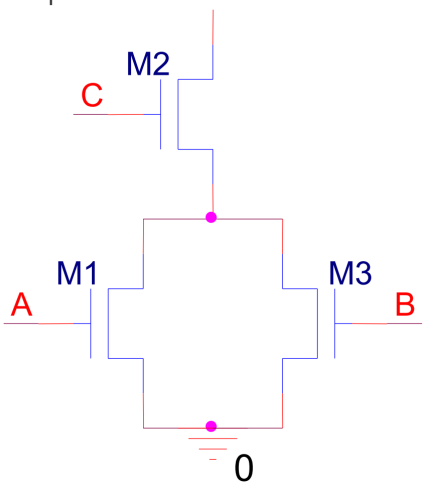
IT eszközök technológiája - 3. előadás

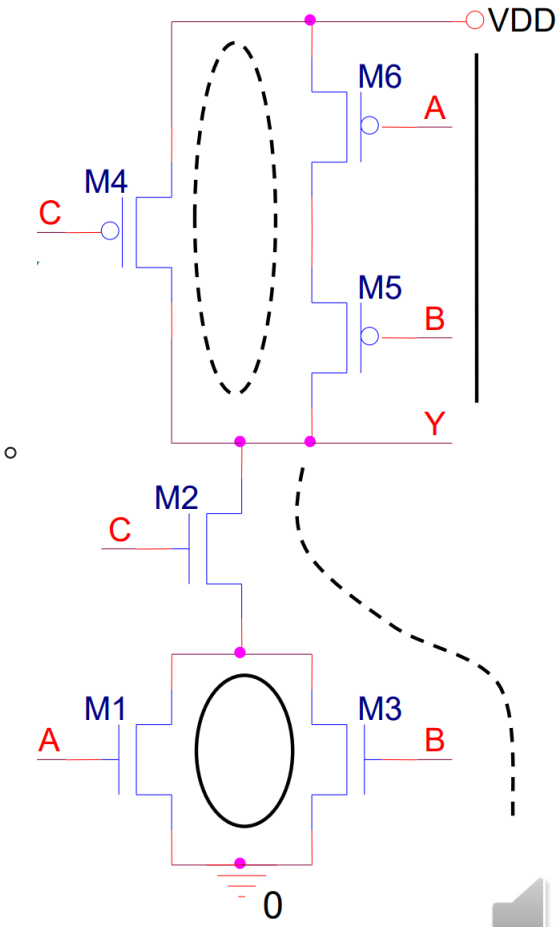
Komplex kapuk

- Tranzisztor szinten tudunk bonyolultabb logikai függvényeket megvalósítani
- Általában max 4 bemenettel, az ÉS illetve VAGY függvényeket kombináljuk
- pl:
  - OAI21 → Y = komplementer((A + B) C)
  - AOI22 → Y = komplementer(AB + CD)
- Általában n bemenet és/vagy kombinációja 2n tranzisztor segítségével megvalósítható
  - bemenet ponált és negált változata is rendelkezésre áll
- Többszintű realizációhoz képest
  - késleltetése kedvezőbb
  - kevesebb tranzisztort tartalmaz
  - hazárdmentes

Példa: Y = komplementer((A + B) C)

- pull-down network tervezése:
  - minden 0 értékhez a kimenet és föld között kell kapcsolat
  - függvényben szereplő összegeknek párhuzamosan, szorzatoknak sorba kapcsoltan kell szerepelniük
  - 

- pull-up network tervezése:
  - minden 1 értékhez a kimenet és a távfeszültség között kell kapcsolat
  - Bool algebrával kiszámolt érték: $Y = \overline{C(A+B)} = \bar{C} + \overline{A+B} = \bar{C} + \bar{A}\,\bar{B}$
  - Reciprok hálózat megközelítésből: pull-down ellenkezőjére kötünk mindent (sorosból párhuzamos, párhuzamosból soros)
- megoldás:



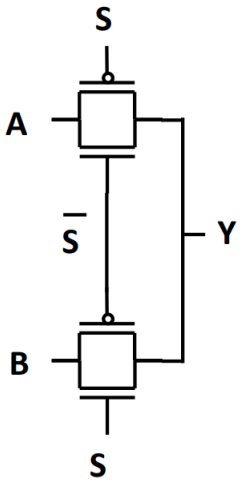
- ha kapuból csinálnánk:
  - nagyobb kérleltetése lenne, több tranzisztor kéne

Példa: teljes összeadó

- kritikus út → **carry** (az átvitt érték mindig kell a következő helyiértékhez)
  - megodás pl: előbb számítunk átvitelt, aztán összeget
  - C_{OUT} = AB + C(A+B) → így a C-t a lehető legkevesebb helyre kell bekötni
  - S = ABC + (A + B + C) kompl(C_{OUT})

CMOS transzfer kapu

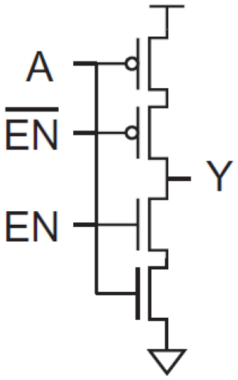
- jelfolyam útjába helyezett kapcsoló
- n és p típusú tranzisztor kapcsolásából áll, amikre ellentétes jelet kötünk
  - C = 0 → szakadás
  - C = 1 → vezetés
- kiválasztók-ra nagyon jó (kétbemenetű multiplexer)



- pozitívum: komplex kapus 8 szükséges tranzisztor helyett 4 is elég!

Órajel vezérelt CMOS

- Háromállapotú (tri-state kapu)
  - EN = 0 → a kimenet lebeg (nagyimpedanciás)
  - EN = 1 → a kimenet a bemenet inverze

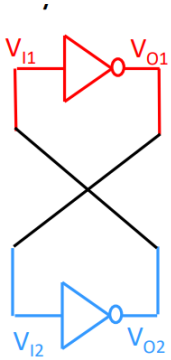


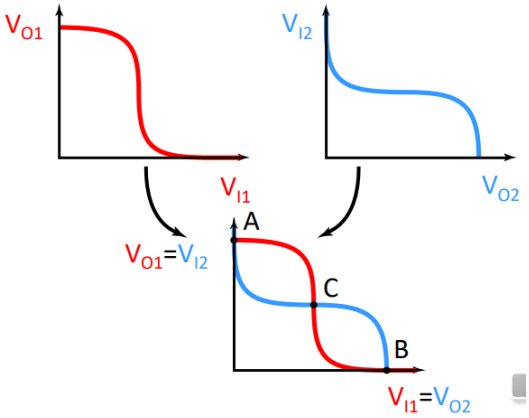
CMOS tárolók

- **Latch** → szintvezérelt
  - EN = 1 → átlátszó
  - bemeneti változás a kimenetre jut (kis késleltetés után)
- **Flip-flip** → élvezérelt
  - beírás az órajel fel vagy lefutó élére történik

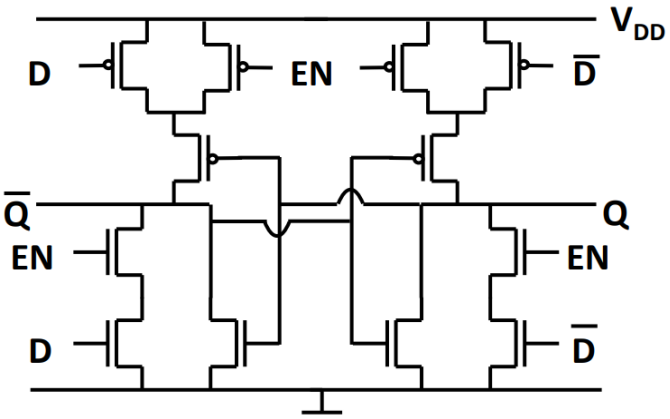
Tárolás alapelve

- két állapotú (bistabli) áramkör alapja: két gyűrűbe kapcsolt inverter

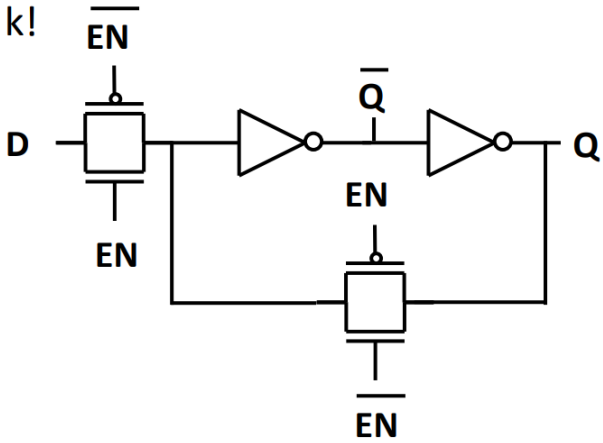




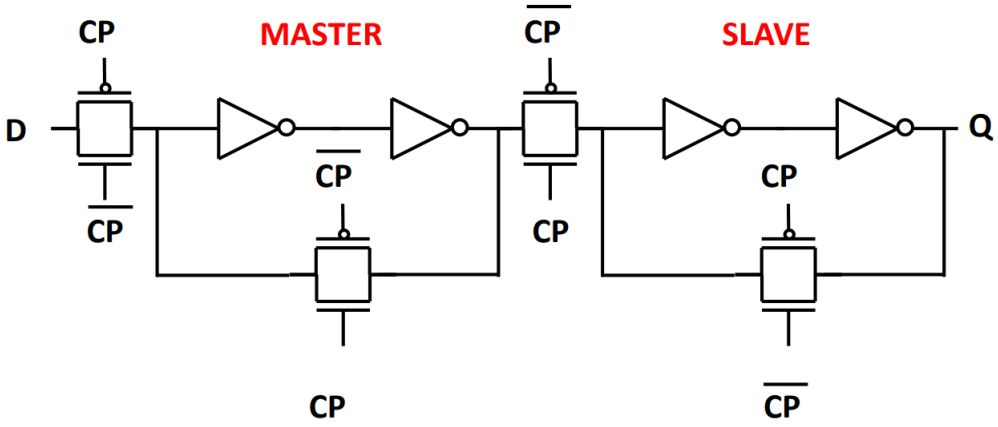
- **D-latch**
  - komplex kapukkal, 12 tranzisztorból



  - transzfer kapuval, 8 tranzisztorból



    - EN = 1, bal kapu nyitva, D = Q, visszacsatoló (lenti) kapu zár
    - EN = 0, bal kapu csukva (nincs írás), visszacsatoló ág él → létrejön a két gyűrűbe kapcsolt inverter
- **D-flipflop**
  - master-slave flip-flop két sorbakötött, ellenütemű órajellel vezérelt latch
  - CP = alacsony szint, első tároló átlátszó
  - CP = felfutó él, master nem átlátszó, tartalom slave-be íródik
  -

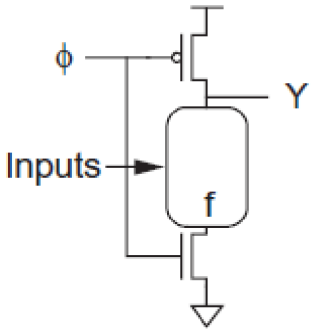


- **t_{setup}** → órajel **aktív éle előtt** a mintavételezett adatnak <u>stabilnak kell lennie</u>
  - $t_{setup} > t_{PGTG} + 2t_{PGINV}$ (1 transzfer kapu és 2 inverter)
- **t_{hold}** → órajel **aktív éle után** ennyi ideig <u>nem szabad megváltoznia</u>
  - $t_{HOLD} > t_{PGTG}$ (míg a master transzfer kapuja elzáródik)
- aszinkron *clear* és *set* is megvalósítható → kétbemenetű NAND kapukkal

Nagysebességű CMOS logika

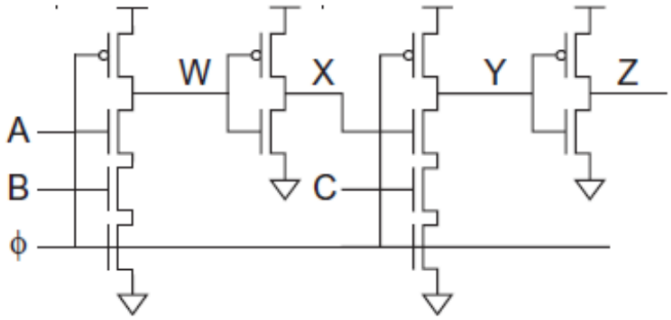
- statikus CMOS logika késleltetée: **t_{pd} ~ CV / I**
  - gyorsítás <u>kapacitás csökkentésével</u>, vagy az <u>áram növelésével</u> (közben méret is csökkenjen)
  - logikai szint távolságának csökkentése → SCL = source-coupled logic
  - differenciális logikai → két feszültségszint különbségének előjele adja a logikai értéket
  - **szórt kapacitás** kihasználása logikai szint **ideiglenes tárolására**
    - területet nyerünk, kapacitást csökkentünk

CMOS dominó logika

- csak pull-down network van
- kimenetet terhelő szórt kapacitást előtöltjük
- 

  - Φ = 0 → előtöltés tápfeszültségre
  - Φ = 1 → kiértékelés (pull-down network az input alapján vagy lehúzza a kimenetet, vagy békén hagyja)
- előnyök:
  - N bemenetű függvényhez N+2 tranzisztor szükséges
  - gyorsabb → előző fokozatot kisebb kapacitás terheli
- hátrányok → kapuk összekapcsolásakor vigyázni kell

- kiértékeléskor egy rövid ideig 1 van akkor is, ha 0-nak kellene lennie (ez kinyitja a következő kapu tranzisztorát → töltés sérülés lesz)
- megoldás: inverteren keresztül kötjük a kimenetet a következő kapu bemenetére



- **dominó logika** 1,5x gyorsabb a CMOS-nál (csak pull-down network dolgozik benne)

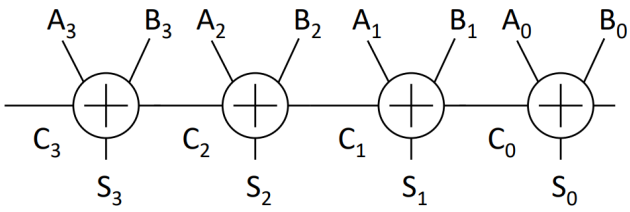
Dinamikus D flip-flop

- 2db órajellel vezérelt invertert összekapcsolunk, ezeket ellentétesen vezéreljük
- info → master és slave közti szórt kapacitásban tárolódik
- következmény → van egy minimális órajel, amit használni kell (ha annál kisebb használunk, akkor elszivárog a kapacitás annyi idő alatt)

Az adatút elemei (*már nem zh anyag!*)

Összeadó

Ripple carry adder

- több bites számok összeadása → sorba kapcsolt teljes összeadók
- nem lesz túl gyors (egymásra várnak) → utolsó carry elkészülése + utolsó összeg elkészülése
- kritikus út = carry terjedése
- egy bizonyos bitszám felett túl lassú lesz
- 

- gyorsítása:
  - generate → G = AB
  - propagate → P = A ⊕ B

Carry skip adder

- számítsuk ki egy blokkra a propagate-t és ezzel vezéreljünk egy multiplexert, ami vagy az előző fokozat átvitelét vagy a generált átvitelt teszi a kimenetre
- propagate = és kapu → könnyen kiszámítható
- késleltetés: O(N/K)

Carry-look-ahead

- fokozatonként történik a carry gyors kiszámítása, minden fokozatra P és G számítása
- késleltetés: P és G kiszámítása (részben párhuzamosan történik)

Carry-select

- fokozatonként kiszámolja az összeget átvitel és átvitel nélkül, amiből az előző fokozat átvitele választja ki a végeredményt egy multiplexer segítségével

Funnel shifter

Kombinációs szorzó