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

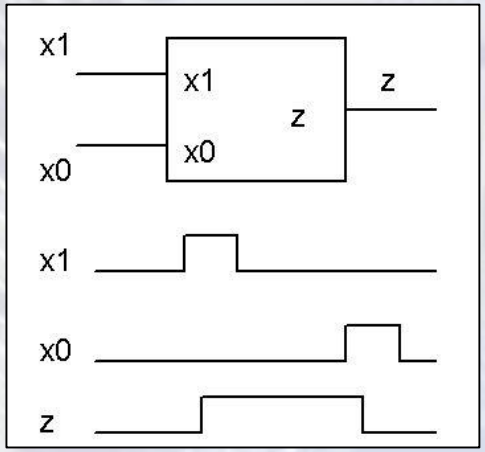
Digitális technika VIMIAA02

Fehér Béla, Benesóczky Zoltán
BME MIT

Sorrendi hálózatok

(Finite State Machine, FSM)

- Az eddigiekben megismert digitális áramkörök ***kombinációs hálózatok*** (a kimenet csak az aktuális bemenetektől függ)
- Sok esetben a feladatok megoldásához ez nem elegendő, ***a*** rendszer viselkedését befolyásolja, hogy az egyes ***bemeneti értékek időben milyen sorrendben érkeztek***. Az ilyen hálózatokat ***sorrendi hálózatok***nak nevezik.
- **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.)



- A sorrendi hálózatok megvalósításához, ***memória*** *(tároló)* funkció is kell.

Szinkron sorrendi hálózatok (SSH)

- A sorrendi hálózat által ***tárolt információt állapot***nak (***state***) nevezzük.
- Léteznek ún. aszinkron sorrendi hálózatok azonban ezek tervezése problémás. Ezért mi csak ***szinkron sorrendi hálózatok***kal foglalkozunk.
- A szinron tároló funkció megvalósításához új alapelemre tárolóra van szükség.

Szinkron sorrendi hálózatok (SSH)

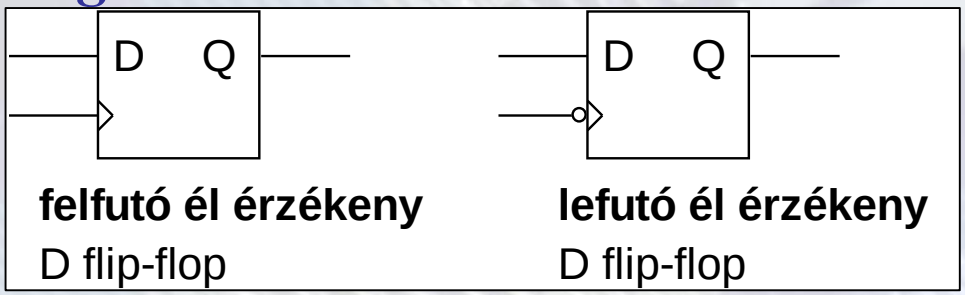
- A szinkron hálózatok ***tárolóiba írást*** az ***órajel (clk, c) ütemezi.***
- Az órajel egy folyamatos periodikus négyszög jel, melyet az ***órajel generátor*** állít elő.

- ***A szinkron működés azt jelenti, hogy az órajel valamelyik élének*** (változásának) ***hatására íródhat új érték a tárolóba***.
- Az ütemezés történhet ***az órajel 0-1 átmenetére (felfutó él)*** vagy 1-0 bemenetére (lefutó él). Mi csak ***felfutó élet*** használunk. Tehát ***az órajel aktív éle a felfutó él.***

- Az órajel **prióduszideje** T_{clk}, ***frekvenciája*** $f_{clk}=1/T_{clk}$, az ***áramkör kívánt sebessége*** alapján állítják be.
- ***Egy sorrendi hálózat minden tárolója ugyanazt az órajelet és kapja és azonos aktív élet használ.*** (Itt csak felfutó élet.)

A D flip-flop

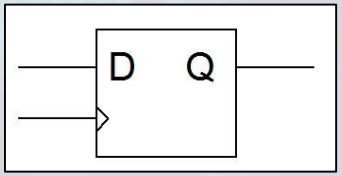
- **A digitális technikában használt legfontosabb bit tároló egység a D flip-flop**
  - A DFF az órajel (clock, clk, c) ***aktív élének*** hatására mintavételezi és tárolja a bemeneti D adatbitet.
  - A tárolt érték jelenik meg a Q kimenetén (Q = D) és ***tárolódik a következő aktív élig***.
- **D ff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

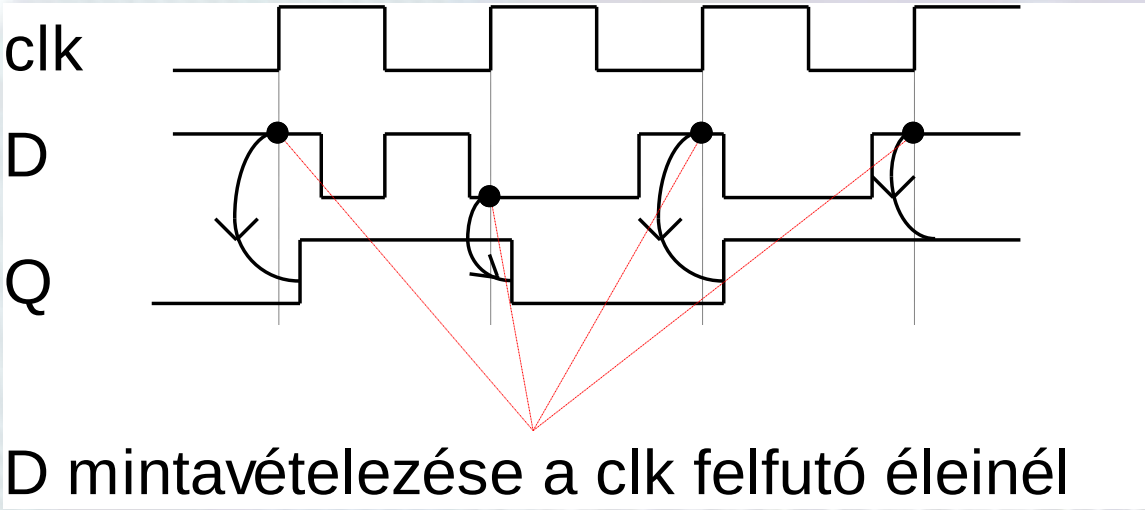


- Az ***élérzékeny órajel bemenetet*** a bemenet melletti > jel jelöli a rajzokon.
- A továbbakban csak felfutó él érzékeny tárolókat fogunk használni.

A D flip-flop

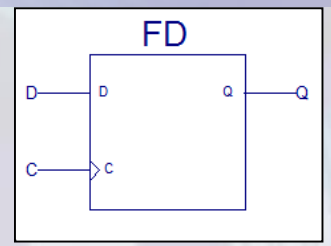
A D flip-flop működését mutató idődiagram:





A felfutó él pillanatában a D bemenet értékét ponttal jelöltük. A D flip-flop ezt az értéket tárolja, ami kis késleltetés múlva megjelenik a Q kimenetén. Ott ez az érték marad addig, amíg a következő felfutó él hatására felülíródik az új D bemeneti értékkel.

A D flip-flop



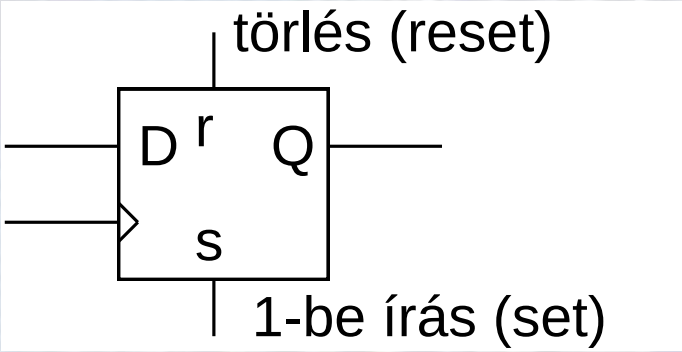
- **A DFF egy kész áramköri egység** (alapelem, mint a kapuk) Nem foglalkozunk a belső felépítésével**.**
  - ***Verilog viselkedési leírással*** építjük be.
  - Az ***always blokk*** érzékenységi listájában a ***felfutó él eseményre hivatkozás***: **posedge jelnév**
  - A Digitális technika tárgyban leírt ***szinkron sorrendi hálózatokban a viselkedési leírása mindig***: **always(posedge clk)** formával kezdődik. (Az érzékenységi listában egy jelek is lehetnének.)
  - **DFF viselkedése**: 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. **Verilog leírás:**

```
reg q;
always@(posedge clk)
  q <= d;
```

A **reg típusú változó <= kifejezés**-ben a „<=” **nem blokkoló értékadás**t jelöl. Ezt nem magyarázzuk, a Digitális technika tárgyban leírt *órajel érzékeny viselkedési leírásokban* ***mindig ezt kell használni***. A *tárgyban leírt* kombinációs hálózatok viselkedési leírásában is használhatjuk az = helyett.

A DFF

- **Bekapcsoláskor a Q értékét a feladatnak megfelelően kell inicializálni.**
- ***Ezért a flip-flopnak alaphelyzet beállító bemenetei is lehetnek:***
  - RESET (r, pr): a jelre Q = 0 az órajel ↑ élre áll be (szinkron)
  - SET (s): a jelre Q = 1 az órajel ↑ élre áll be (szinkron)
  - Ha mindkettő van és egyszerre aktívak, az hatásos amelyiknek a prioritása nagyobb.

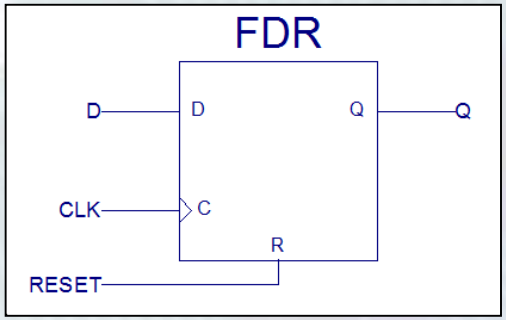


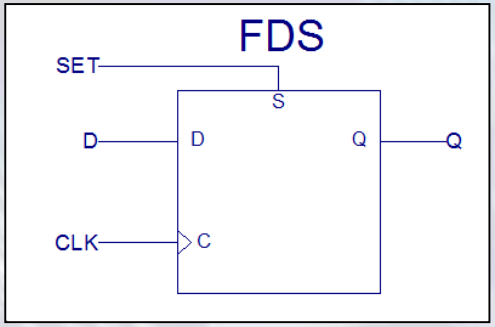
A DFF

- A sorrendi hálózatok tárolóit a bekapcsoláskor és esetleg nyomógomb hatására alaphelyzetbe kell állítani. Erre szolgál a **RESET (rst)** jel.
- Az RESET jelet 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 csak az általuk előállított **clk** és **reset** (res) jeleket használjuk.

A DFF alaphelyzet beállító jelei

- **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;
```

- 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.

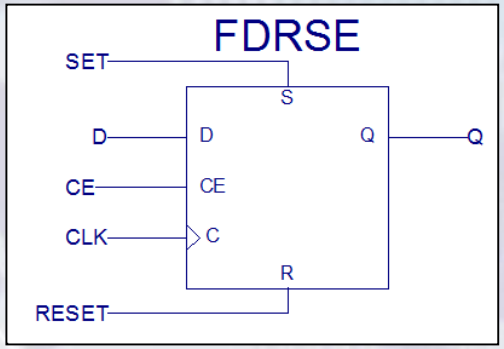
- **Verilog HDL kódminta** (a rst prioritása a legnagyobb):

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

- Ha egyszerre aktív rst és set, akkor q = 0 lesz az órajel felfutó él hatására. Ha rst=0 és set=1, akkor q = 1 lesz. Ha rst = 0 és set = 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 SET bemenetekre nincs!



Verilog kód minta:

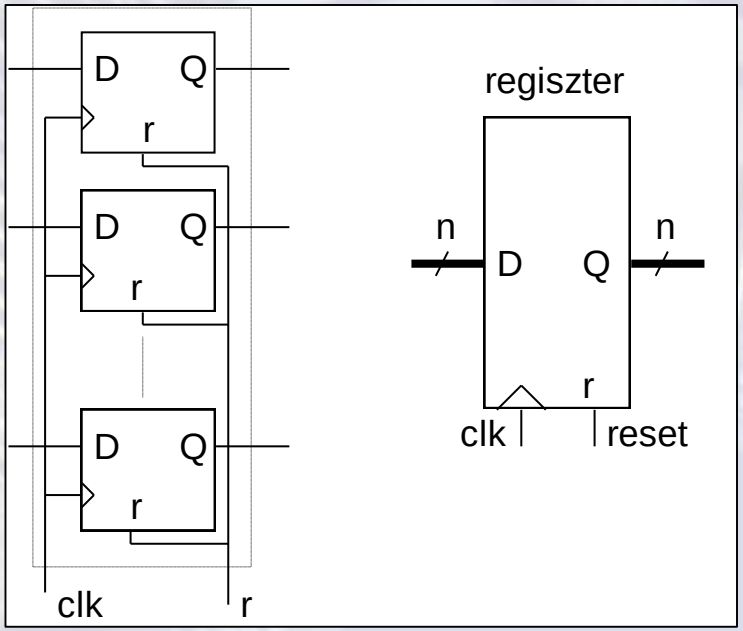
```
reg Q;
always @ (posedge clk)
   if (RESET)          Q <= 1'b0;
   else if (SET)       Q <= 1'b1;
        else if (CE)   Q <= D;
```

  - A ***több bites regisztereknél*** ugyanezt a funkciót ellátó jel neve **LD** (LOAD), azaz adatbetöltés vezérlőjel.

A regiszter

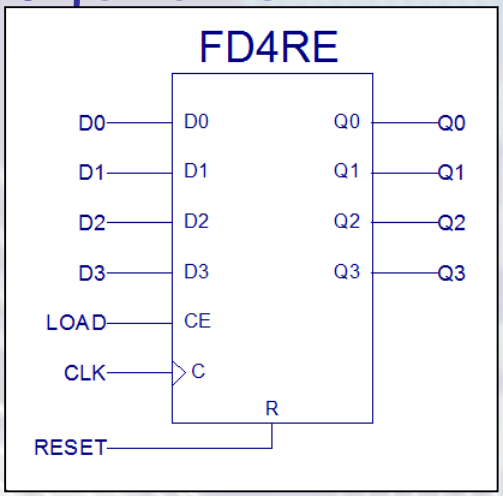
- **A regiszterek DFF-okból felépített több bites tárolók**
- A regiszterek felépítése olyan, hogy a ***D adatbemeneti bitek kivételével az összes többi vezérlőjelük közösített, azaz egyszerre működik.*** (Pl. a reset minden bitet töröl.)
- A regiszterek mérete 2-től akár 64 bitig terjed
- A legegyszerűbb 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(rst) Q <= 4'b0;
  else    Q <= D;
```



Tölthető regiszter

- A regiszter **LOAD** (LD) **adat betöltő** jellel is rendelkezhet



Verilog HDL kódminta:

```
reg [3:0] Q;
always@(posedge clk)
if(rst) Q <= 4'h0;
else if(LOAD) Q <= D;
```

- Itt a RESET magasabb prioritású, mint a LOAD, mert az if szerkezetben az van előbb.
- Ha egy **sorrendi hálózatban** nem változik egy tároló regiszter értéke, azt **Verilogban** nem kell leírni (Q <= Q), mert **amíg egy reg típusú változó nem kap új értéket, addig megtartja a régit**. Ezért az utolsó sor után **nem kell ezt írunk:** else **Q <= Q;**

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

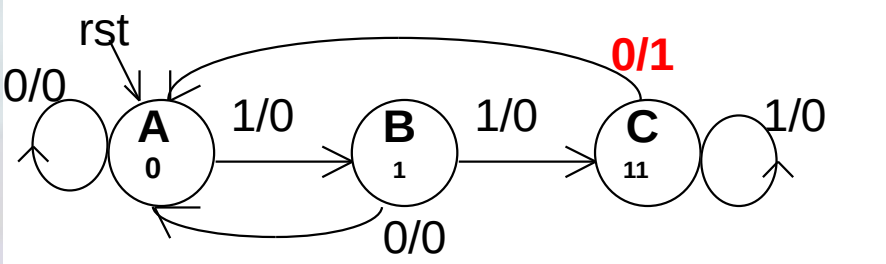
- Az állpotgépeket megadhatjuk ***szöveges specifikáció***val. Pl. Ismerje fel az **x** bemenetére az órajellel szinkronban érkező bitsorozatban, ha a ***legutolsó*** 3 bit **110** és a **z** kimenetén jelezzen 1-el az utolsó bit beérkezésével egyidőben.

  x: 10**110**01**110**0...
  z: 0000**1**0000**1**0...

- Megadható ***állapotgráffal*** (állapot diagramnak is nevezik). (Az állapotgráfot általában a 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,*** 0-ra az ***A***).
- 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

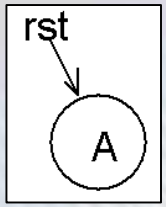
- Az állapotgráf rajzolása során ***egy olyan kezdőállapotból indulunk ki, ami azt az információt tárolja, hogy a bekapcsolás (reset) óta nem jött adat***. Ezt az állapotot általában A-val jelöljük és ***berajzolunk egy bele mutató rst (reset) feliratú nyilat***.
- Ezután a már meglévő állapot(ok)ból 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 abba irányítjuk a gráfot***. ***Ha nem, a már meglévő megfelelő állapotba***.
- Az állapotokba az első felíráskor azt az információt is beírhatjuk, 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

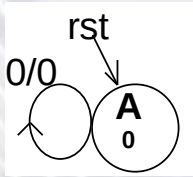
Példa: A mintafeladat gráfjának megtervezése:

Ismerje fel az x bemenetére az órajellel szinkronban érkező bitsorozatban, ha a ***legutolsó 3 bit 110*** és ***1-el jelezze az utolsó bit beérkezésével egyidőben***.

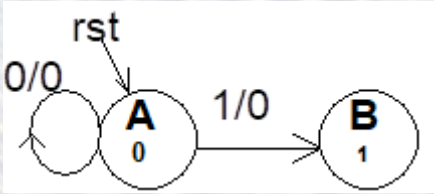
1. Felvesszük ***A*** állapotot és berajzoljuk, hogy reset-re ez lesz a kezdő állapot.



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

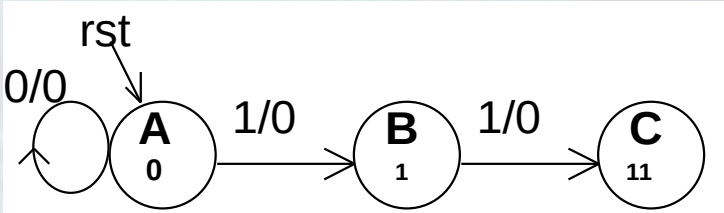


3. Ha x = 1, akkor megjött a sorozat első bitje, ezt meg kell jegyezni, új állapotra van szükség. Felvesszük **B** állapotot (új név ABC sorrendben). Ez megjegyzi, hogy bejött 1 db 1-es bit.

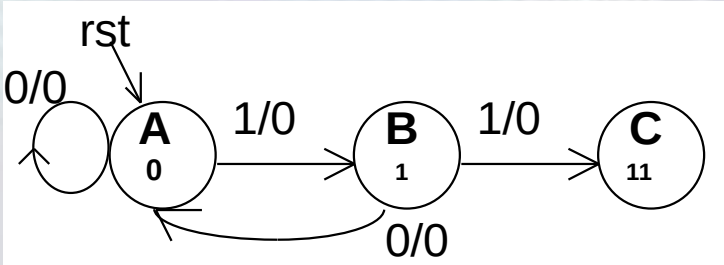


Állapotgépek specifikálása

4. A **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.

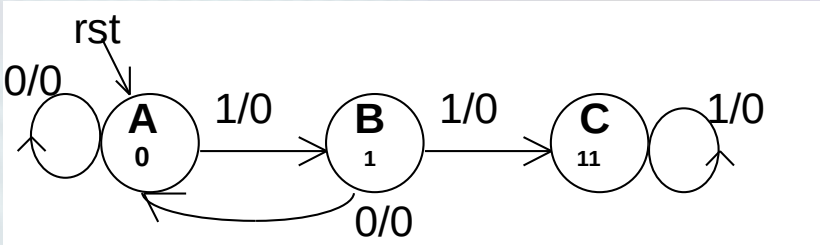


5. **B** állapotban, ha x = 0, akkor 1 után 0 jött, elölről kell kezdeni a figyelést, mert ez már nem lehet a jó sorozat. Vissza kell menni **A**-ba ami csak annyit jegyez meg, hogy ideáig csak 0 jött.

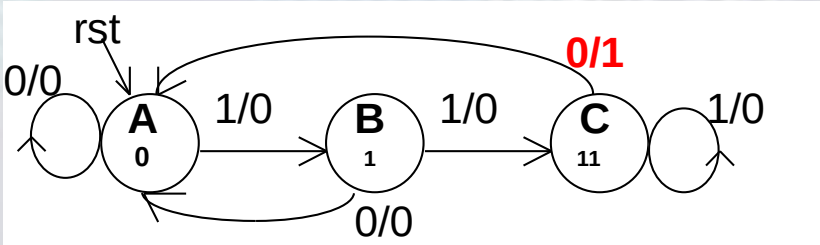


Állapotgépek specifikálása

6. **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.



7. **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 **C**-ből **A**-ba mutató nyilat rajzolunk és ráírjuk, hogy **0/1**.



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

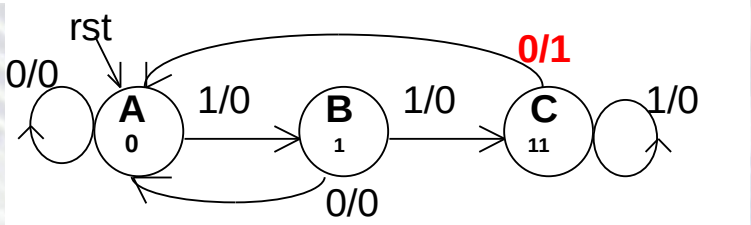
Állapottábla

- 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 aktuális 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	A/0	C/0
C	A/1	C/0

Ha C az aktuális állapot, a C állapot után következõ állapot és a kimenet

ha x =0 ha x =1

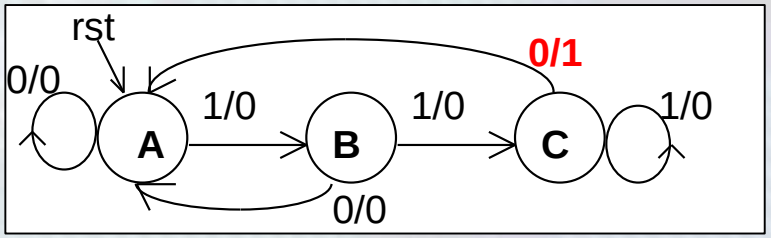


Állapotminimalizálás

- 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 **állapotminimalizálás**nak nevezik.
- A kevesebb állapotszámú állapotgép sokszor egyszerűbben valósítható meg.
- A példa gráfja minimális. Az állapotminimalizálási algoritmusokkal nem foglalkozunk.

Állapotkódolás

- A ***minimalizált állapotgráf*** alapján elkészítjük a ***minimalizált állapottáblát***.



	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 állapotkódolási módszerek léteznek***. (Ilyeneket nem tanulunk.)

Állapotkódolás

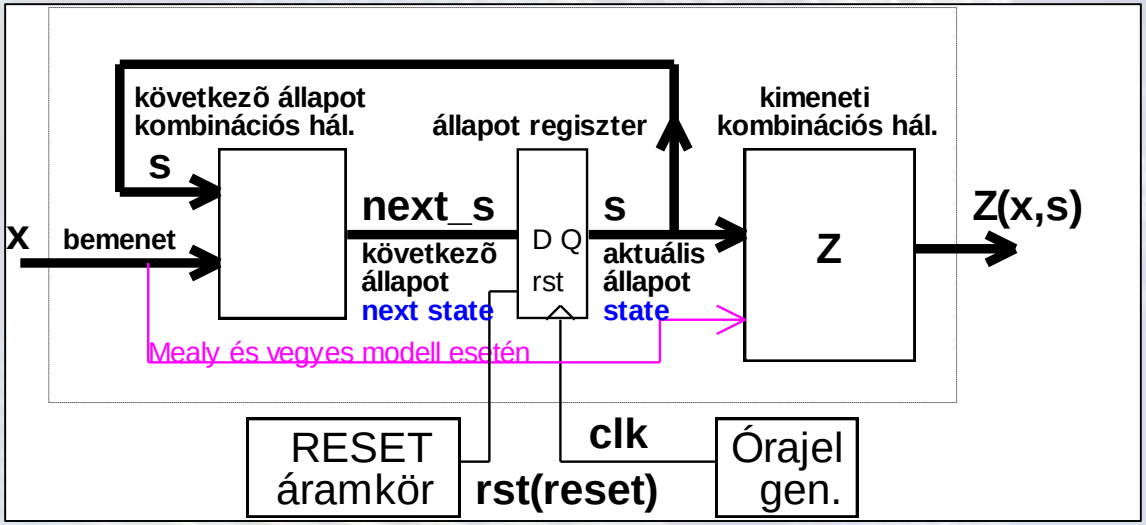
- Most minimális hosszúságú kódolást alkalmazunk. A 3 állapothoz 2 bit elegendő.
- Véletlenszerűen ***kódoljuk az állapotokat és kitöltjük a kódolt állapottáblát***.

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

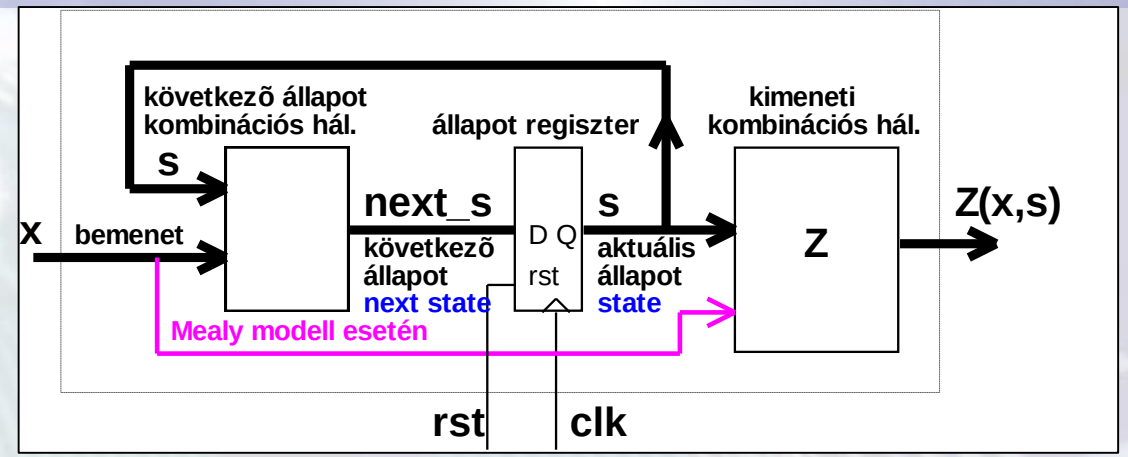
- A 2 biten 4 lehetséges kódszó van. Ebből csak 3-at használunk. A maradék 1 db kód tranziens hiba esetén előállhat. Ezért az utolsó sort úgy töltöttük ki, hogy ebben az esetben a kezdő (A) állapotba kerüljön a hálózat.
- Az az elv, hogy a nem használt állapotkódokból használt állapotkódokba jusson a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

Az állapotgép felépítése

- Mielőtt folytatnánk a tervezést nézzük meg a szinkron sorrendi hálózatot megvalósító logika általános 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

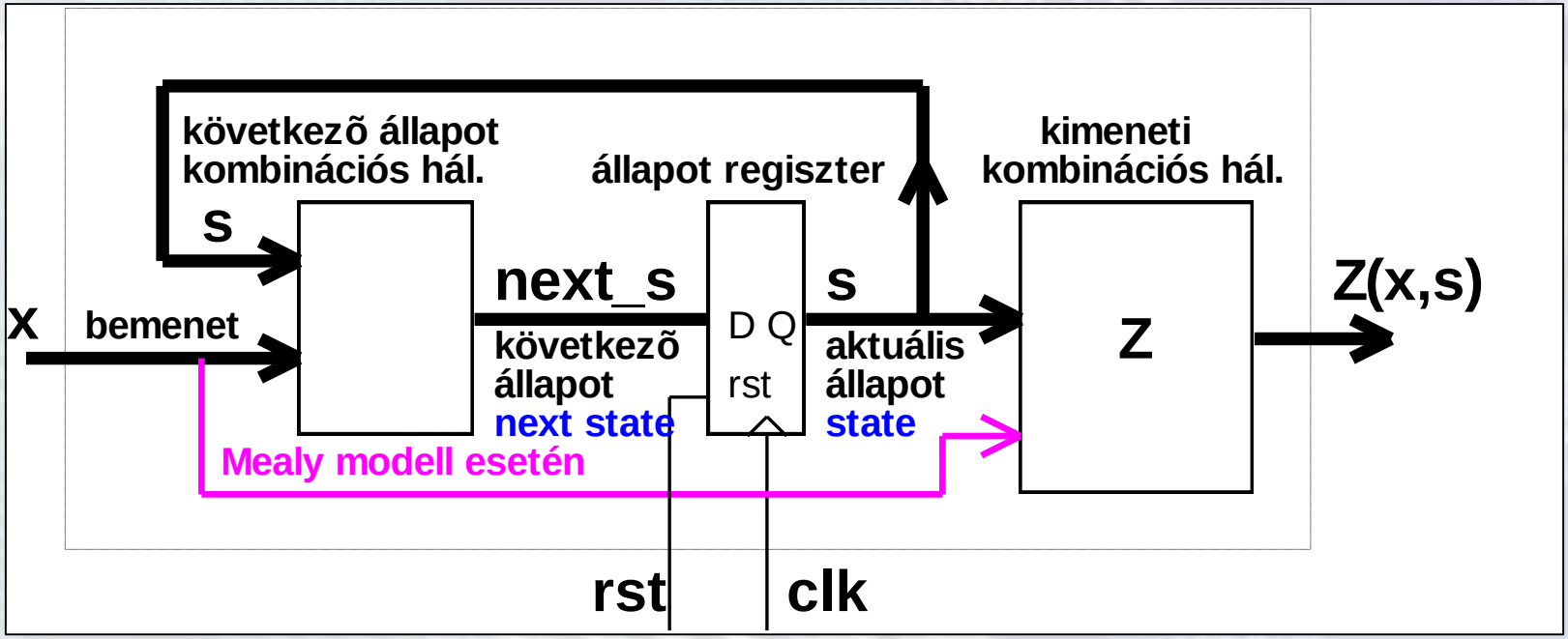


- Az ***állapot regiszter órajelenként*** tárolja az aktuális állapot (s) kódját.
- A **következő állapot** (next_s) **kódját** a regiszter bemenetére kapcsolódó logika (kombinációs hálózat) állítja elő, az aktuális állapot (s) és az aktuális bemenet (x) alapján: **next_s(s, x)**
- A **kimeneti logika**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

Az állapotgép felépítése

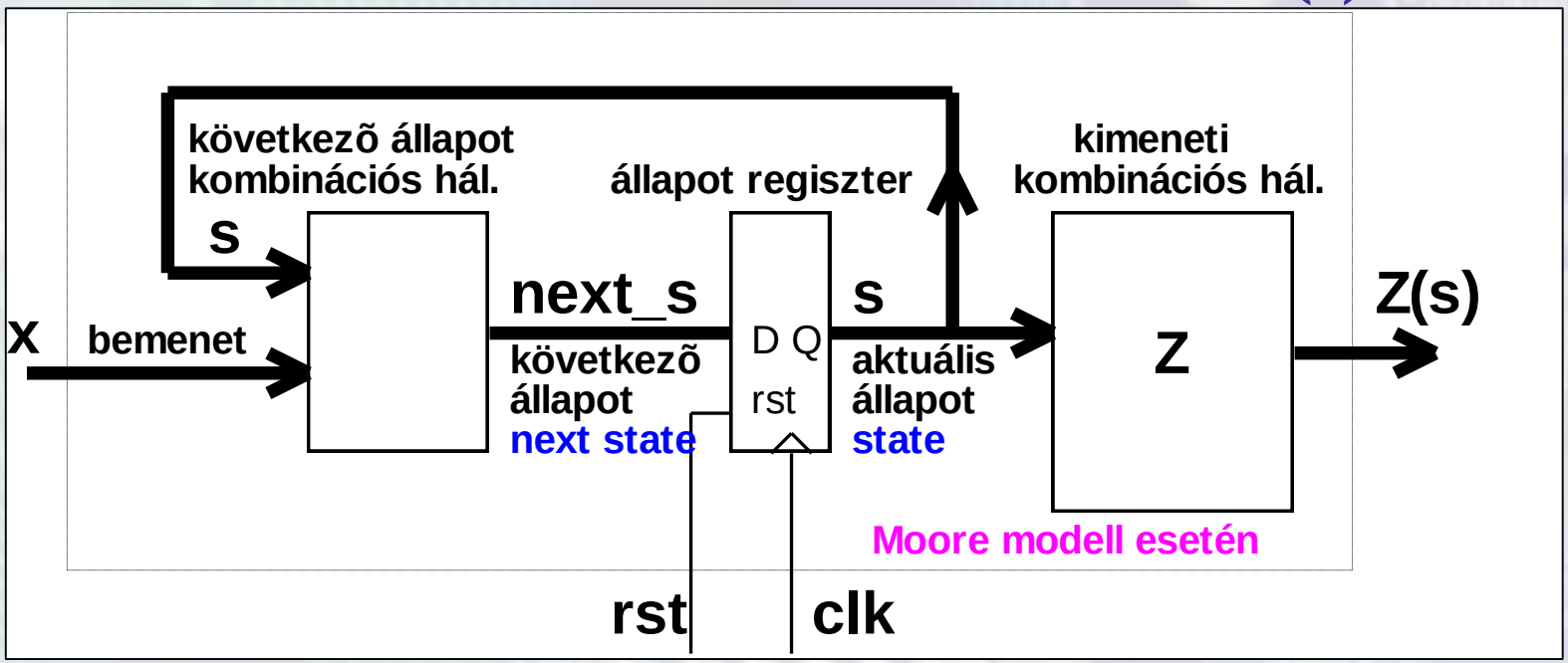
Mealy modell

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



Az állapotgép felépítése

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

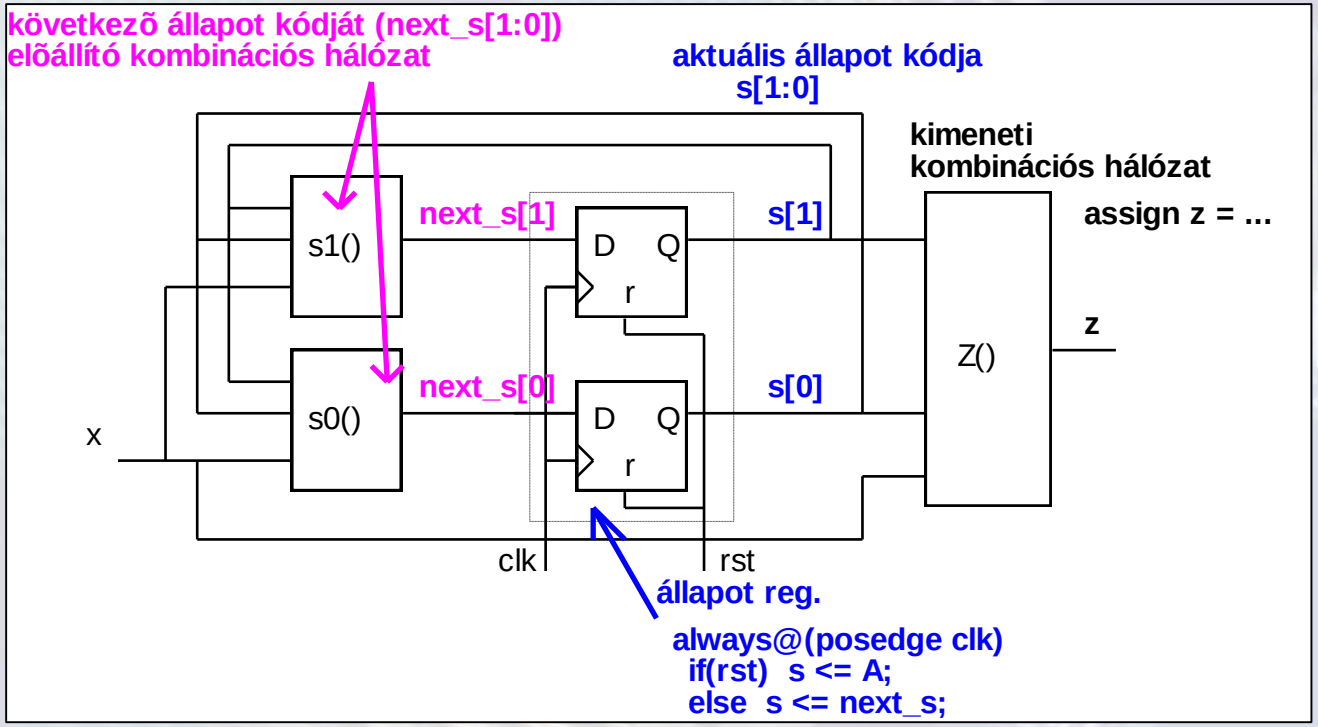


- **Vegyes modell** esetén egyes kimeneti bitek Mealy, $z_i(s,x)$, mások Moore jellegűek $z_j(s)$

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 következő állapot és a kimenet igazságtáblái alapján meghatározhatók az egyszerűsített logikai függvények**.
  Ezt nevezzük ***hagyományos tervezési módszer***nek.
  A logikai függvények egyszerűsítését nem részletezzük, csak az eredményeket adjuk meg.

  - A ***következő állapot*** egyszerűsített logikai függvényei SOP alakban:

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

  - A ***kimeneti logika*** egyszerűsített logikai függvénye:

    z = s1 s0 /x;

Az állapotgép leírása Verilog-ban

Az állapotkódolás és állapot regiszter Verilog leírása

```
reg [1:0] s;
//Állapotkódolás megadása
// a parameter-rel konstansokhoz nevek rendelehetők:
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 beíródik a kezdő állapot
    else       s <= next_s;   //  egyébként beíródik a következő
end                           // állapot kódja (amit a next_s
                              // logika állít elő)
```

Az állapotgép leírása Verilog-ban

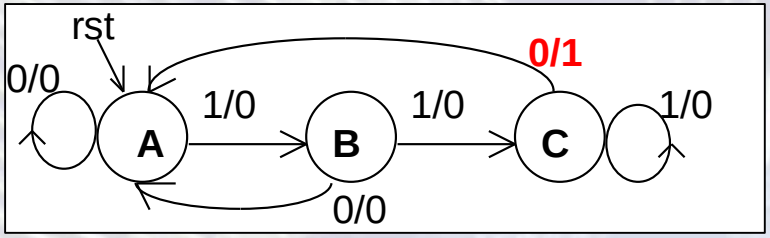
- A következő állapot logikát többféleképpen is megadhatjuk.
- Megadhatjuk a minimalizált függvények alapján ***Boole algebrai alakban assign-al***. (Hagyományos módszer.)

```
//next state logika:
wire [1:0] next_s; // mert most assign-al adjuk meg
    assign next_s[1] = s[0]&x;
    assign next_s[0] = s[0]&x | ~s[1]&x;
```

Az állapotgép leírása Verilog-ban

- Azonban a gyakorlatban az **állapotgráf vagy állapottábla alapján** *az* ***viselkedési leírással adjuk meg az*** *állapotátmeneteket és a minimalizálást a tervezőrendszerre bízzuk*. (Ezt ***általános FSM tervezési mód***nak nevezzük.)

```
//next state logika
reg [1:0] next_s;    // mert most always blokkban adunk neki értéket
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;          // esetleges nem használt állapotkód esetén
endcase                       // (tranziens hiba) az A-ba megy
```

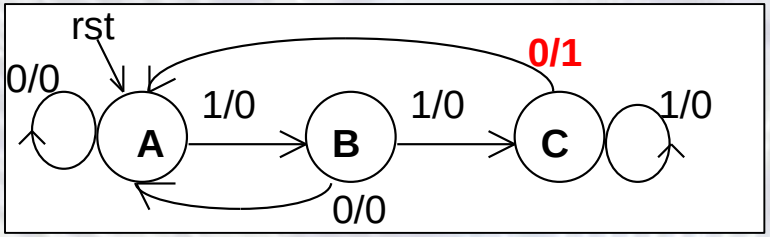


Az állapotgép leírása Verilog-ban

A kimeneti függvény megadása

- A kimeneti függvény megadására is több lehetőségünk van.
- Megadhatjuk a minimalizált függvényt SOP ***Boole algebrai alakban***:

  assign z = s[1]&s[0]&~x; //hagyományos tervezési mód
- Megadhatjuk az ***állapotgráf/állapottábla alapján az állapotkódok felhasználásával és az egyszerűsítést a tervező rendszerre bízzuk.***



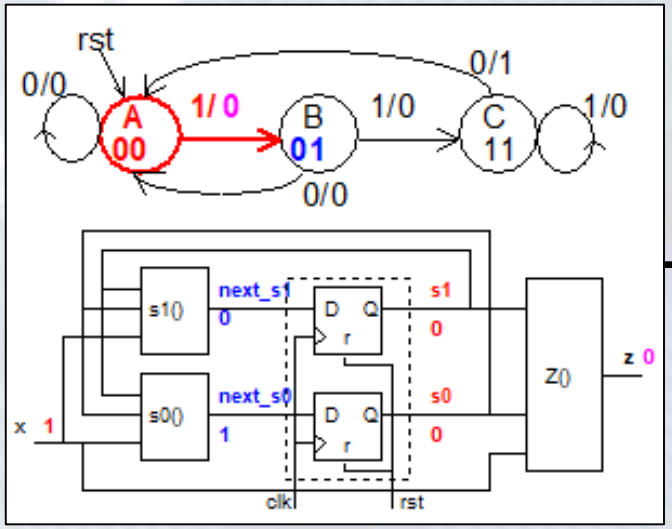
A z kimenet akkor 1, ha C állapotban van és X=0:

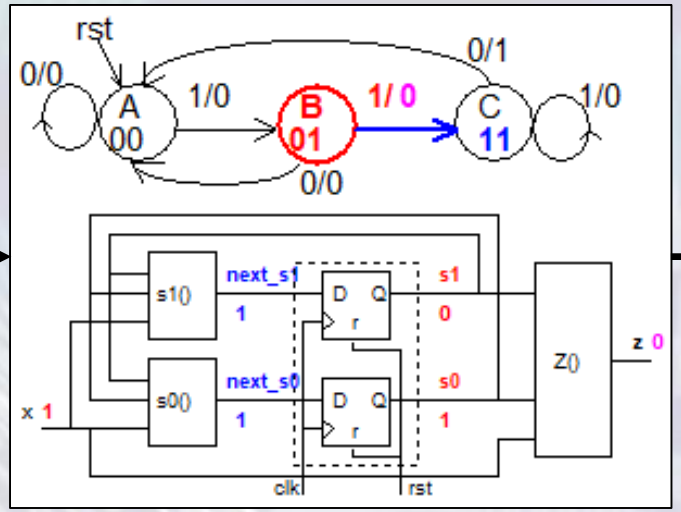
assign z = (s == C)&~x;

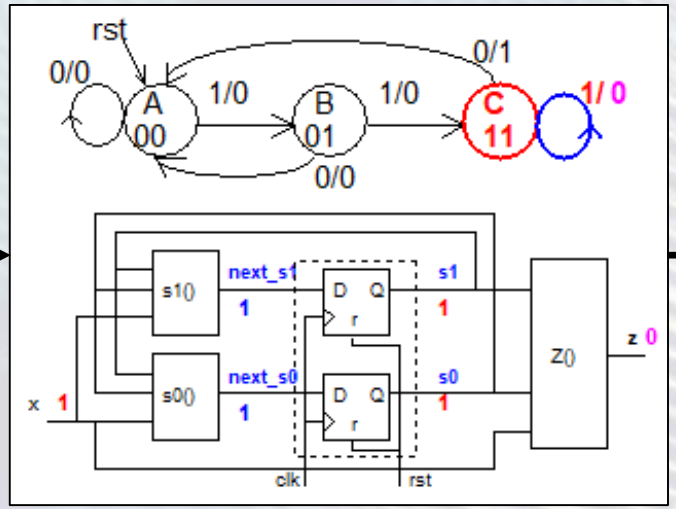
vagy

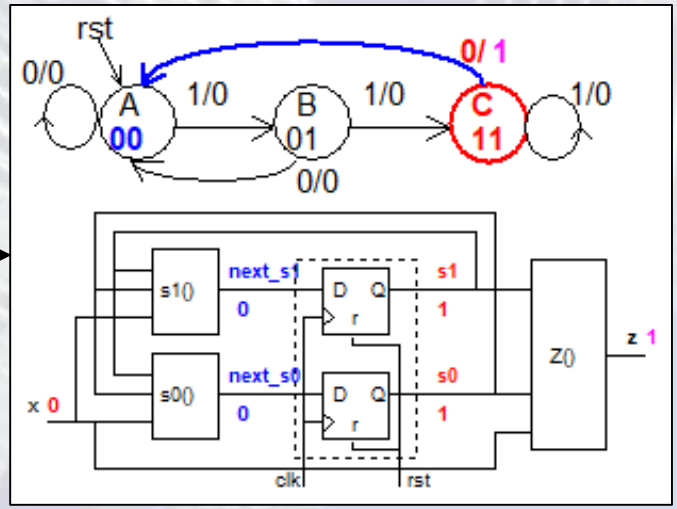
assign z = (s == C)&(x == 1'b0);

Működés reset után az X=1110 bemeneti sorozat hatására



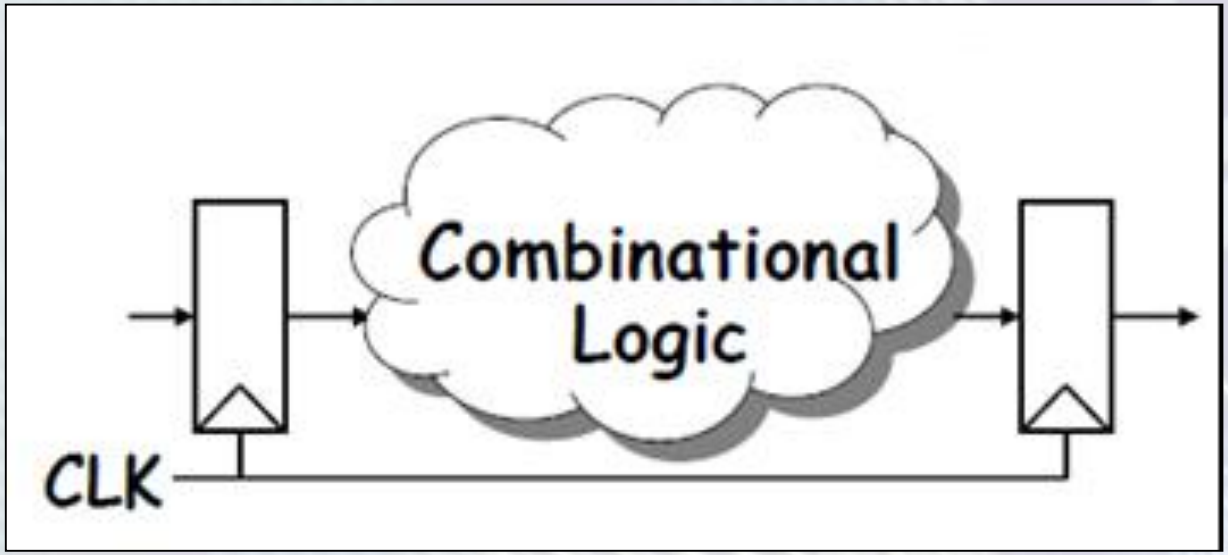






Szinkron sorrendi logikák működési feltételei

- **Adott számú DFF párhuzamos működtetése, közös, globális, időben és térben egyidejű CLK órajellel történik** (az összes DFF órajele nagyon pontosan azonos, az aktív él egyszerre vált).

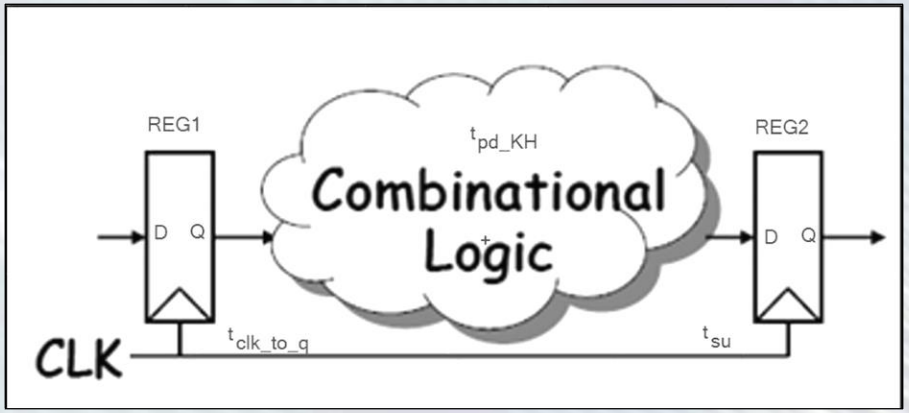


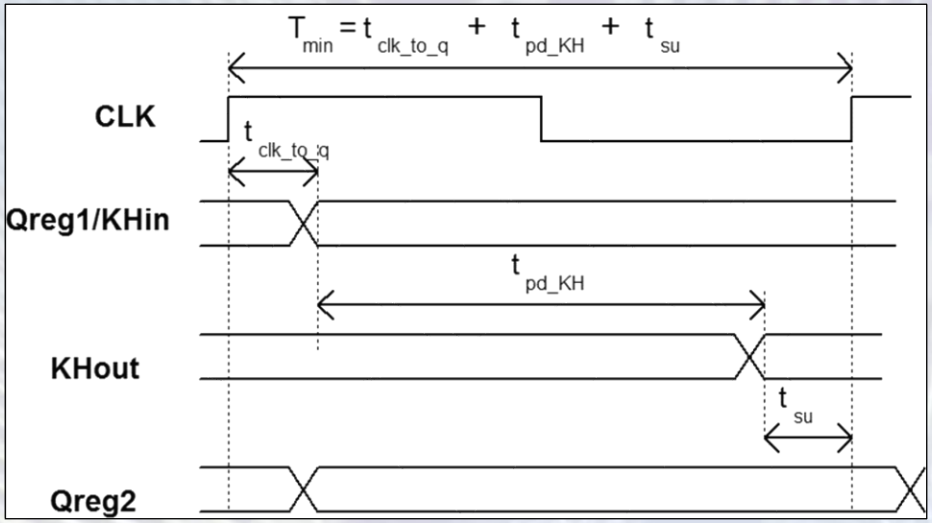
- **A műveleteket (számításokat) a kombinációs logika végzi.**

Szinkron sorrendi logikák működési feltételei

A szinkron digitális tervezői paradigma biztosítása

- **Két órajel között elegendő idő kell a feladatok elvégzésére. (A kombinációs hálózat műveletvégzésére és a DFF-ok beírására.)**





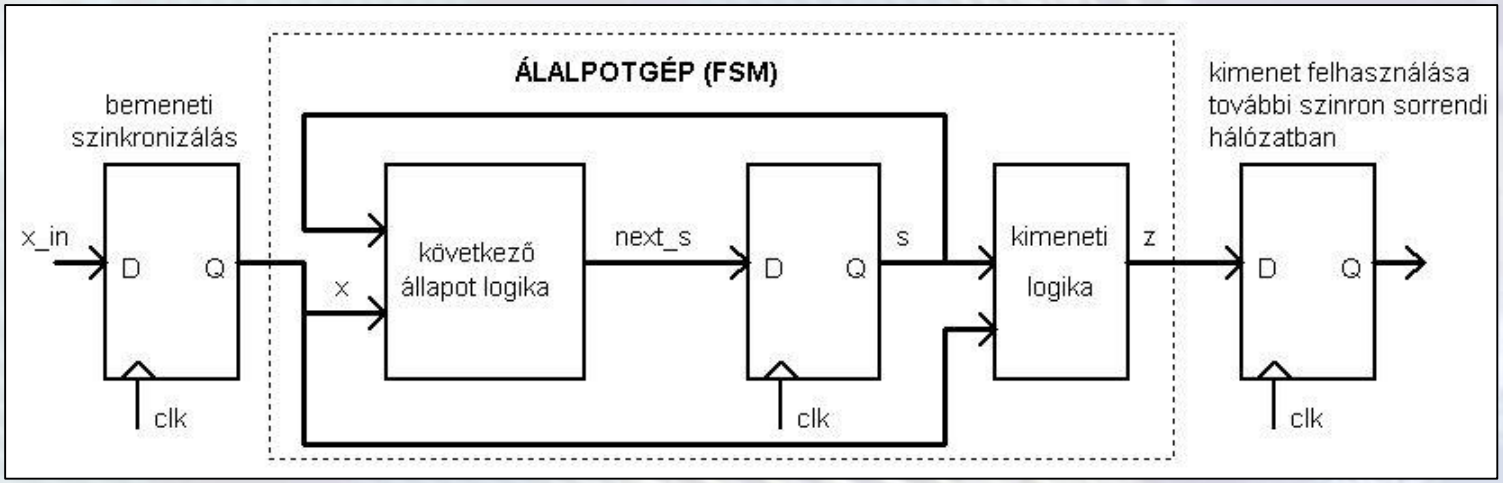
- $T_{clk_to_q}$ **:** az órajel akítv élétől a Q kimenet beállásáig eltelt idő
- T_{pd_KH} **:** a kombinációs hálózat jelterjedési ideje (bementi változás eljut a kimenetre)
- T_{su} **:** egy regiszter adat bementén a jelnek az órajel aktív éle előtt legalább ennyi idővel stabilnak kell lenni, különben nem garantált a helyes működése.

Szinkron sorrendi logikák működési feltételei

- **Az idődiagram magyarázata**
  - Az órajel aktív éle után kis késéssel ($t_{clk_to_q}$) megváltozik az 1. regiszter kimenete, vagyis a kombinációs hálózat bemenete.
  - A kombinációs hálózaton keresztül is véges idő alatt terjed a jel (t_{pd_KH}). A jelterjedési idő logikai áramkörök sebességétől és a kombinációs hálózat megvalósításától is függ. Itt a leghosszabb jelterejedési idő érdekes.
  - Ha megfelelően nagy az órajel periódusideje (T_{clk} >= Tmin), akkor a következő aktív éle előtt előírt idővel (t_{su}, adat előkészítési idő) a 2. regiszter bemenetén már stabil a kombinációs hálózat kimenetéről érkező adat és ez az él hatására hibátlanul beíródik a regiszterbe (ha még megfelelő ideg stabil marad). Teljesül, ha $T_{clk} >= T_{clk_to_q} + T_{pd_KH} + T_{su}$
    Ennél jobban nem részletezzük.
- **Két órajel él között stabil kimeneti értékek biztosítása.**
  - A regiszter biztosítja, ha az előbbi feltételek teljesülnek. Ha a bemenetén a beírandó adat az aktív órajel él előtt megfelelő idővel már stabil (és utána is megfelelő ideig még stabil). Egyébként hibás értéket tárolhat. (Fontos az időzítési előírások betartása.)

Szinkron sorrendi logikák működési feltételei

- A ***szinkron álapotgép*** (FSM) esetében ***a kombinációs hálózat kimenete ugyanazon regiszter bemenetére (is) kapcsolódik, mint amely a bemenetét meghajtja***:



- A bemenetét az órajelhez kell szinkronizálni, így tarthatók be az időzítési követelmények.

Digitális technika 4. EA vége