Chapter 7. Designing Sequential Logic Circuits

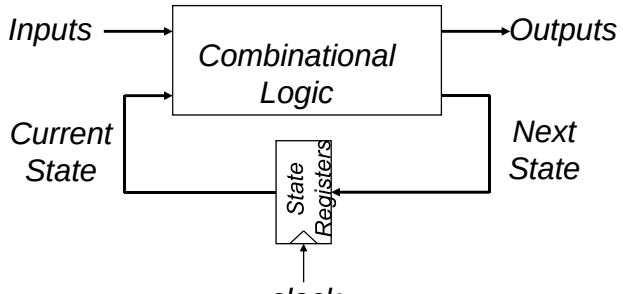
[Adapted from Rabaey's *Digital Integrated Circuits*, ©2002, J. Rabaey et al.]

Sequential Logic

- Combinational circuits: output is only a function of current input values.
- Sequential circuits: outputs depend on not only current input values, but also preceding input values. It remembers the history of the system.

Sequential Logic

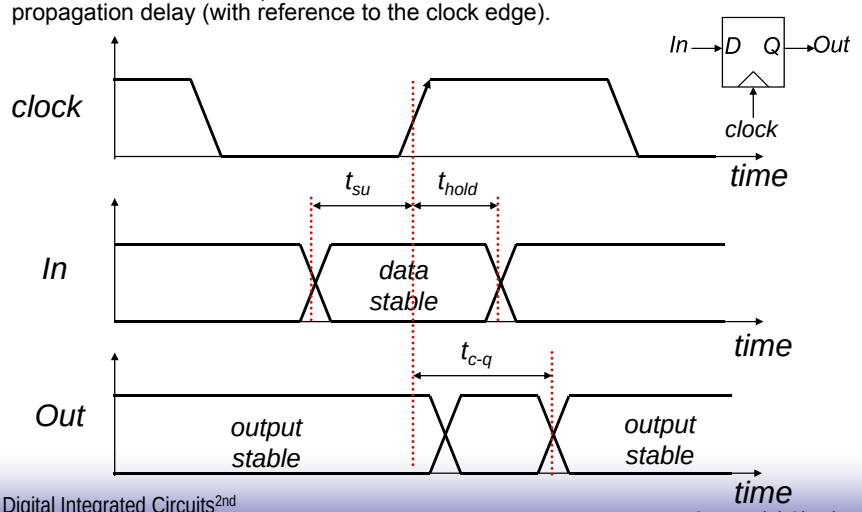
- Two storage mechanisms:
  - ✓ Positive feedback
  - ✓ Charge based
- Registers: used to hold the system state
  - ✓ positive edge triggered
  - ✓ negative edge triggered



Block diagram of a finite-state machine (FSM)

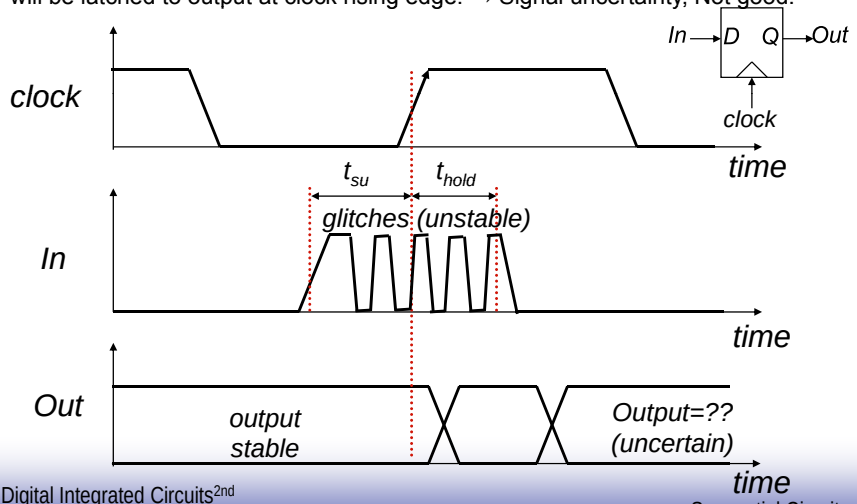
Timing Metrics

- setup time (t_{su}): the time that data inputs (D) must be valid before clock transition.
- hold time (t_{hold}): the time data input must remain valid after the clock edge.
- propagation delay ($t_{c\text{-}q}$): Data at D input is copied to Q output after a worst case propagation delay (with reference to the clock edge).



Timing Metrics

❑ Why do we have setup time (t_{su}) and hold time (t_{hold}) requirements for registers: If input data to register is not stable (e.g. there are glitches) before and after clock rising edge, it is very difficult to predict what input value (0 or 1) will be latched to output at clock rising edge. → Signal uncertainty, Not good!



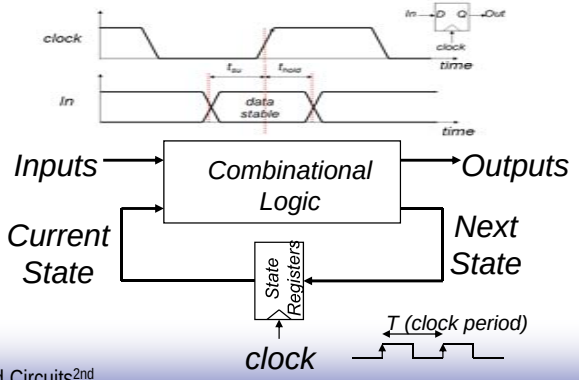
System Timing Constraint #1

❑ Minimum clock period T required for proper operation of sequential circuit (due to requirement of setup time):

$$T \geq t_{c\text{-}q} + t_{plogic} + t_{su}$$

Where: t_{plogic}: worst case propagation delay of the logic
t_{cd}: contamination delay, minimum propagation delay of logic

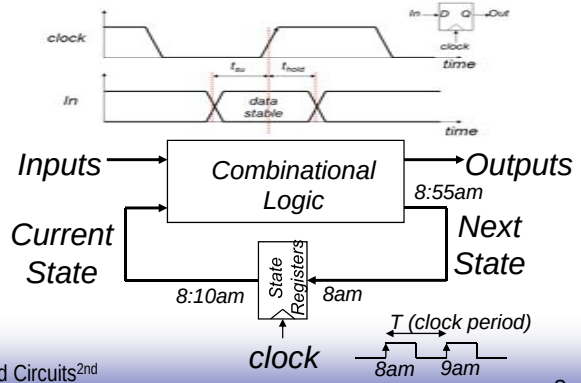
✓ Reason: After current clock edge, the slowest signal for next state should be stable at least t_{su} before next clock edge.



System Timing Constraint #1

❑ How to understand: $T \geq t_{c\text{-}q} + t_{plogic} + t_{su}$?

❑ Example: Assume $t_{c\text{-}q}$=10min, t_{plogic}=45min, t_{su}=20min, first clock rising edge arrives at 8am.If T=1hour, setup time requires next state should arrive register input 20 minutes before next clock rising edge (9:00am), i.e. before 8:40am. However, due to delay ($t_{c\text{-}q}$+t_{plogic}=10+45=55min), next state arrives register input at 8:55am. → too late. It violates setup time requirement. If T>$T_{c\text{-}q}$+t_{plogic}+t_{su}=75min, such violation will not occur.



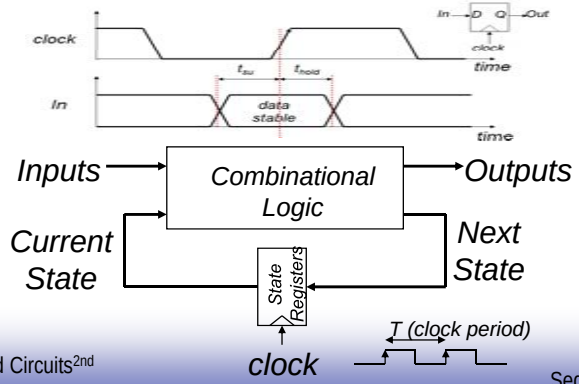
System Timing Constraint #2

❑Timing requirement due to hold time of register:

$$t_{cdreg} + t_{cdlogic} \geq t_{hold}$$

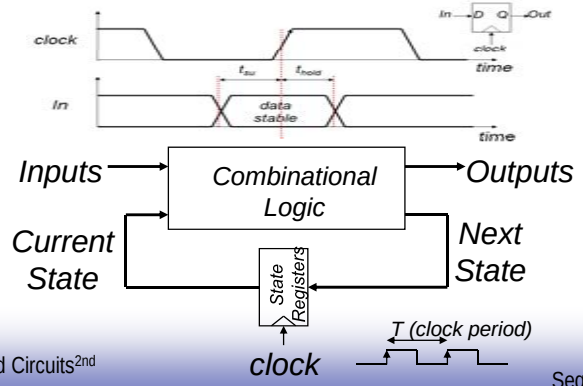
Where: t_{cdreg}: minimum propagation delay (contamination delay) of register.
$t_{cdlogic}$: minimum propagation delay (contamination delay) of the combinational logic.

✓ Reason: After current clock edge, the fastest signal for next state should not arrive input D before t_{hold} to change current values of input D.



System Timing Constraint #2

- How to understand: $t_{cdreg} + t_{cdlogic} \geq t_{hold}$?
- Assume t_{cdreg}=10min, $t_{cdlogic}$=15min, t_{hold}=30min, first clock rising edge arrives at 8am. Hold time requires current register inputs stay unchanged until 8:30am. However, signal passes through register and combinational logic, becomes next state and appears at register input at 8:25am, i.e., new next-state will change the current register input value at 8:25am. → too early, it violates hold time requirement!



Static vs Dynamic Storage

- Static storage
  - preserve state as long as the power is on
  - have positive feedback (regeneration) with an internal connection between the output and the input
  - useful when updates are infrequent (clock gating)
- Dynamic storage
  - store state on parasitic capacitors
  - only hold state for short periods of time (milliseconds)
  - require periodic refresh
  - usually simpler, so higher speed and lower power

Latches vs Flipflops

- Latches (level-sensitive)
  - level sensitive circuit that passes inputs to Q when the clock is high (or low) - transparent mode
  - input sampled on the falling (or rising) edge of clock is held stable when clock is low (or high) - hold mode
- Flipflops (edge-triggered)
  - edge sensitive circuits that sample the inputs on a clock transition
    - positive edge-triggered: 0 → 1
    - negative edge-triggered: 1 → 0
  - built using latches (e.g., master-slave flipflops cascading a positive and negative latch)

Sequential Definitions

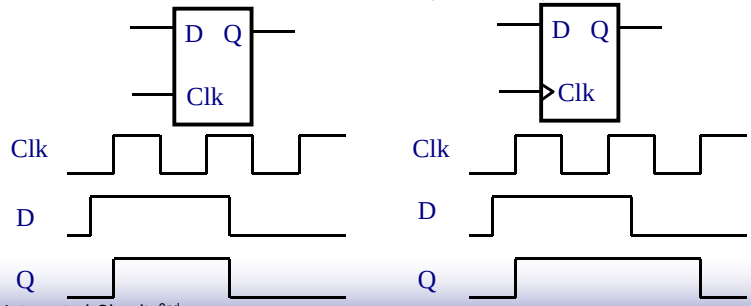
- Static versus dynamic storage
  - static uses a bistable element with feedback (regeneration) and thus preserves its state as long as the power is on
  - static is preferred when updates are infrequent (clock gating)
  - dynamic stores state on parasitic capacitors so only holds the state for a period of time (milliseconds) and requires periodic refresh
  - dynamic is usually simpler (fewer transistors), higher speed, lower power
- Latch versus flipflop
  - latches are level sensitive with two modes: transparent - inputs are passed to Q, and hold - output stable.
  - fliplflops are edge sensitive that only sample the inputs on a clock transition

Latch versus Register

- 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

✓ Latch (positive): stores data when clock is high (Clk=1)

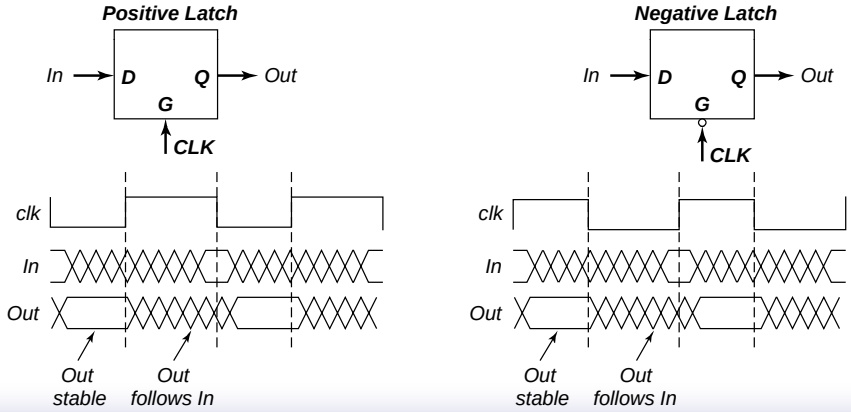
✓ Register (positive edge-triggered): stores data when clock rises (Clk=0→1)



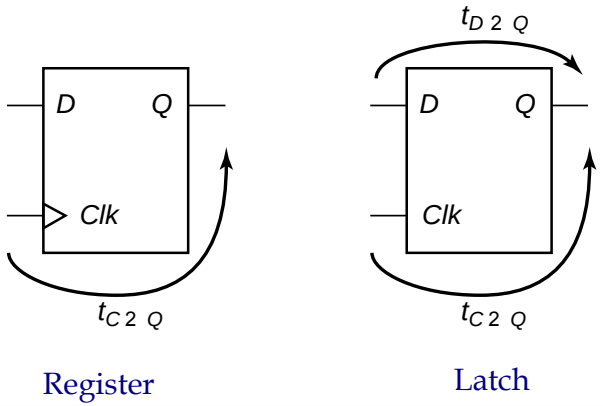
Latches

❑ Positive latch: transparent high - passes D input to Q output when clock is high

❑ Negative latch: transparent low - passes D input to Q output when clock is low.



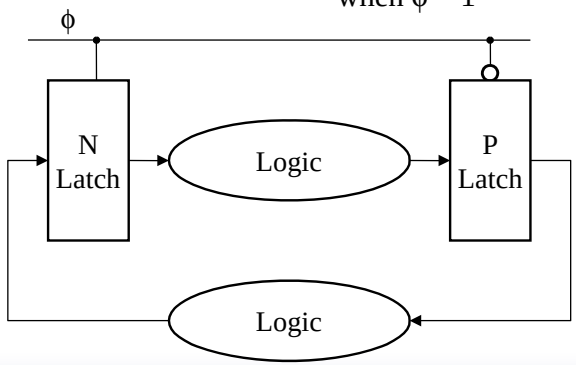
Characterizing Timing



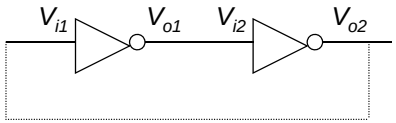
Latch-Based Design

• N latch is transparent when $\phi = 0$

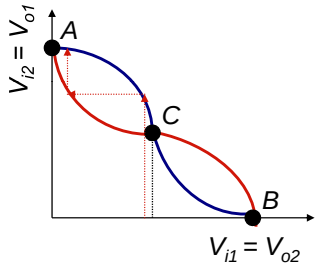
• P latch is transparent when $\phi = 1$



Review: The Regenerative Property



cascaded inverters

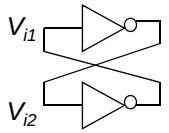


If the gain in the transient region is larger than 1, only A and B are stable operation points. C is a metastable operation point.

❑ A bistable circuit has two stable states. In absence of any triggering, the circuit remains in a single state and thus remembers a value.

Bistable Circuits

❑ The cross-coupling of two inverters results in a bistable circuit (a circuit with two stable states)



❑ *Have to be able to change the stored value by making A (or B) temporarily unstable by increasing the loop gain to a value larger than 1*

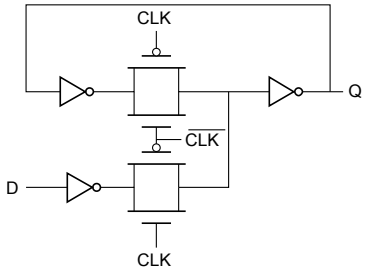
- *done by applying a trigger pulse at V_{i1} or V_{i2}*
- *the width of the trigger pulse need be only a little larger than the total propagation delay around the loop circuit (twice the delay of an inverter)*

❑ *Two approaches used*

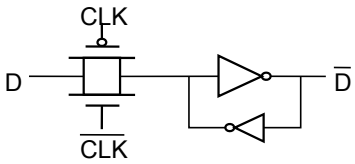
- *cutting the feedback loop (mux based latch)*
- *overpowering the feedback loop (as used in SRAMs)*

Writing into a Static Latch

Use the clock as a decoupling signal,
that distinguishes between the transparent and opaque states



Converting into a MUX

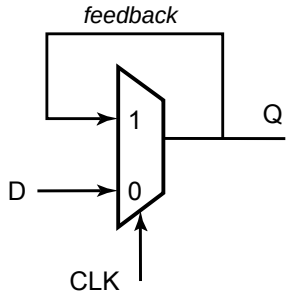


Forcing the state
(can implement as NMOS-only)

MUX-Based Latches

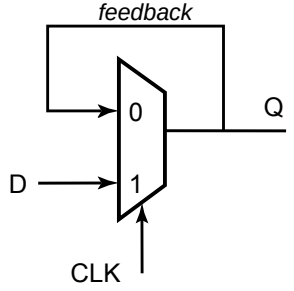
❑ Change the stored value by cutting the feedback loop

Negative latch
(transparent when CLK= 0)



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

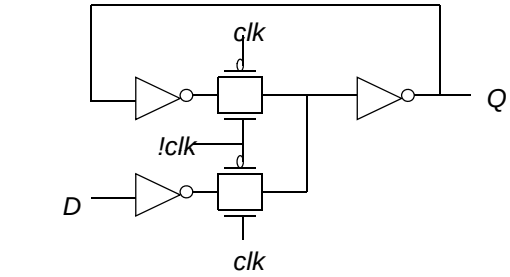
Positive latch
(transparent when CLK= 1)

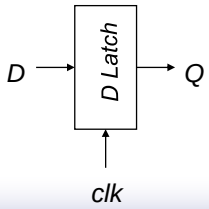


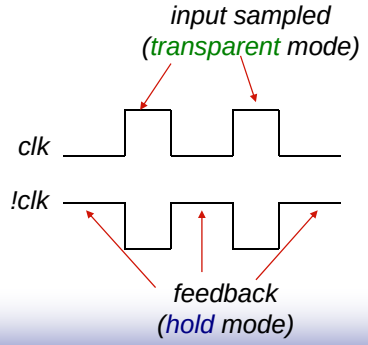
$$Q = Clk \cdot Q + \overline{Clk} \cdot In$$

TG MUX Based Latch Implementation

- Positive latch based on transmission gate multiplexers:

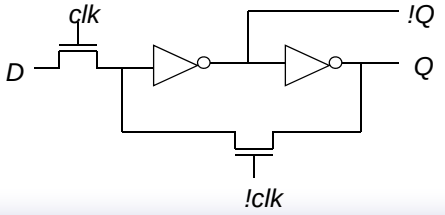


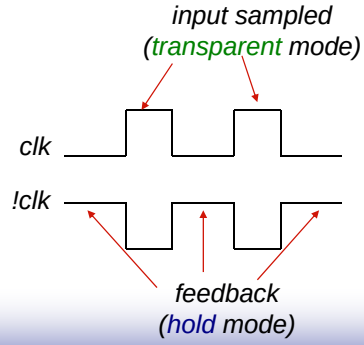




PT MUX Based Latch Implementation

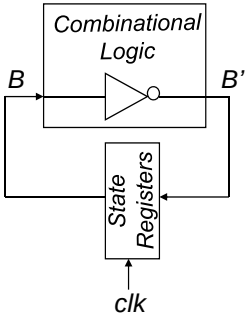
- Multiplexer-based NMOS latch by using NMOS-only pass transistors for multiplexers:
  - ✓ Reduced clock load, but threshold drop at output of pass transistors so reduced noise margins and performance.
  - ✓ It also causes static power in 1st inverter, because maximum input voltage to inverter is V_{DD}-V_{Tn}, and PMOS device of inverter is never fully turned off.

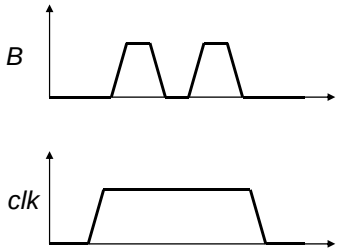




Latch Race Problem

- Signal race problem of latches: If input signal keep changing (e.g. glitches) when clk=1, it may cause uncertainty on the signal being latched.





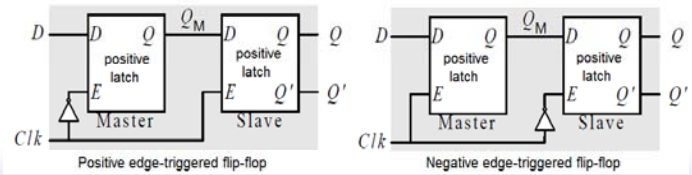
Which value of B is stored?

Two-sided clock constraint

$$T \geq t_{c\text{-}q} + t_{plogic} + t_{su}$$
$$T_{high} < t_{c\text{-}q} + t_{cdlogic}$$

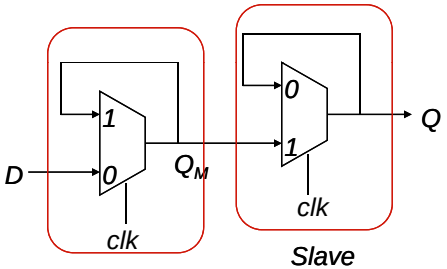
Constructing Flip-flop from Latches Using Master-Slave Two-stage Design

- Edge-triggered flip-flops can be constructed by cascading two level-sensitive latches using master-slave two-stage design.
- Ex: Positive edge-triggered flip-flop by cascading 2 positive latches with opposite clocks
  - ✓ When clk=0, master latch is transparent, Q_M=Q, but slave stage is hold, Q_M cannot pass to Q;
  - ✓ When clk=0→1, master latch is hold, slave stage is transparent, Q_M value latched at the end of clk=0 phase is passed to Q, $Q=Q_M|_{clk=0\rightarrow1}=D|_{clk=0\rightarrow1}$→rising edge triggered.
  - ✓ When clk=1, master latch is hold, D cannot change Q_M and Q.



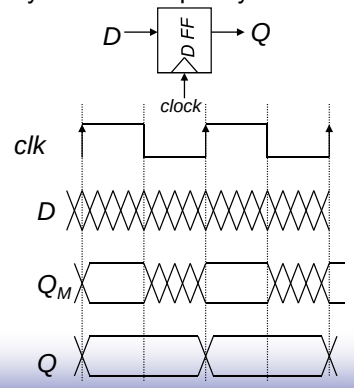
Master Slave Based ET Flipflop

- Positive edge-triggered register with master-slave configuration: cascading a negative latch (master) with a positive latch (slave).
  - ✓ When clk=0, master stage is transparent, D input is passed to Q_M, slave stage is in hold mode, keeping its previous value by feedback.
  - ✓ When clk=0→1, master is hold, slave is transparent, Q_M remains the value of D right before clock rising edge, and its value is passed to Q.
  - ✓ Q_M is constant when clk=1 → Q makes only 1 transition per cycle.



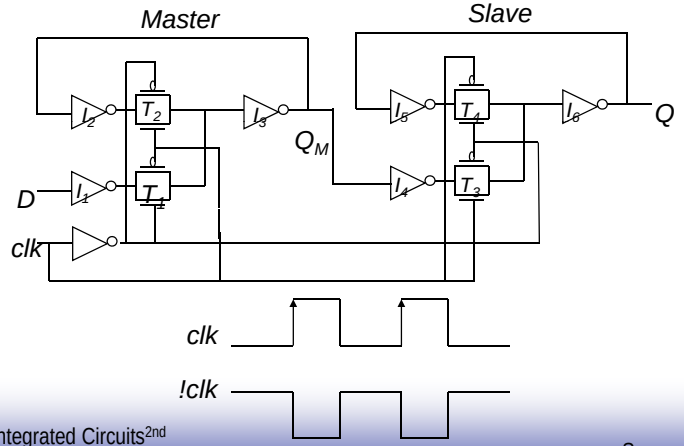
clk = 0 transparent hold

clk = 0→1 hold transparent

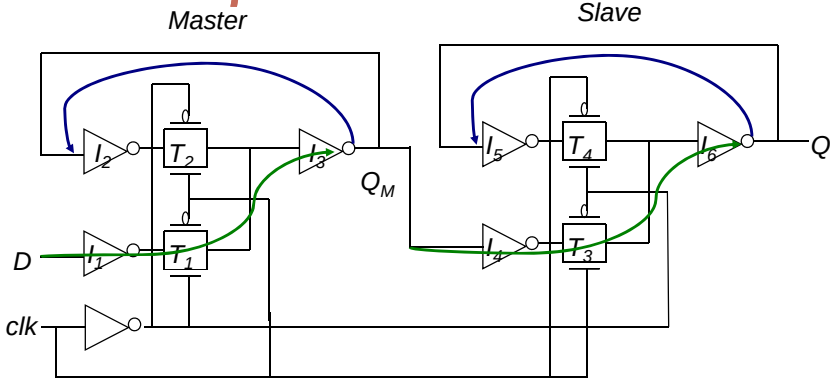


MS ET Implementation

- Master-slave positive edge-triggered register using multiplexers
  - ✓ When clk=0, T_1: on, T_2: off, D is sampled to Q_M, T_3: off, T_4: on, I_5 and I_6 hold the state of slave latch.
  - ✓ When clk=0→1, T_1: off, T_2: on, I_2 and I_3 hold state of Q_M. T_3: on, T_4: off, Q_M is copied to output Q.

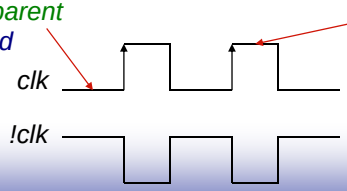


MS ET Implementation



master transparent
slave hold

master hold
slave transparent



MS ET Timing Properties

- Assume propagation delays of inverter and TG are t_{pd_inv} and t_{pd_tx}, that the contamination delay is 0, and that the inverter delay to derive !clk is 0
- Set-up time - time before rising edge of clk that D must be valid

$$t_{su}=3 * t_{pd\_inv} + t_{pd\_tx}$$

- Propagation delay - time for Q_M to reach Q

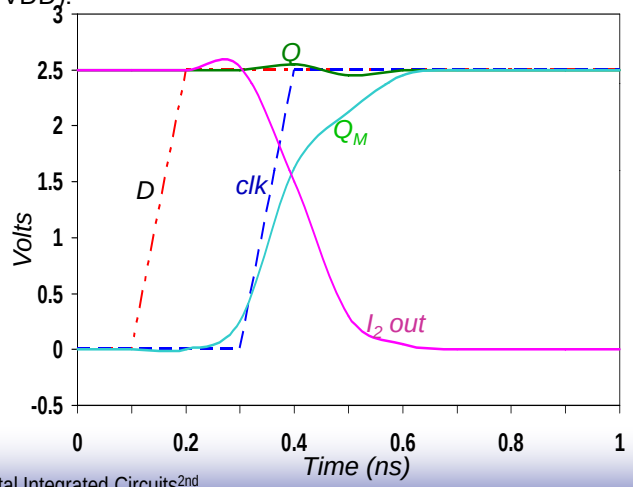
$$t_{c\text{-}q}=t_{pd\_inv} + t_{pd\_tx}$$

- Hold time - time D must be stable after rising edge of clk -

$$t_{hlod}=0$$

Set-up Time Simulation

- PSPICE set-up time simulation: we progressively skew the input with respect to the clock edge until the circuit fails.
  - ✓ Case #1: skew=210ps: correct value of input D is sampled (Q remains at VDD).

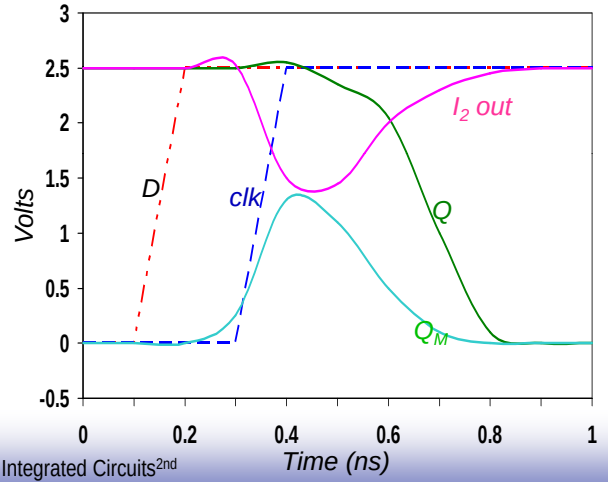


t_{setup} = 0.21 ns

works correctly

Set-up Time Simulation

- PSPICE set-up time simulation: we progressively skew the iput with respect to the clock edge until the circuit fails.
  - ✓ Case #2: skew=200ps: incorrect value of D propagates to Q (Q changes to 0).

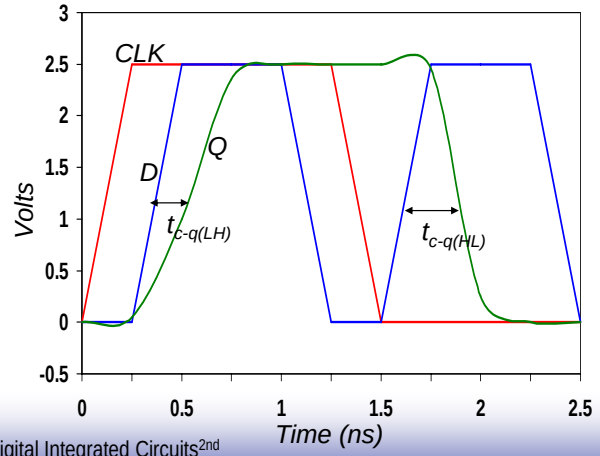


t_{setup} = 0.20 ns

fails

Propagation Delay Simulation

- PSPICE propagation delay simulation
  - ✓ propagation delay: time from 50% point of CLK edge to 50% point of Q output
  - ✓ $t_{c\text{-}q(lh)}$=160ps, $t_{c\text{-}q(hl)}$=180ps.

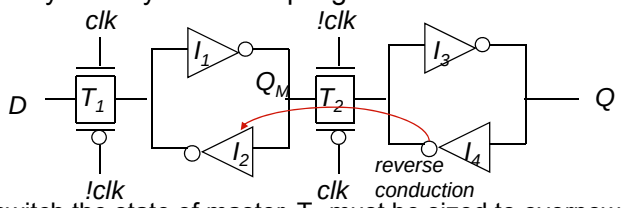


$t_{c\text{-}q(LH)}$ = 160 psec

$t_{c\text{-}q(HL)}$ = 180 psec

Reduced Load MS ET FF

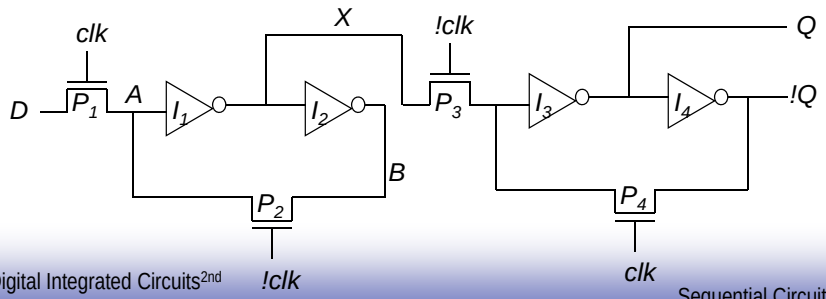
- Clock load per register is important since it directly impacts the power dissipation of the clock network.
- Transmission-gate register design: clock load of 8 transistors (ignoring inverters for clock)
- Can reduce the clock load (at the cost of robustness) by making the circuit ratioed: eliminate feedback transmission gate by directly cross-coupling the inverters.



- to switch the state of master, T_1 must be sized to overpower the feedback inverter I_2 to switch the state of cross-coupled inverter. (I_2: small W/L → larger R_{on}, weaker device).
- to avoid reverse conduction, I_4 must be weaker than I_1 (I_4: small W/L, larger R_{on}, weaker device).

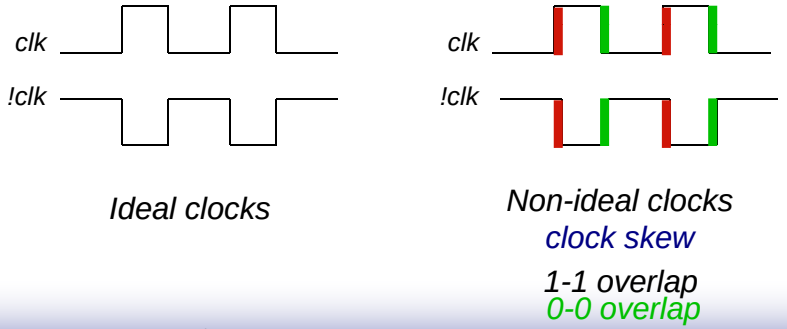
Master-slave Register based on NMOS-only Pass Transistors

❑ Negative master-slave two-stage register based on NMOS-only pass transistors and cascade-inverter latches:
✓ When clk=1, P1 on, P2 off, P3 off, P4 on, X=D', master stage on, slave stage off, Q remains its previous value,
✓ When clk=1→0, P1 off, P2 on, P3 on, P4 off, master stage off, slave stage on, Q=X'=(D')'=D$|_{clk=1\rightarrow0}$: falling-edge triggered.



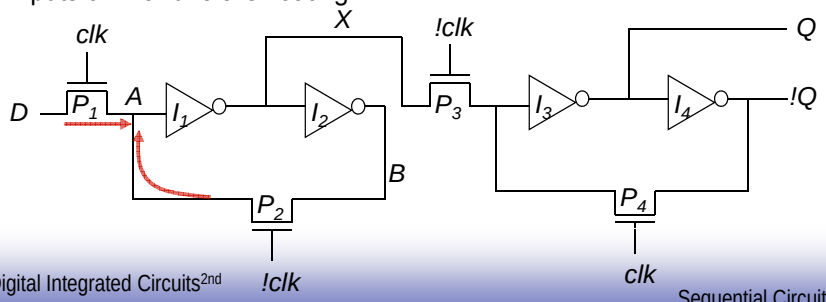
Non-Ideal Clocks

❑ Clock skews come from
✓ delay for clk' generated from clk,
✓ Variations in wires used to route clk and clk',
✓ load capacitance vary based on data in connecting latches.
❑ Non-ideal clocks:
✓ 1-1 overlap
✓ 0-0 overlap



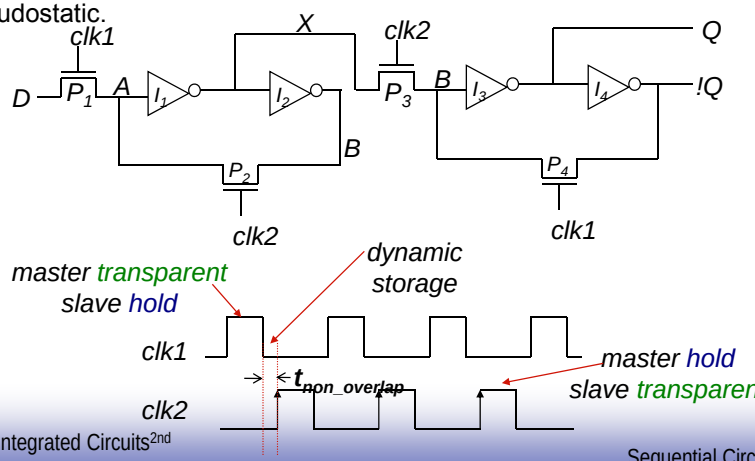
Example of Clock Skew Problems

❑ Problems caused by clock overlap:
✓ Race condition – direct path from D to Q during the short time when both clk and !clk are high (1-1 overlap). Output changes in both clock rising/falling edges instead of one edge. Further, value of output Q depends on whether input D arrives node X before or after falling edge of clk'.
✓ Undefined state – both B and D are driving A when clk and !clk are both high
✓ Dynamic storage – when clk and !clk are both low (0-0 overlap): inputs of I1 and I3 are floating.



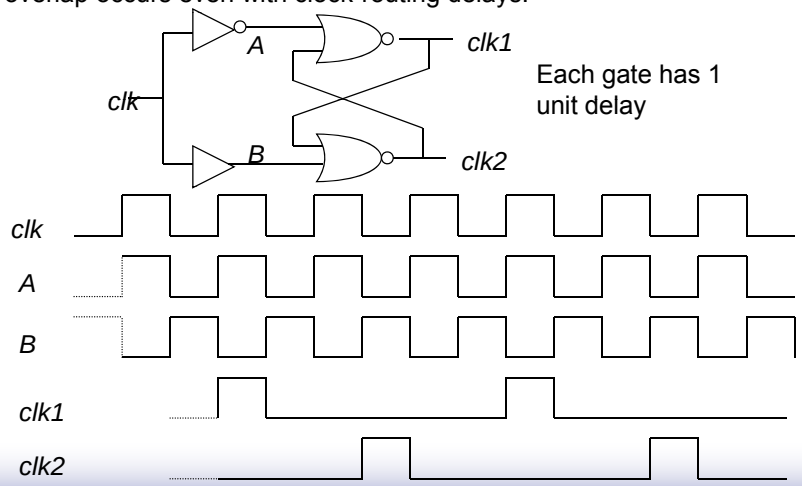
Pseudostatic Two-Phase ET FF

❑ Solution: use two nonoverlapping clocks (clk1 and clk2) instead of one clock and its inversion (clk and clk').→No 1-1 overlap, no signal race. But during non-overlap time (0-0 overlap), FF is in high-impedance state: feedback loop is open, nodes A and B are floating. Leakage will destroy the state if this condition holds for too long – pseudostatic.



Two Phase Clock Generator

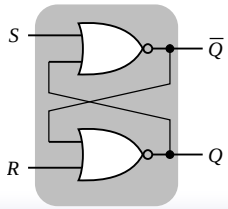
- Circuit for generating two-phase non-overlapping clock
- By keeping nonoverlap time $t_{non\_overlap}$ large enough so that no overlap occurs even with clock-routing delays.

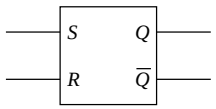


Overpowering Feedback Loop : Cross-Coupled Pairs

- Traditional way to switch bistable element: overpower feedback loop.
- Static SR (set-reset) flip-flop: writing data by pure force. Cross-coupled NOR gates, with the 2nd input of NOR gates connected to trigger inputs (S and R) for forcing output Q and Q' to a given state.
  - ✓ SR=00: cross-coupled inverter ((A+0)'=A', NOR gate with input "0" is like an inverter), Q and Q' retain their values.
  - ✓ SR=10: Q is forced to 1, Q'=0.
  - ✓ SR=01: Q is forced to 0, Q'=1.
  - ✓ SR=11: QQ'=00, Q' is not inversion of Q → forbidden state.

NOR-based set-reset



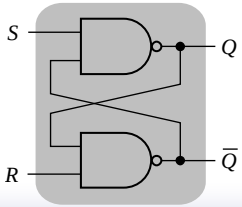


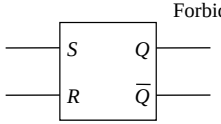
S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

Forbidden State

SR flip-flop: Cross-Coupled NAND

- Another SR flip-flop implementation: cross-coupled NAND structure
- Static SR (set-reset) flip-flop based on cross-coupled NAND gates, with the 2nd input of NAND gates connected to trigger inputs (S and R, active low) for forcing output Q and Q' to a given state.
  - ✓ SR=00: QQ'=11, Q' is not inversion of Q → forbidden state.
  - ✓ SR=10: Q is forced to 0, Q'=1.
  - ✓ SR=01: Q is forced to 1, Q'=0.
  - ✓ SR=11: cross-coupled inverter ((A·0)'=A', NAND gate with input "1" is like an inverter), Q and Q' retain their values.



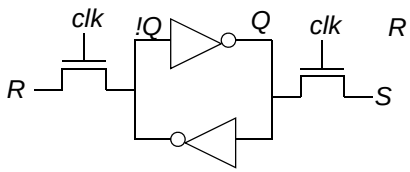


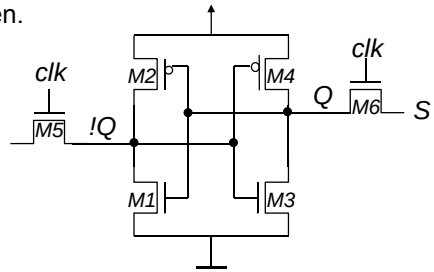
Forbidden State

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

6 Transistor CMOS SR Latch

- Previous SR flip-flop: asynchronous.
- Synchronous clocked 6-transistor CMOS SR latch: use clock for synchronous behavior.
  - ✓ When clk=0: R and S are isolated from cross-coupled inverters, Q and Q' remains their values.
  - ✓ When clk=0→1, S and R are connected to Q and Q'.
    - If SR=00: Q=0, Q'=0, forbidden.
    - If SR=11: Q=1, Q'=1, forbidden.
    - If SR=01: Q=0, Q'=1,
    - If SR=10: Q=1, Q'=0.



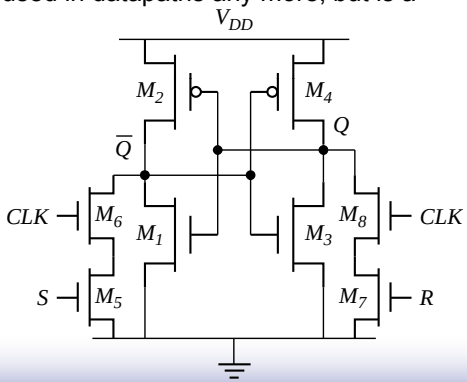


Clocked ratioed CMOS SR Latch

❑ Clocked 6-transistor CMOS SR latch: S and R are applied to source/drain terminals (low impedance nodes): not good.

❑ Clocked ratioed 8-transistor CMOS SR latch: R and S applied to gate terminals. It includes a cross-coupled inverter pair, plus 4 extra transistors to drive the flip-flop from one state to another, and to provide synchronization. (It's not used in datapaths any more, but is a basic building memory cell)

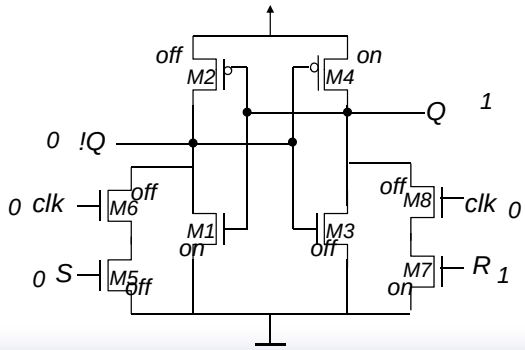
❑ In steady state, one inverter in high, while the other one is low. No static path between Vdd and Gnd, but tranistor sizing is essential to ensure that flip-flop can transition from one state to the other when needed.



Ratioed CMOS Clocked SR Latch

❑ Clocked ratioed 8-transistor CMOS SR latch: working principle

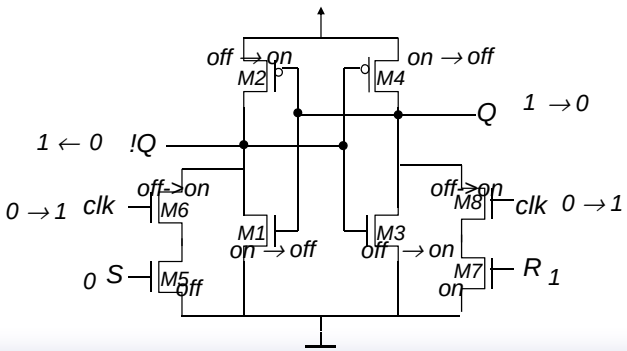
✓ Assume initial state: Q=1, Q'=0, now R=1, S=0, M1 is on, M5 is off, but clk=0, M6 and M8 are off, R and S are disconnected to Q and Q', cross-coupled inverters M2,M1, M4, M3 retain their current state.



Ratioed CMOS Clocked SR Latch

❑ Clocked ratioed CMOS SR latch: working principle

✓ Now if clk=0→1, M6 and M8 are off→on. Since M4, M7 and M8 are on, they should be properly sized so that Q is brought below switching threshold of inverter M2/M1, so that Q'=0→1, which in turn causes inverter M4/M3 to switch state, Q=1→0. Finally: Q=0, Q'=1.

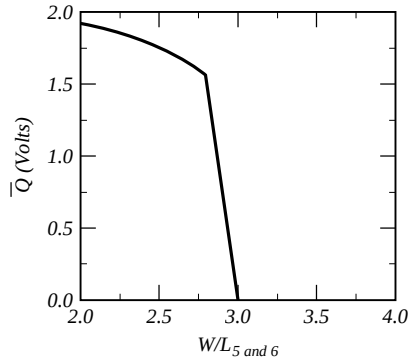


Sizing Issues

❑ Sizing issues for SR flip-flop:

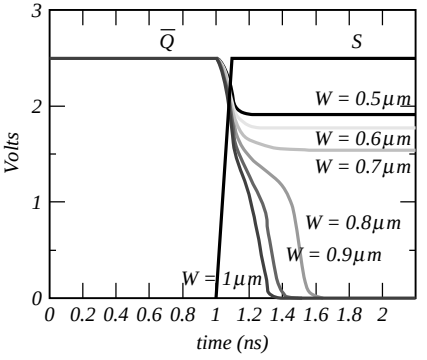
(a). DC output Q' v.s. pull-down device size $M_{5\text{-}6}$ (with W/L_2=1.5μm/0.25μm).

(b). Transient response showing that M_5 and M_6 must each have W/L larger than 3 to switch SR flip-flop



(a)

Output voltage dependence on transistor width



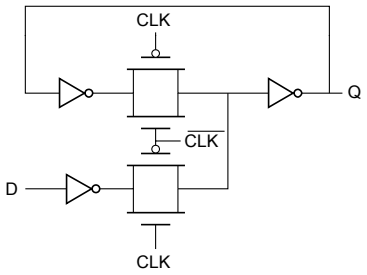
(b)

Transient response (L=0.25μm)

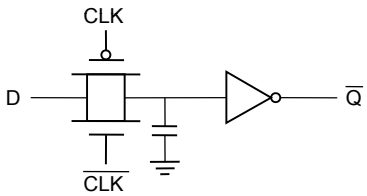
Storage Mechanisms

❑ Static storage: cross-coupled inverter pair forming a bistable element. A stored value remains valid as long as supply voltage is applied.

❑ Dynamic storage: utilize charge stored on capacitor to represent logic signal (absence: 0, presence: 1). Due to charge leakage, periodic refreshing is needed to maintain the signal.



Static



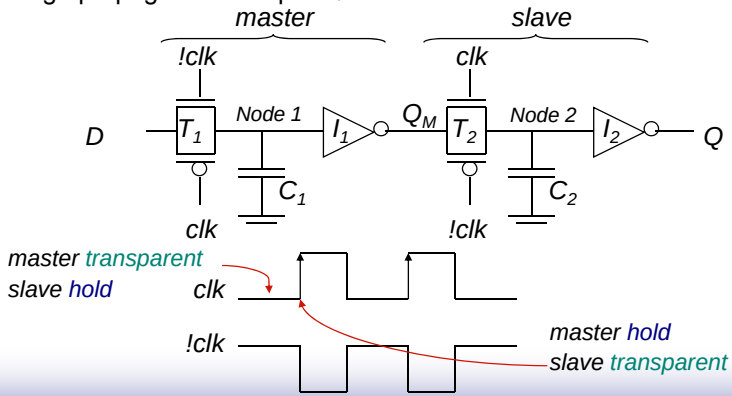
Dynamic (charge-based)

Dynamic ET Flipflop

❑ Dynamic TG positive edge-triggered register (8 transistors):

✓ When clk=0, T1 is on, T2 is off, master stage transparent, input D is sampled to storage node 1, slave stage is hold, node 2 is floating.

✓ When clk=0→1, T1 on→off, T2 off→on, master stage is hold, slave stage is transparent, the value sampled on node 1 before clock rising edge propagates to output Q.

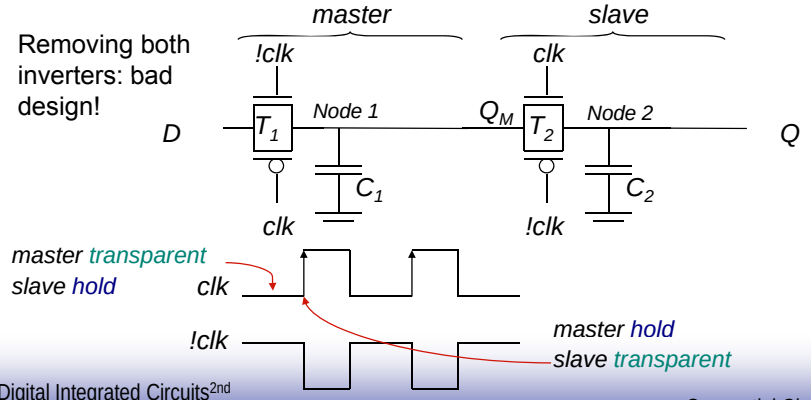


Dynamic ET Flipflop

❑ Question: Why do we need two inverters? Can we remove them?

❑ Answer: No. If two inverters are removed, when T_1 is off, both nodes 1 and 2 are floating. We only rely on charge stored in capacitors C_1 and C_2 to maintain them to be "1", which is very weak. If V(D)=5V, C_1=C_2, when clk=0→1, we expect V(Q)="1"=5V, but due to charge sharing, V(Q)=2.5V. Further, due to charge leakage, V(Q) is keep dropping.

Removing both inverters: bad design!

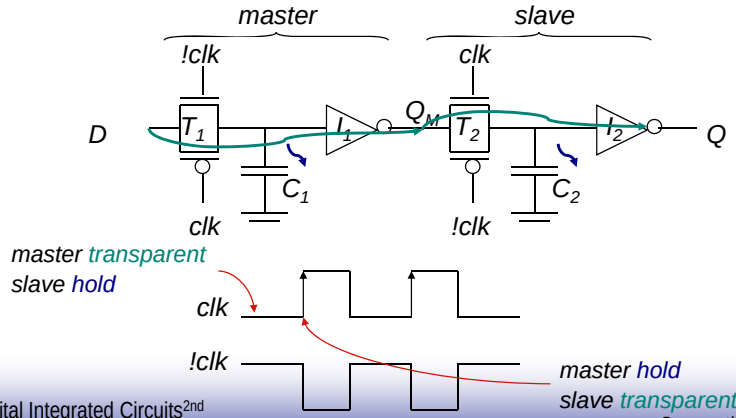


Dynamic ET Flipflop

❑ setup time: $t_{su}=t_{pd\_tx}$

❑ hold time: $t_{hold}=0$

❑ propagation delay $t_{c\text{-}q}=2t_{pd\_inv}+t_{pd\_tx}$

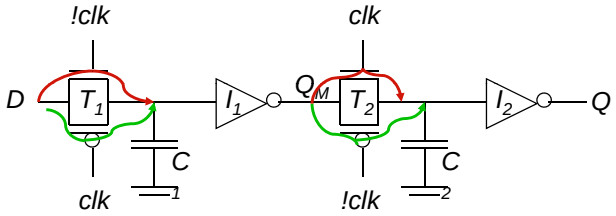


Dynamic ET FF Race Conditions

❑ Clock overlap causes race problem for dynamic edge-triggered flip-flop:

✓ 0-0 overlap: both PMOS of T1 and PMOS of T2 are simultaneously on, creating a direct path for data to flow from D input to Q output.

✓ 1-1 overlap: both NMOS of T1 and NMOS of T2 are simultaneously on, creating a direct path for data to flow from D input to Q output.



0-0 overlap race condition

$t_{overlap0\text{-}0} < t_{T1} + t_{I1} + t_{T2}$

1-1 overlap race condition

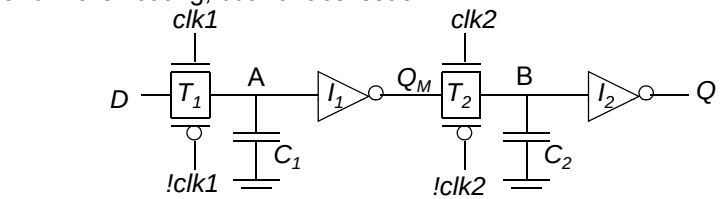
$t_{overlap1\text{-}1} < t_{hold}$

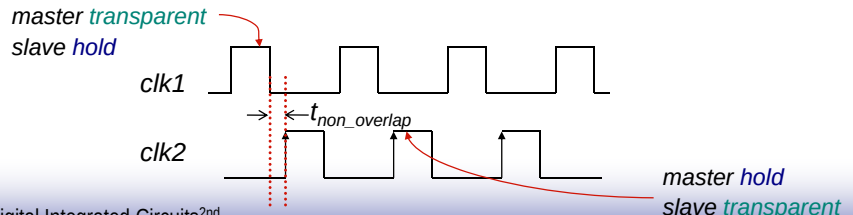
Dynamic Two-Phase ET FF

❑ Solution #1 to race problem: use two-phase nonoverlap clocks (clk1 and clk2) instead of only one clock and its inversion (clk and clk').

✓ No 1-1 overlap between clk1 and clk2: T1 and T2 will never be simultaneously on, no race problem.

✓ 0-0 overlap between clk1 and clk2: both T1 and T2 are off, nodes A and B are floating, but no race issue.

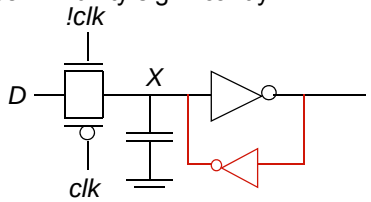




Making a Dynamic Latch Pseudo-Static

❑ Robustness considerations limit the use of dynamic FF's
- coupling between signal nets and internal storage nodes can inject significant noise and destroy the FF state
- leakage currents cause state to leak away with time
- internal dynamic nodes don't track fluctuations in V_{DD} that reduces noise margins

❑ A simple fix is to make the circuit pseudostatic by adding a weak feedback inverter. → Node X is driven by the output of an inverter, it will never be floating again. slightly increase the delay, but improves the noise immunity significantly.



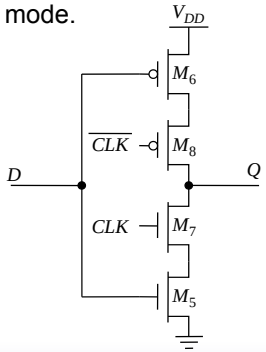
❑ *Add above logic (weak feedback inverter) to all dynamic latches*

Other Latches/Registers: C²MOS

❑ Positive C²MOS (clocked CMOS) latch:

✓ When clk=0: M7,M8 are off, Q is floating, latch in hold mode.

✓ When clk=1: M7, M8 are on, Q=D', latch is in transparent mode.

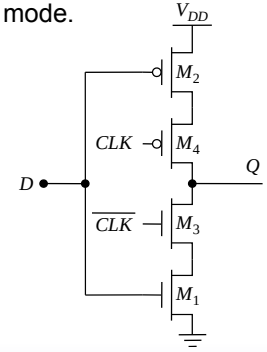


Positive C²MOS latch

❑ Negative C²MOS (clocked CMOS) latch:

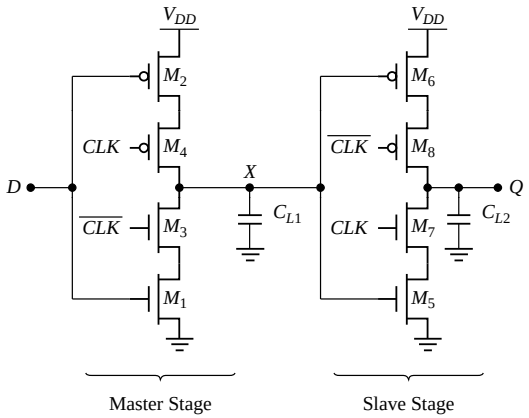
✓ When clk=1: M3,M4 are off, Q is floating, latch in hold mode.

✓ When clk=0: M3, M4 are on, Q=D', latch is in transparent mode.



Negative C²MOS latch

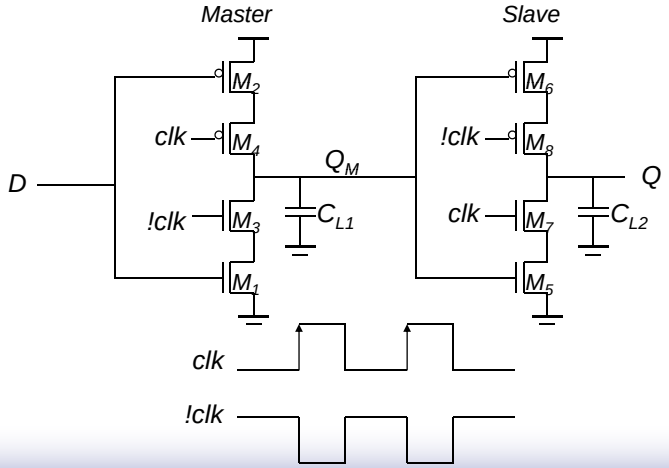
C²MOS (Clocked CMOS) ET Flipflop



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

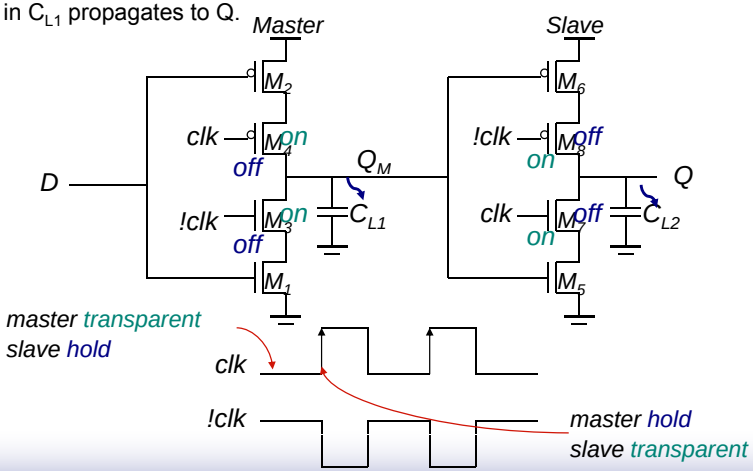
C²MOS (Clocked CMOS) ET Flipflop

- C²MOS (clocked CMOS) positive edge-triggered flipflop: clock-skew insensitive, an ingenious master-slave register insensitive to clock overlap (8-transistors).



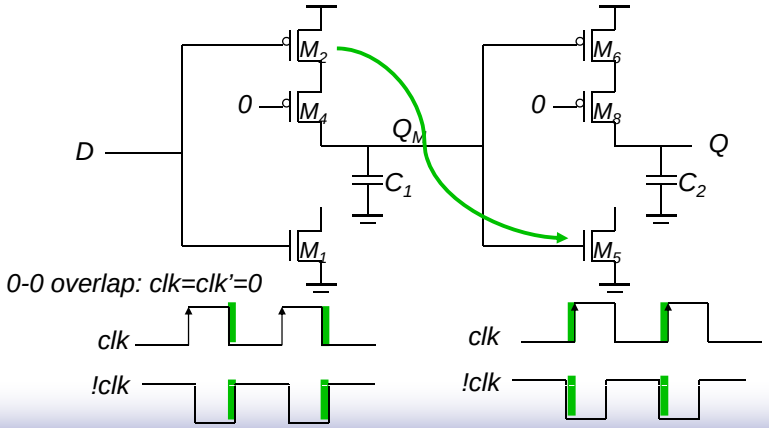
C²MOS (Clocked CMOS) ET Flipflop

- Working principle: operates in two phases (positive edge-triggered register).
  - ✓ clk=0: M_3, M_4 on; M_7, M_8 off, master stage acts as inverter, Q_M=D' (evaluation); slave stage in high-impedance (hold), Q remains previous value stored on C_{L2}.
  - ✓ clk=1: master stage in hold mode, slave stage evaluates, Q=Q_M', value stored in C_{L1} propagates to Q.



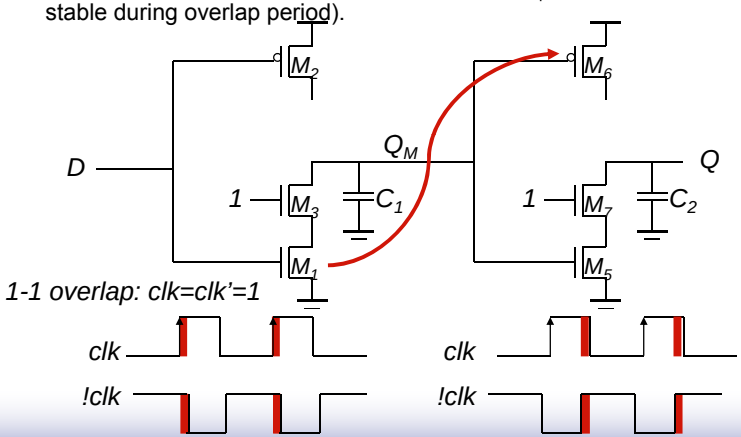
C²MOS FF 0-0 Overlap Case

- C²MOS register with clk-clk' clocking is insensitive to clock overlap as long as rise and fall times of clock edges are sufficiently small.
- Ex: For (0-0) clock overlap, M_3 M_7 are off, M_4 M_8 are on. If D=1, M_2 is off, D cannot pass to Q_M. If D=0, Q_M=1, but M_6 is off, Q is floating, Q_M cannot pass to Q. Thus D cannot pass to Q during (0-0) overlap.



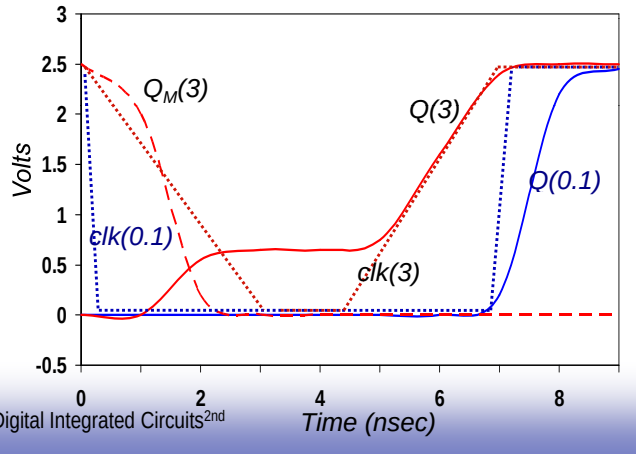
C²MOS FF 1-1 Overlap Case

- Ex: For (1-1) clock overlap, M_4 M_8 are off, M_3 M_7 are on. If D=0, M_1 is off, D cannot pass to Q_M. If D=1, Q_M=0, but M_5 is off, Q is floating, Q_M cannot pass to Q. Thus D cannot pass to Q during (1-1) overlap.
- However, right after overlap, clk'=0 and M8 is on, Q_M=0 propagates to Q. → Not good. Solution: 1-1 overlap constraint: $t_{overlap1-1} < t_{hold}$ (data D should be stable during overlap period).



C²MOS Transient Response

- In sum, C²MOS register is insensitive to clock overlaps because overlaps activate either pull-up or pull-down networks of latches, but never both of them simultaneously.
- However, if t_r and t_f of clock is large (slow clocks), there exists a time slot where both NMOS and PMOS are on. → Both master and slave stages are on, Input can directly go to output → signal race condition.



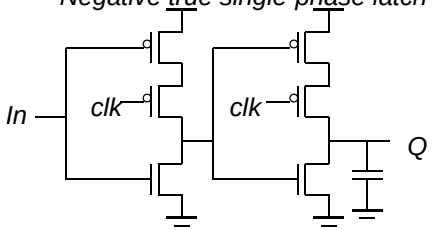
For a 0.1 ns clock: no race

For a 3 ns clock (race condition exists)

Other Latches/Registers: TSPC

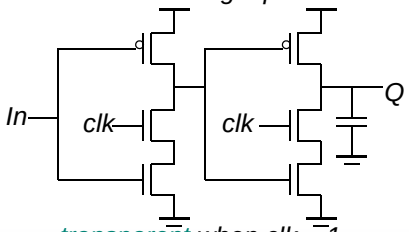
- Two-phase clocked latches: clock overlap may cause race and floating node problem.
- True single-phase clocked (TSPC) latches: use only a single clock, never have clock overlap, no need to worry about clock overlap problem.
- Negative true single-phase latch: when clk=0, two cascaded inverters, Q=In (non-inverting), transparent mode; when clk=1, both inverters are disabled: hold mode.
- Positive true single-phase latch: when clk=1, two cascaded inverters, Q=In (non-inverting), transparent mode; when clk=0, both inverters are disabled: hold mode.

Negative true single-phase latch



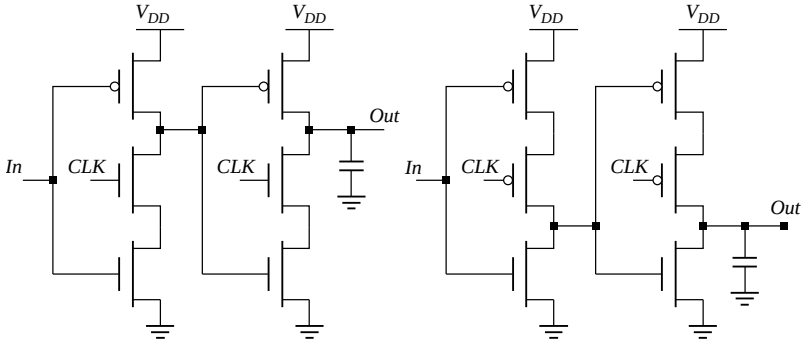
hold when clk = 1
transparent when clk = 0

Positive true single-phase latch



transparent when clk = 1
hold when clk = 0

True Single Phase Clocked (TSPC) Latches

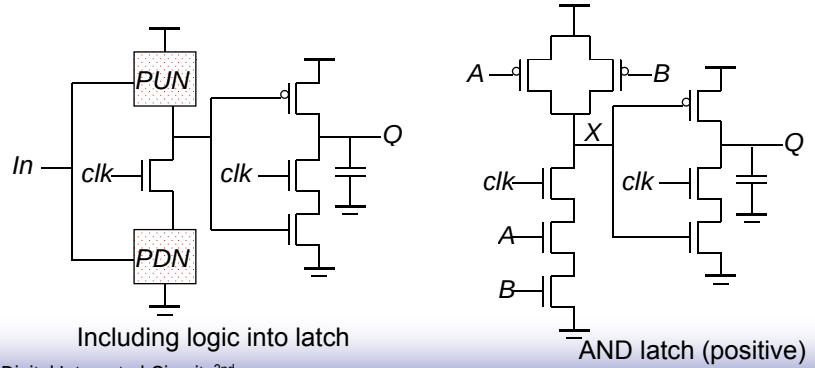


Positive latch (6 transistors) (transparent when CLK= 1)

Negative latch (6 transistors) (transparent when CLK= 0)

Embedding Logic in TSPC Latch

- TSPC latch offers possibility of embedding logic functionality into latches.→ reduce delay overhead due to latches.
- Ex: AND latch. Instead of implement it as “CMOS NAND + CMOS inverter + positive TSPC latch”, we can implement it as “AND latch”.
- ✓ When clk=0, hold mode, Q remains its current value,
- ✓ When clk=1, transparent mode, Q=X’=((AB)’)’=AB

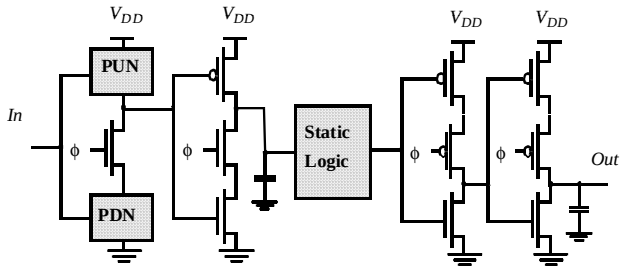


Including logic into latch

AND latch (positive)

TSPC - True Single Phase Clock Logic

- ❑ TSPC – True Single Phase Clock Logic
- ✓ Virtually no design constrains: no even-inversion constrains between 2 latches or between a latch and a dynamic block. Dynamic and static circuits can mix freely.
- ✓ Logic functions can be included in negative TSPC or positive TSPC latches, or placed between them.
- ✓Disadvantage: more transistors per latch (6 instead of 4)

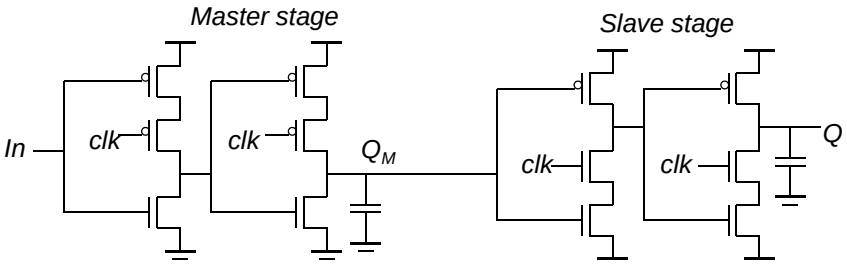


Including logic into the latch

Inserting logic between latches

True Single Phase Clocked Registers (TSPCR)

- True single-phase clocked registers (TSPCR) can be constructed by cascading positive and negative TSPC latches (12 transistors).
- Positive edge-triggered TSPCR: negative latch + positive latch
- ✓ When clk=0, master stage is transparent, Q_M=In; slave stage is hold,
- ✓ When clk=0→1, master stage hold, slave stage transparent, Q=Q_M, QM value is passed to Q: positive edge-triggered.
- TSPC latch is dynamic: when latch in hold mode, output may be floating. → vulnerable to signal coupling and charge sharing.

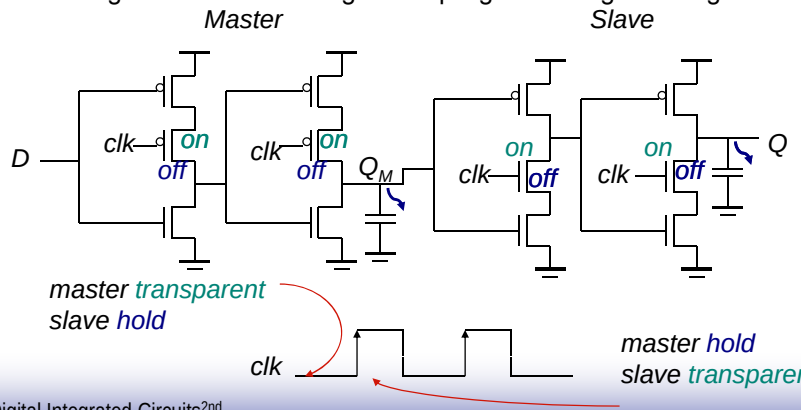


hold when clk = 1
transparent when clk = 0

transparent when clk = 1
hold when clk = 0

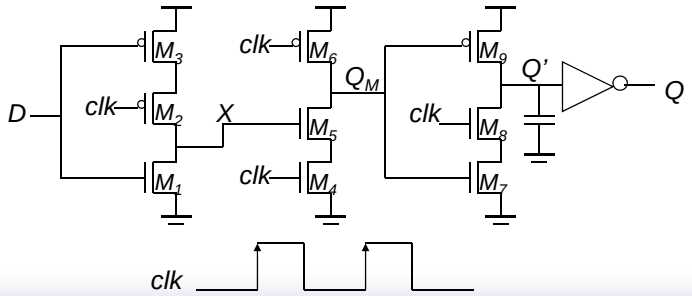
TSPC ET FF

- Positive edge-triggered TSPCR: negative latch + positive latch
- ✓ When clk=0, master stage is transparent, Q_M=In; slave stage is hold,
- ✓ When clk=0→1, master stage hold, slave stage transparent, Q=Q_M, Q_M value is passed to Q: positive edge-triggered.
- TSPC latch is dynamic: when latch in hold mode, output may be floating. → vulnerable to signal coupling and charge sharing.



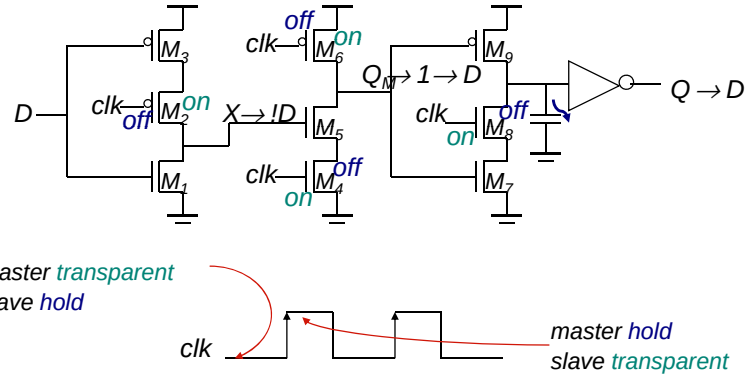
Simplified TSPC ET FF

- Simplified positive edge-triggered TSPCR (9 transistors): replace the 2 intermediate clocked inverters with a dynamic Φn inverter.
- ✓ When clk=0, X=D', Φn inverter in precharge phase, Q_M=1, M8 M9 off, Q' floating, Q remain its current value → hold mode.
- ✓ When clk=0→1, 1st clocked inverter is off, Φn inverter evaluates, Q_M=X', 2nd clocked inverter is on, Q'=Q_M'=D'(rising-edge), Q=D(rising-edge). → Positive edge-triggered register.



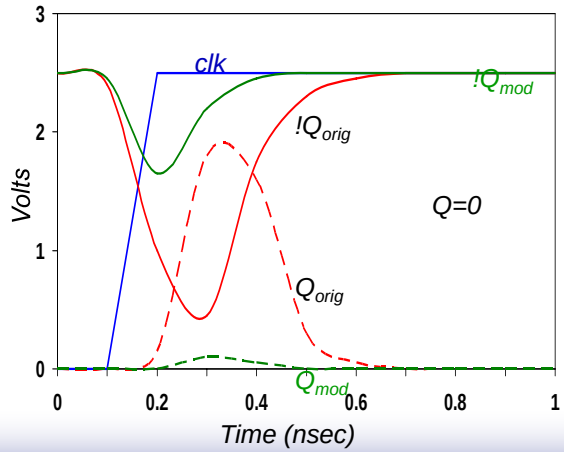
Simplified TSPC ET FF

- Working principle of simplified positive edge-triggered TSPCR:



Sizing Issues in Simplified TSPC ET FF

- Transistor sizing is critical for achieving correct functionality in TSPC register.
- With improper sizing, glitches may occur at output due to race condition when clk transitions from 0 to 1.



Transistor sizing

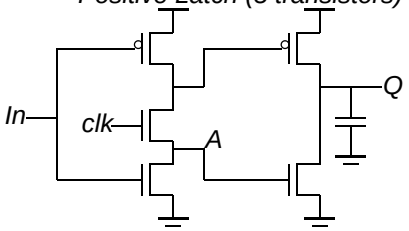
Original width
M_4, M_5 = 0.5μm
M_7, M_8 = 2μm

Modified width
M_4, M_5 = 1μm
M_7, M_8 = 1μm

Split-Output TSPC Latches

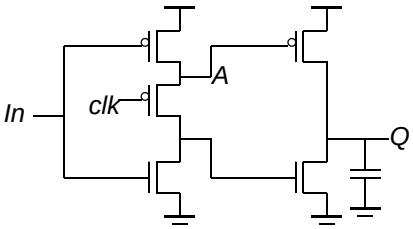
- TSPC latches can be further simplified to 5-transistor design, also called split-output TSPC latches.
- ✓ only the first inverter is controlled by clock.
- ✓ clock load is reduced by half.
- Ex: positive split-output TSPC latch.
- ✓ When clk=0, Q is floating, hold mode;
- ✓ when clk=1, two cascaded inverters, Q=In, transparent mode.

Positive Latch (5 transistors)



transparent when clk = 1
hold when clk = 0

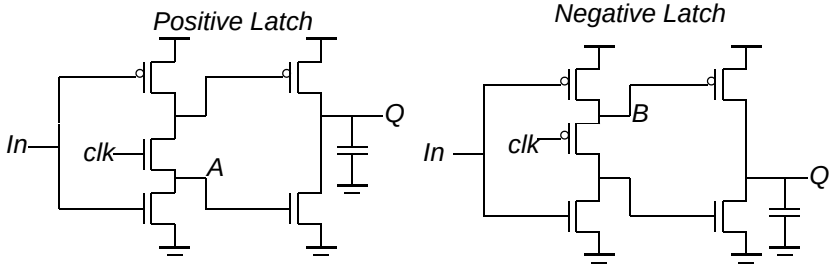
Negative Latch (5 transistors)



hold when clk = 1
transparent when clk = 0

Split-Output TSPC Latches

- Problem of split-output TSPC latches: not all node voltages in latch experience full logic swing.
  - ✓ For positive split-output TSPC latch: When In=0, clk=1, V_A=1, but $V_{A\_max}=V_{DD}-V_{Tn,}$ Otherwise, NMOS for clk is off.
  - ✓ For negative slpit-output TSPC latch: When In=1, clk=0, V_B=0, but $V_{B\_min}=|V_{Tp}|$, otherwise, PMOS for clk is off.

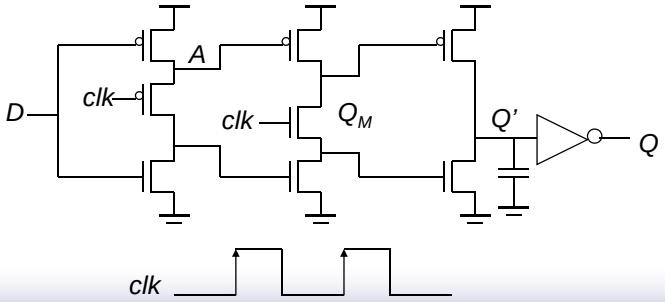


transparent when clk = 1
hold when clk = 0

hold when clk = 1
transparent when clk = 0

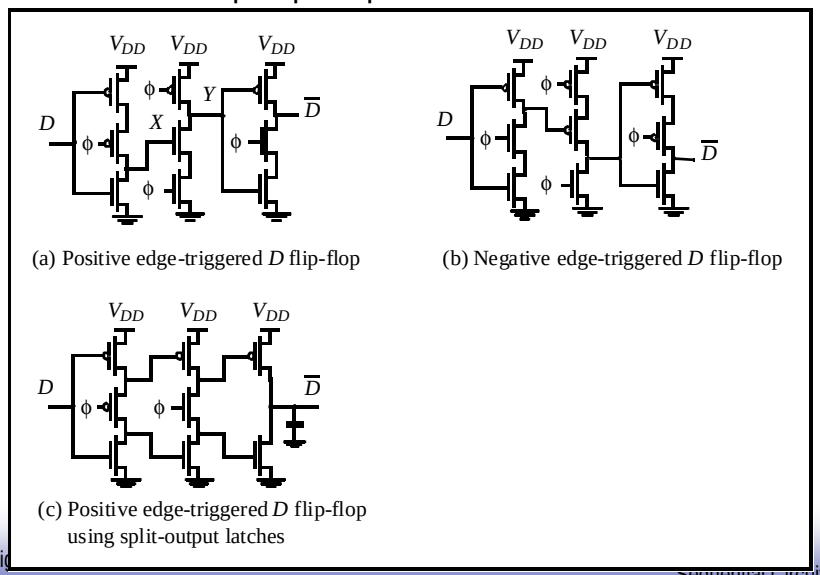
Split-Output TSPC ET FF

- Split-output TSPC register: cascading positive and negative split-out TSPC latches, but further share one inverter in between.
- Ex: positive split-output TSPCR.
- ✓ When clk=0, 1st inverter is on (transparent), A=D', 2nd inverter is off (hold), Q_M floating, Q' and Q remains previous state,
- ✓ When clk=0→1, 1st inverter is off(hold), 2nd and 3rd inverters are on (transparent), Q_M=A', Q'=Q_M'=A=D'(rising-edge), Q=D(rising-edge)→positive edge-triggered register.



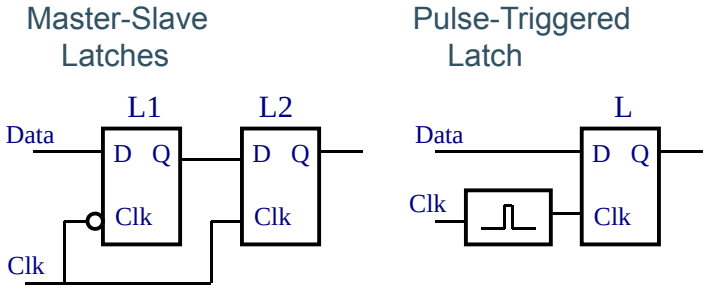
Master-Slave Flip-flops

- Master-slave flip-flops implemented with TSPC

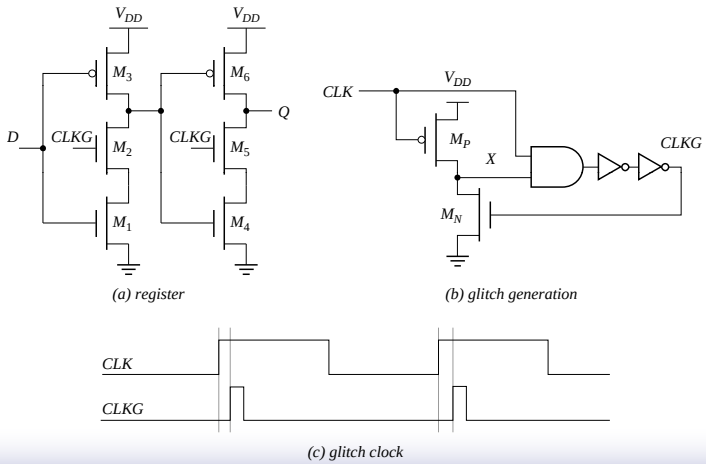


Optional: Pulse-Triggered Latches An Alternative Approach

Ways to design an edge-triggered sequential cell:

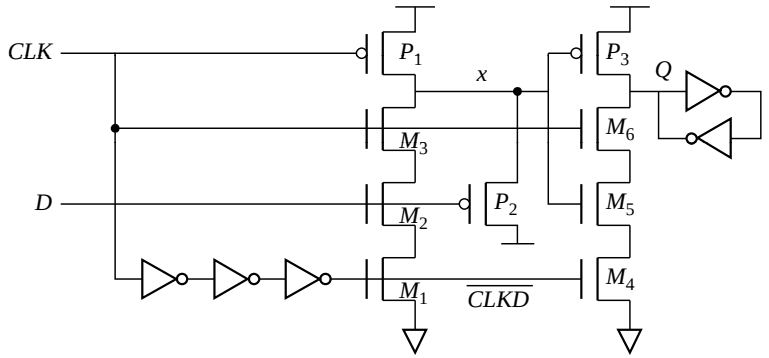


Optional: Pulsed Latches

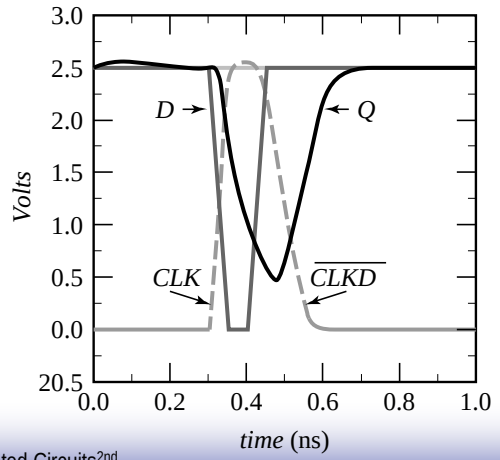


Optional: Pulsed Latches

Hybrid Latch – Flip-flop (HLFF), AMD K-6 and K-7 :

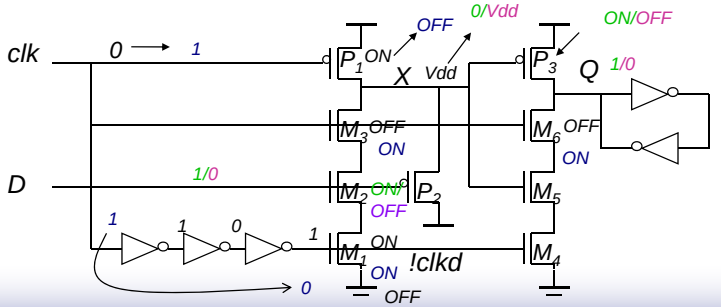


Optional: Hybrid Latch-FF Timing



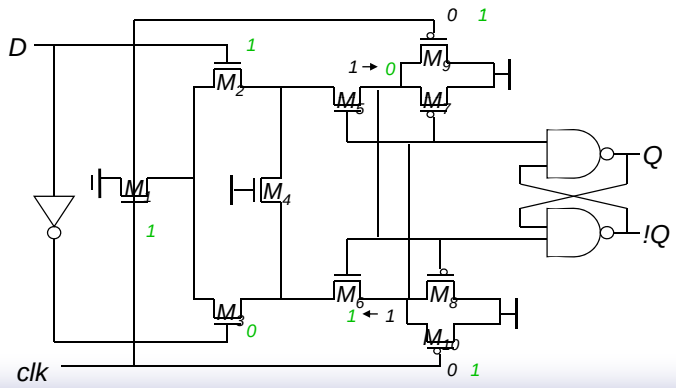
Optional: Pulsed FF (AMD-K6)

- *Pulse registers - a short pulse (glitch clock) is generated locally from the rising (or falling) edge of the system clock and is used as the clock input to the flipflop*
  - *race conditions are avoided by keeping the transparent mode time very short (during the pulse only)*
  - *advantage is reduced clock load; disadvantage is substantial increase in verification complexity*



Optional: Sense Amp FF (StrongArm SA100)

- *Sense amplifier (circuits that accept small swing input signals and amplify them to full rail-to-rail signals) flipflops*
  - *advantages are reduced clock load and that it can be used as a receiver for reduced swing differential buses*



Flipflop Comparison Chart

Name	Type	#clk ld	#tr	t_{set-up}	t_{hold}	t_{pFF}
Mux	Static	8 (clk-!clk)	20	$3t_{pinv}+t_{ptx}$	0	$t_{pinv}+t_{ptx}$
PowerPC	Static	8 (clk-!clk)	16			
2-phase	Ps-Static	8 (clk1-clk2)	16			
T-gate	Dynamic	4 (clk-!clk)	8	t_{ptx}	t_{o1-1}	$2t_{pinv}+t_{ptx}$
C²MOS	Dynamic	4 (clk-!clk)	8			
TSPC	Dynamic	4 (clk)	11	t_{pinv}	t_{pinv}	$3t_{pinv}$
S-O TSPC	Dynamic	2 (clk)	10			
AMD K6	Dynamic	5 (clk)	19			
SA 100	SenseAmp	3 (clk)	20			

Choosing a Clocking Strategy

- Choosing the right clocking scheme affects the functionality, speed, and power of a circuit
- Two-phase designs
  - + robust and conceptually simple
  - - need to generate and route two clock signals
  - - have to design to accommodate possible skew between the two clock signals
- Single phase designs
  - + only need to generate and route one clock signal
  - + supported by most automated design methodologies
  - + don't have to worry about skew between the two clocks
  - - have to have guaranteed slopes on the clock edges

Pipelining

❑ Pipelining: a popular design technique to accelerate operation of datapaths in digital processors.

❑ Ex: Datapath to compute log(|a+b|). (a,b: streams of numbers).

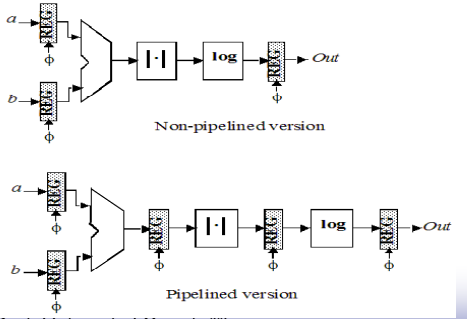
Without pipeline, minimum clock period T_{min} for correct evaluation is:

$T_{min,non-pipe}=t_{c-q}+t_{pd,logic}+t_{su}$

t_{c-q}, t_{su}: propagation delay and setup time of register,

$t_{pd,logic}$: worst case delay path through combinational network,

$t_{pd,logic}=t_{pd,add}+t_{pd,abs}+t_{pd,log}$



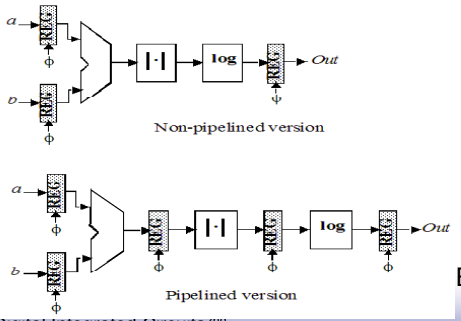
Clock Period	Adder	Absolute Value	Logarithm
1	a_1+b_1		
2	a_2+b_2	$\lvert a_1+b_1\rvert$	
3	a_3+b_3	$\lvert a_2+b_2\rvert$	$\log(\lvert a_1+b_1\rvert)$
4	a_4+b_4	$\lvert a_3+b_3\rvert$	$\log(\lvert a_2+b_2\rvert)$
5	a_5+b_5	$\lvert a_4+b_4\rvert$	$\log(\lvert a_3+b_3\rvert)$

Example of pipelined computations

Pipelining

❑ Pipeline improves resource utilization and increases functional throughput, with the cost of additional pipeline registers and increased latency.

❑ Ex: Datapath to compute log(|a+b|). (a,b: streams of numbers). With pipeline, minimum clock period T_{min} for correct evaluation is: $T_{min,pipe}=t_{c\text{-}q}+\max(t_{pd,add}, t_{pd,abs}, t_{pd,log})+t_{su}$



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	$\lvert a_4+b_4\rvert$	$\log(\lvert a_3+b_3\rvert)$

Example of pipelined computations (assembly-line fashion)

Pipelining

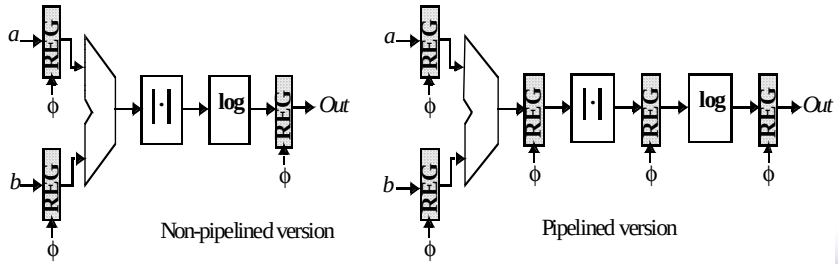
❑ Assume latch delay and setup time are ignorable with respect to logic delays, and all logic blocks have approximately the same propagation delay

$t_{p,add}=t_{p,abs}=t_{p,log}=t_p$

✓ Minimum clock period for non-pipelined design: $T_{min\_non\_pipe}=3t_p$

✓ Minimum clock period for pipelined design: $T_{min\_pipe}=t_p$

✓ The pipelined network outperformed the non-pipelined version by 3 times: $T_{min\_pipe}=T_{min\_non\_pipe}/3$



Latch-Based Pipeline

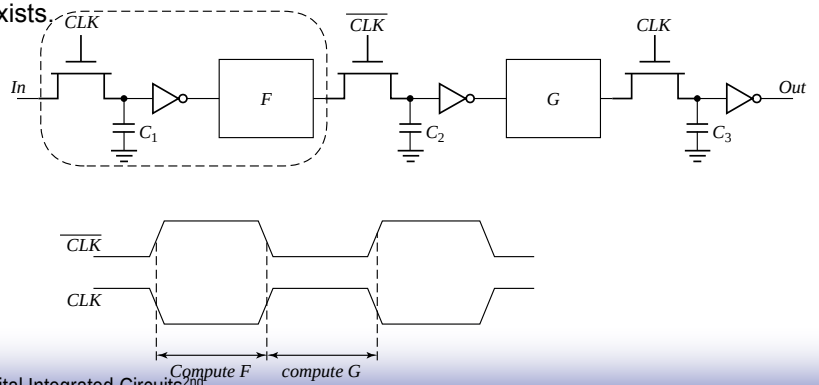
❑ Pipeline circuits can be constructed with level-sensitive latches or edge-triggered registers.

❑ Ex: Pipeline based on pass-transistor-based positive and negative latches (using clk-clk' two-phase non-overlapping clocks)

✓ when clk=0→1, input data is sampled on C_1, computation of F starts,

✓ when clk=1→0, result of F is stored on C_2 and computation of G starts.

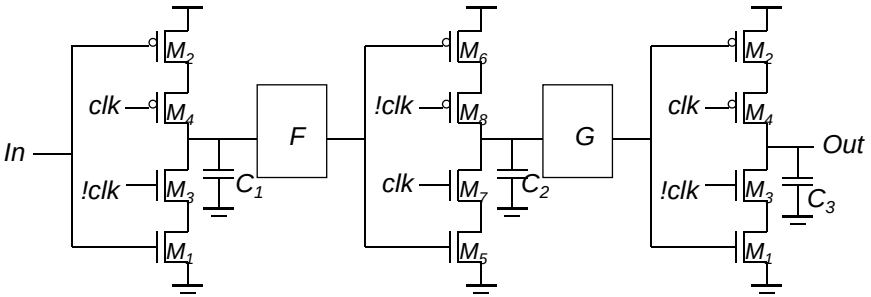
❑ Latch-based pipeline has race condition if overlap between clk and clk' exists



Pipelined Logic using C²MOS Latches

❑ Pipelining datapath using C²MOS latches

❑ Design constraint: A C²MOS-based pipelined circuit is race-free as long as all the logic functions F, G (implemented using static logic) between the latches are noninverting.



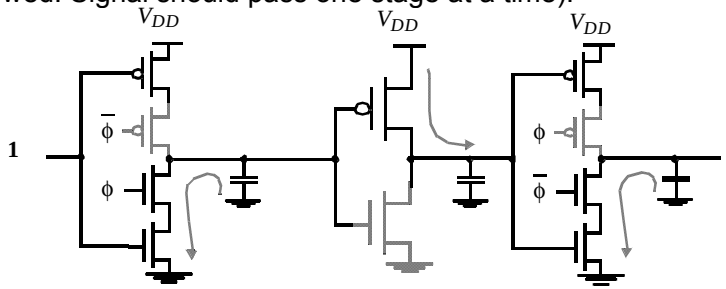
NORA Logic

What are the constraints on F and G?

Pipelined Logic using C²MOS Latches

❑ In a C²MOS based pipelined circuit, the only way a signal can race from stage to stage is when logic function F is inverting.

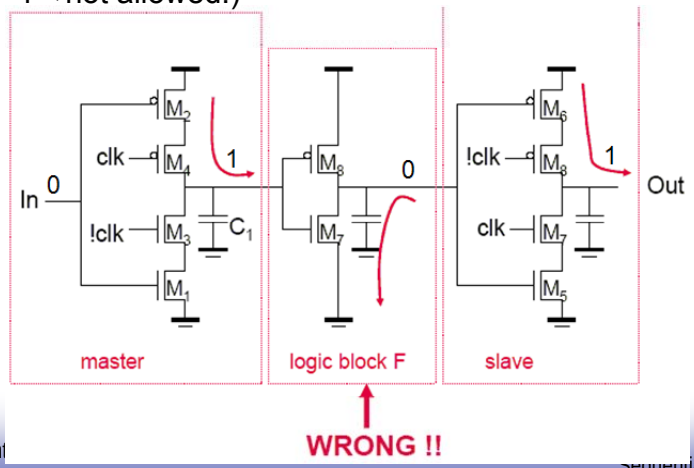
❑ Ex. If F is an inverter, in (1-1) clock overlapping, all C²MOS latches simplify to pure pull-down networks: input races to output. (clk=clk'=1, when In=1, it directly passes to Out, Out=0→not allowed! Signal should pass one stage at a time).



Number of a static inversions should be even

Pipelined Logic using C²MOS Latches

❑ Ex. If F is an inverter, in (0-0) clock overlapping, all C²MOS latches simplify to pure pull-up networks: input races to output. (clk=clk'=0, when In=0, it directly passes to Out, Out=1→not allowed!)



NORA CMOS Modules

❑ NORA-CMOS: NO-Race logic, used for fast pipelined datapaths using dynamic logic.

✓ It combines C²MOS pipeline registers and np-CMOS dynamic logic function blocks.

✓ Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a C²MOS latch.

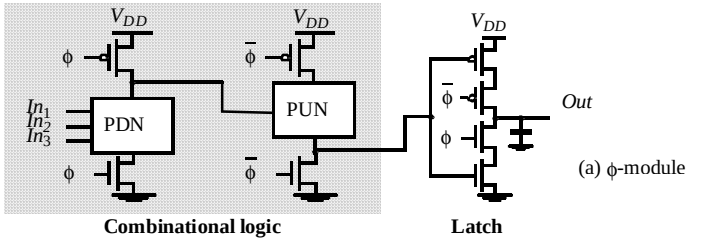
✓ Logic and latch are clocked such that both are simultaneously in either evaluation, or hold (precharge) mode.

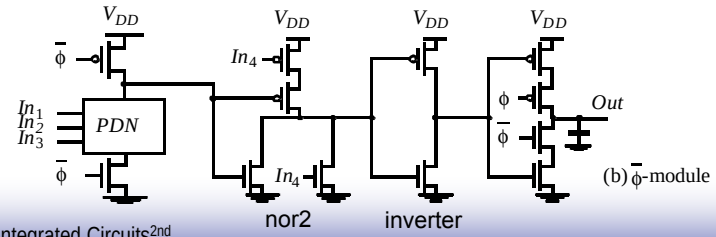
✓ A block that is in evaluation during Φ=1 is called a Φ-module, while the inverse is called a $\overline{\Phi}$-module.

	Φ block		$\overline{\Phi}$ block	
	Logic	**Latch**	**Logic**	**Latch**
Φ=0	Precharge	Hold	Evaluate	Evaluate
Φ=1	Evaluate	Evaluate	Precharge	Hold

NORA CMOS Modules

❑ Examples of NORA-CMOS modules



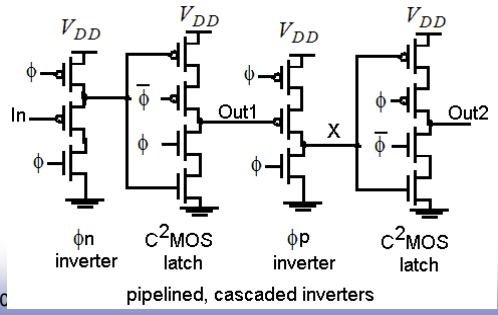


NORA CMOS Modules

❑ NORA datapath consists of a chain of alternating Φ and Φ modules

❑ While one class of module is precharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating.

❑ Data is passed in a pipelined fashion from module to module. Due to its design flexibility, extra inverter stages, as required in Domino-CMOS, are most often avoided.



pipelined, cascaded inverters

NORA CMOS Modules

❑ Design Rules

✓ The dynamic-logic rule: inputs to a dynamic Φn (Φp) block are only allowed to make a single 0→1 (1→0) transition during the evaluation period.

✓ The C²MOS rule: In order to avoid races, the number of static inversions between C²MOS latches should be even.

❑ Revised C²MOS rule:

✓ The number of static inversions between C²MOS latches should be even (in the absence of dynamic nodes); if dynamic nodes are present, the number of static inversions between a latch and a dynamic gate in the logic block should be even. The number of static inversions between the last dynamic gate in a logic block and the latch should be even as well.

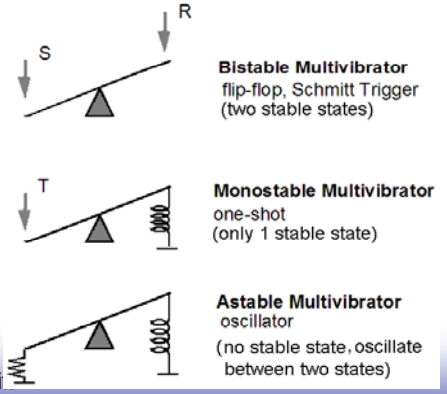
Multivibrator Cricuits

❑ Bistable circuits: two stable states. (e.g. register, latch)

❑ Astable circuits: oscillators (e.g. on-chip clock generation)

❑ Monostable (one-shot) circuits: used for pulse generators

❑ Schmitt trigger: hysteresis in DC characteristics, switching threshold depends on transition direction (low-to-high or high-to-low), good in noisy environments.



Non-Bistable Sequential Circuits—Schmitt Trigger

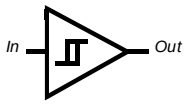
❑ Schmitt trigger:

✓ Slowly changing input leads to fast transition time at output

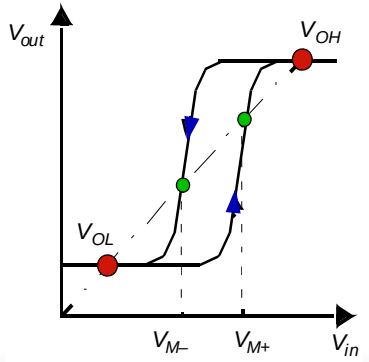
✓ VTC displays different switching thresholds for positive- and negative-going input signals - hysteresis.

✓ Switching thresholds for low-to-high and high-to-low transitions are called V_{M+} and V_{M-}.

✓ Hysteresis voltage = V_{M+}-V_{M-}



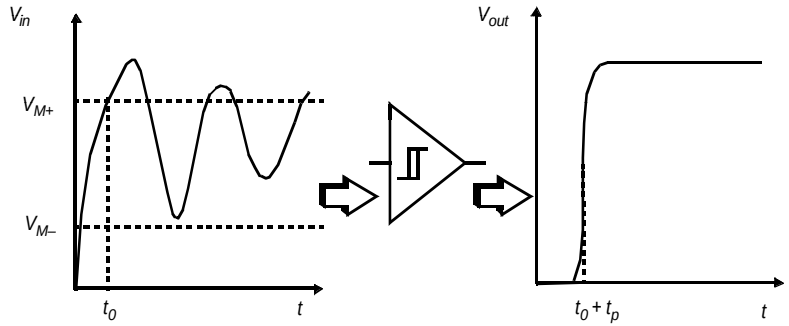
Schematic symbol



VTC of non-inverting Schmitt trigger

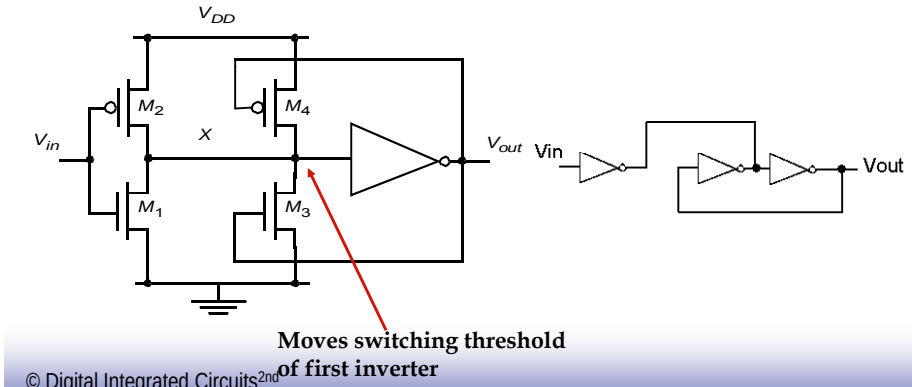
Noise Suppression using Schmitt Trigger

- Schmitt trigger can turn a noisy or slowly varying input to a clean digital output.
- Reason for hysteresis: positive feedback.

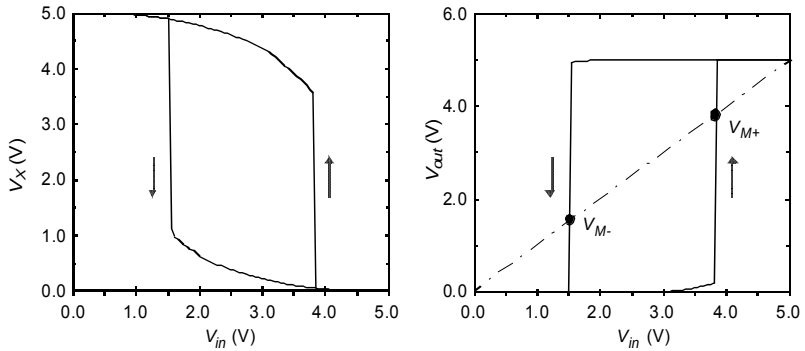


CMOS Schmitt Trigger

- CMOS Schmitt trigger design
- ✓ Switching threshold V_M of CMOS inverter depends on (k_n/k_p)
- ✓ Increasing (k_n/k_p) ratio reduces V_M, and vice versa
- ✓ Adapting the ratio depending on the direction of the transition results in a shift in switching thresholds → hysterisis

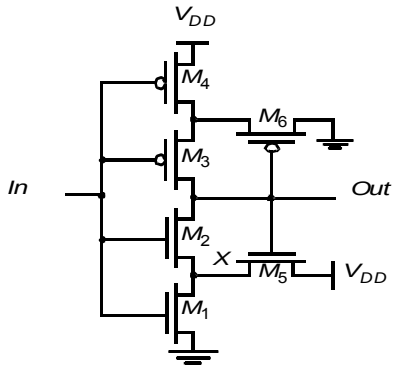


Schmitt Trigger Simulated VTC



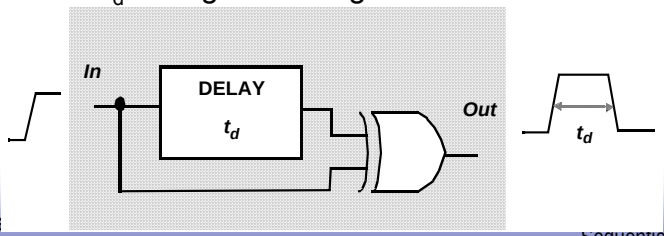
Alternative CMOS Schmitt Trigger Design

- Alternative CMOS Schmitt trigger design



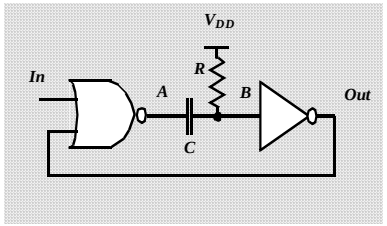
Transition-Triggered Monostable

- ❑ Monostable circuit: generates a pulse of a predetermined width every time the quiescent circuit is triggered by a pulse or transition event.
- ❑ Only 1 stable state (the quiescent one)
- ❑ Used to generate pulses of known length
- ❑ Monostable circuit using a delay element to control the duration of the pulse
  - ✓ In quiescent state, both inputs to XOR are identical → Out=0
  - ✓ An input transition causes XOR inputs to differ temporarily → Out=1 for t_d then goes low again.

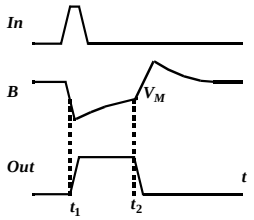


Monostable Trigger (RC-based)

- ❑ Monostable circuit using feedback combined with RC timing network to produce output pulse of fixed width



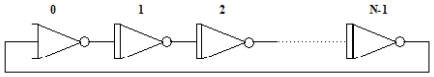
(a) Trigger circuit.



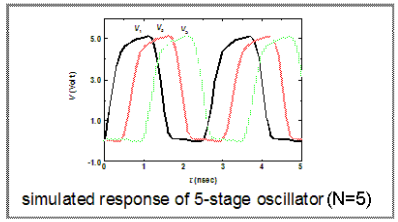
(b) Waveforms.

Astable Multivibrators (Oscillators)

- ❑ Astable circuits: no stable states. Output oscillates back and forth between two quasi-stable states with a certain period.
- ❑ Application: on-chip generation of clock signals.
- ❑ Example: ring oscillator
  - ✓ odd number of inverters connected in a circular chain
  - ✓ oscillates with period $T=2\times t_p\times N$



N-stage Ring Oscillator

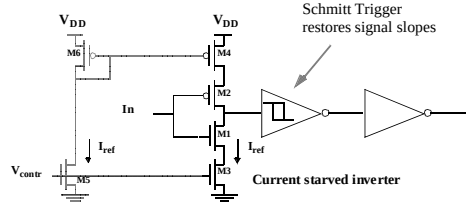


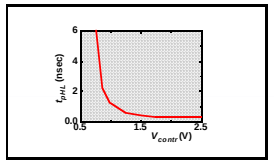
simulated response of 5-stage oscillator (N=5)

oscillation period T=2.57ns
propagation delay of each inverter: tp=T/2N=2.57ns/10=257ps

Voltage Controller Oscillator (VCO)

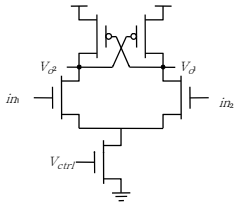
- ❑ Voltage-controlled oscillator (VCO): oscillation frequency is programmable (proportional to control voltage)
- ❑ Just replace inverters in ring oscillator with current-starved inverters
- ❑ Delay of current starved inverter can be adjusted by controlling the current available to (dis)charge C_L of the gate



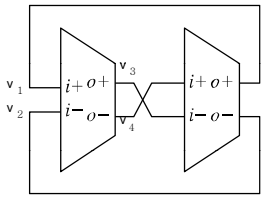


propagation delay as a function of control voltage

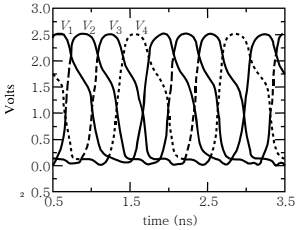
Differential Delay Element and VCO



delay cell



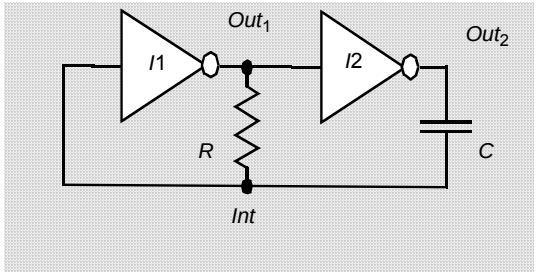
two stage VCO



simulated waveforms of 2-stage VCO

Relaxation Oscillator

- Relaxation oscillator: astable circuit composed of an RC network combined with feedback
- It has no stable state and oscillates with period T=2(log3)RC



$T = 2\ (\log 3)\ RC$