Why Testing?

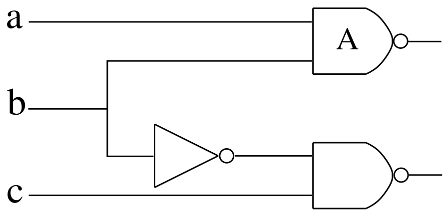
- Fault may occur anytime and anywhere.
  - Design
  - Manufacturing
  - Package
  - Field
- Larger circuits dramatically increase defect rates.
  - n: # of transistors; p: probability that a transistor is faulty
  - Yield: $Y = (1-p)^n$
  - $p = 10^{-6}, n = 10^6 \Rightarrow Y = 36.8\%$
  - $p = 10^{-6}, n = 4 \times 10^6 \Rightarrow Y = 1.8\%$!
- Faults are usually costly.

Physical Failures

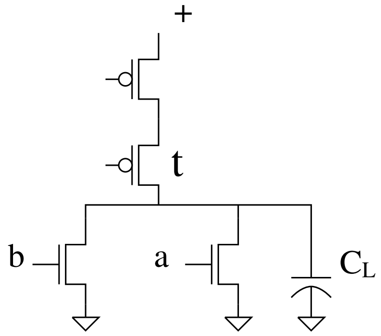
- Manufacturing defects
  - Silicon defects
  - Lithographic problems
  - processing problems, etc
- Wear-out
  - High current densities
  - Corrosion
  - Ion migration
  - "Hot" electron trapping, etc
- These result in faulty devices, opens, shorts, improper thresholds of devices.
- Failures are technology-dependent.

Fault Models

- **Fault model: Stuck-at,** stuck-open/closed, delay faults, etc
  - Possible locations of faults
  - I/O behavior produced by the faults
- Good news: If we have a fault model, we can test the network for every possible instantiation of that type of fault.
- Bad news: It is difficult to enumerate all types of manufacturing faults.



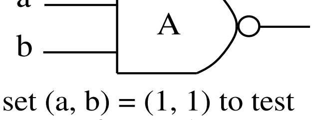
E.g., logic gate A is stuck at 0



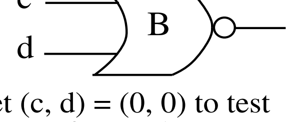
E.g., transistor t is stuck–open

Stuck-at-0/1 Faults

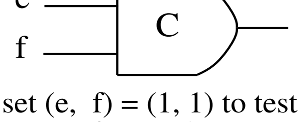
- **Stuck-at-0/1:** Logic gate output is always stuck at 0 or 1, independent of input values.
  - Most popular fault model: single stuck-at fault.
- Testing procedure
  - Set gate inputs
  - Observe gate output
  - Compare fault-free and observed gate outputs
- Key: Pick the gate inputs that should lead to 1 at the gate output to test stuck-at-0 fault. Similar trick works for stuck-at-1 faults.



set (a, b) = (1, 1) to test gate A for stuck–at–1



set (c, d) = (0, 0) to test gate B for stuck–at–0

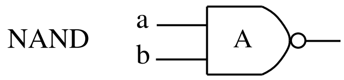


set (e, f) = (1, 1) to test gate C for stuck–at–0

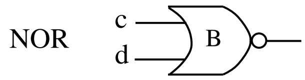
Stuck-at Faults in Gates

- Three ways to test NAND for stuck-at-0, only one way to test it for stuck-at-1.

- Three ways to test NOR for stuck-at-1, only one way to test it for stuck-at-0.



a	b	fault–free	S–A–0	S–A–1
0	0	1	0	1
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1



c	d	fault–free	S–A–0	S–A–1
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	0	1

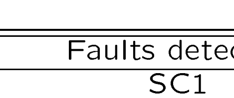
Fault Coverage

- **Fault coverage:** % of faults that can be detected

- Test vector: set of gate inputs applied for testing defects

  – Test vectors for combinational circuits can be done automatically by CAD programs, called **automatic test pattern generators (ATPGs)**.

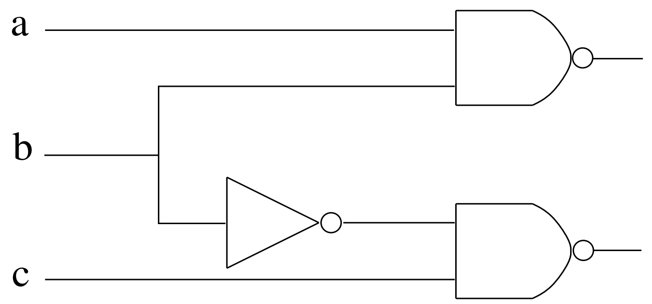
- 6 stuck-at faults for a 2-input AND gate: SA0 (Stuck-At-0), SA1, SB0, SB1, SC0, SC1



Test vector	Faults detected	Fault coverage
{(0, 0)}	SC1	16.7%
{(0, 1)}	SA1, SC1	33.3%
{(1, 1)}	SA0, SB0, SC0	50.0%
{(0, 0), (0, 1)}	SA1, SC1	33.3%
{(0, 0), (1, 1)}	SA0, SB0, SC0, SC1	66.7%
{(0, 1), (1, 1)}	SA0, SA1, SB0, SC0, SC1	83.3%
{(0, 0), (0, 1), (1, 1)}	SA0, SA1, SB0, SC0, SC1	83.3%
{(0, 1), (1, 0), (1, 1)}	ALL	100.0%

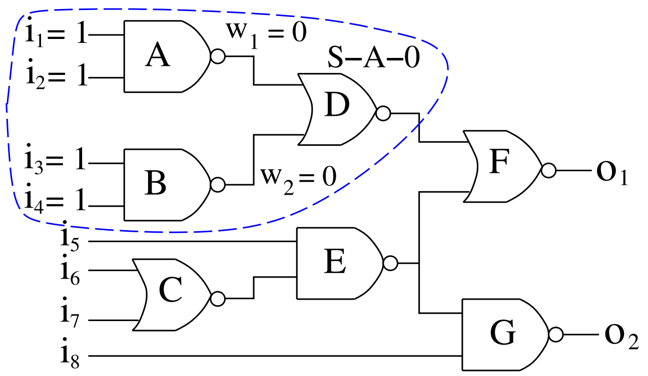
Testing Combinational Networks

- 100% fault coverage: test every gate for stuck-at-0 and stuck-at-1.
- Assume that there is only one faulty gate per network.
- Example
  - Set (a, b, c) = (0, 0, 0) to test both NANDs for stuck-at-0 simultaneously.
  - Cannot test both NANDs for stuck-at-1 simultaneously due to inverter. Must use two vectors (1, 1, x) and (x, 0, 1).
  - Must also test inverter.



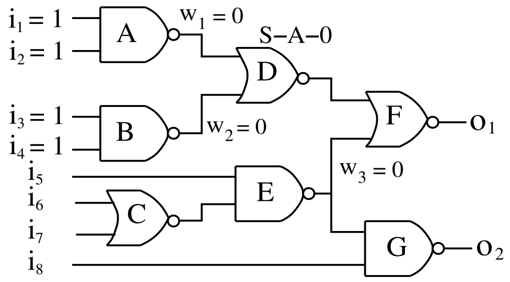
Testing Procedure

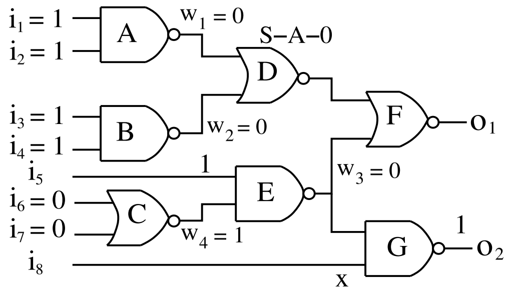
- Two tasks to testing
  - **Controlling** the gate's inputs by applying values to primary inputs.
  - **Observing** the gate's outputs by inferring its values from the values at primary outputs.
- Goal: Test gate D for stuck-at-0 fault.
- Step 1: Justify 0 values on gate inputs; work backward from gate to primary inputs.
  - $w_1 = 0 \Rightarrow i_1 = i_2 = 1$; $w_2 = 0 \Rightarrow i_3 = i_4 = 1$.



Testing Procedure (cont'd)

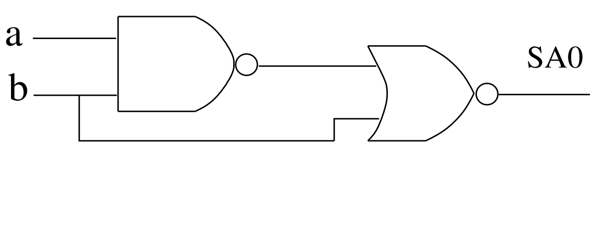
- Step 2: Observe the fault at a primary output
  - o_1 gives different values if D is true ($o_1 = 0$) or faulty ($o_1 = 1$) $\Rightarrow w_3 = 0$.
- Work forward and backward
  - $w_3 = 0 \Rightarrow i_5 = w_4 = 1 \Rightarrow i_6 = i_7 = 0$. $w_3 = 0 \Rightarrow o_2 = 1, i_8 =$ don't care.





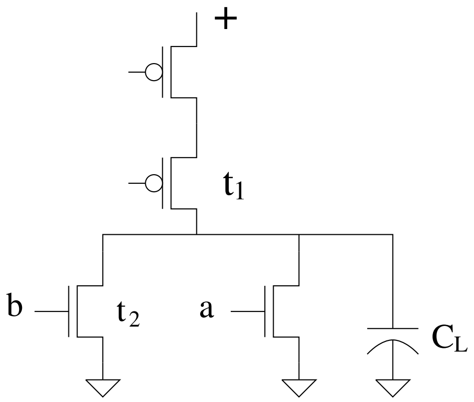
Fault Masking and Redundancy

- Redundant logic may mask faults.
  - Testing NOR for stuck-at-0 requires setting both inputs to 0.
  - Network topology ensures that one NOR input will always be 1.
  - Function reduces to 0: $F = \overline{\overline{ab} + b} = 0$.
- Minimize redundant logic for design-for-testability.



Stuck-at-open/closed Model

- Model transistor always ON/OFF.
- Stuck-open example
  - If t_1 is stuck open (switch cannot be closed), there can be no path from V_{DD} to output capacitance.
  - Testing requires two cycles
    - * Must discharge capacitor
    - * Try to operate t_1 to see if capacitor can be charged.



Delay Fault

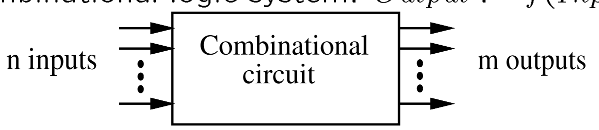
- Delay falls outside acceptable limits.
  - Gate delay fault: assume that all delay errors are lumped into one gate.
  - Path delay fault: model delay errors along path through network.
- Delay problems reduce yield
  - Performance problems
  - Functional problems in some types of circuits.

Chapter 5: Sequential Circuits & CAD Systems

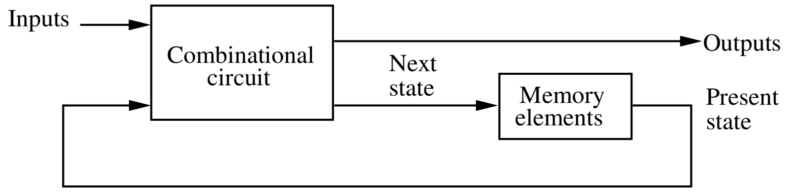
- **Reading: Wolf**
  - Chapter 5.1–5.2: Latches and flip-flops
  - Chapter 5.3: Sequential systems and clocking
  - Chapter 5.4: Sequential system design
  - Chapter 10.8: Test generation
  - Chapter 10.9: Sequential machine optimizations

Sequential Systems

- **No state:** Combinational logic system: $Output := f(Inputs)$.

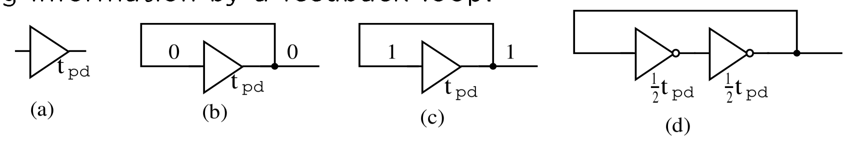


- **State present:** Sequential logic system: $Output := f([Inputs,]State)$.
  - **Synchronous** sequential system: State updated at **discrete time**.
  - **Asynchronous** sequential system: State updated at **any time**.
  - Two components: **combinational circuit** and **memory elements**.
  - Signals: inputs, outputs, **state**.

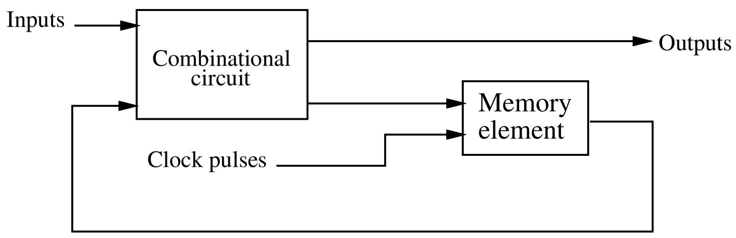


Sequential Circuits

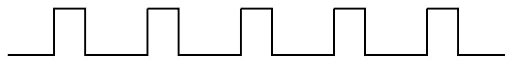
- Storing information by a feedback loop:



- **Synchronous clocked sequential circuit:** Use clock pulses as inputs to memory elements.



(a) Block diagram



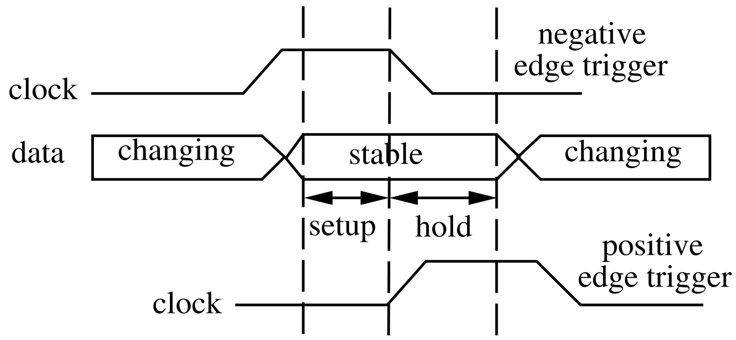
(b) Timing diagram of clock pulses

Timing Parameters

- **Clock edge:** rising or falling transition.

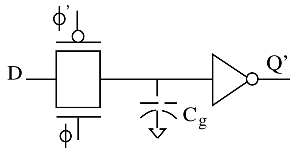
- **Setup time:** time before clock during which data input must be stable.

- **Hold time:** time after clock event for which data input must remain stable.

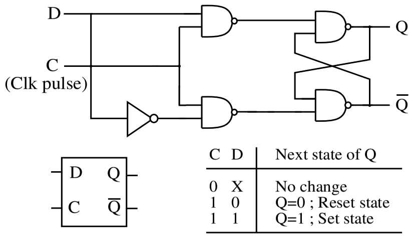


Memory Elements

- In CMOS, memory is created by
  - Capacitance (dynamic)
  - Feedback (static)
- Types of memory elements
  - **Latch:** transparent when internal memory is being set from the input.
  - **Flip-flop:** not transparent, reading input and changing output are separate events.

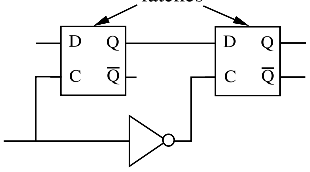


dynamic D–type latch



C	D	Next state of Q
0	X	No change
1	0	Q=0 ; Reset state
1	1	Q=1 ; Set state

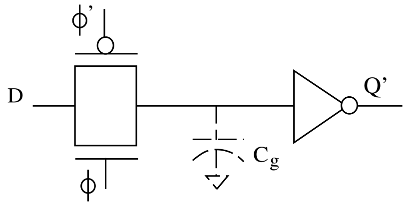
D–type latch



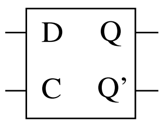
master–slave flip–flop

Dynamic Latch

- Use complementary transmission gate to ensure that storage node is always strongly driven.
- Latch is transparent when transmission gate is closed.
- Storage capacitance comes primarily from inverter gate capacitance.
- $\phi = 1$: transmission gate is ON, inverter output follows D input.
- $\phi = 0$: transmission gate is OFF, inverter output is determined by storage node.
- Setup and hold times determined by transmission gate must ensure that value stored on transmission gate is solid.
- Must handle the problem for stored charge leakage. Load latch every cycle?



dynamic D–type latch



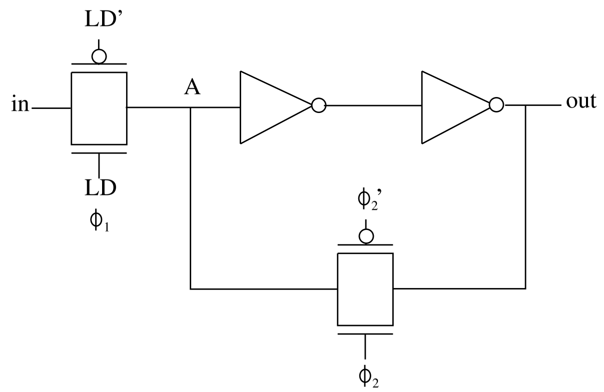
Q(t+1) = D

Q	D	Q(t+1)	Comment
0	0	0	No change
0	1	1	Set Q
1	0	0	Clear Q
1	1	1	Set Q

excitation table

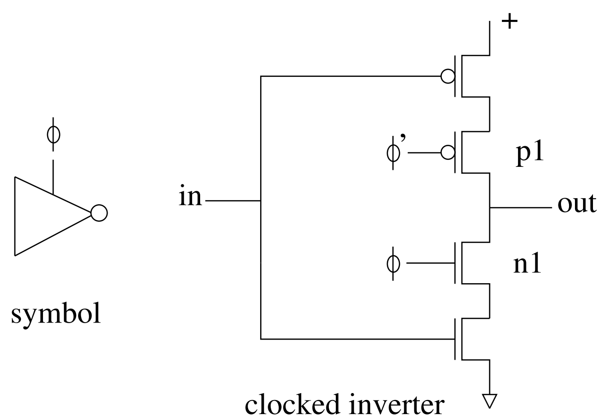
Recirculating Latch

- The value stored on the capacitor of dynamic latch leaks away over time.
- Some latches are static on one phase (**quasi-static**): use feedback to restore value.
- A **recirculating latch** supplies current to constantly refresh the stored value.
  - Load on one phase: during ϕ_1, LD is set to 1 to load new data.
  - Activate feedback on the other phase: during ϕ_2, the latch can be repeatedly refreshed.



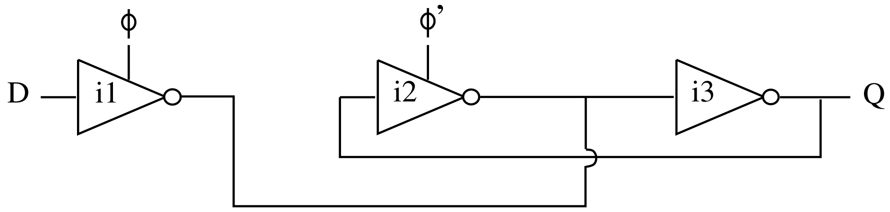
Clocked Inverter

- clocked inverter
  - $\phi = 0$: both clocked transistors are OFF, output is floating.
  - $\phi = 1$: both clocked transistors are ON, acts as an inverter to drive output.

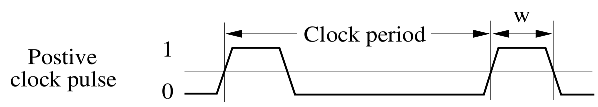


Clocked Inverter Latch

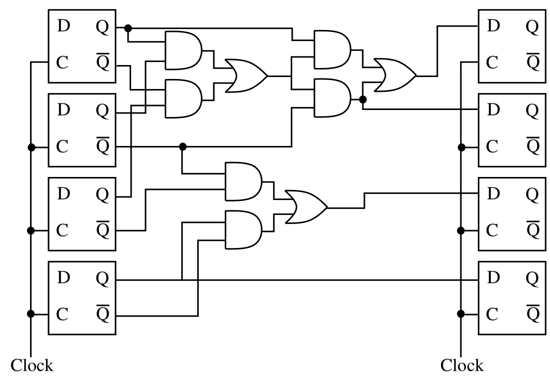
- clocked inverter latch
  - $\phi = 0$: i_1 is OFF, i_2, i_3 form feedback circuit.
  - $\phi = 1$: i_2 is OFF, breaking feedback; i_1 is ON, driving i_3 and output.
  - Latch is transparent when $\phi = 1$.



Problem with Latch

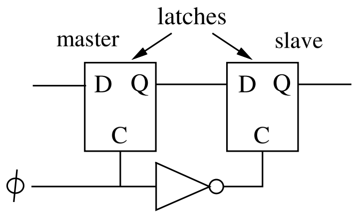


- If the clock **period** is **too short**, some data changes will not propagate through the circuit.
- If the clock **pulse width** is **too long**, some data will propagate through the second rank of flip-flops (shoot through).
- Solutions to the clocking problem? **Master-slave flip-flop, edge triggering.**

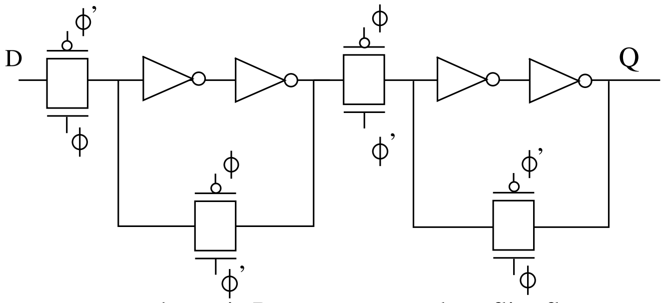


Flip-flops

- Major varieties
  - Master-slave
  - Edge-triggered
- Master-slave operation
  - $\phi = 1$: master latch is enabled, loading value from input; slave latch is disabled, maintaining old output value.
  - $\phi = 0$: master latch is disabled; slave latch is enabled, master's stable value is passed to output Q.



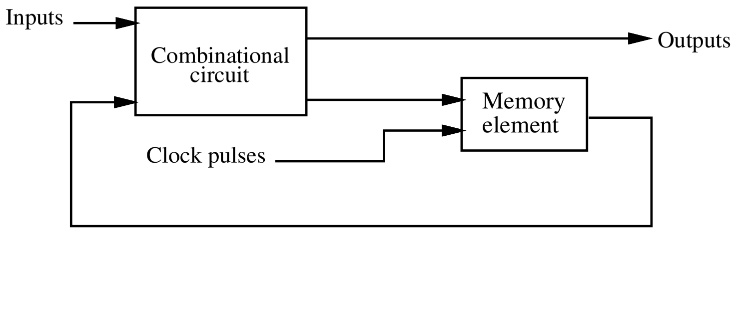
master–slave flip–flop



dynamic D–type master–slave flip–flop

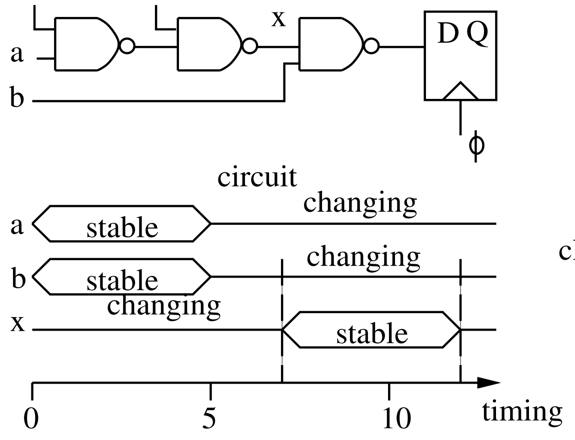
Constraints on Sequential System

- No combinational cycles (may be relaxed).
- All components must have bounded delay.

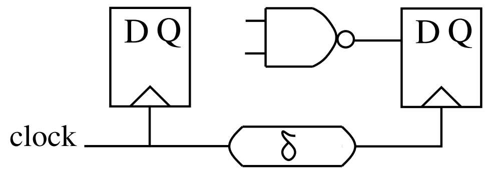


Signal and Clock Skew

- **Signal skew:** data signals must obey setup and hold time constraints and avoid signal skew.
- **Clock skew:** difference in the minimum and the maximum arrival times of the clock.



data signal skew

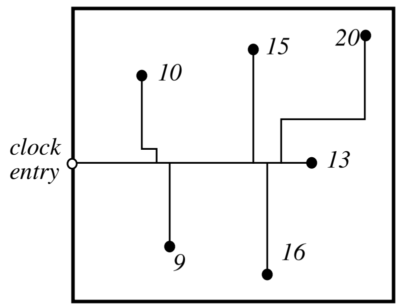


delay: clock skew

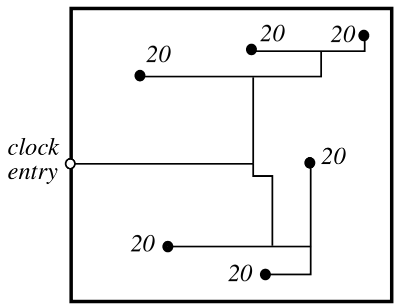
Clock Routing Problem

- Routing clock nets such that
  1. clock skew is minimized (or clock signals arrive simultaneously)
  2. clock delay is minimized

  * Other issues: total wirelength (area), power consumption.
- Approaches
  1. H-tree
  2. Zero-skew clock tree
  3. Wire sizing and buffering

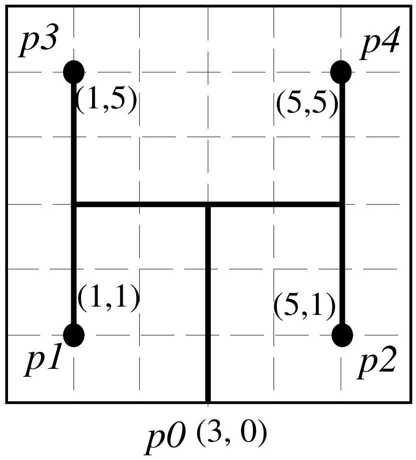


clock skew = 20 – 9 =11

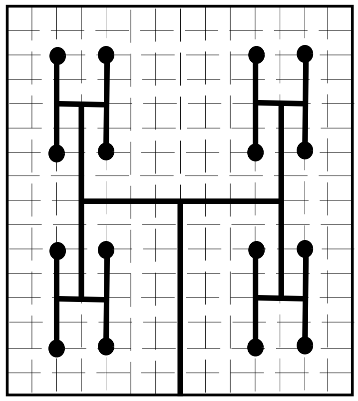


clock skew = 0

H Trees



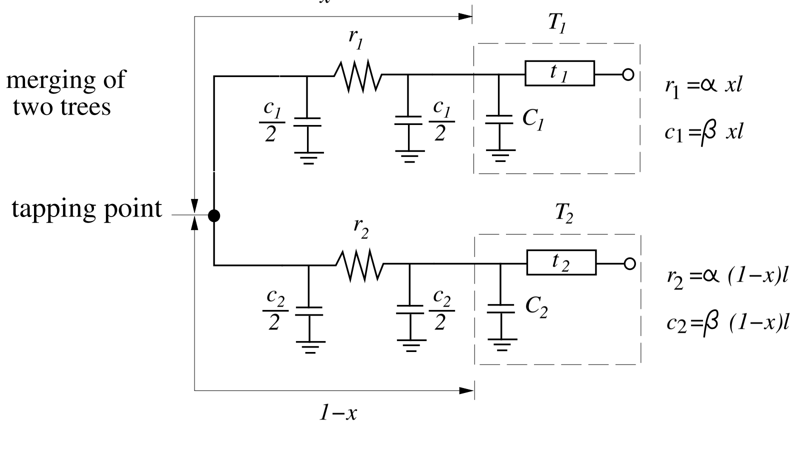
H−tree over 4 points



H−tree over 16 points

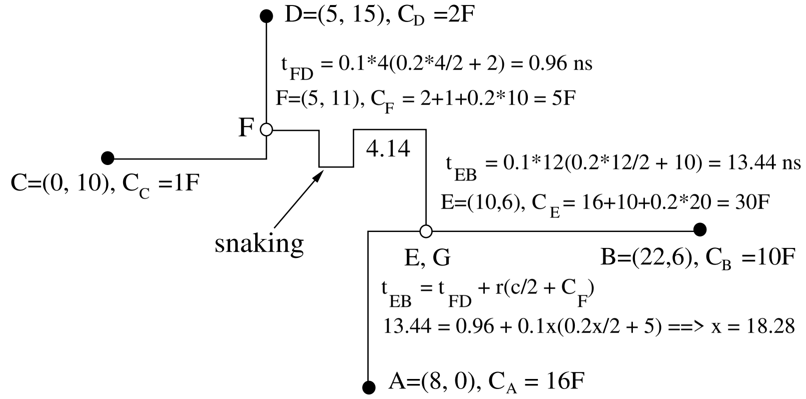
Zero-Skew Clock Tree

- To ensure the delay from the **tapping point** to leaf nodes of subtrees T_1 and T_2 being equal, it requires that
$$r_1(\frac{c_1}{2} + C_1) + t_1 = r_2(\frac{c_2}{2} + C_2) + t_2.$$
- Solving the above equation, we have
$$x = \frac{(t_2 - t_1) + \alpha l \left(C_2 + \frac{\beta l}{2}\right)}{\alpha l(\beta l + C_1 + C_2)},$$
where α and β are the per unit values of resistance and capacitance, l the length of the interconnecting wire, $r_1 = \alpha x l, c_1 = \beta x l,\ r_2 = \alpha(1 - x)l, c_2 = \beta(1 - x)l$.



Zero-Skew Computation

- **Balance delays:** $r_1(\frac{c_1}{2} + C_1) + t_1 = r_2(\frac{c_2}{2} + C_2) + t_2$.
- **Compute tapping points:** $x = \frac{(t_2 - t_1) + \alpha l \left(C_2 + \frac{\beta l}{2}\right)}{\alpha l(\beta l + C_1 + C_2)}$, α (β): per unit values of resistance (capacitance); l: length of the wire; $r_1 = \alpha x l, c_1 = \beta x l$; $r_2 = \alpha(1-x)l, c_2 = \beta(1-x)l$.
- If $x \notin [0, 1]$, we need **snaking** to find the tapping point.
- Exp: $\alpha = 0.1\Omega/unit, \beta = 0.2F/unit$. (Find tapping points E for A and B, F for C and D, and G for E and F.)



Wire Sizing & Buffering

- Sizing wires and inserting/sizing buffers for clock-skew and -delay minimization.
  - Can consider other objectives simultaneously: power, area, timing, clock-skew sensitivity, crosstalk, crosstalk sensitivity
  - Discrete wire/buffer sizes: dynamic programming
  - Continuous wire/buffer sizes: mathematical programming (e.g., Lagrangian relaxation)