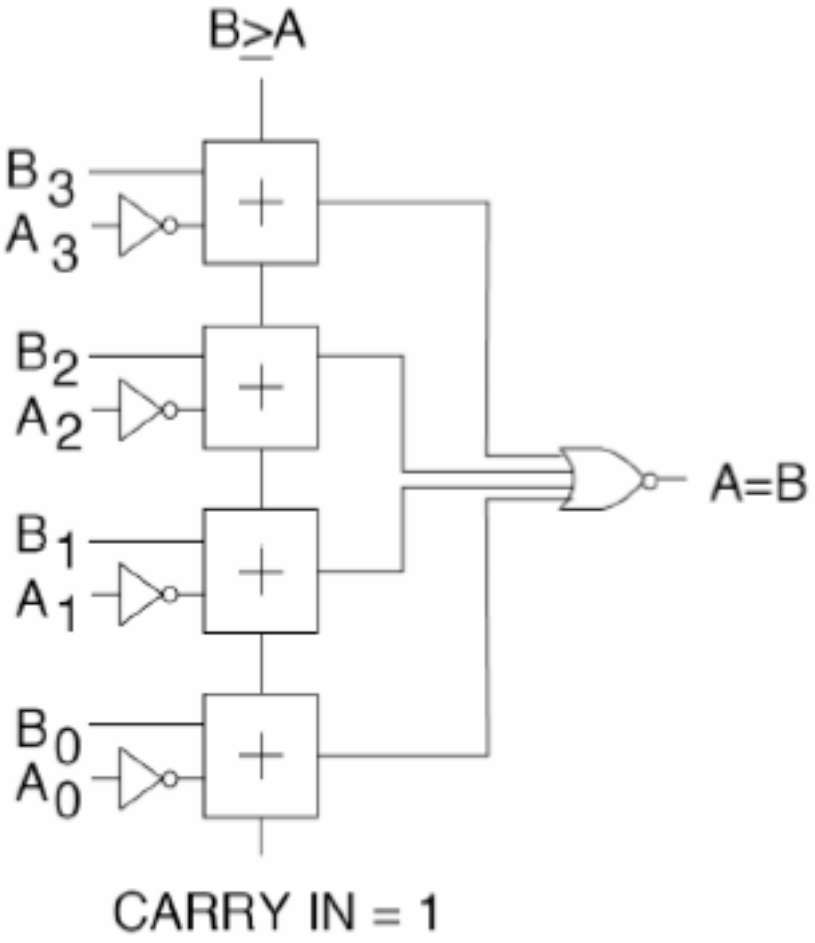
Subtractor is then adder with one input inverted and carry in = 1

- Can build adder/subtractor by XORing each bit of one input with subtract signal and using subtract as carry in

Otros Subsistemas

Comparison

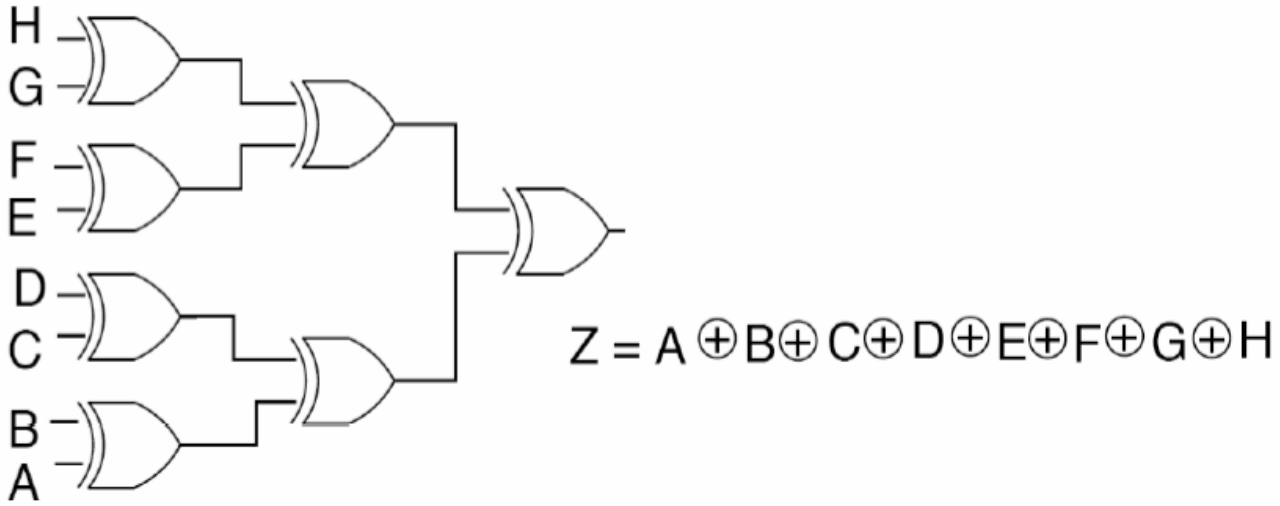
- Want to compare two numbers, A and B
- If A=B, A-B=0
- If B>=A, A-B will generate a carry out in the last bit (overflow)
- Comparison can be implemented using a substractor



Parity

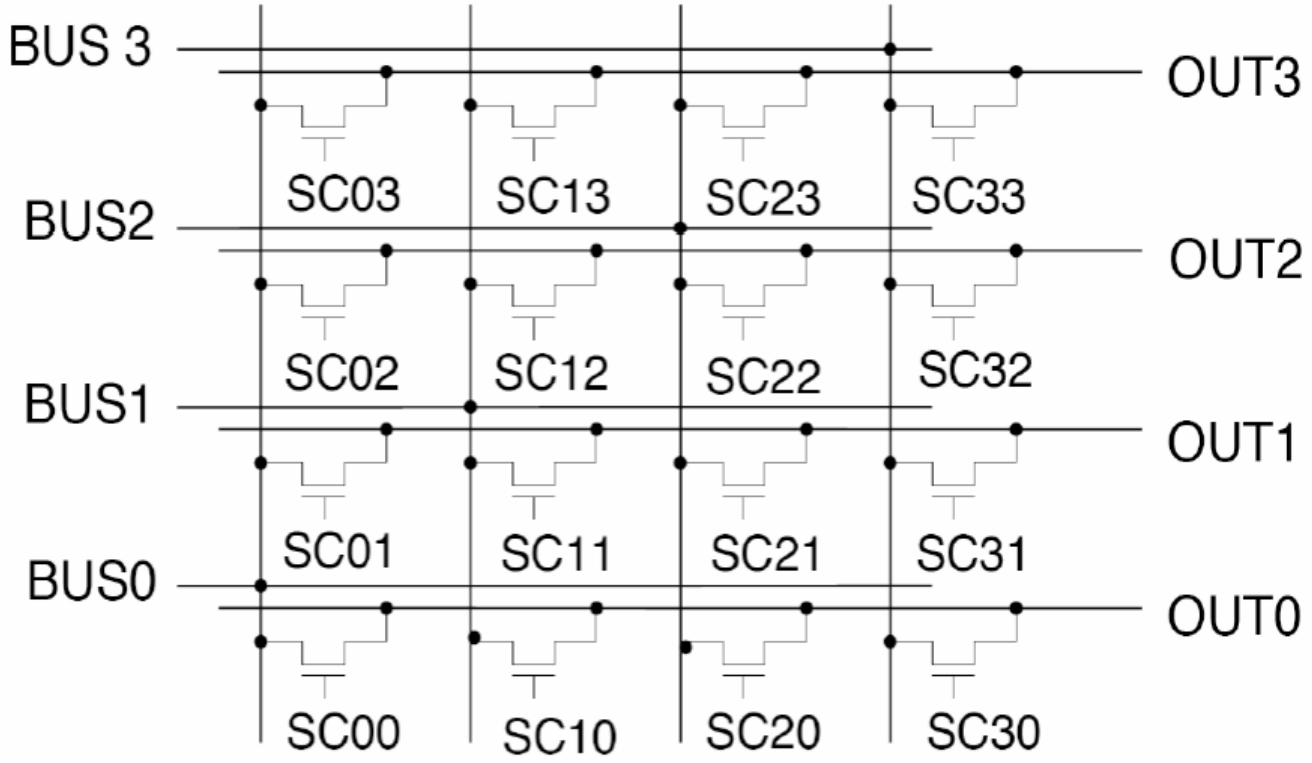
- A binary word is said to have even (odd) *parity* if it has an even (odd) number of 1 bits.
- A parity generator indicates the parity of a binary word.
- Functionally, parity generation is as simple as an exclusive-OR:

$$P_X = x_{n-1} \oplus x_{n-2} \oplus \cdots \oplus x_1 \oplus x_0$$



Crossbar as shifter

- Can use a crossbar network to implement a shifter (and any other permutation of input bits
  - Requires n^2 control inputs for n-bit data



Counters

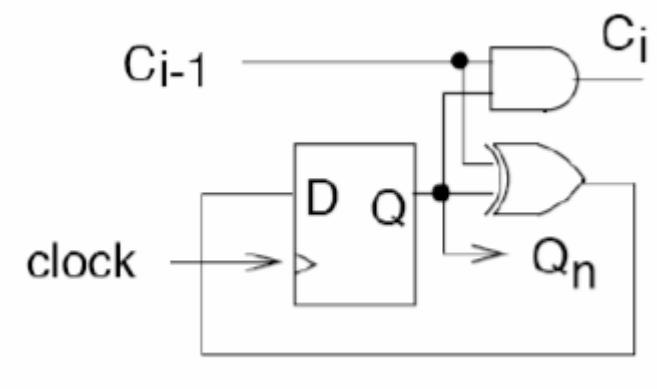
Counts the number of events on its input

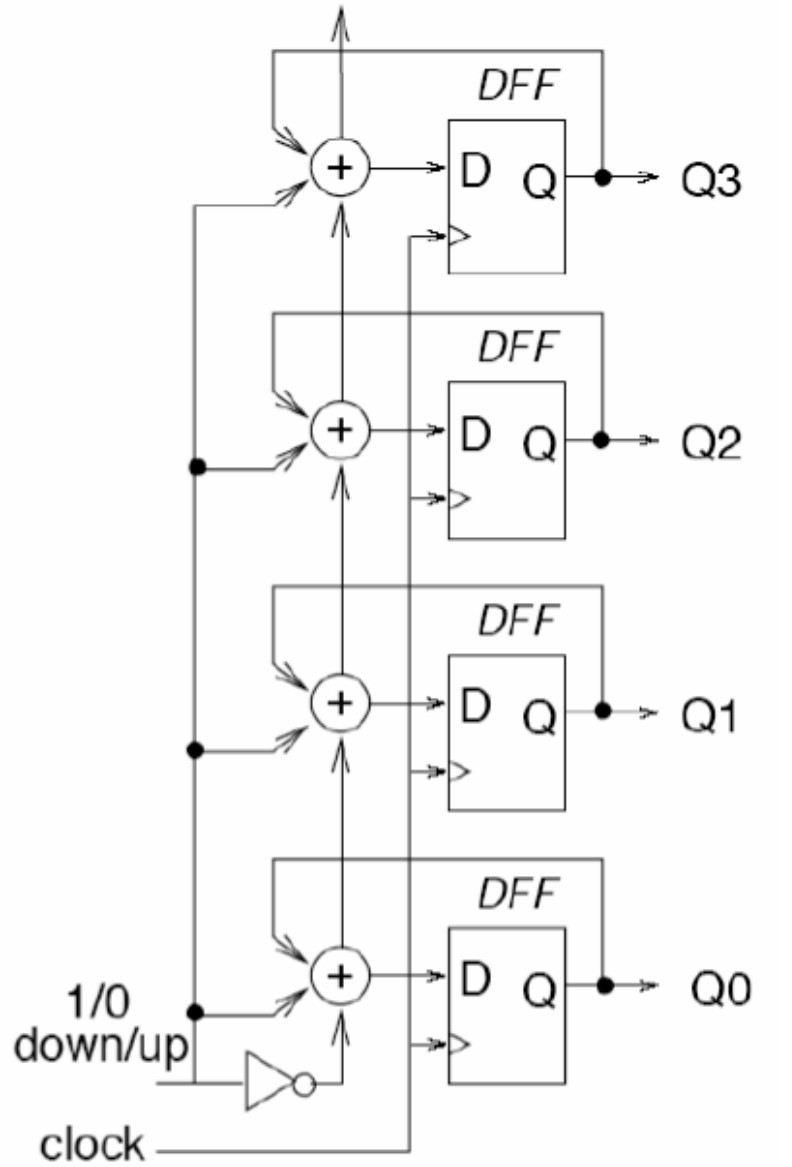
RULE # 1 of making a digital system work right: never gate clock signal !!

– Bits don't switch in synchrony
– Count ripples through the bits

Counter

- Adder with a constant input
- Selection used to define up (sum) or down (substraction)
- Synchronous – all signals change with clock





Multiplication

- Single-bit multiplication is just AND

a	b	$a \times b$
0	0	0
0	1	0
1	0	0
1	1	1

- Multi-bit multiplication can be done by combination of AND, ADD, shift

Performance question #1: How many bit multiplications do you do at the same time?

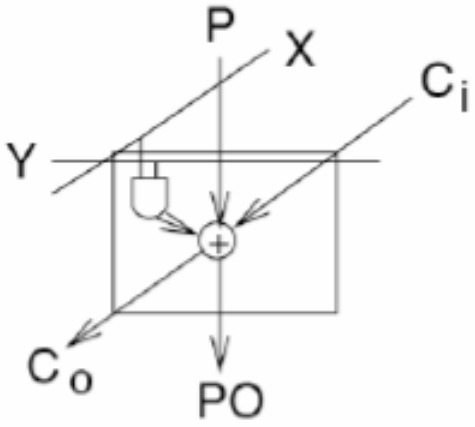
- 1 (serial)
- Between 1 and n^2 (serial-parallel)
- All of them (parallel)

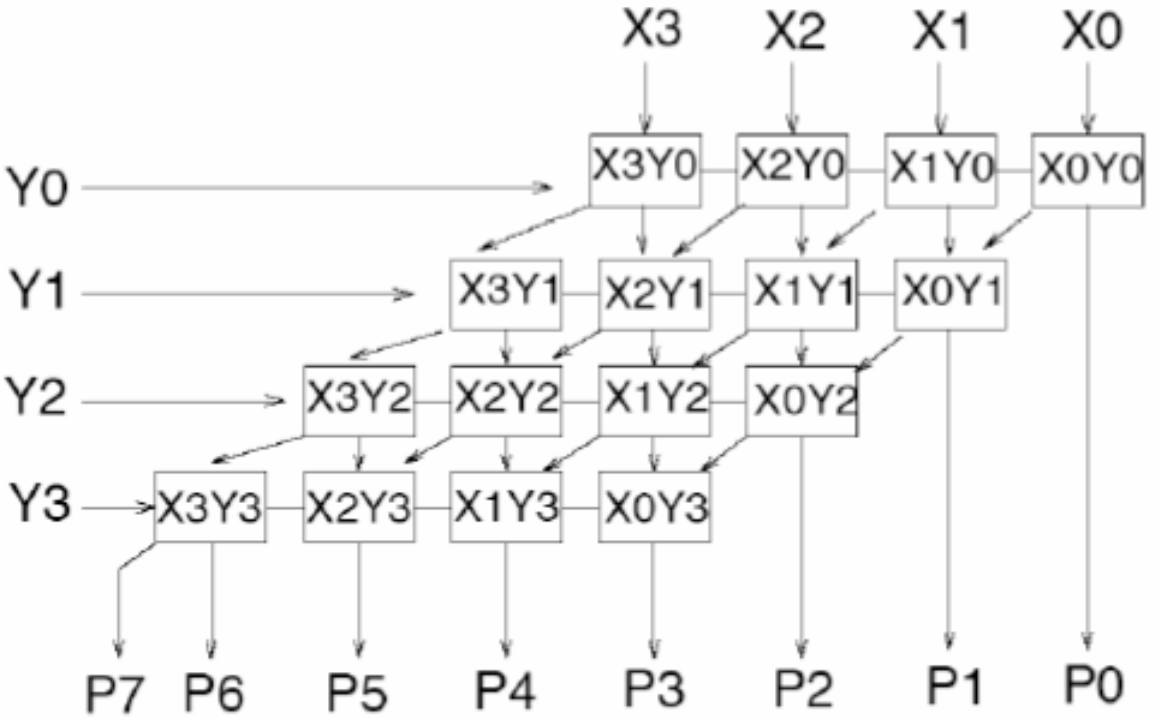
Trade-off between area and speed

Parallel Multiplication

Each *partial product* bit of the multiplication can be computed in parallel

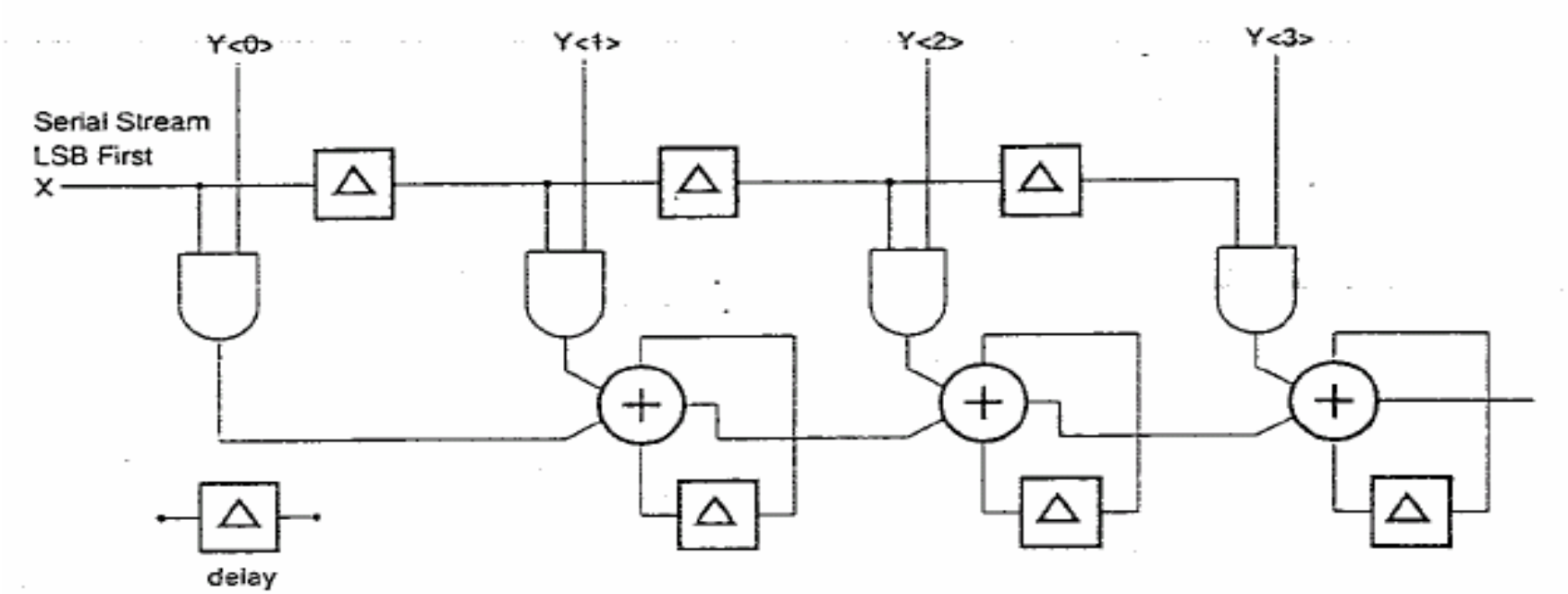
Then, perform *n* *n*-bit adds to sum the partial products



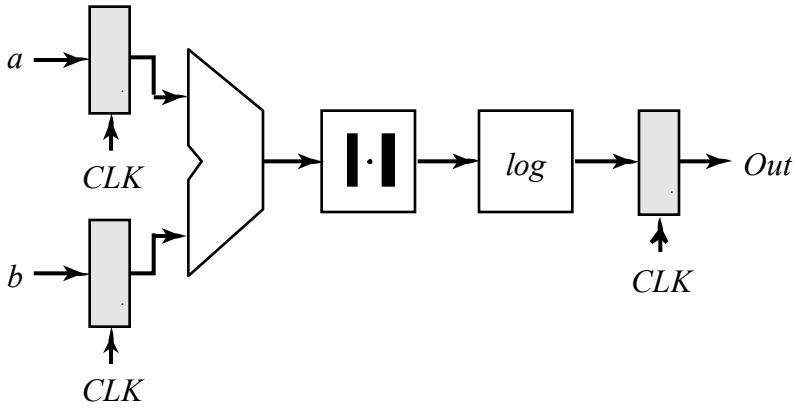


Serial-Parallel multiplier

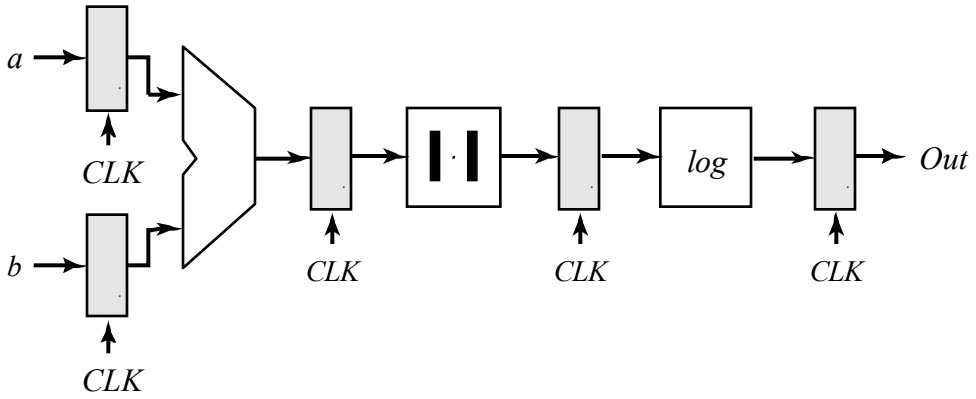
- Requires M+N clock cycles



Pipeline technique



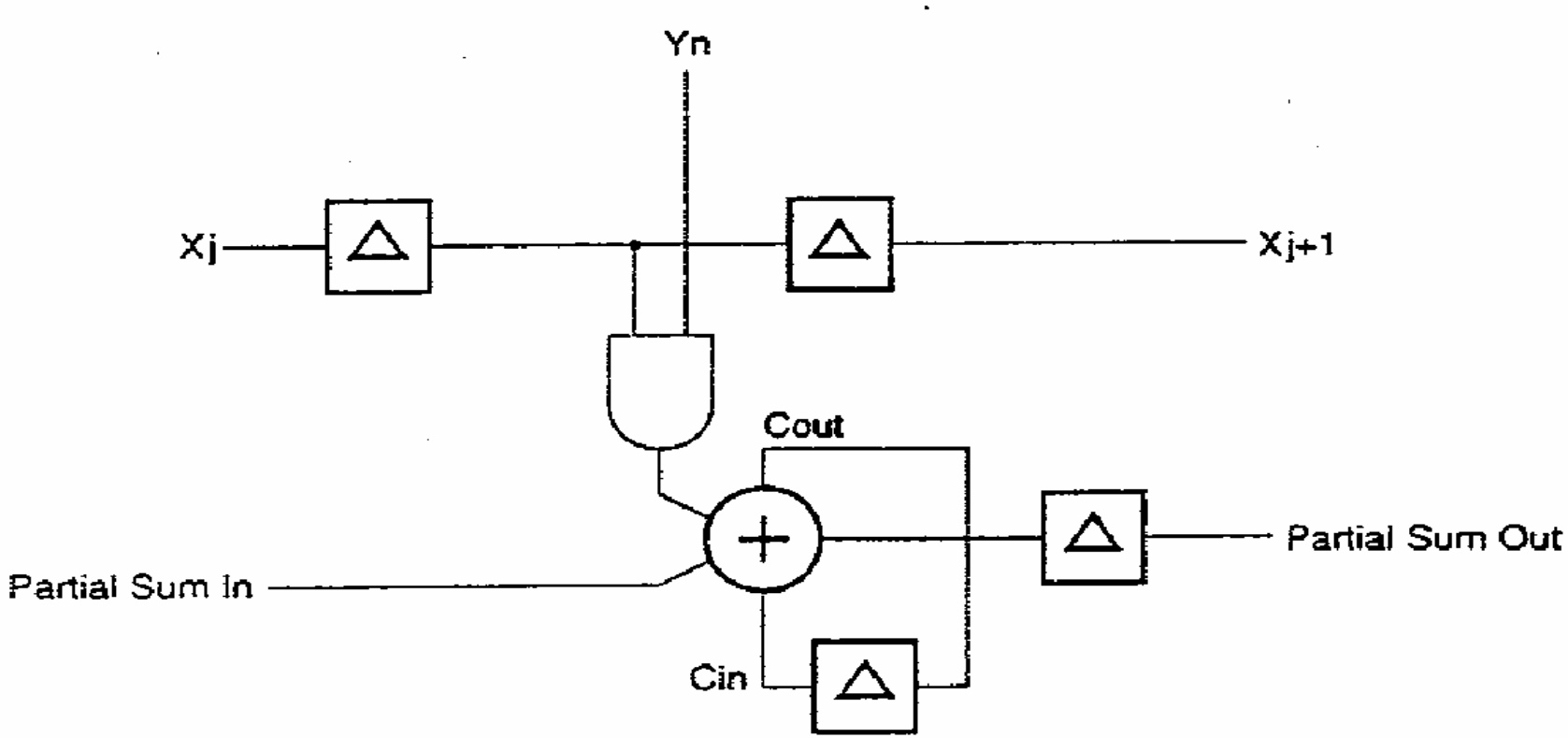
Reference



Pipelined

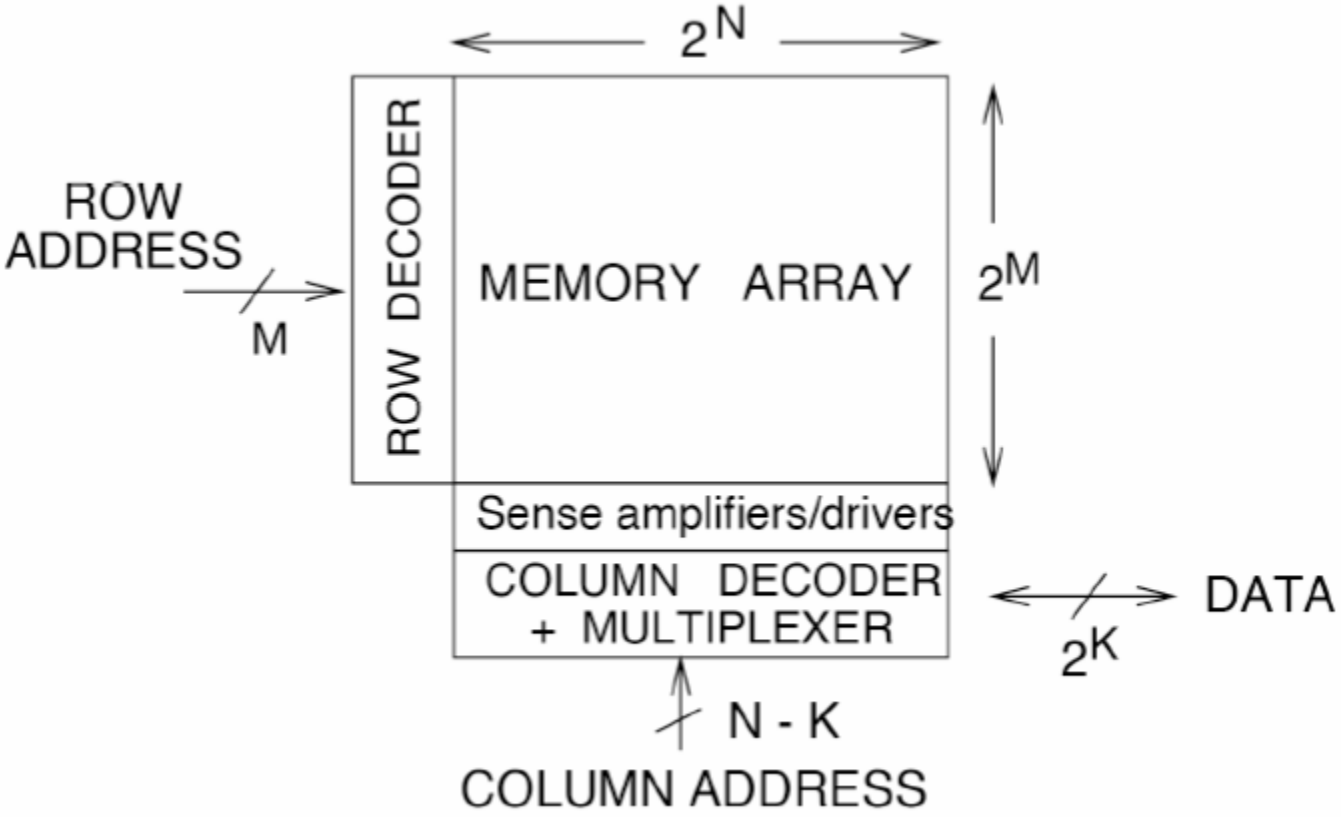
Clock Period	Adder	Absolute Value	Logarithm
1	$a_1 + b_1$		
2	$a_2 + b_2$	$\lvert a_1 + b_1 \rvert$	
3	$a_3 + b_3$	$\lvert a_2 + b_2 \rvert$	$\log(\lvert a_1 + b_1 \rvert)$
4	$a_4 + b_4$	$\lvert a_3 + b_3 \rvert$	$\log(\lvert a_2 + b_2 \rvert)$
5	$a_5 + b_5$	$\lvert a_4 + b_4 \rvert$	$\log(\lvert a_3 + b_3 \rvert)$

Pipelined Serial to Parallel



Memories

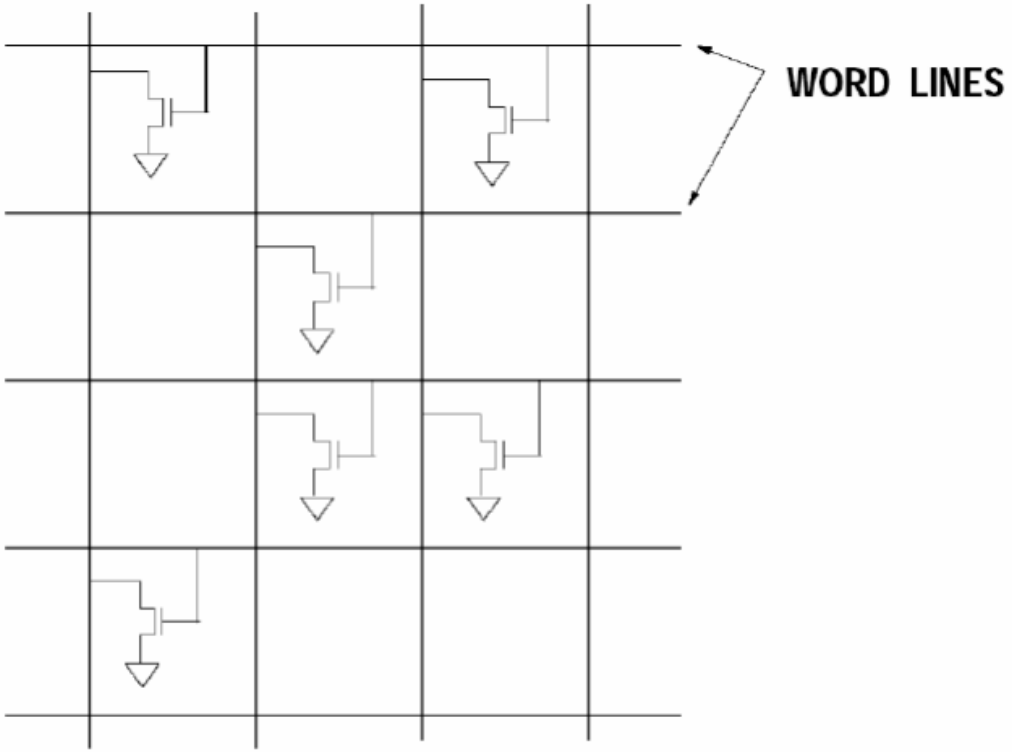
Organization



ROM

Simplest ROM cell is a single transistor

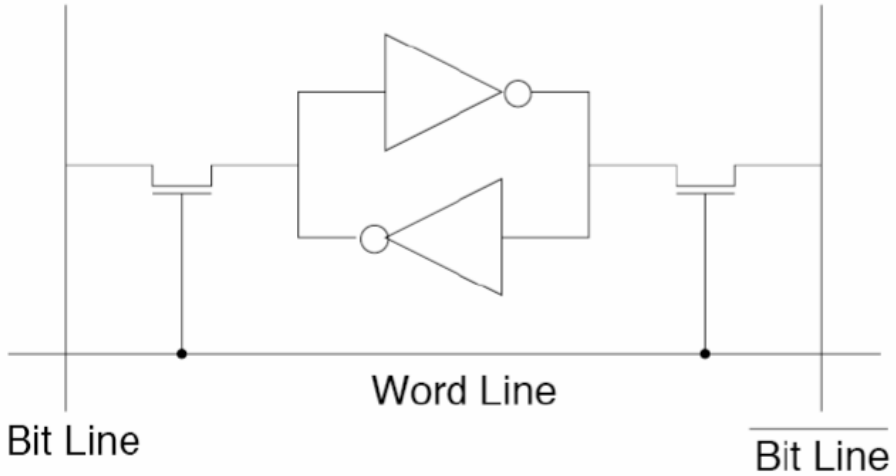
Precharge all bit lines high, assert word line. Transistors pull bits whose output = 0 low.

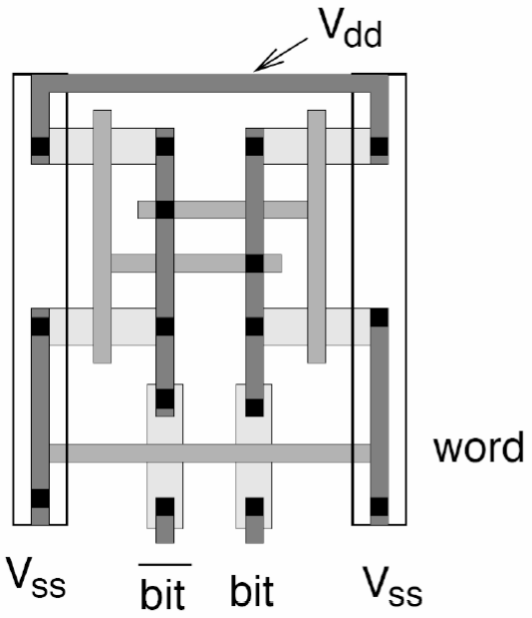


SRAM

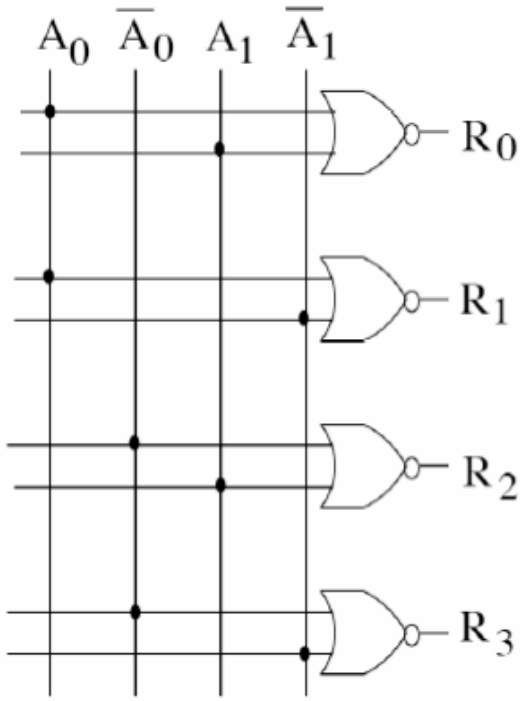
Cross-coupled inverters hold state

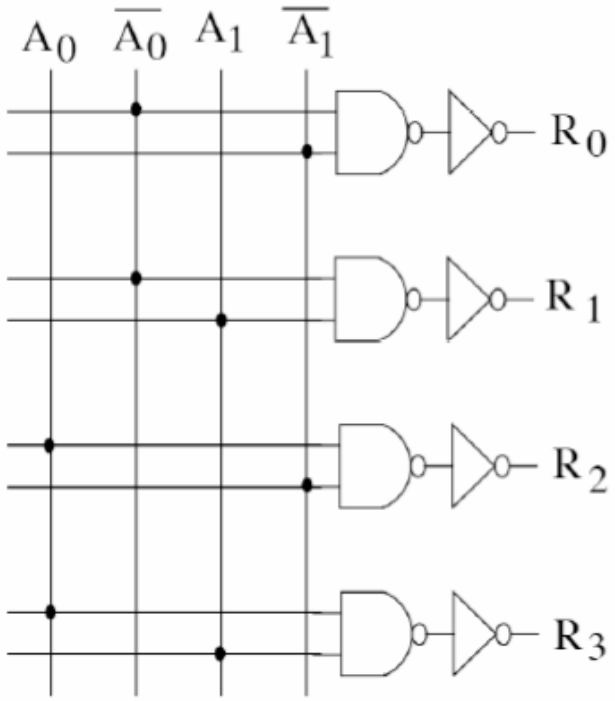
To write, drive data strongly on bit lines, such that you overwhelm the stored value





Static row decoder

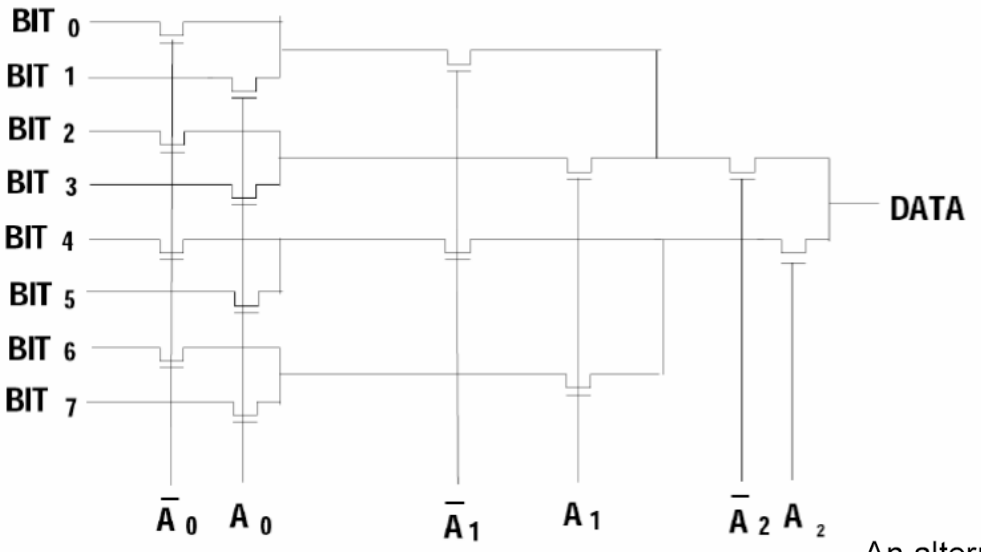




Due to the high capacitance of the word lines and the high number of address lines, these circuits have to be designed very carefully.

Row decoder

simple tree decoder selects one out of 8 bits:



An alternative design is to use a structure similar to the row decoder to generate a one-hot drive signal based on the address bits

- Put full pass-gate at the output of each bit line, control signal to pass-gates are the outputs of the decoder
- Gives full output swings
- Fewer transistors in series --> faster
- May take more area, especially when you consider the number of control wires