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

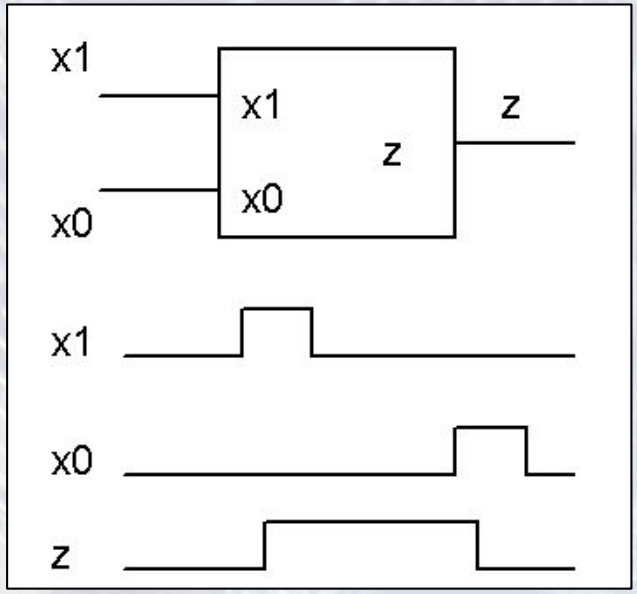
Digitális technika
VIMIAA02

Fehér Béla, Benesóczky Zoltán
BME MIT

Sorrendi hálózatok

(Finite State Machine, FSM)

- Az eddigiekben megismert digitális áramkörök ***kombinációs hálózatok*** (a kimenet csak az aktuális bemeneteltől függ)
- Sok esetben a feladatok megoldásához ez nem elegendő, ***a rendszer viselkedését befolyásolják a bememententek előző értékei***.
- **Példa:** A logika kimenete adjon 1-et, ha x1 =1, majd maradjon 1-ben, ha x1 = 0 lesz. A kimenet adjon 0-át, ha x0 = 1 lesz, és maradjon 0-ban, ha x0 = 0 lesz. (Ha x1 és x0 egyszerre 1, akkor szintén menjen 1-be. X1 az erősebb.)



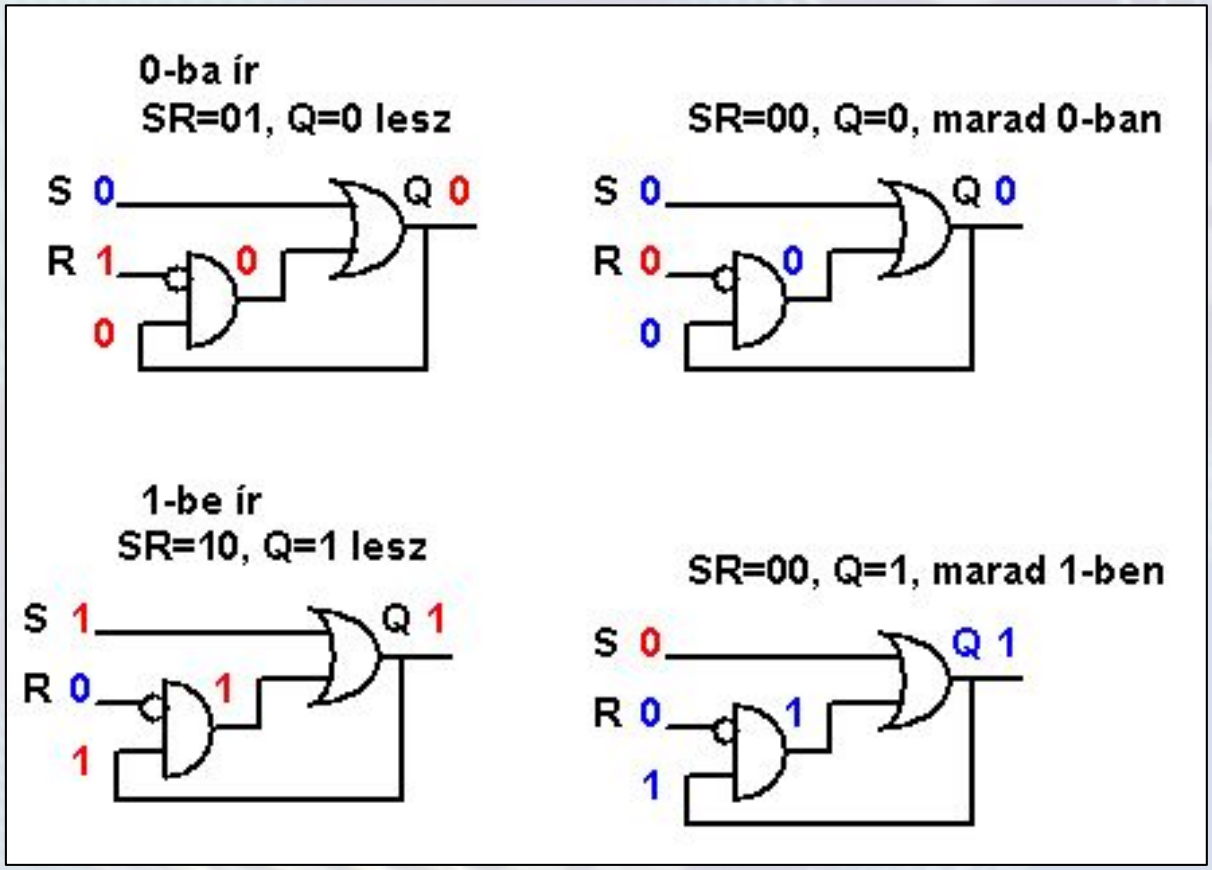
- **Emlékező, memória funkció kell**
  - Az ***állapotot*** meg kell jegyezni és az aktuális bemenet alapján, ha a feladathoz szükséges, meg kell változtatni

Elemi bit tárolók

- **Visszacsatolás kombinációs hálózatban**
  - A sorrendi hálózat tárolni is képes információt. Az általa tárolt információt ***állapotnak*** (state) nevezzük.
  - A sorrendi hálózat kimenete az aktuális bemenettől és az állapotától függ.
  - Kombinációs hálózat közvetlen visszacsatolásával is létre lehet hozni sorrendi hálózatot.
  - ***A kombinációs hálózat visszacstolásával létrehozott sorrendi hálózatokat aszinkron sorrendi hálózatoknak nevezzük.***

Elemi bit tárolók: SR latch

- SET - RESET (SR) latch
  (A SET 1-be állítást, a RESET alaphelyzetbe állítást jelent
- A jellemző működés:

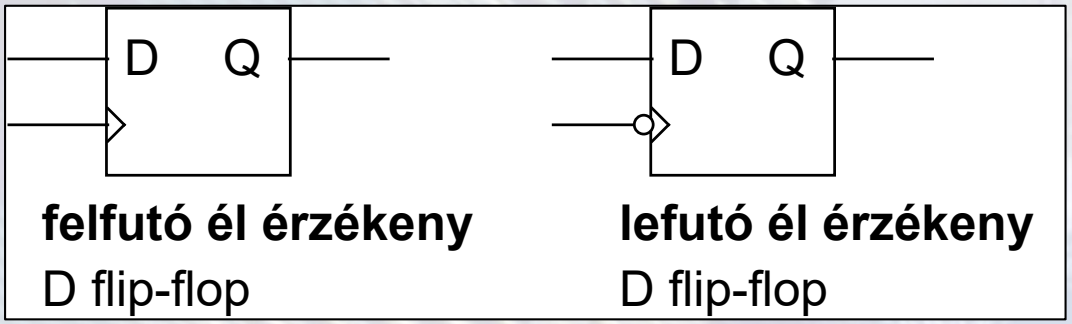


Szinkron sorrendi hálózatok (SSH)

- Aszinkron sorrendi hálózatokból nagyméretű sorrendi hálózatok nagyon nehézkesen építhetők fel.
- Ezért ***a továbbiakban csak*** az órajel éllel ütemezett ún. ***szinkron sorrendi hálózatokkal foglalkozunk***.
- A ***szinkron hálózatok tárolóit*** (az adat beírását) ***órajel él ütemezi***. Az ***ütemezés történhet az órajel 0-1 átmenetére (felfutó él)***, vagy 1-0 átmenetére (lefutó él)
- Az órajel vezérelt működést ***szinkron működés***nek nevezzük.
- A továbbiakban ***csak*** **felfutó élet** ***használunk az ütemezésre*** (lefutó él is lehetne) Ezt úgy is szokás mondani, hogy ***az órajel aktív éle a felfutó él.***
- Az órajelet (clock, clk, c) ***órajel generátor*** állítja elő***. Ez egy periodikus négyszög jel (0101...).*** A jel ***frekvenciáját*** (f_{clk}) az ***áramkör kívánt sebessége*** alapján állítják be.

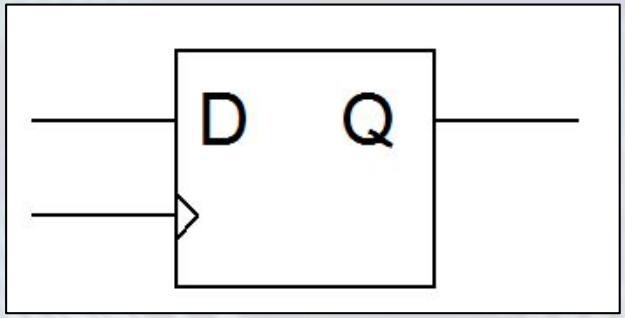
A D flip-flop

- **A digitális technikában használt legfontosabb bit tároló egység a D flip-flop**
  - A DFF egy órajel (clock, clk, c) aktív élének hatására mintavételezi és tárolja a bemeneti D adatbitet
  - Ez az érték jelenik meg a Q kimeneten (Q = D) és tárolódik a következő aktív élig.
- **D ff rajzjele:**

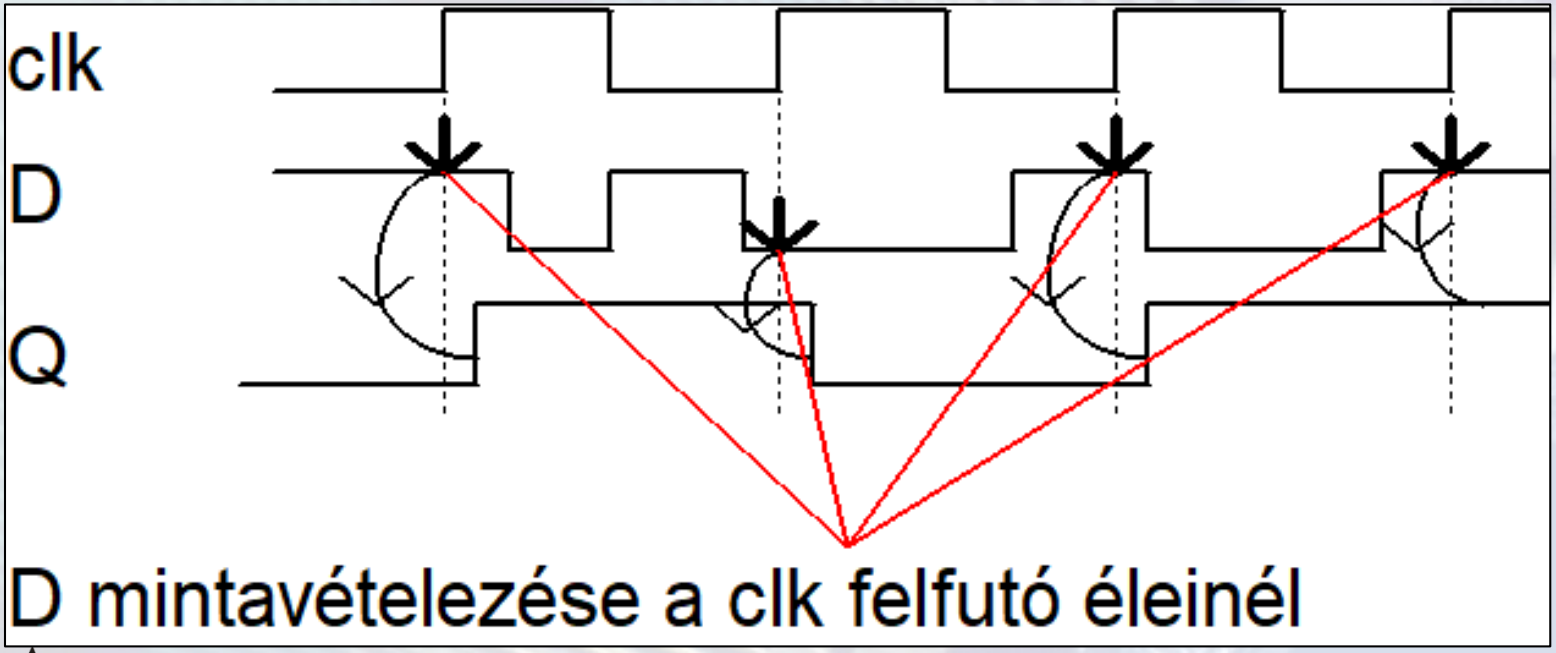


- Az órajel bemenetet a bemenet melletti > jel jelöli a rajzokon.
- A Digitális technika tárgyban csak felfutó él érzékeny tárolókat fogunk használni.

A D flip-flop



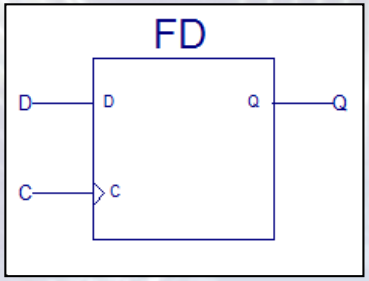
A D flip-flop működését mutató időidiagram:



A D flip-flop

- **A DFF egy kész áramköri egység** (alapelem, mint a kapuk)
  - Nem foglalkozunk a belső felépítésével.
  - Verilog viselkedési leírással építjük be.
  - **DFF funkció**: Az órajel minden felfutó élénél (***posedge clk***) vegyen mintát a D adat bitből és ezt tartsa a kimenetén a teljes órajel periódusban a következő felfutó élig.

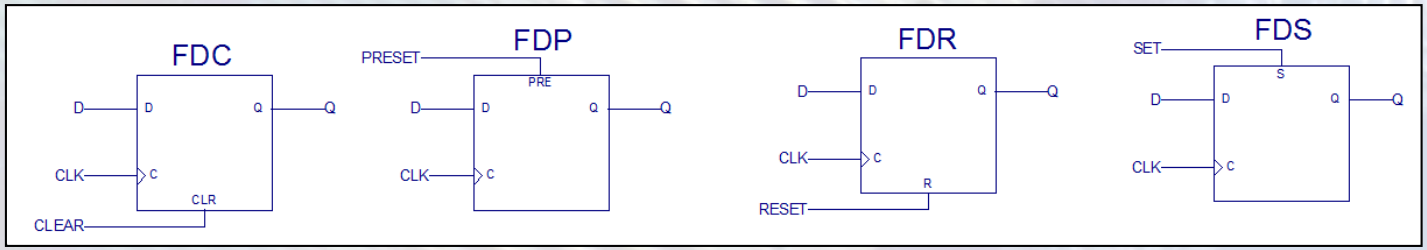
```
reg Q;
always @ (posedge clk)
   Q <= D;
```



  - Szó szerint ezt a kódot írtuk le, így kell kiolvasni is.

A DFF

- **Bekapcsoláskor a Q értékét a feladatnak megfelelően kell inicializálni.**
- ***Ezért a flip-flopnak alaphelyzet beállító bemenetei is lehetnek:***
  - RESET (PR): a jelre Q = 0 az órajel ↑ élre áll be (szinkron)
  - SET (S): a jelre Q = 1 az órajel ↑ élre áll be (szinkron)
  - CLEAR (CLR, Cl): a jelre Q = 0 és azonnal beáll (aszinkron)
  - PRESET (PR, P): a jelre Q = 1 és azonnal beáll (aszinkron)
  - Aszinkron CLEAR / PRESET Szinkron RESET / SET:



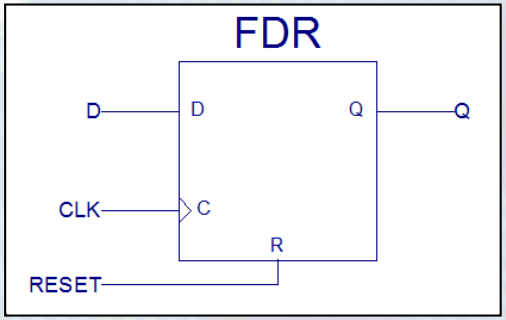
  - A továbbiakban csak szinkron jeleket használunk (PR, S)

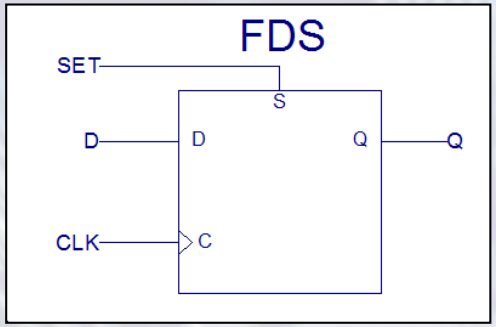
A DFF

- **Az inicializáló jelet (RESET) külön áramkör allítja elő, az órajelhez hasonlóan**.
- A logikai rajzokon általában sem az órajel generátort, sem a reset áramkört nem fogjuk a továbbiakban feltüntetni. (Csak nagyon kivételes esetben.)
- Helyettük az általuk előállított **clk** és **reset** (res) jeleket használjuk

A DFF alaphelyzet beállító jelei

- **A továbbiakban a SZINKRON vezérlést használjuk.**
  DFF szinkron RESET-tel **DFF szinkron SET-el**





- **Verilog HDL kódminta:**

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

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

  - Az érzékenységi listában csak a **posedge clk** jel szerepel, ez jelzi a **szinkron felfutó él** vezérelt működési módot
  - A **RESET/SET** jelek kiértékelése megelőzi a **Q <= D** értékadást

A DFF alaphelyzet beállító jelei

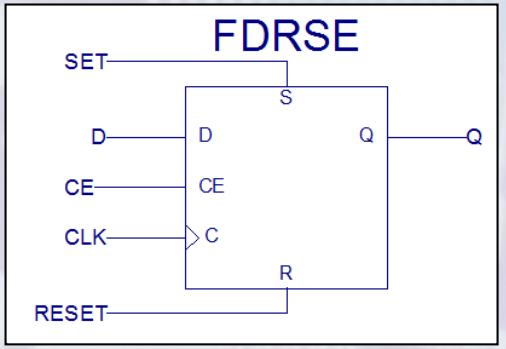
- Ha ***mindkét alaphelyzetbe állító jelet használjuk***, akkor az if else if szerkezetben előbb szereplő jel prioritása nagyobb lesz az utána levőnél.
- **Verilog HDL kódminta, az rst prioritása a legnagyobb:**

```
always@(posedge clk)
 if(rst) q <= 1'b0;
 else if(pr) q <= 1'b1;
      else  q <= D;
```

- Most, ha egyszerre aktív rst és pr, akkor q = 0 lesz az órajel felfutó él hatására. Ha rst=0 és pr=1, akkor q = 1 lesz. Ha rst = 0 és pr = 0, akkor q = D lesz.

A DFF órajel engedélyezése (CE)

- A D flip-flopnak létezik olyan változata, amelynél ***a D bemenet mintavételezése tiltható/engedélyezhető egy CE jellel***
- **Tehát az órajelre történő mintavétel feltételes, a CE függvényében történik.**
  - Az elemi DFF-nál ezt CE, azaz órajel engedélyezés funkciónak nevezzik, de ***csak a D bemenetre van hatással***, a RESET és PR bemenetekre nincs!
  - A több bites regisztereknél ugyanezt a funkciót ellátó jel neve LD (LOAD), azaz adatbetöltés vezérlőjel.

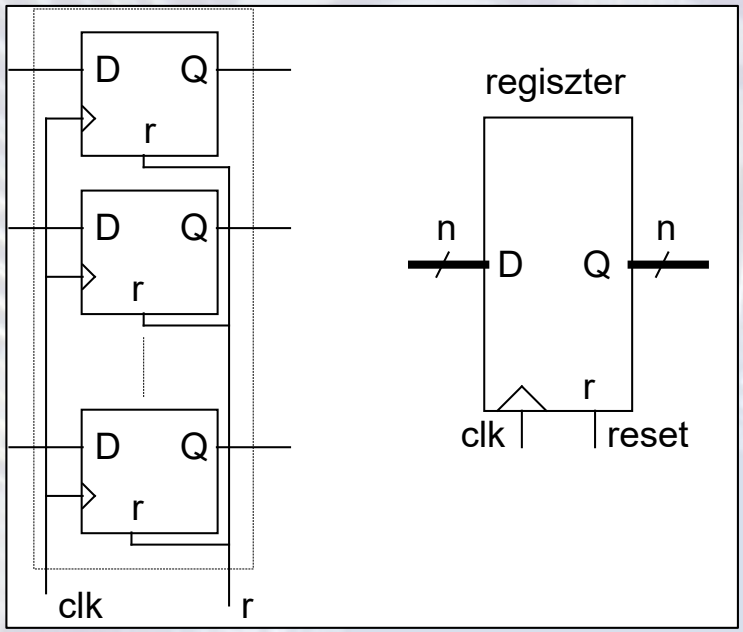



```
reg Q;
always @ (posedge clk)
   if (RESET)          Q <= 1'b0;
   else if (SET)       Q <= 1'b1;
        else if (CE)   Q <= D;
```

A regiszter

- **A regiszterek DFF-okból felépített több bites tárolók**
- A regiszterek felépítése olyan, hogy a ***D adatbemeneti bitek kivételével az összes többi vezérlőjelük közösített, azaz egyszerre működik.*** (Pl. a reset minden bitet töröl.)
- A regiszterek mérete 2-től akár 64 bitig terjed
- A legegyszerűbb regiszter csak RESET vezérlő jellel rendelkezik:
  - **RESET** (rst): tartalom törlése
  - Ha nem aktív, akkor a bemenet mintavételezése
  - **Verilog kódminta**:

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

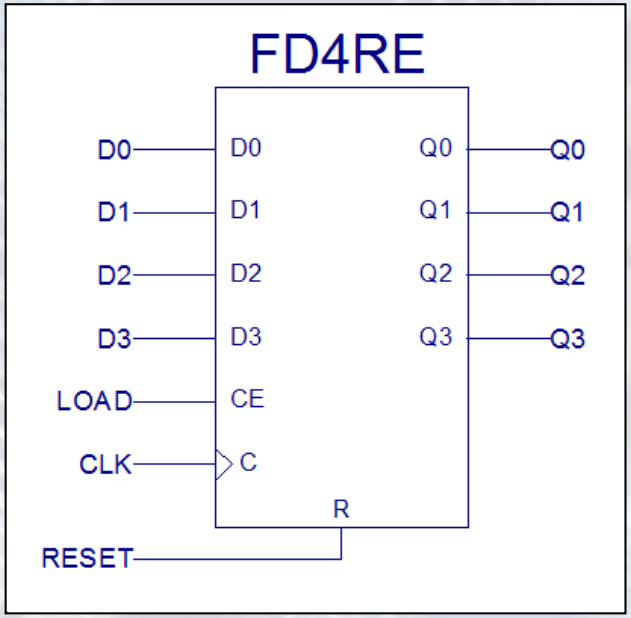


A regiszter

- A regiszter **LOAD adat betöltő jel**lel is rendelkezhet

Verilog HDL kódminta:

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



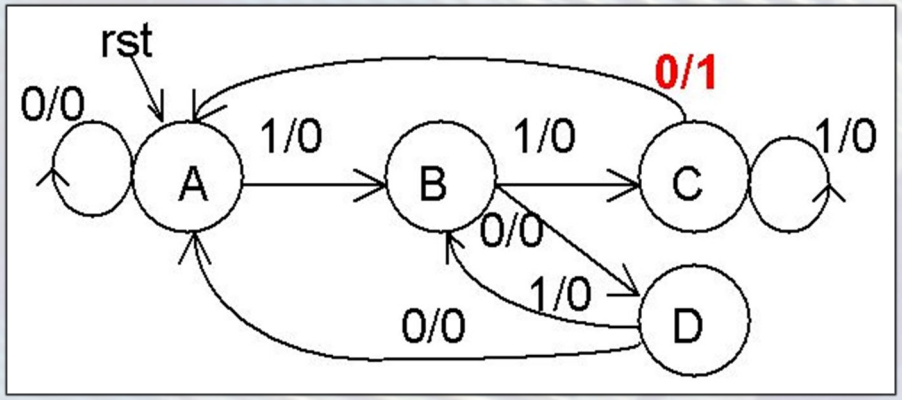
- A sorrend határozza meg a prioritást (ha egyszerre több vezérlőjel aktív, melyik hatásos)
- Az, amelyik előbb van az **if else if ...** szerkezetben
- Itt a RESET magasabb prioritású, mint a LOAD

Általános szinkron sorrendi hálózat (SSH, állapotgép, FSM)

- Az állapotgéppel olyan feladatokat lehet megoldani, amelyeknél ***a kimenet előállításához nem elég a bemenet aktuális ismerete, hanem szükséges az előző bemenetek alapján eltárolt információ*** (állapot, state, s) ismerete is.
- Az állapotgép ***az aktuális állapot*** (state, s) ***és az aktuális bemeneti kombináció*** alapján állítja elő a kimeneti jelet.
- A szinkron FSM ***állapota az órajel aktív élére változhat***, állapotának kódját regiszterben tároljuk.

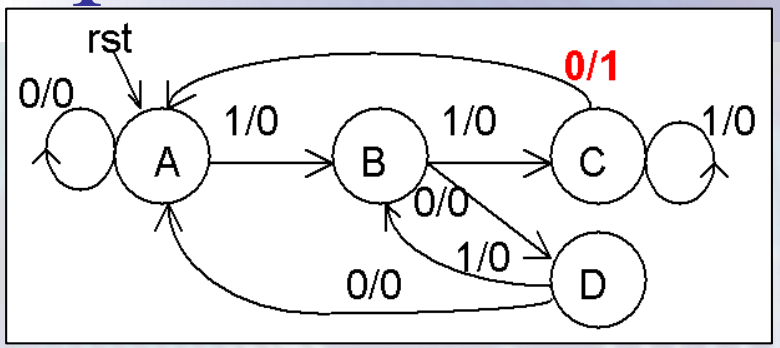
Állapotgépek specifikálása

- Az állpotgépeket megadhatjuk ***szöveges specifikáció***val. Pl. Ismerje fel az x bemenetére az órajellel szinkronban érkező bitsorozatban, ha a legutolsó 3 bit 110 és a z kimenetén jelezzen 1-el az utolsó bit beérkezésével egyidőben.
- Megadható ***állapotgráffal*** (állapot diagramnak is nevezik). (Az állapotgráfot általában szöveges specifikáció alapján készítjük.)



Állapotgépek specifikálása

Állapotgráf



- A ***gráf pontokhoz*** (itt körrel jelöljük) rendeljük az állapotgép ***állapotait***.
  Beleírjuk az állapot nevét. (ABC betűi vagy beszédes név)
  A ***kezdő állapotot*** rst és nyíl jelöléssel látjuk el (itt az A-nál).
- Az állapotgráf ***irányított gráf*** (éleinek iránya van).
- A ***gráf élek*** jelölik az adott bemenet hatására bekövetkező ***állapot átmenteket*** (pl. ***A*** állapotból 1 bemenetre a következő állapot ***B***).
- A ***gráf élekre írjuk***, hogy milyen bemenetre történik az állapot átmenet és milyen kimenetet ad a hálózat (***bemenet/kimenet***).
- Ha a kimenet csak az állapottól függ (lásd később Moore modell), akkor a kimenetet az állapot neve alá is írhatjuk.

Állapotgépek specifikálása

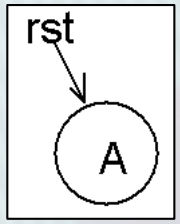
- Az állapotgráf rajzolása során ***egy olyan kezdőállapotból indulunk ki, ami azt az információt tárolja, hogy a bekapcsolás (reset) óta nem jött adat***. Ezt az állapotot általában A-val jelöljük és ***berajzolunk egy bele mutató rst (reset) felíratú nyilat***.
- Ezután a már meglévő állapot(ok)ból kiindulva ***a feladat leírása alapján minden bemeneti kombinációra megvizsgáljuk***, hogy ***az adott állapotban milyen kimenetet kell adni*** és, hogy ***fel kell-e venni új állapotot***, vagy egy már meglévő tárolja azt az információt, amire az adott bemeneti kombináció után emlékeznie kell a hálózatnak. ***Ha kell felvesszük az új állapotot és az adott bementre ide irányítjuk a gráfot. Ha nem, a már meglévő megfelelő állapotba***.
- Az állapotokba az első felíráskor azt az információt is beírjuk, hogy mit jegyez meg a bemeneti sorozatból (ha ez egyszerűen megtehető).
- Előbb-utóbb eljutunk oda, hogy ***nem kell új állapotot felvennünk és minden állapotban minden bemeneti kombiációra meghatároztuk a kimenetet és a következő állapotot***. Ekkor elkészült **az *előzetes állapotgráf***.

Állapotgépek specifikálása

Példa: A mintafeladat gráfjának megtervezése:

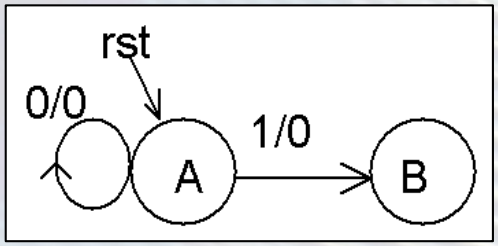
Ismerje fel az x bemenetére az órajellel szinkronban érkező bitsorozatban, ha a legutolsó 3 bit 110 és jelezze az utolsó bit beérkezésével egyidőben.

1. Felvesszük A állapotot és berajzoljuk hogy reset-re ez lesz a kezdő állapot.



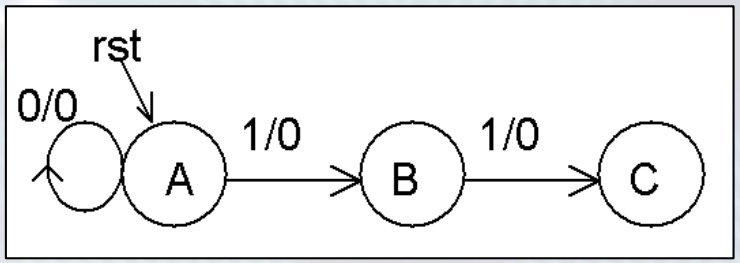
2. Amíg az x bemenet 0, addig maradjon A állapotban, mert a bitsorozat 1-el kezdődik és z = 0. (A–ból A-ba vezető nyíl, x/z (bemenet/kimenet): 0/0) Tehát az A azt jegyzi meg, hogy nem jött még meg az első várt bit.

3. Ha x = 1, akkor megjött a sorozat első bitje, ezt meg kell jegyezni, új állapotra van szükség. Felvesszük B állapotot (új név ABC sorrendben). Ez megjegyzi, hogy bejött 1 db 1-es bit.

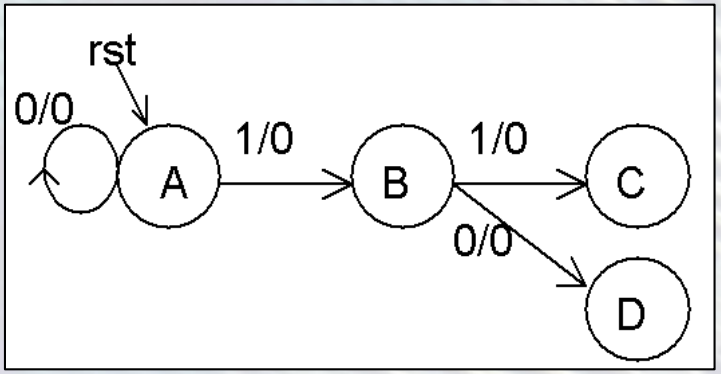


Állapotgépek specifikálása

4. B állapotban, ha x =1, akkor a várt sorozatból bejött 2 bit: 11 és ezt meg kell jegyezni. Felvesszük C állapotot, B-ből C-be mutató nyilat rajzolunk és ráírjuk 1/0.



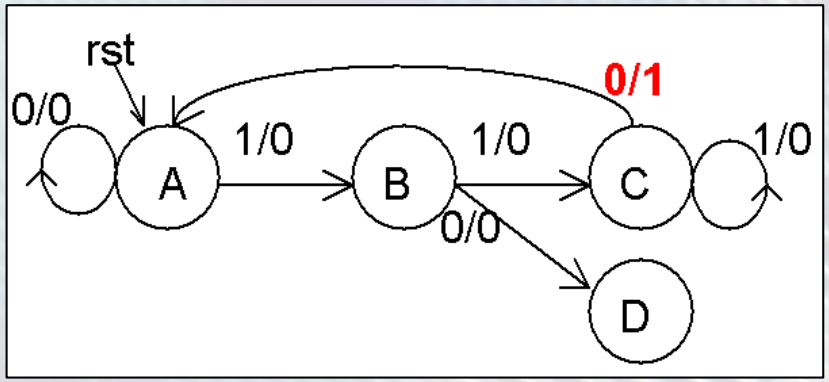
5. B állapotban, ha x = 0, akkor felvesszük D állapotot (ha még kevés a tapasztalatunk és nem jövünk rá, hogy ha 1 után 0 jött akkor elölről lehet kezdeni a figyelést, vissza lehet menni A-ba) De most x = 0-ra D állapotba mutató nyilat rajzolunk és ráírjuk, hogy 0/0. D azt jegyzi meg, hogy az utolsó 2 bit 10 volt.



Állapotgépek specifikálása

6. C állapotban, ha x = 0 jött, akkor ***megjött a teljes várt sorozat 110***. A C állapotban ***z= 1-et adunk*** és megyünk az A állapotba, hogy ott várjuk az új sorozat első bitjét. Tehát D-ből A-ba mutató nyilat rajzolunk és ráírjuk, hogy 0/1.

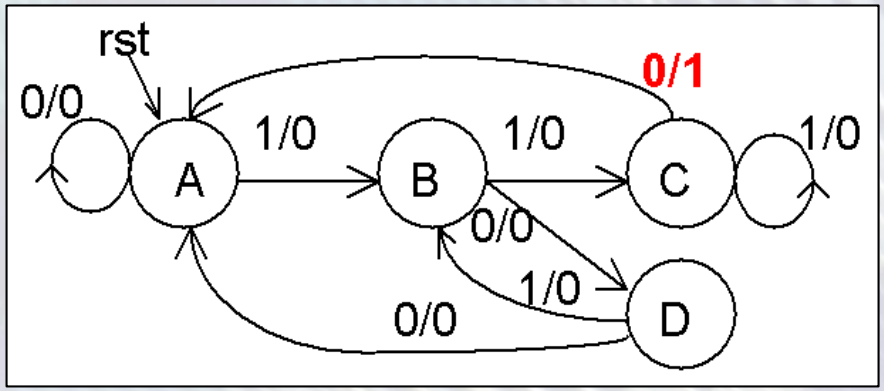
7. C állapotban, ha x = 1 jött, akkor csak azt kell megjegyeznünk, hogy a várt sorozatból már bejött két bit:11. A C állapot pont ezt jegyzi meg. C-ből C-be mutató nyilat rajzolunk és ráírjuk, hogy 1/0.



Állapotgépek specifikálása

8. D állapotban, ha x = 0, akkor ez már nem lehet jó sorozat, csak azt kell megjegyezni, hogy még nem jött meg az első várt bit (1). Ezt a már meglévő A állapot jegyzi meg, tehát oda kell menni. D-ből A-ba mutató nyilat rajzolunk és ráírjuk 0/0.

9. D állapotban, ha x =1, akkor azt kell megjegyezni, hogy bejött a várt sorozat első bitje. Ilyen állapotunk már van, a B pont ezt jegyzi meg. D-ből B-be húzunk egy nyilat és ráírjuk 1/0.



Mivel minden állapotban minden bemenethez megadtuk, hogy mi lesz a következő állapot és kimenet, továbbá nincs több állapotra szükségünk, készen vagyunk a feladatot leíró előzetes állapotgráffal.

Állapotgépek specifikálása

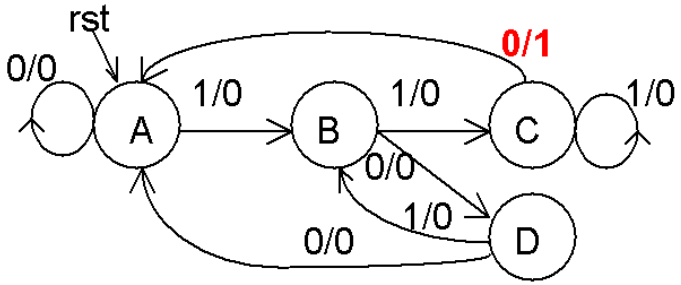
Állapottábla

- Az állapottábla ***ugyanazokat az információkat tartalmazza, mint az állapotgráf, csak táblázatos formában***
- Sorainak bal oldalán soroljuk fel az állapotokat (itt: A,B,C)
- Az állapotok oszlopától ***jobbra levő oszlopok különböző bemeneti kombinációkhoz tartoznak*** (itt: x=0, x=1)
- Egy-egy ***az állapottól jobbra levő rubrikákba azt írjuk be***, hogy ha az ***adott állapotban*** van az automata és ehhez ***az oszlophoz tartozó a bemenet értéke***, akkor ***mi lesz a következő állapot és a kimenet*** (következő állapot/kimenet formában)

aktuális állapot	bemeneti kombinációk	
	x = 0	x = 1
A	A/0	B/0
B	D/0	C/0
C	A/1	C/0
D	A/0	B/0

Ha D az aktuális állapot, a D állapot után következő állapot és a kimenet

ha x =0 ha x =1

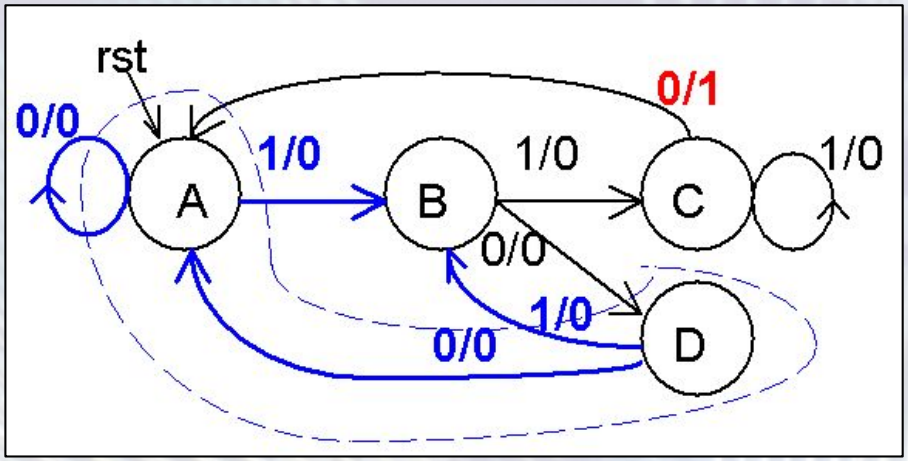


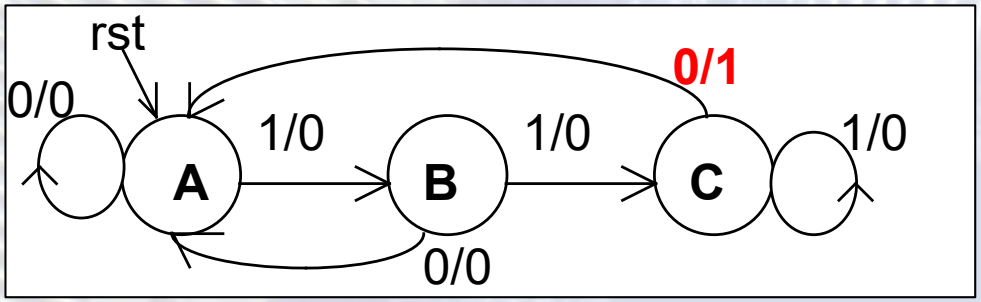
Állapotminimalizálás

- A specifikáció alapján készített állapotgráf felesleges állapotokat tartalmazhat. (Olyan állapotokat, amelyek a feladat szempontjából ugyanazt az információt jegyzik meg.)
- Ezeket meg kell keresni és egyetlen állapottal helyettesíteni. Ezt állapotminimalizálásnak nevezik.
- A kevesebb állapotszámú állapotgép sokszor egyszerűbben valósítható meg.

Állapotminimalizálás

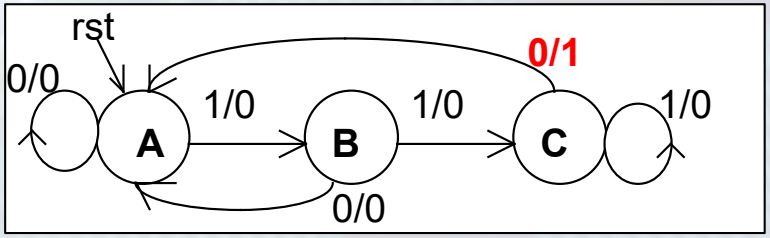
- Belátható, hogy az *A* és D állapotok egyetlen állapottal helyettesíthetők, mivel tetszőleges de ugyanazon bemeneti kombinációra a kimenetük megegyezik és a következő állapotuk ugyanaz (x=1-re B) vagy {A,D} halmazban marad (x=0-ra A-ból A, D-ből A).
- Az A és D állapotot helyettesítettük egy új *A* elnevezésű állapottal és felrajzoljuk az így kapott **minimalizált állapotgráfot**.
- Szisztematikus minimalizálási módszerek léteznek, de nem tanuljuk.





Állapotkódolás

- A ***minimalizált állapotgráf*** alapján elkészítjük a ***minimalizált állapottáblát***. Ez még az állapot minimalizáláshoz tartozik.



	x=0	x=1
A	A/0	B/0
B	A/0	C/0
C	A/1	C/0

- Az **állapotokhoz kódokat** (fix hosszúságú bináris számokat) rendelünk, ezt nevezzük ***állapotkódolás***nak.
- A hálózat bonyolultsága az állpotkódtól is függ, ezért ***szisztematikus módszerek léteznek***. (Ilyeneket nem tanulunk.)

Állapotkódolás

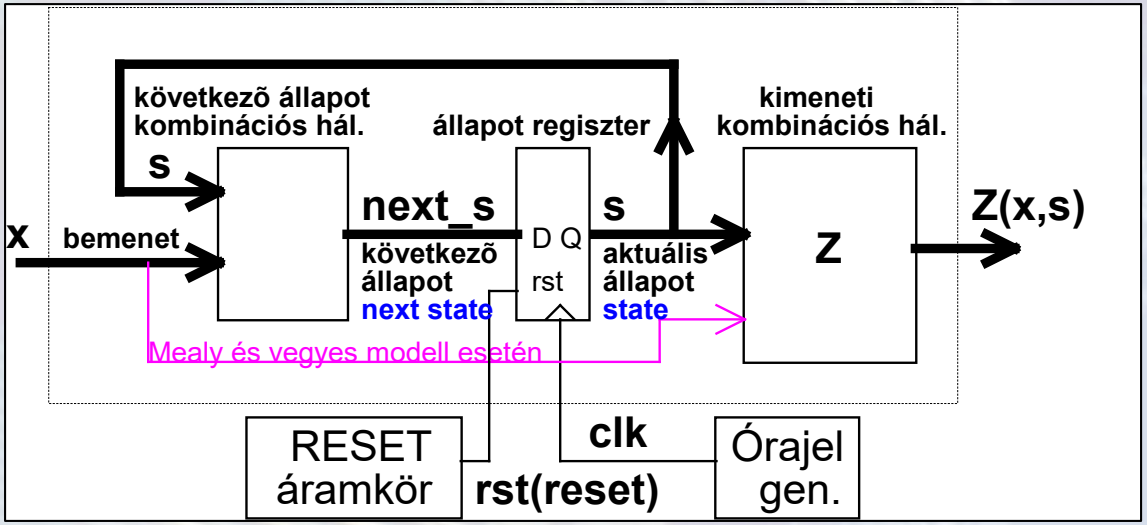
- Most minimális hosszúságú kódolást alkalmazunk. A 3 állapothoz 2 bit elegendő.
- Most véletlenszerűen ***kódoljuk az állapotokat és kitöltjük a kódolt állapottáblát***.

s[1:0]	x=0	x=1
A 00	A 00/0	B 01/0
B 01	A 00/0	C 11 /0
C 11	A 00/1	C 11 /0
10	A 00/0	A 00/0

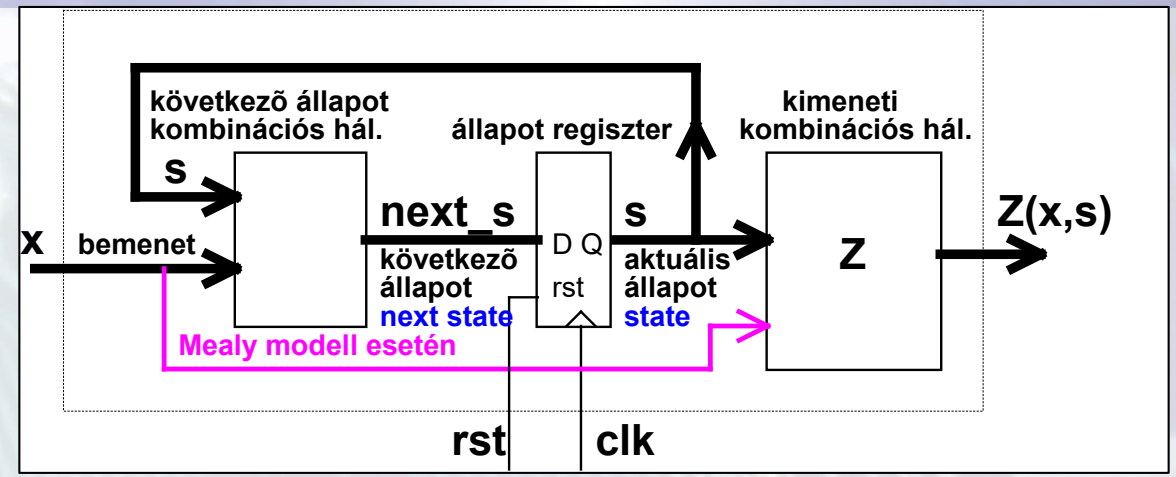
- A 2 biten 4 lehetséges kódszó van. Ebből csak 3-at használunk. A maradék 1 db kód tranziens hiba esetén előállhat. Ezért az utolsó sort úgy töltöttük ki, hogy ebben az esetben a kezdő (A) állapotba kerüljön a hálózat.
- Az az elv, hogy a nem használt állapotkódokból használt állapotkódokba jusson a hálózat.

Az állapotgép felépítése

- Mielőtt folytatnánk a tervezést nézzük meg a megvalósító logikai hálózat felépítését
- Az óregenerátor állítja elő az órajelet
- A RESET generátor (reset áramkör) állítja elő induláskor (pl. bekapcsoláskor) a reset impulzust, melyet a kiinduló állapot beállítására használunk.
- A továbbiakban ezeket nem rajzoljuk le, csak a clk és rst jeleket



Az állapotgép felépítése

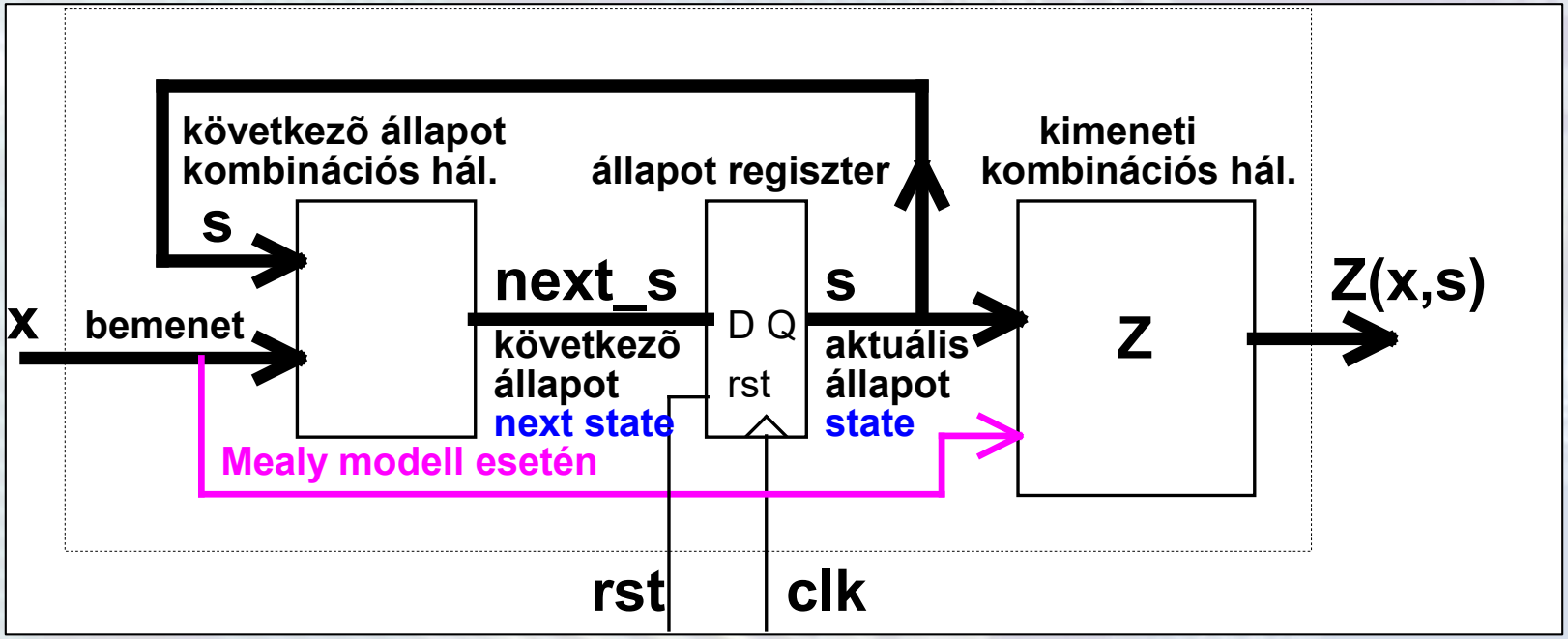


- Az ***állapot regiszter órajelenként*** tárolja az aktuális állapot (s) kódját.
- A **következő állapot** (next_s) **kódját** a regiszter bemenetére kapcsolódó logika (kombinációs hálózat) állítja elő, az aktuális állapot (s) és az aktuális bemenet (x) alapján.
- A **kimeneti logika** (kombinációs hálózat) állítja elő a kimenetet (Z), mely általános esetben az aktuális állapottól (állapotregiszter tartalma) és az aktuális bemenettől (x) függ.

Az állapotgép felépítése

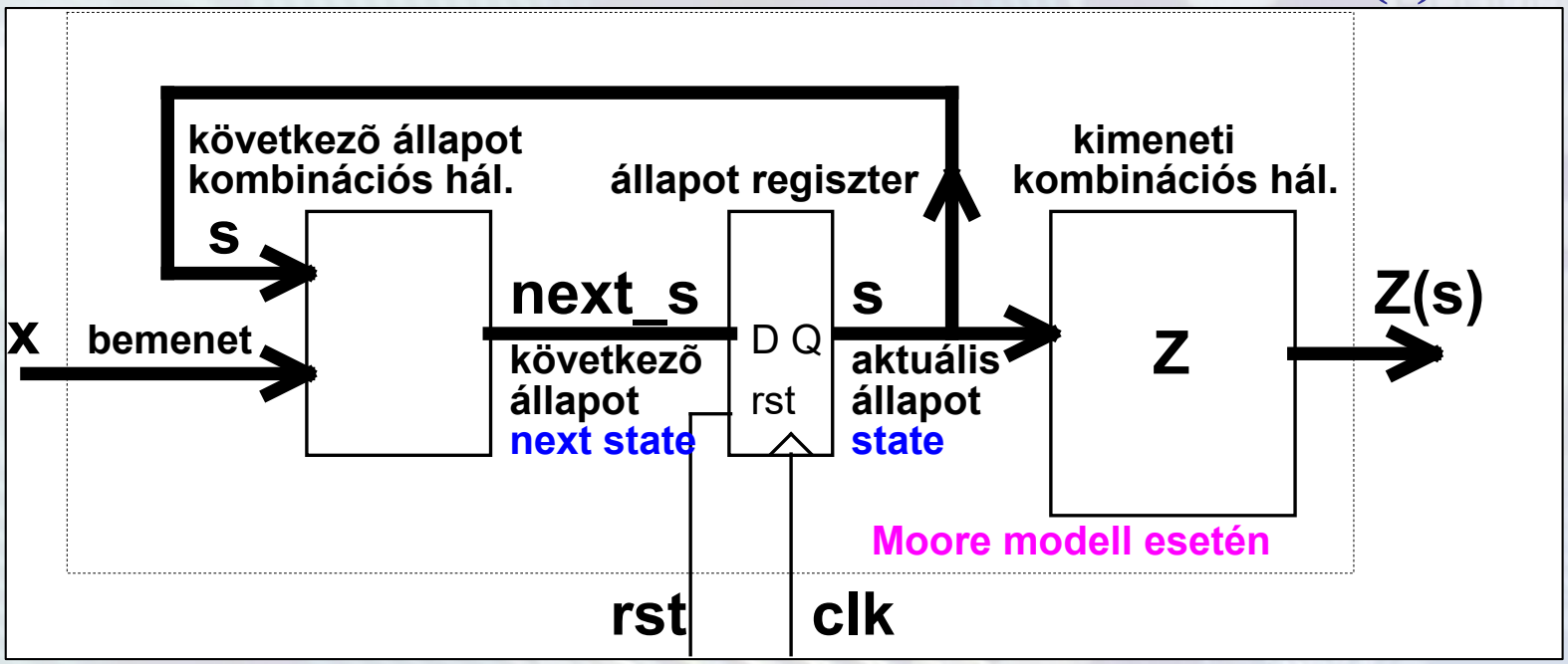
Mealy modell

- A szinkron sorrendi hálózat *kimenete* az *aktuális állapot és az aktuális bemenet függvénye* ún. ***Mealy modell*** szerinti működésnél: **z=Z(s ,x)**



Az állapotgép felépítése

- **Moore modell**
- A kimenet csak az aktuális állapottól függ az ún. ***Moore modell*** szerinti működés esetén: **z=Z(s)**

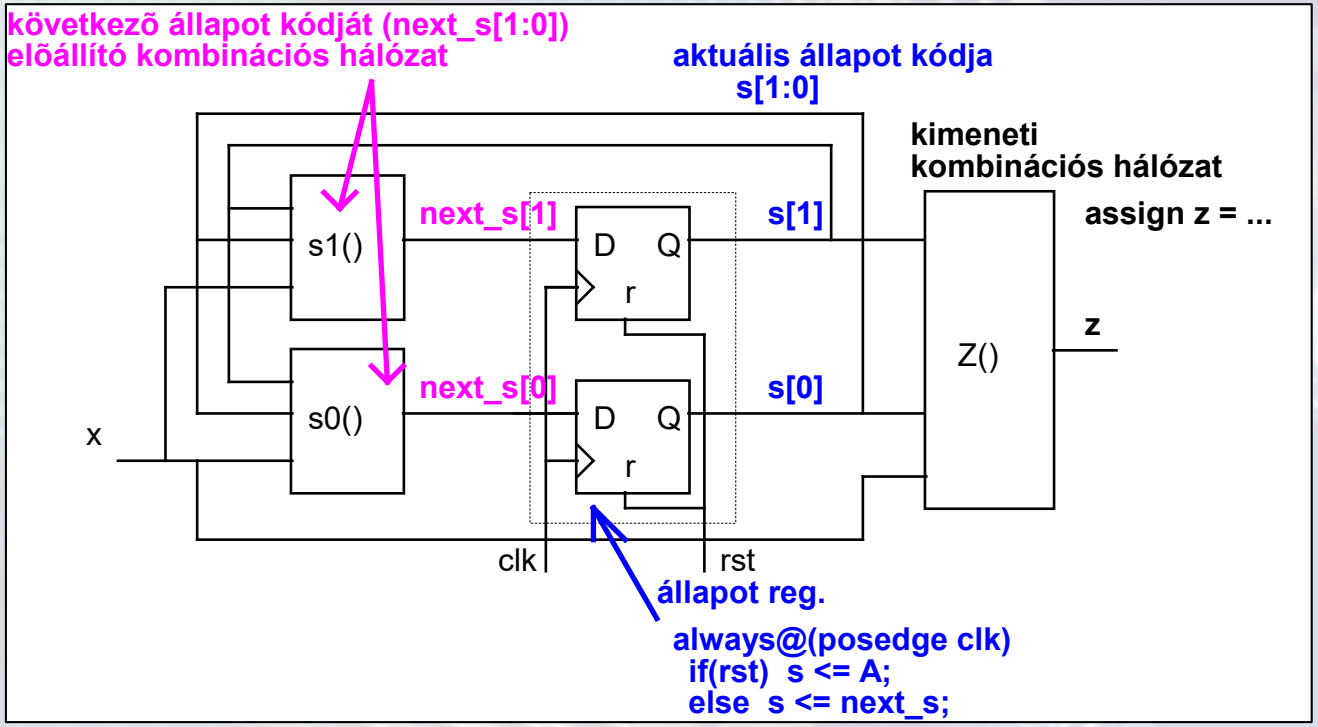


- **Vegyes modell** esetén egyes kimeneti bitek Mealy, $z_i=Z_i(s,x)$, mások Moore jellegűek $z_j=Z_j(s)$

Az állapotgép felépítése

Következő állapot kódját előállító logika (next_s) és a kimeneti függvény (z) meghatározása

- Az aktuálisan megtervezendő automata struktúrája (2 bites állapotregiszter, egyetlen kimenet, Mealy):



Az állapotgép kombinációs hálózatainak tervezése

Hagyományos módszer

- Megtervezendők a következő állapot logika **next_s1(s1,s0,x)** **next_s0(s1,s0,x)** és a kimenet **Z(s1 ,s0 ,x)** függvénye
- Igazságtáblájukat egyszerre tartalmazza a **kódolt állapottábla**.

s1s0	**x=0** next_s1, next_s0/z	**x=1** next_s1, next_s0/z
A 00	A 00/0	B 01/0
B 01	A 00/0	C 11/0
C 11	A 00/1	C 11/0
10	A 00/0	A 00/0

Az állapotgép kombinációs hálózatainak tervzése

- Az igazságtáblák (Logic Friday-ban megadva):

Term	s1	s0	x	=>	next_s1	next_s0	z
0	0	0	0		0	0	0
1	0	0	1		0	1	0
2	0	1	0		0	0	0
3	0	1	1		1	1	0
4	1	0	0		0	0	0
5	1	0	1		0	0	0
6	1	1	0		0	0	1
7	1	1	1		1	1	0

- A Logic Friday-val minimalizált függvények:
  (Abban a formában, ahogy az eredményt adja. Az x' az x negáltját jelöli)

  next_s1 = s0 x; next_s0 = s0 x + s1' x;

  z = s1 s0 x';

Az állapotgép leírása Verilog-ban

```
reg [1:0] s, next_s;
//Állapotkódolás megadása
parameter A = 2'b00, B = 2'b01, C = 2'b11;
//Állapotregiszter viselkedési leírása
always @ (posedge clk)        //órajel felfutó élre működik
begin
    if (rst)    s <= A;    // rst-re kezdő állapot
    else        s <= next_s;        // egyébként következő
end
```

Az állapotgép leírása Verilog-ban

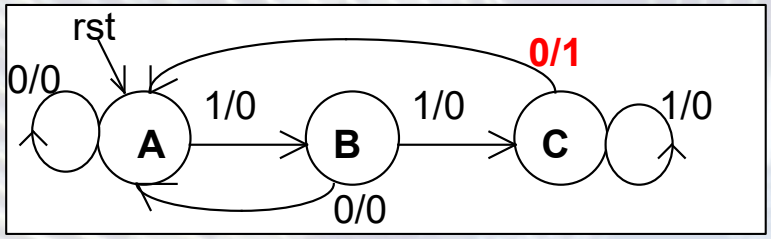
- A következő állapot logikát többféleképpen is megadhatjuk.
- Megadhatjuk a minimalizált függvények alapján ***Boole algebrai alakban assign-al***. (Ezt hagyományos módszernek nevezzük.)

```
//next state logika
assign next_s[1] = s[0]&x;
assign next_s[0] = s[0]&x | ~s[1]&x;
// s[0]&x közös szorzat tagok, csak egyszer lesz megvalósítva
```

Az állapotgép leírása Verilog-ban

- Azonban sokkal *célszerűbb* az **állapotgráf vagy állapottábla alapján** *az állpotátmeneteket megadni* ***viselkedési leírással*** *és a minimalizálást a tervezőrendszerre bízni*. (Ezt általános FSM tervezési módnak nevezzük.)

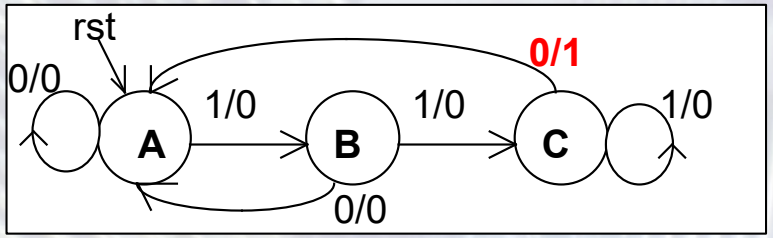
```
//next state logika
always @(*)
case(s)
A: if(~x) next_s <= A;
    else  next_s <= B;
B: if(~x) next_s <= A;
    else  next_s <= C;
C: if(~x) next_s <= A;
    else  next_s <= C;
default:next_s <= A;
endcase
```



Az állapotgép leírása Verilog-ban

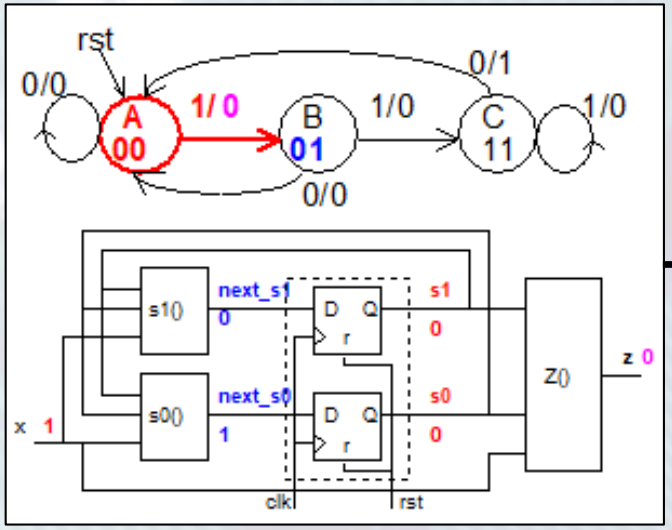
A kimeneti függvény megadása

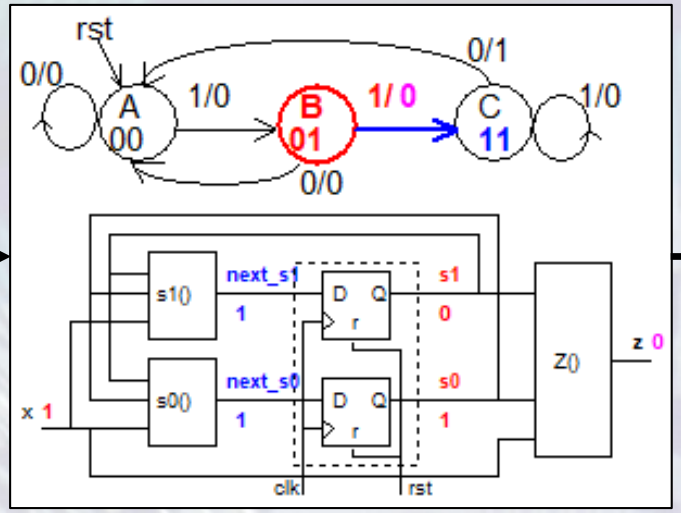
- A kimeneti függvény megadására is több lehetőségünk van.
- Megadhatjuk a Logic Friday által minimalizált függvényt SOP ***Boole algebrai alakban***:
  assign z = s[1]&s[0]&~x; //hagyományos tervezési mód
- Megadhatjuk az ***állapotgráf/állapottábla alapján az állapotkódok felhasználásával***:

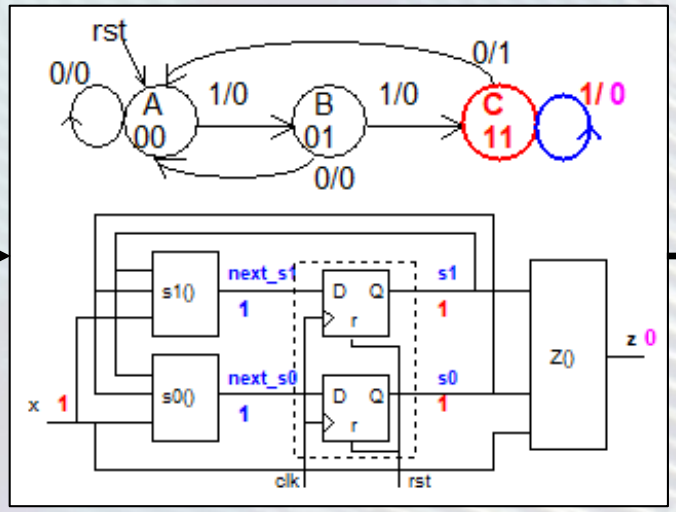


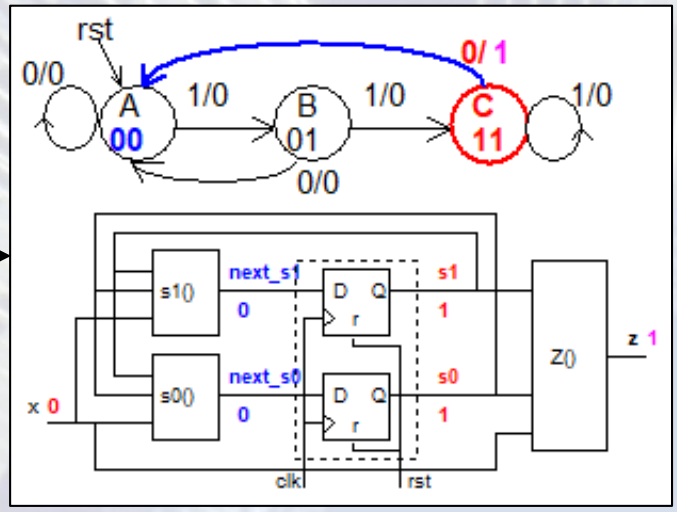
C állapotban van és X=0, Z=1
assign z = (s == C)&~x; //általános FSM
vagy
assign z = (s == C)&(x == 1'b0); //általános FSM

Működés reset után az X=1110 bemeneti sorozat hatására



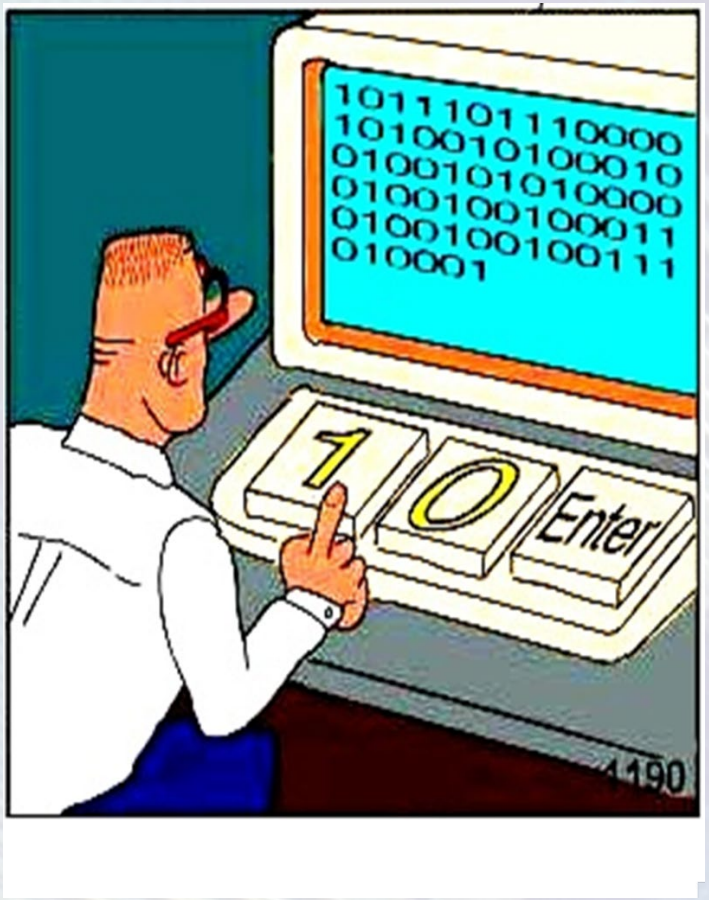






Kérdések, észrevételek

- **A kérdéseket a benes@mit.bme.hu címre várjuk**



Digitális technika 4. EA vége
A további diák nem szerepelnek a vizsgakövetelményekben

FSM formális definíciója

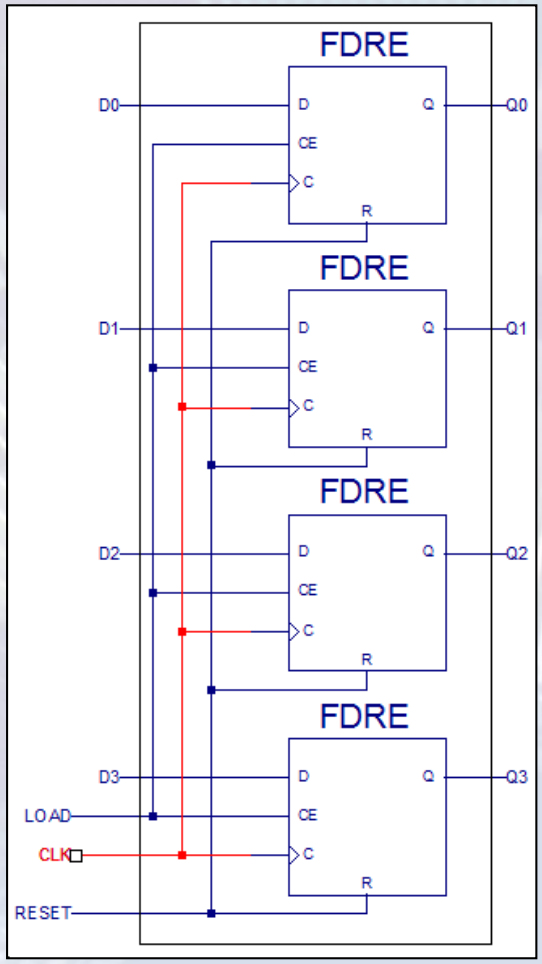
- **Véges állapotú gépek jellemzése**
- **Az FSM formális definíciója a következő:**

{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 esetleg a bemeneti jelek) alapján a ***kimenet értékét*** állítja elő
- **O** a lehetséges kimeneti értékek halmaza

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

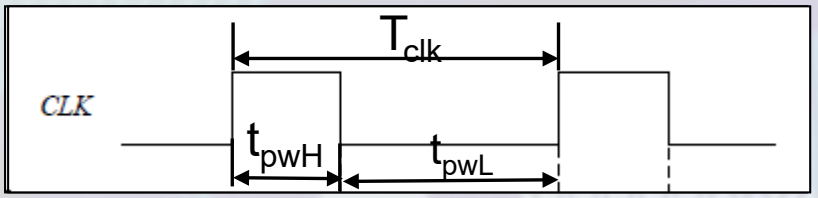


Regiszterek időzítési jellemzői

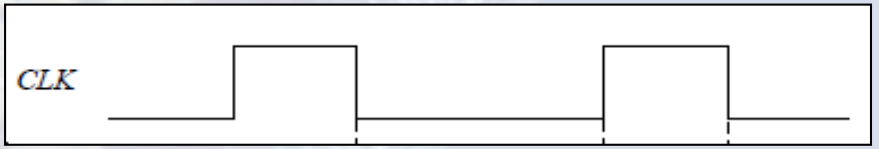
- **Az időzítési adatok többsége az órajellel kapcsolatos**
- **Az órajel:**
  - Az órajel egy speciális jel
  - Normál esetben nem tartozik a logikai jelek közé
  - Nem része a logikai kifejezéseknek
  - Egyetlen (de nagyon fontos) feladata a rendszerben lévő szinkron működésű elemek egységes, azonos 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)
    - Időben: Az órajel minden periódusának stabilitása jó
      Rövid idejű stabilitás → Jitter

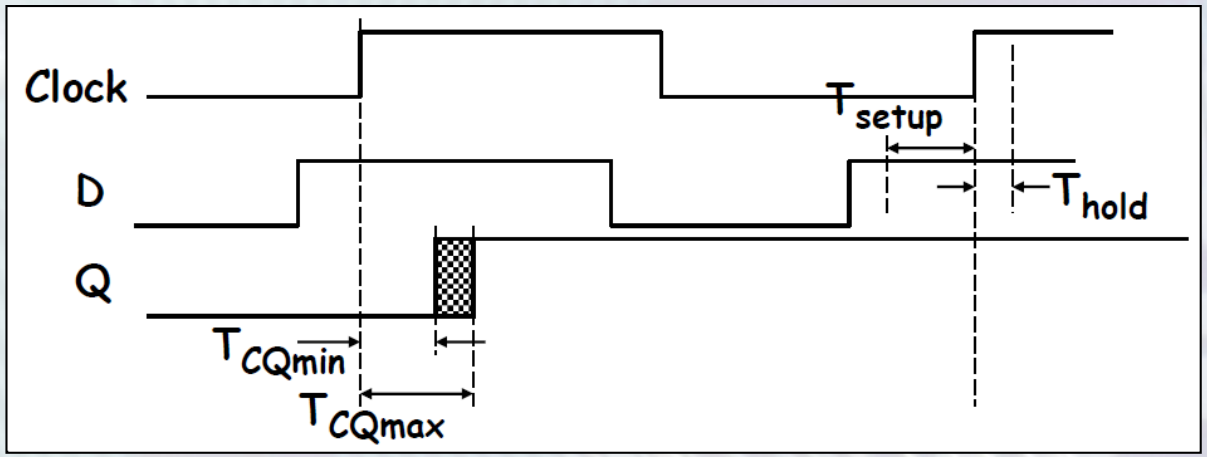


Hosszú idejű stabilitás → Lassú frekvencia változás



Regiszterek időzítési jellemzői

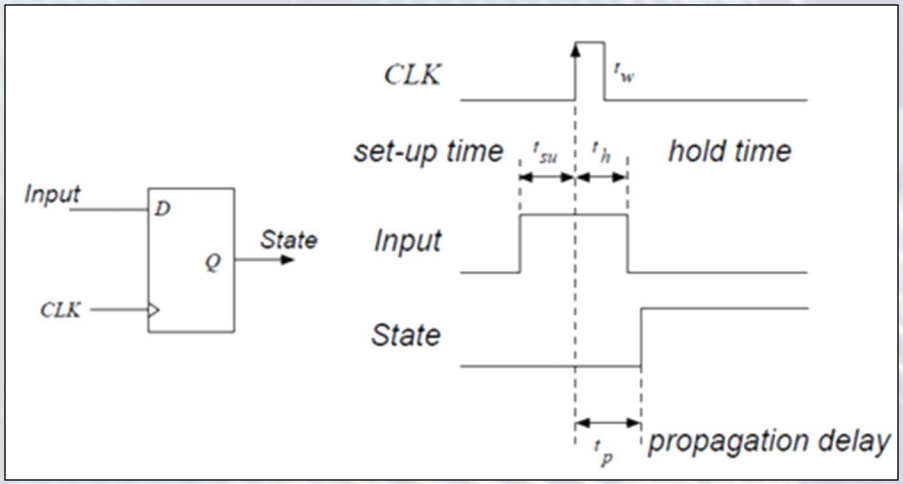
- **A FF-ok, regiszterek időzítési jellemzői a következők**



- 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

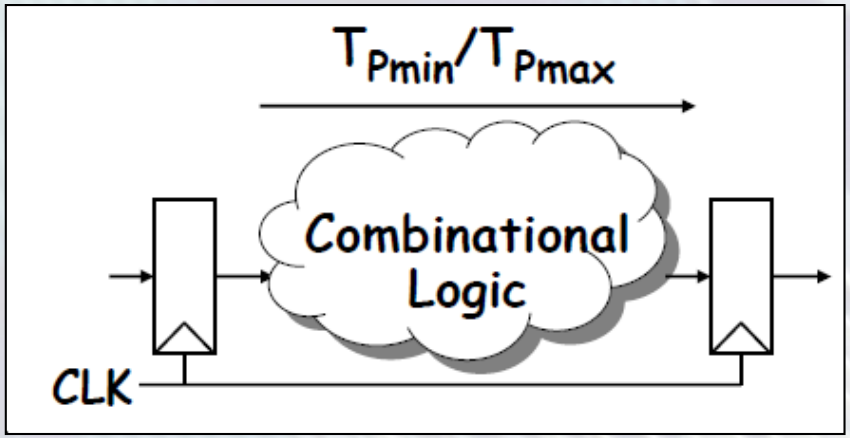
- **A helyes működés feltétele ezen időzítési paraméterek, követelmények betartása**
- **Amennyiben az előkészítési és tartási idők teljesülnek, a kimenet a megadott időn belül reagál a vezérlésre**



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

Regiszterek időzítési jellemzői

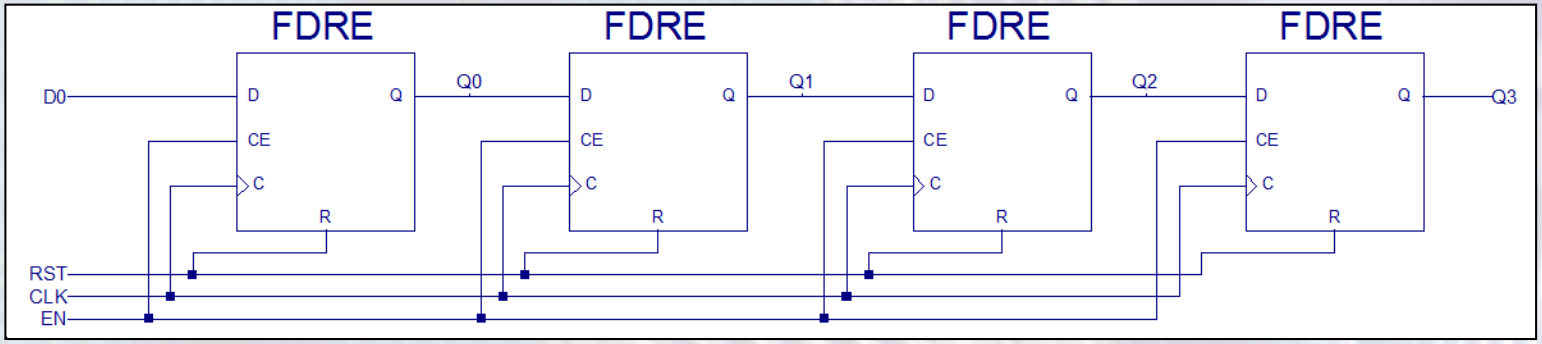
- **A időzítési paraméterek határozzák meg az elérhető működési sebességet**
- **Egy összetett rendszer működésének időbeli jellemzői**



- **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

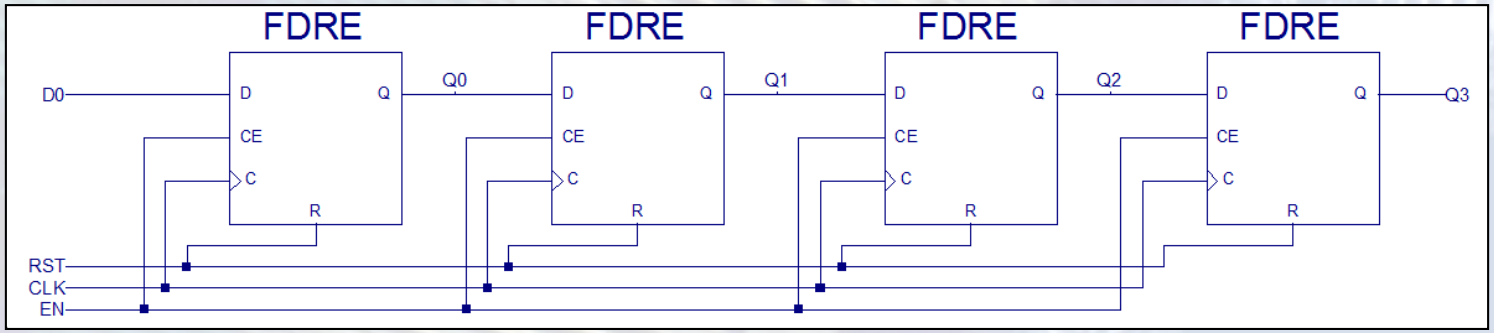
- **SHIFTREGISZTER**
  - A időzítési követelmények elemzését a shift regiszterrel mutatjuk be
  - A shiftregiszter a legegyszerűbb sorrendi hálózat, egyben az egyik legfontosabb funkcionális egység



  - 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 órajelfrekvenciával 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!!



Digitális technika 4. EA vége