BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika
VIMIAA01

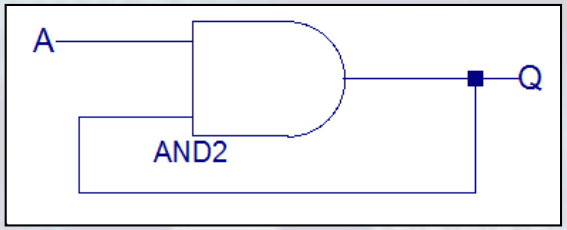
Fehér Béla
BME MIT

Sorrendi hálózatok

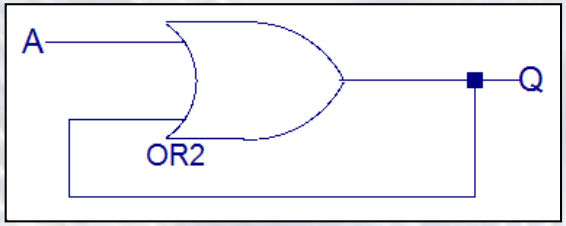
- **Az eddigiekben megismert digitális áramkörök egyszerű, állapot ill. memóriamentes egységek voltak**
- **Sok esetben a feladatok megoldásához ez nem elegendő, a rendszer viselkedését befolyásolja az addigi előélete is**
- **Példa: Laboratórium főkapcsoló**
  - BE: Bekapcsol
    - Bekapcsolva marad
  - KI: Kikapcsol
    - Kikapcsolva marad
- **Emlékező, memória funkció**
  - Korábbi állapot fenntartása
    → Kimenet visszacsatolásával

Elemi bit tárolók

- **Visszacsatolás**
  - **ÉS** kapu visszacsatolva
  - TFH kezdetben Q = 0
  - Ha A = 0, Q = 0 marad, de A = 1-re is Q = 0 marad
  - TFH kezdetben Q = 1
  - Amíg A = 1, Q = 1 marad, de A = 0-ra Q = 0 lesz …

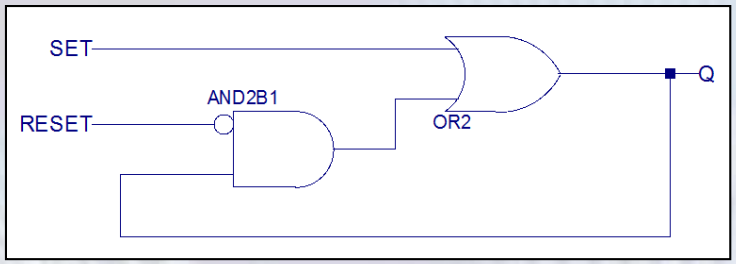


  - **VAGY** kapu visszacsatolva
  - TFH kezdetben Q = 0
  - Ha A = 0, Q = 0 marad, de A = 1-re Q = 1 lesz!
  - Amíg A = 1, Q = 1 marad, de ha A = 0 lesz, Q akkor is 1 marad....
- **Nem vezérelhetőek**



Elemi bit tárolók: SR latch

- **A visszacsatoló kört is vezérelni kell!**
  - SET - RESET latch
  - A bemeneti jel és a visszacsatolás is vezérelhető
  - A működés jellemzése vezérlési táblával, amit itt **állapotátmeneti tábl**ának hívunk
  - **Időbeli információt is tartalmaz Q(t) → Q(t⁺)**
    - Azaz a bemenetek és a jelenlegi állapot függvényében soronként adja meg a „következő” állapot értékét
    - Tömörítve is használjuk, könnyebb értelmezni
      - Első sor: Tartja az értékét
      - További három sor: Bármi (x) a Q(t) kimenet, a jelölt értéket kapja

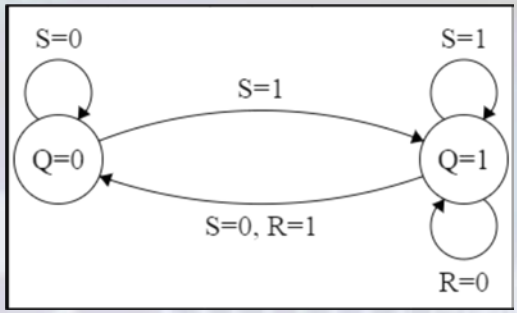


S(t)	R(t)	Q(t)	Q(t⁺)
0	0	Q(t)	Q(t)
0	1	x	0
1	0	x	1
1	1	x	1

S(t)	R(t)	Q(t)	Q(t⁺)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

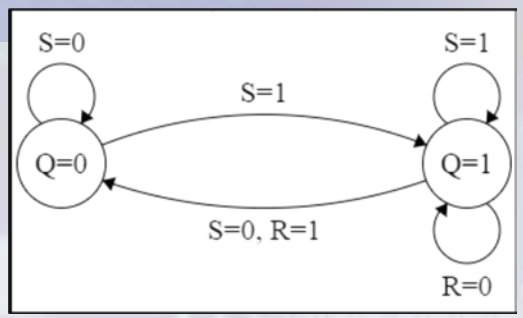
Állapotdiagram

- **Állapotdiagram: Grafikus leírási forma**
  - Az állapotátmeneti tábla és az állapot-diagram egymással ekvivalens leírás
- **Az állapotdiagram elemei**
  - **Kör:** A rendszer egy **állapotát** reprezentálja. Beleírva az állapot neve, kódja, értéke vagy más, az állapothoz tartozó információ. (Pl. az adott állapotban a kimeneti jel értéke.)
  - **Nyilak:** Az **átmeneteket** jelzik két állapot között. A nyílra írjuk, hogy az adott átmenet milyen bemeneti feltételek mellett következik be. A feltételnek egyértelműnek kell lennie. Érvényes a saját magába visszatérő átmenet is. Ez a stabil, fennmaradó állapotot jelzi.

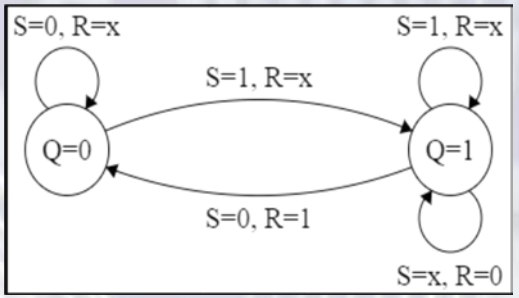


S(t)	R(t)	Q(t)	Q(t+)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

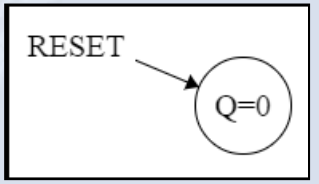
Állapotdiagram



- **Állapotdiagram elemei:**
- **Az átmenetek feltételeit megadhatjuk teljes formában is.**
  - Pl. A Q=1 állapot fennmarad az SR = 00, 10 és 11 feltételek teljesülése esetén

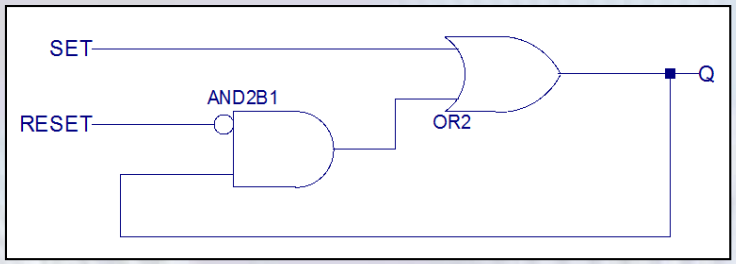


- **Szabályok (nincs túl sok):**
  - Minden állapotban az összes átmeneti feltétel VAGY kapcsolata 1 → **Az állapotgép teljesen specifikált**
  - Minden állapotban az átmeneti feltételek páronkénti ÉS kapcsolata 0 → **Az állapotgép determinisztikus**
  - Lehet egy kitüntetett kezdeti RESET állapot, ezt egy forrás nélküli nyíl jelzi

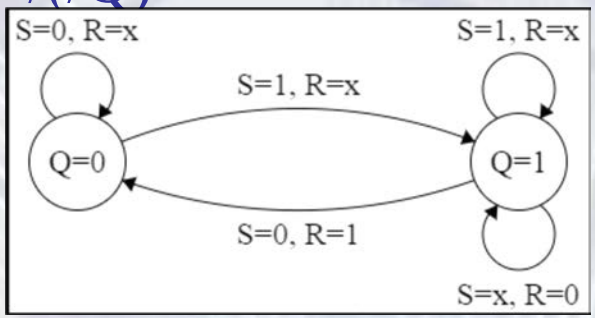


Elemi bit tárolók:SR latch

- **A visszatérve az eredeti áramkörre**
  - SET - RESET latch
  - A SET bemenet domináns
    - Így értelmezett az SR=11 bemeneti kombináció is
    - Ha szükséges /Q kimenet, azt egy külön inverterrel kell előállítani → Így mindig teljesül a Q = /(/Q)

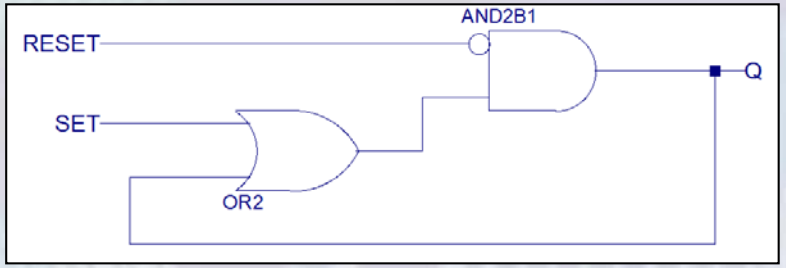


  - Persze lehetnek problémák
  - Az állapotdiagram szerint Q=1-ben maradunk, ha SR = 00, 10,vagy 11
  - De ha SR=00-ból SR=11-be kapcsolnánk, és esetleg a bemeneten a 00-01-11 sorozatot érzékeli, akkor bármi történhet! $Q(t^+) = ?$
  - **A BEMENETEN EGYIDŐBEN CSAK EGY JEL VÁLTOZHAT!**

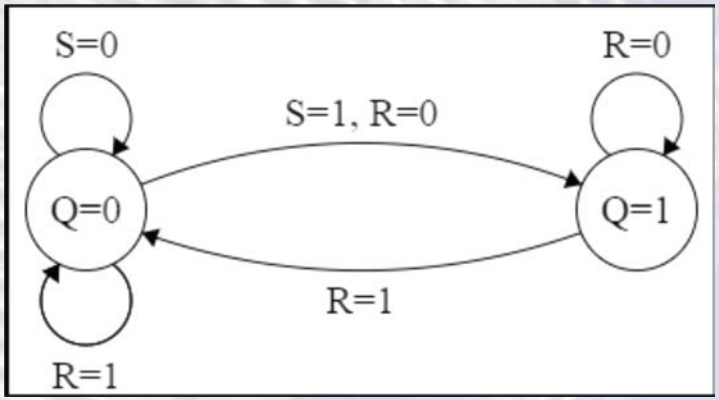


Elemi bit tárolók:SR latch

- **A dualitási szimmetria miatt**
  - RESET - SET latch
  - A RESET bemenet domináns
    - Itt is értelmezett az SR=11 bemeneti kombináció is
    - Egyébként tulajdonságai (pro és kontra) ugyanazok, az SR= 00 – 11 átmenet problémás lehet 00-10-11 $Q(t^+)$=?
  - **A BEMENETEN EGYIDŐBEN CSAK EGY JEL VÁLTOZHAT!**



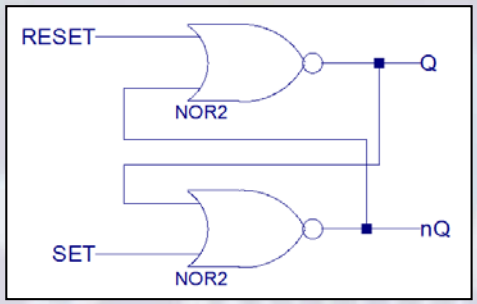
S(t)	R(t)	Q(t)	$Q(t^+)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

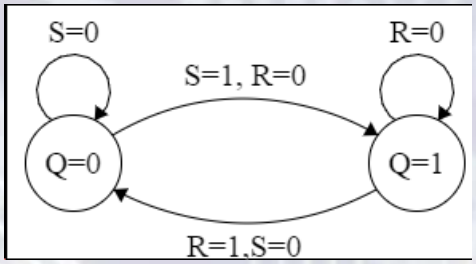


S(t)	R(t)	Q(t)	$Q(t^+)$
0	0	Q(t)	Q(t)
0	1	x	0
1	0	x	1
1	1	x	0

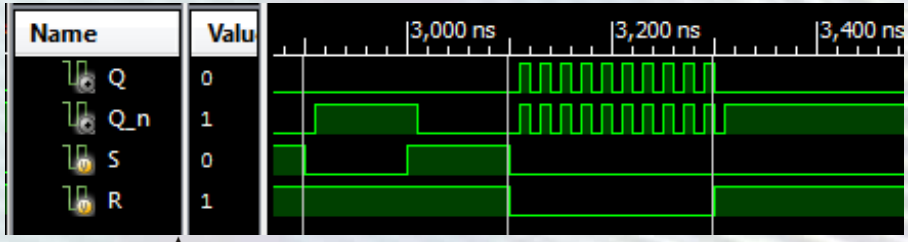
Elemi bit tárolók: NOR SR latch

- **Szimmetrikusan visszacsatolt latchek**
  - **NOR** kapuval, ponált vezérlés
  - Szép szimmetrikus felépítés
  - Egy komoly hiba: **Q** és **nQ** nem egymás negáltjai, ha SR = 11
    - Boole axióma sérül, de „visszahozható”
    - Lehet, hogy nem is 2 állapot van, hanem 3? (Q,nQ → 01, 10, 00) Ez már egy bonyolultabb állapotdiagram lenne
  - Viszont az SR=11 → 00 átmenet itt is hibás működést okozhat





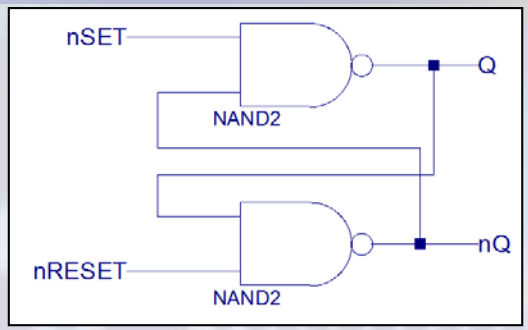
S(t)	R(t)	Q(t)	$Q(t^+)$	$/Q(t^+)$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

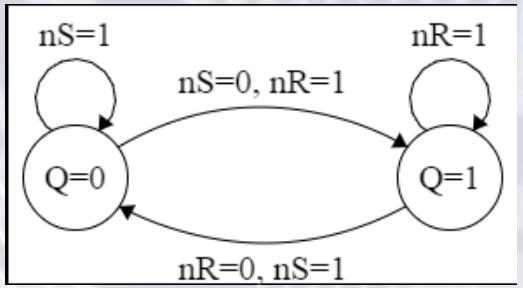


S(t)	R(t)	Q(t)	$Q(t^+)$	$/Q(t^+)$
0	0	Q(t)	Q(t)	/Q(t)
0	1	x	0	1
1	0	x	1	0
1	1	x	0	0

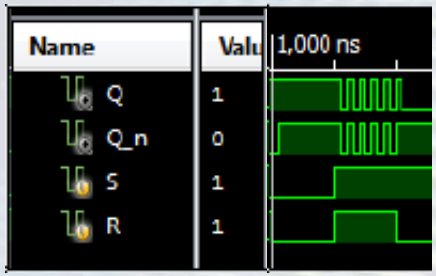
Elemi bit tárolók: NAND SR latch

- **Szimmetrikusan visszacsatolt latchek**
  - **NAND** kapuval, negált vezérlés (nS, nR aktív alacsony)
  - Ez is szép szimmetrikus
  - **Q** és **nQ** itt is problémás
  - Kritikus bemenet a nSnR=00, bizonytalan lehet a kilépés
  - Általában az SR latchek „veszélyesek”, zárjuk ki a nem megengedett vezérlést!





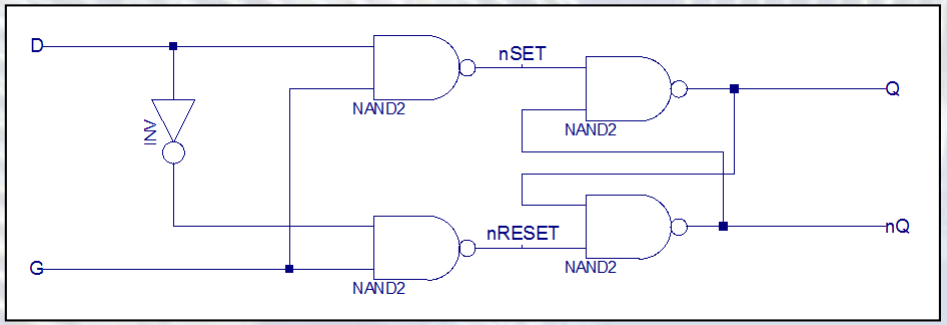
nS(t)	nR(t)	Q(t)	$Q(t^+)$	$nQ(t^+)$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0



nS(t)	nR(t)	Q(t)	$Q(t^+)$	$nQ(t^+)$
0	0	Q(t)	1	1
0	1	x	1	0
1	0	x	0	1
1	1	x	Q(t)	nQ(t)

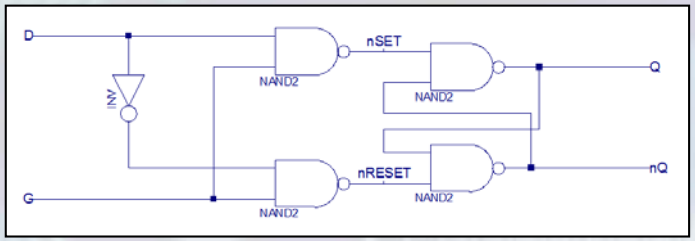
Elemi bit tárolók: DG latch

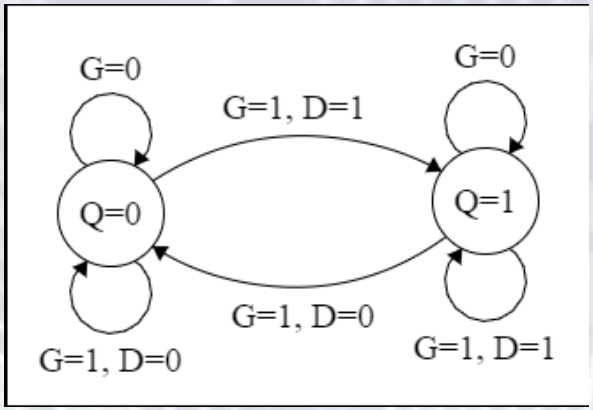
- **SR latch probléma: A bemenetek azonos vezérlése**
- **Megoldás: A bemeneti jeleket „kapuzzuk” egy G kapuzó jellel, továbbá biztosítjuk, hogy az SR vezérlőjelek mindig ponált/negált vezérlést kapjanak**
- **A DG latch : Kapuzott adatbit tároló**
  - **A G kapuzó jel szabályozza, hogy a D adatbemenet mikor befolyásolhatja a kimenetet**
    - A bemeneti inverter miatt SR =01 vagy 10 fordulhat csak elő, illetve SR = 00, ha G=0
    - Mindig legális vezérlés



Elemi bit tárolók: DG latch

- **DG latch állapotdiagram:**
  - Jól látható a G jel domináns szerepe
  - Sajnos az egység a G=1 magas szint esetén transzparens, azaz D minden változása közvetlenül átjut a kimenetre
  - Jobb lenne egy olyan elem, ami egy adott pillanatban mintavételez és utána azt az értéket tartja, egy újabb mintavétel idejéig
    → Ez lesz az élvezérelt D flip-flop

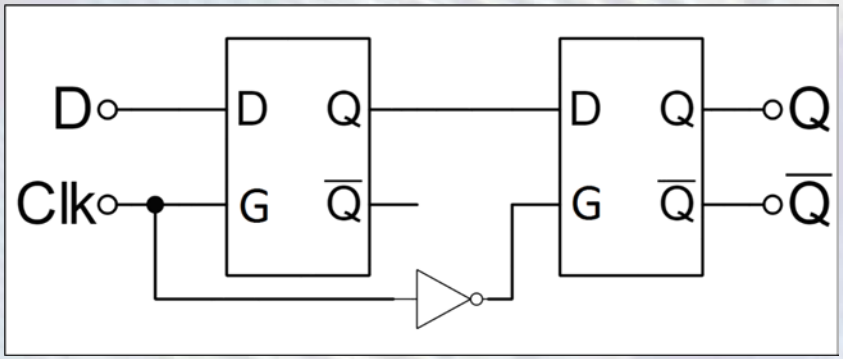




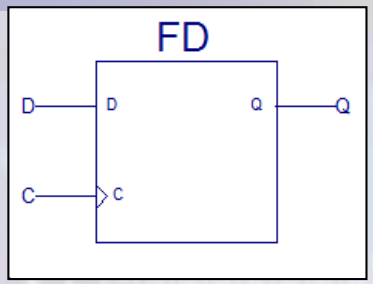
G(t)	D(t)	Q(t)	$Q(t^+)$
0	0	Q(t)	Q(t)
0	1	Q(t)	Q(t)
1	0	x	0
1	1	x	1

Az igazi bit tároló: a DFF

- **A DFF, azaz D flip-flop**
  - Tehát amit várunk: Egy C (Clk, ütemező óra) vezérlőjel ↑ élének hatására mintavételezze a bemeneti D adatbitet és ezt az értéket tárolja a következő ilyen eseményig.
  - Egy ilyen egység felépíthető pl. a már ismert DG latch áramkörből → Élvezérelt Master-Slave DFF
    - A belső felépítést nem tárgyaljuk



A DFF



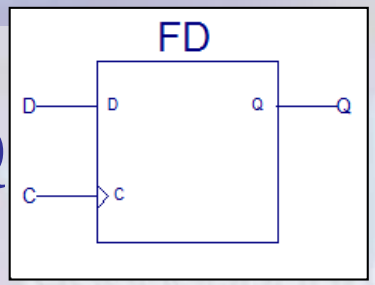
- **A továbbiakban a DFF egy kész egység**
  - Nem gondolkodunk a belső felépítésén
  - Verilog viselkedési leírással építjük be:
  - **DFF funkció**: Az órajel minden felfutó élénél vegyen mintát a D adat bitből és ezt tartsa a kimenetén a teljes órajel periódusban a következő felfutó élig

```
reg Q;
always @ (posedge clk)
   Q <= D;
```

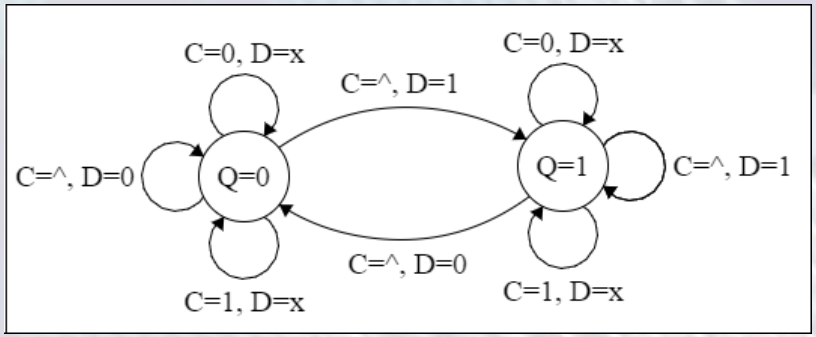
  - Szó szerint ezt a kódot írtuk le, így kell kiolvasni is
  - A működés tömör állapottáblája:
    - Ha az órajel 0 vagy 1, akkor a kimenet stabil, nem változik, de a felfutó élnél, bármilyen állapotban van, a következő állapota a D értéke legyen.

C	D(t)	Q(t)	$Q(t^+)$
0	x	Q(t)	Q(t)
↑	0	x	0
↑	1	x	1
1	x	Q(t)	Q(t)

A DFF



- **A DFF állapotdiagramja**
  - 2 bemenete van: C, D, és 1 állapotváltozó Q
  - Ha mindent jelölni akarunk, akkor az állapotdiagram bonyolult. Nem segíti a megértést

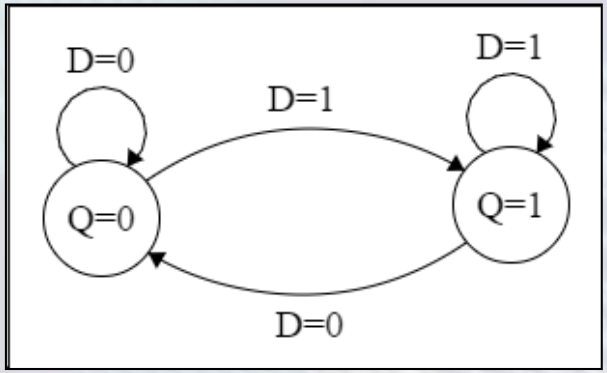


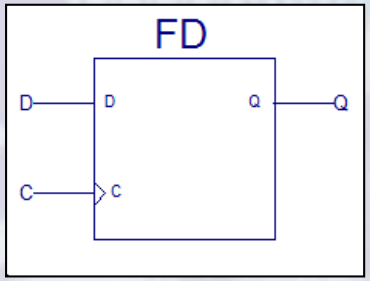
- **A továbbiakban egyszerűsítünk:**

  Minden további állapotdiagram órajelre működik, azaz szinkron működésű. Ezért a C órajelet az átmenetek feltételeinek felírásából elhagyjuk, de gondolatban hozzá képzeljük!

A DFF

- **A DFF állapotdiagramja**
- **Az egyszerűsített állapotdiagram áttekinthetőbb.**



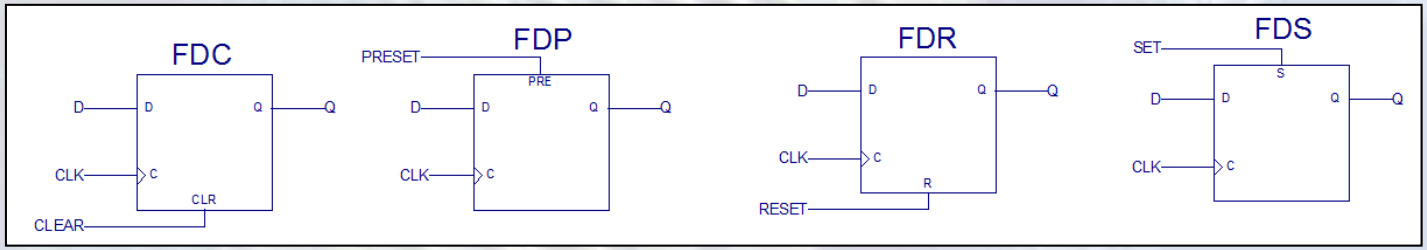


- Értelmezés: Az órajel felfutó élénél, ha D(t)=1, akkor a $Q(t^+) = 1$ lesz, ha D(t)=0, akkor $Q(t^+) = 0$.
  Egyesítve $Q(t^+) = D(t)$
- Tehát az állapotváltás nem akkor van, amikor a D adatbemeneten a jel 0-ba vagy 1-be vált, hanem akkor amikor ezt a változást az órajel ↑ él érvényesíti.

A DFF

- **Egy apró probléma: Bekapcsoláskor a Q 0 vagy 1 lesz?**
  - Ha a rendszert újra akarjuk indítani, hogyan lehet az induló állapotot beállítani?
  - Tehát hiányzik valami alaphelyzet beállító bemenet.
  - Összesen 4 lehetőségünk van:
  - Az alaphelyzet beállító jelre Q = 0 és azonnal beáll CLEAR
  - Az alaphelyzet beállító jelre Q = 1 és azonnal beáll PRESET
  - Az alaphelyzet beállító jelre Q = 0 az órajel ↑ élre RESET
  - Az alaphelyzet beállító jelre Q = 1 az órajel ↑ élre SET

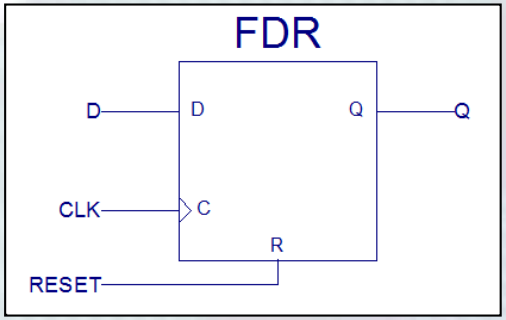
Aszinkron CLEAR / PRESET Szinkron RESET / SET



A DFF alaphelyzet beállító jelei

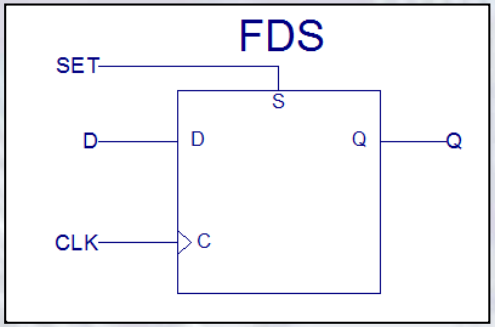
- **A továbbiakban a SZINKRON vezérlést használjuk.**

DFF + szinkron RESET




```
reg Q;
always @ (posedge clk)
   if (RESET)   Q <= 1'b0;
   else         Q <= D;
```

DFF + szinkron SET

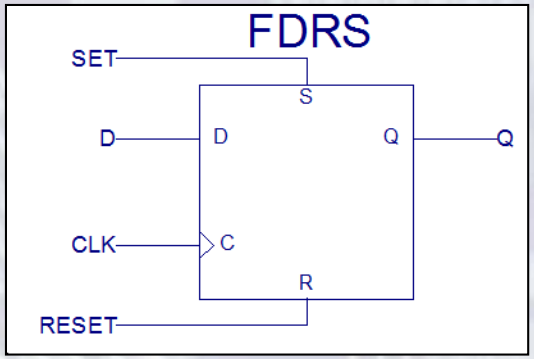



```
reg Q;
always @ (posedge clk)
   if (SET)     Q <= 1'b1;
   else         Q <= D;
```

- **Verilog HDL kódminta:**
  - Az érzékenységi listában csak a **CLK** jel szerepel → **SZINKRON** működési mód minden vezérlőjelre
  - A **RESET/SET** jelek kiértékelése megelőzi a **Q <= D** értékadást → Prioritás

A DFF alaphelyzet beállító jelei

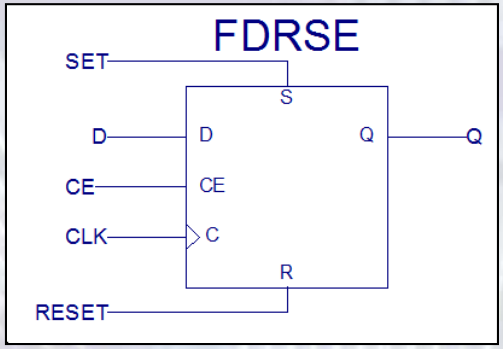
- **Lehetséges a RESET és SET együttes használata is**
  - Ebben az esetben a RESET jel nagyobb prioritású
- **DFF + szinkron RESET és SET**
- **Verilog HDL kódminta:**
  - Az érzékenységi listában csak a **CLK** jel szerepel → **SZINKRON** működési mód
  - A **RESET/SET** jelek felsorolása a kódban állítja be a jelek kiértékelési és így az érvényre jutási sorrendjét → Prioritás **RESET > SET > D**




```
reg Q;
always @ (posedge clk)
   if (RESET)      Q <= 1'b0;
   else if (SET)   Q <= 1'b1;
   else            Q <= D;
```

A DFF órajel engedélyezése

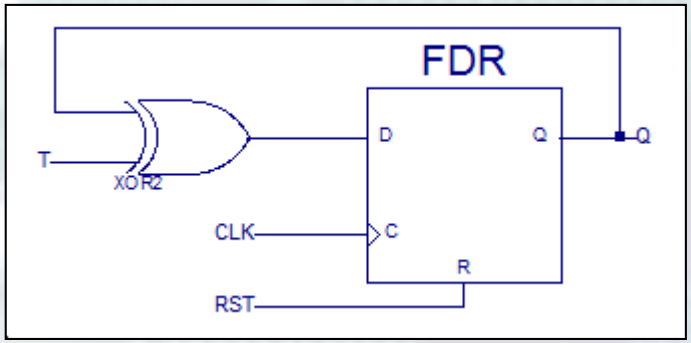
- **Eddig: A DFF minden órajel felfutó élnél a vezérlő-jelek állapota alapján új értéket vett fel, az alaphelyzet beállító jelek vagy az adatbemenet értéke alapján.**
  - Az aktív alaphelyzet beállító jelek (RESET, SET) mindig érvényre jutnak
  - A D bemenet mintavételezése azonban tiltható/engedélyezhető a CE jellel
- **Tehát az órajelre történő mintavétel feltételes, a CE függvényében történik.**
  - Az elemi DFF-nál ez CE, azaz órajel engedélyezés funkció
  - A több bites regisztereknél ez LD (LOAD), azaz töltés vezérlőjel

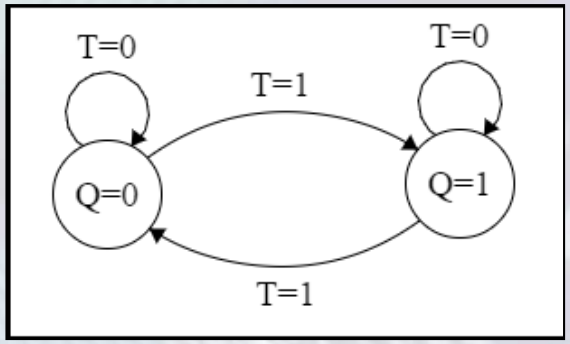



```
reg Q;
always @ (posedge clk)
   if (RESET)          Q <= 1'b0;
   else if (SET)       Q <= 1'b1;
        else if (CE)   Q <= D;
```

További FF típusok

- **TFF → XOR kapuval visszacsatolt DFF**





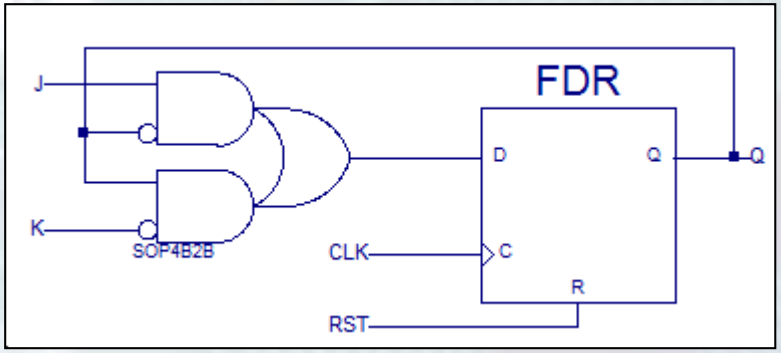
T(t)	Q(t)	$Q(t^+)$
0	Q(t)	Q(t)
1	Q(t)	/Q(t)

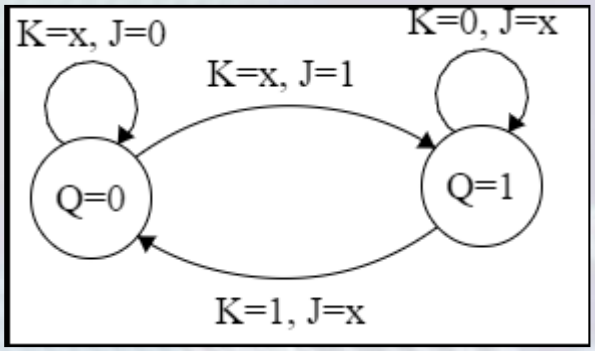
- **Működés: A TFF állapotot vált (toggle), ha T = 1, egyébként T=0 esetén, tartja az előző értékét**
- **Hagyományos technológiában külön könyvtári elem**
- **FPGA-ban közvetlenül kódoljuk a funkciót, ha szükségünk van rá**

```
reg Q;
always @ (posedge clk)
   Q <= T ^ Q;
```

További FF típusok

- **JK FF → SOP által visszacsatolt DFF**





J(t)	K(t)	Q(t)	$Q(t^+)$
0	0	Q(t)	Q(t)
0	1	Q(t)	0
1	0	Q(t)	1
1	1	Q(t)	/Q(t)

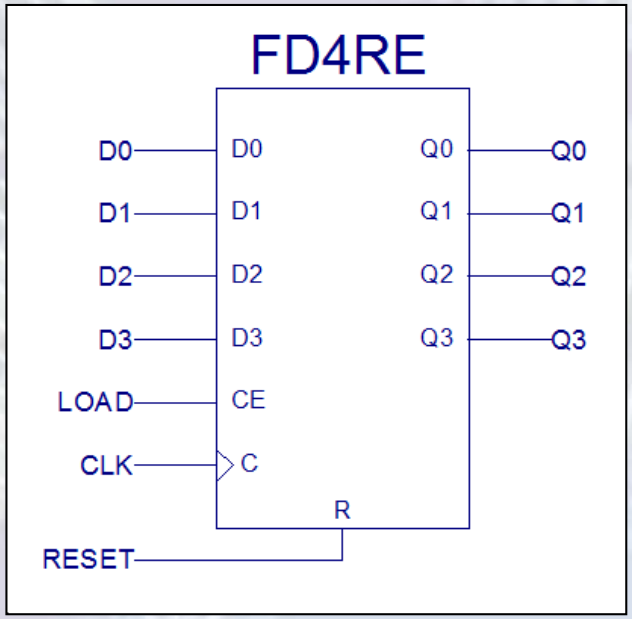
- **Működés: A JK FF az SR tároló javított verziója. Értelmezett a JK = 11 vezérlés is, ekkor invertálja az aktuális állapotát (mint a TFF)**
- **Hagyományos technológiában külön könyvtári elem**
- **FPGA-ban közvetlenül kódoljuk a funkciót, ha szükségünk van rá**

```
reg Q;
always @ (posedge clk)
   if (RST) Q <= 1'b0;
   else     Q <= J & ~Q | ~K & Q;
```

```
reg Q;
always @ (posedge clk)
   if (RST) Q <= 1'b0;
   else
      case ({J,K})
         2'b00: Q <= Q;     // Tart
         2'b01: Q <= 1'b0; // Töröl
         2'b10: Q <= 1'b1; // Beír
         2'b11: Q <= ~Q;    // Invertál
      endcase
```

A regiszter

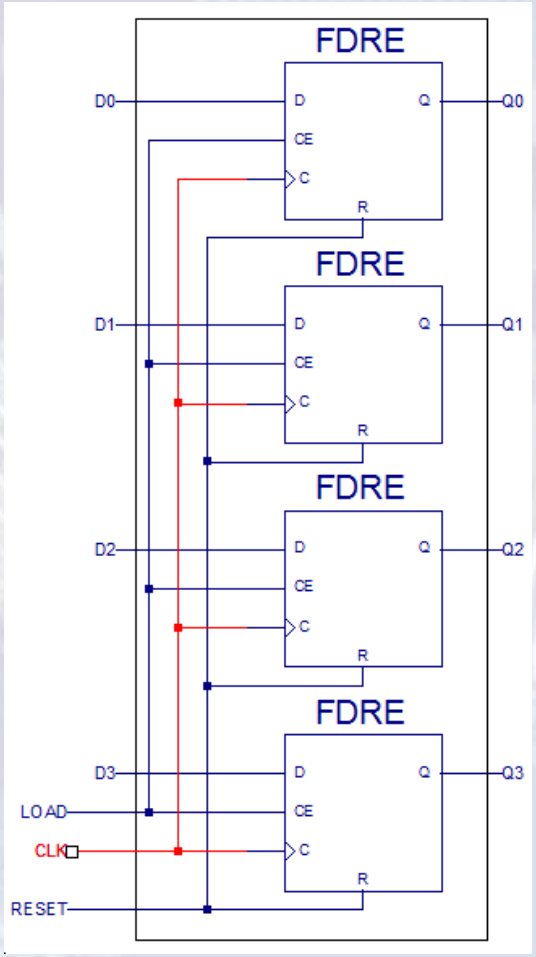
- **A regiszterek a DFF-okból felépített több bites tárolók**
- A regiszterek felépítése olyan, hogy a D adatbemeneti bitek kivételével az összes többi vezérlőjelük közösített, azaz egy ütemben működik a CE/LOAD, RESET és CLK
- A regiszterek mérete 2-től akár 64 bitig terjed
- Az egyszerű regiszter 3 funkcióval rendelkezik/rendelkezhet:
  - RESET, tartalom törlése
  - LOAD, a D[3:0] adat betöltése
  - HOLD, a jelenlegi tartalom tartása
- **Verilog HDL kódminta**




```
reg [3:0] Q;
always @ (posedge clk)
   if (RESET)      Q <= 4'b0;
   else  if(LOAD)  Q <= D;
```

Regiszterek időzítési jellemzői

- **A regiszterben lévő DFF-ok a közös CLK órajel miatt azonos időben, szinkron módon, az órajel ↑ élére vesznek mintát az adatbemeneteikből**
- **A regiszterek helyes működését az előírt időzítési feltételek betartása biztosítja**
- **A fontosabb időzítési paraméterek:**
  - Maximum f_{clk} (= 1/ T_{clk})
  - Minimum órajel periódusidő T_{clk}
  - Órajel pulzusszélesség t_{pwH}, t_{pwL}
  - Kimeneti válaszidő t_{cq}
  - Bemeneti előkészítési idő t_{su}
  - Bemeneti tartási idő t_h



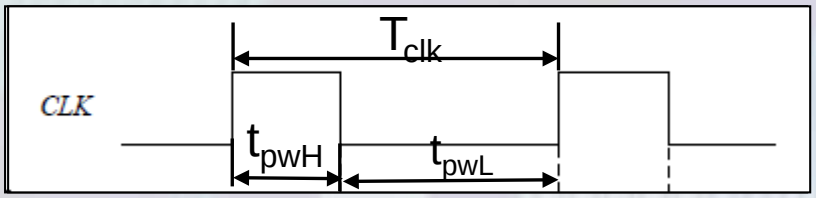
Regiszterek időzítési jellemzői

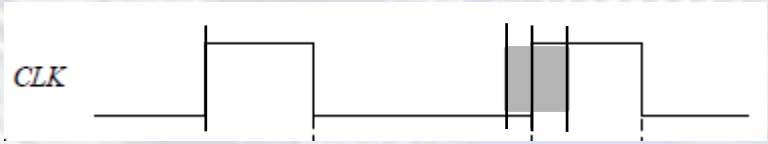
- **Regiszterek időzítési jellemzői részben hasonlítanak a logikai kapuk késleltetéseire, részben attól eltérőek**
- **Az időzítési adatok többsége az órajellel kapcsolatos**
- **Az órajel:**
  - Az órajel egy speciális jel
  - Normál esetben nem tartozik a logikai jelek közé
  - Nem része a logikai kifejezéseknek
  - Egyetlen (de nagyon fontos) feladata a rendszerben lévő szinkron működésű elemek egységes, azonos idejű ütemezéssel történő vezérlése
  - Az órajel (a legtöbb esetben) nagy pontosságú, periodikus jel, amelynek mind a rövid, mind a hosszú idejű stabilitása kiemelkedően jó → kvarcoszcillátor

Órajel paraméterek

- **Periódikus jel, jellemzői:**
  - T_{clk} periodusidő
  - $f_{clk} = 1/ T_{clk}$ frekvencia
  - D kitöltési tényező: t_{pwH} / T_{clk} (Jellemzően 50%)
  - A minőségi órajel időben és térben állandó
    - Térben: Az áramkör minden pontján az órajel minden FF-hoz ugyanabban az időben jut el
      → Ha nem, órajel elcsúszás (Skew) Védekezés: H elosztás
    - Időben: Az órajel minden periódusának stabilitása jó
      Rövid idejű stabilitás → Jitter

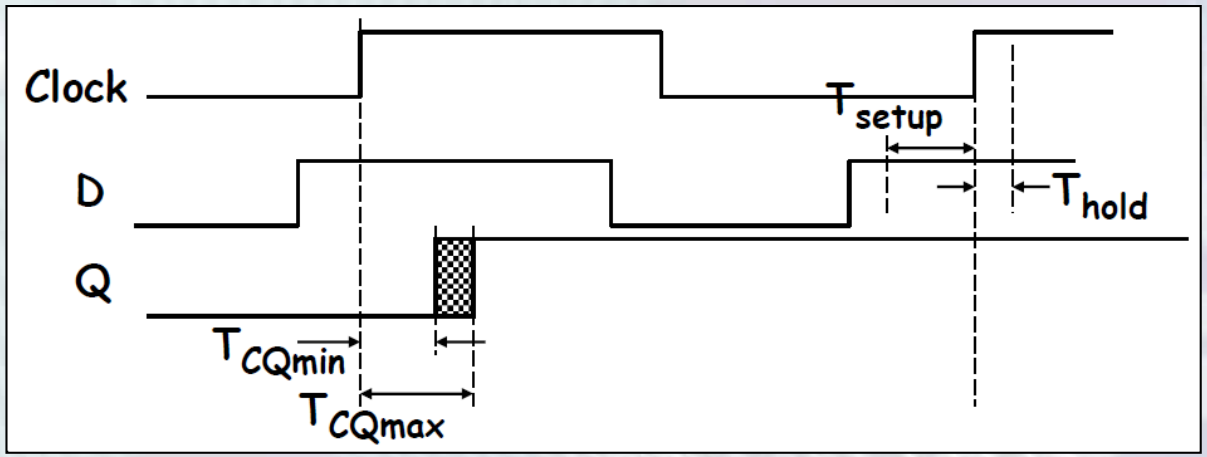
      Hosszú idejű stabilitás → Lassú frekvencia változás





Regiszterek időzítési jellemzői

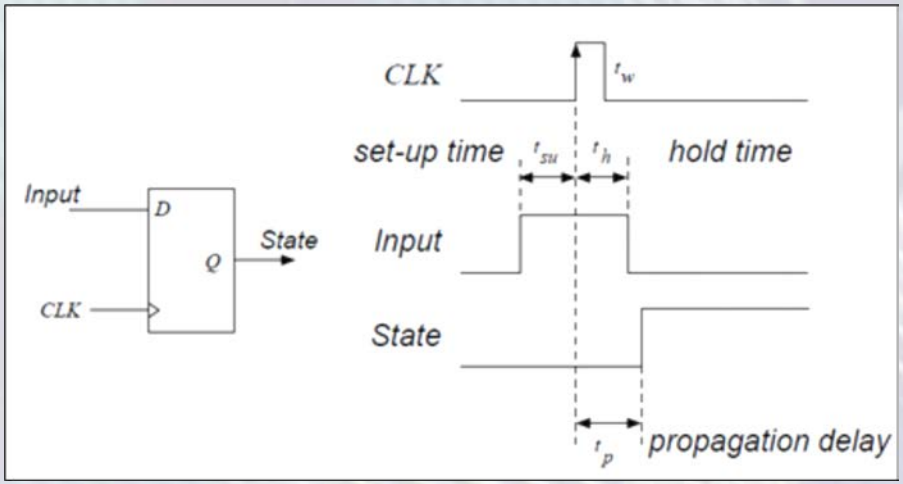
- **A FF-ok, regiszterek időzítési jellemzői a következők**



- A $\mathbf{T_{cq}}$ kimeneti kapcsolási idő (a DFF reakció ideje az órajel felfutó élre, min és max értékekkel)
- A $\mathbf{t_{su}}$ és $\mathbf{t_h}$ előkészítési és tartási idők (a DFF adat bemenetének előírt stabil mintavételezési időablaka a CLK ↑ éléhez képest (ezalatt D nem változhat)
- Technológiailag garantált, hogy $\mathbf{T_{cqmin} > t_h}$

Regiszterek időzítési jellemzői

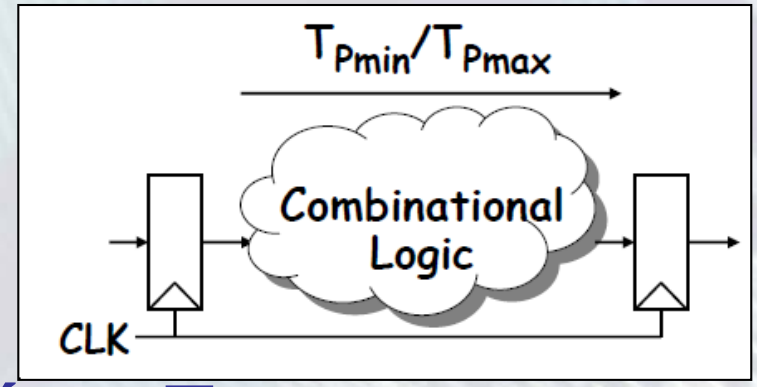
- **A helyes működés feltétele ezen időzítési paraméterek, követelmények betartása**
- **Amennyiben az előkészítési és tartási idők teljesülnek, a kimenet a megadott időn belül reagál a vezérlésre**



- **A RESET/SET/CE bemenetekre hasonló időzítési feltételek vonatkoznak, esetleg eltérő értékekkel**

Regiszterek időzítési jellemzői

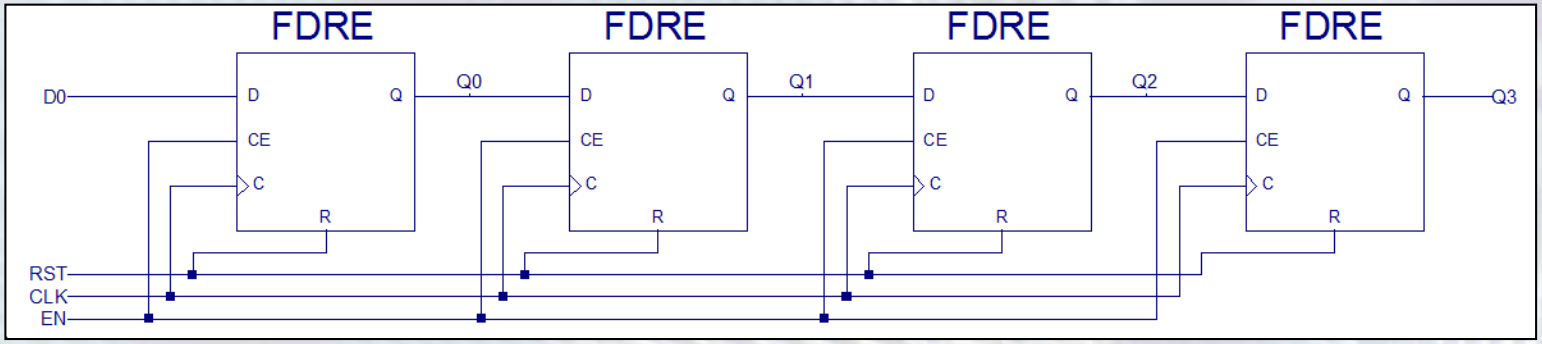
- **A időzítési paraméterek határozzák meg az elérhető működési sebességet**
- **Egy összetett rendszer működésének időbeli jellemzői**



- **Leglassabb jelútra: $T_{clk} \geq t_{cqmax} + t_{pmax} + t_{su}$**
  - Csökkentve f_{clk}-t (növelve T_{clk}-t), mindig kielégíthető
- **Leggyorsabb jelútra: $t_{cqmin} + t_{pmin} \geq t_h$**
  - Ez technológiailag teljesül, ha nem, az komoly gond

Regiszterek időzítési jellemzői

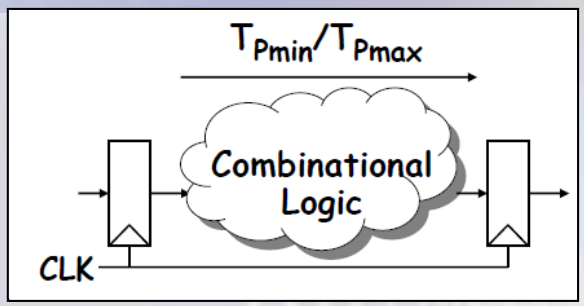
- **SHIFTREGISZTER**
  - A időzítési követelmények elemzését a shift regiszterrel mutatjuk be
  - A shiftregiszter a legegyszerűbb sorrendi hálózat, egyben az egyik legfontosabb funkcionális egység

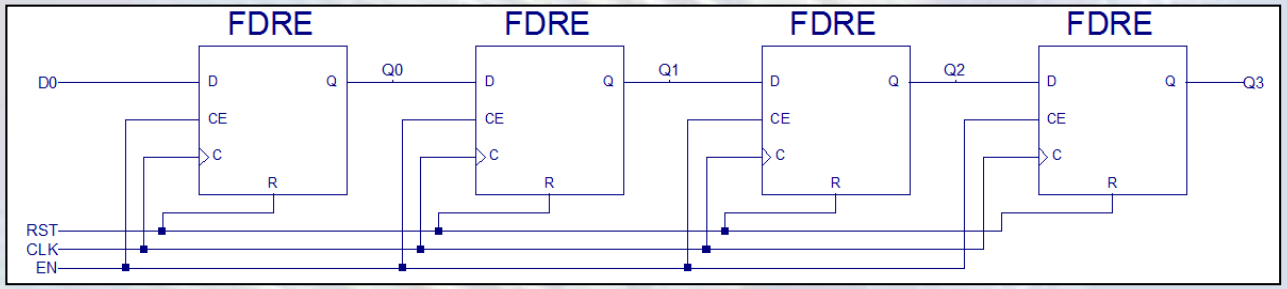


  - Soros bemenet D0 → Párhuzamos kimenet az utolsó 4 ütemből {Q0,Q1,Q2,Q3} rendezéssel
  - Soros bemenet D0 → Soros kimenet bármelyik kimenetről Q0,Q1,Q2,Q3, 1,2,3,4 órajel késleltetéssel

Regiszterek időzítési jellemzői

- **SHIFTREGISZTER**
  - Ez a lehetséges legnagyobb órajel-frekvenciával működtethető, több, mint 1 DFF-ot tartalmazó áramkör
    → nincs kombinációs logika a DFF-ok között
    $\mathbf{T_{clk} \geq t_{cqmax} + t_{pmax} + t_{su}}$, ahol $\mathbf{t_{pmax} = 0}$
  - Egy adott technológiában: $\mathbf{f_{max} = 1/ (t_{cqmax} + t_{su})}$
  - Leggyorsabb jelút $\mathbf{t_{cqmin} + t_{pmin} \geq t_h}$, ahol $\mathbf{t_{pmin} = 0}$, tehát ha $\mathbf{t_{cqmin} \geq t_h}$ nem teljesül, akkor nem működik!!

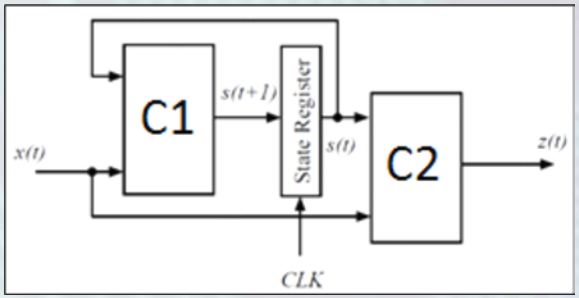




Sorrendi hálózatok

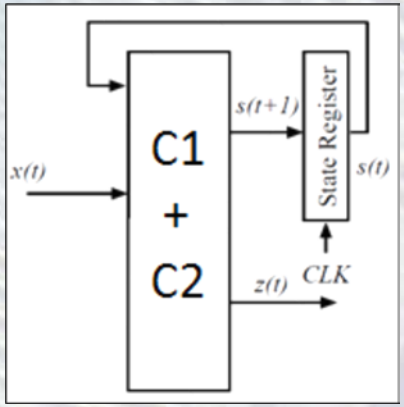
- **Sorrendi hálózatok általános modellje:**
  - Kombinációs logikai blokkok (C1 és C2 függvények):
    - Állapotátmeneti logika s(t+1) = C1(s(t), x(t));
    - Kimeneti logika z(t) = C2(s(t), x(t));
  - Állapotregiszter s(t) → s(t+1) CLK ↑ élre

Mealy modell

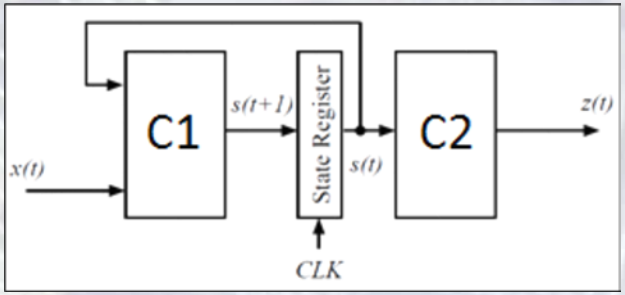


A kimenet közvetlenül függ a bemenettől is

Közös modell



Moore modell



A kimenet csak az állapottól függ

Sorrendi hálózatok

- **Sorrendi hálózatok fontosabb típusai**
  - Véges állapotú rendszerek
    - Mintagenerátor, mintafelismerő egységek
    - Vezérlőegységek, automaták
  - Általános szekvenciális funkcionális elemek
    - Számlálók, shift regiszterek, más funkciók
- **Véges állapotú gépek**
  - Angol nevük Finite State Machines (FSM)
    - Gyakran hivatkozunk rájuk ezen a néven
  - Jellemzően kevés állapottal rendelkező, tetszőleges állapotszekvenciát realizáló egységek
- **Sorrendi hálózatok specifikálása:**
  - Időtartománybeli viselkedés
  - Állapottartománybeli viselkedés
    - A két leírási mód egymásba konvertálható

Véges állapotú gépek

- **Véges állapotú gépek jellemzése**
- **Az FSM formális definíciója a következő:**

{I, S, S_0, C1, C2, O}, ahol

- **I** a bemeneti vezérlőjelek halmaza (a CLK és RESET jeleket nem értjük ezek közé)
- **S** az állapotgép állapotainak halmaza
- **S_0** a kezdeti állapot, bekapcsolás vagy RESET után
- **C1** az állapot átmeneti függvény, amely megadja, hogy egy adott állapotból, a bemeneti jelek milyen feltételei mellett melyik következő állapotot veszi fel az FSM.
- **C2** a kimeneti függvény, ami az állapotok (és a bemeneti jelek) alapján a kimenet értékét állítja elő
- **O** a lehetséges kimeneti értékek halmaza

Sorrendi hálózatok

- **Sorrendi hálózatok jellemzői:**
  - A sorrendi hálózatok kimenetét az állapotregiszter értéke és (esetleg) a bemeneti jelek aktuális értékei határozzák meg
  - Szinkron sorrendi hálózatok: Az állapotregiszter változását egy órajel ütemezi. A bemeneti jelek hatása az állapotregiszterre csak a CLK ↑ él pillanatában jut érvényre.
    - A kimenetek a Moore típusú realizációnál ezért csak az órajel felfutó éle után változnak (mert csak az állapot-regiszter aktuális értékétől függenek).
    - A Mealy típusú realizációnál a kimenet közvetlen logikai függvénye a bemeneteknek is, ezért bármikor változhatnak, ha a bemenet is változik. (De elvárjuk a bemenet helyes időzítését!)
    - Aszinkron sorrendi hálózatok tervezésével nem foglalkozunk

Sorrendi hálózatok tervezése (1)

- **Sorrendi hálózatok tervezése:**
  - A sorrendi hálózatok tervezésének első lépése a specifikáció alapján a szükséges állapotok felvétele, a bemeneti változások hatására fellépő átmenetek definiálása. (A kezdeti állapotkijelölés későbbi analízis során módosulhat.)
  - Általában a specifikációból következik a Moore-Mealy típusválasztás. Ha nem, akkor dolgozzunk Moore modell alapján, illetve Mealy esetén mindig mintavételezzük a független bemeneti jeleket.
  - A kezdeti specifikációt felvehetjük állapotátmeneti táblában (állapotok, kimenetek és állapotátmenetek listázása), vagy állapotdiagramban.

Sorrendi hálózatok tervezése (2)

- **Sorrendi hálózatok tervezése:**
  - A kezdeti állapotfelvétel gyakran redundáns, tartalmaz összevonható állapotokat, ami a komplexitást jelentősen csökkentheti. Ezért az állapotminimalizálási lehetőségeket érdemes megvizsgálni.
  - A következő lépés a „magas” szintű specifikáció leképezése a digitális hardverre. Állapot kódolási módszer megválasztása, állapotregiszter méretének előírása, állapotkódolás. Ez a lépés jelentősen befolyásolja a teljes FSM komplexitását, jellemzőit.
    - Hagyományos tervezésben a következő lépés a kódolt állapotátmeneti tábla elkészítése, azaz az állapotátmeneti és a kimeneti függvény definiálása.

Sorrendi hálózatok tervezése (3)

- **Sorrendi hálózatok tervezése:**
  - Verilog HDL tervezési technológiában közvetlenül definiálhatjuk az állapotregiszterre és az állapotátmeneti logikára vonatkozó előírásokat.
  - Ezek alapján a szükséges vezérlési függvényeket a rendszer generálja.
  - A tervezőrednszer esetleg az „általános technológiai ismeretei” alapján állapotminimalizálást, módosított állapotkódolást is alkalmazhat.

Digitális technika 4. EA vége