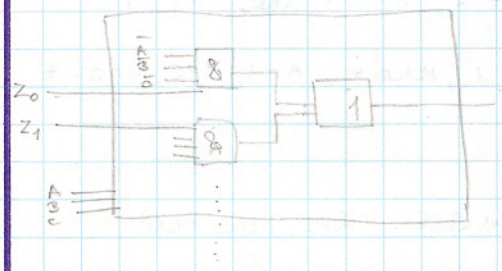
Multiplexer:



- Minden verzió benne van $(\bar{A}, \bar{B}, C ; \bar{A}B\bar{C} \ldots$ stb.$) \Rightarrow 8$ db & kapu kell felhasználva
- Bármilyen kombinációt kapcsolunk rá $\Rightarrow$ valamelyik & kapu igazat fog adni $\Rightarrow$ a VAGY kapu mindig 1-et fog adni
- Ezek egyben vannak betokozva, amelybe bevezetjük az A,B,C-t. Így a negyedik vezetéken múlik, hogy az & 0-t vagy 1-et ad-e. Ha a negyedik vezeték $0 \Rightarrow$ 0-t ad az & kapu, ha $1 \Rightarrow$ 1-et ad.
- És a logikai szint, amely a kiválóból belépő jelenik meg, a végén is meg fog jelenni

A = 0 B = 0 C = 0 Y = Z_0

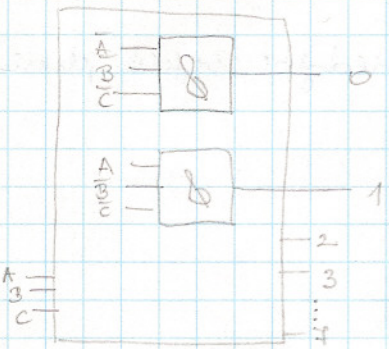
$Z_0 \rightarrow 0,0,0$ $Z_1 = 0,0,1$ $Z_2 = 0,1,0$ stb.

- Megmondja az A,B,C-n, hogy melyik Z-vel legyen egyenlő a kimenet.

A	B	C	sorok	
0	0	0	0	
1	0	0	1	
0	1	0	2	
1	1	0	3	megszámozzuk, A = 1, B = 2, C = 4
0	0	1	4	
1	0	1	5	
0	1	1	6	
1	1	1	7	

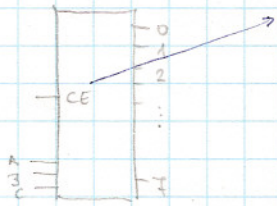
- A legfelső & kapu kimenete akkor van 1-es, ha $A, B, C = 0$. Minden más esetben 0 lesz a kapu kimenete
- A másodikra azt írjuk, hogy $A, \bar{B}, \bar{C}$. Akkor lesz az & kapu kimenete 1, ha mindhárom 1, tehát $A=1, B=0, C=0$-t kell rákapcsolni.
- A 3 vezetékkel kiválasztjuk a negyediket. Ha az első & kapu 1, a többi 0, így a végső kimenet eredménye a z_0-tól függ.
- és a vezérlést az **ADATSZELEKTOR** → A, B, C-vel kiválasztja, hogy melyik adat haladjon át rajta. Másik neve: **MULTIPLEXER.** A példában egy 8-ból 1-re multiplexer van. Ha 4 változót használunk: 16-ból 1-re multiplexer
  Ha 2 -"- : 4-ből 1-re multiplexer.

Demultiplexer:

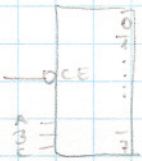


- demultiplexer 3-ból 8-ra. Ha 4 változót használunk: 4-ből 16-ra demultiplexer.
- itt nincs mit kiválogatni, de néha szükség a felhasználásnál

Gyakorlatban:



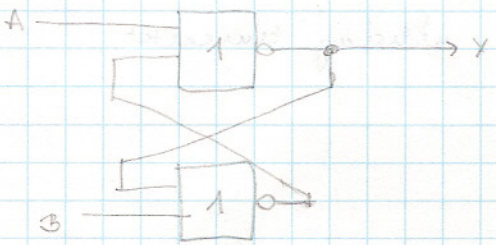
csip engedélyező láb:
ezek egy csipen vannak rajta, azt engedélyezi hogy működjön (1), vagy ne működjön (0).
Ha működjön, az A,B,C-ének az 8 kapu valamel 1-et ad.



ilyenkor fordítva működik. Mindegyik 1-et ad, csak az nem, amelyik A,B,C alapján ki lett választva. A csip engedélyező is fordítva működik.

- Ezek az ún. KOMBINÁCIÓS ÁRAMKÖRÖK. A kimenetek a bemenet értékei meghatározzák.

———— o ————



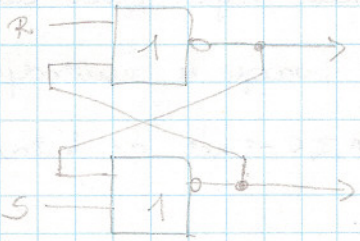
(felhasználva: 2db 2 bemenetű NOR kapu)

A	B	Y	
0	0	0	✓ (fentről)
0	1	1	✓ (kettő)
0	0	1	(lentről)
1	0	0	(fentről)

- Ez egy olyan rendszer, amely a „0,0"-nál el tudja képzelni a kimenetet 0-ként, és 1-ként is.
Megjegyzi azt az állapotot, hogy „1,0", vagy „0,1" után lett „0,0".
És a set és a reset állapotot jelenti.

set: 1 ; reset: 0.

- Ez egy RS (vagy SR) FLIP-FLOP.



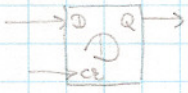
Mind a kettő ki van vezetve.

A	B	Y
1	1	–

Ez az állapot tiltott.

- A kimenet nem kizárólag a bemenet állapota definiálja, hanem a belső állapot is. Tehát a bemenetek és a bemenetek érvényre jutási sorrendje határozza meg, ezért őket úgy hívjuk, hogy SORRENDI ÁRAMKÖRÖK.
- Ez is kapuáramkör, mert két kaput használtunk.
- Sorrendi áramkör, mert néhány kimenetet visszavezettünk bemeneteikre.

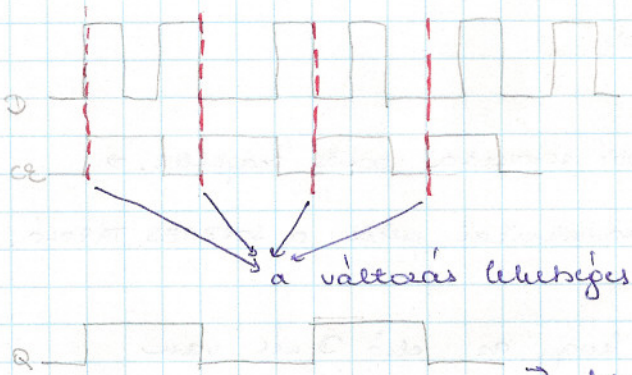
- D-FLOP:



- van egy D bemenete, Q kimenete, és egy órajel (Ck). A clock időbeli befolyását jelenti, azaz, hogy a jel kihatását változni fog. Ehhez a változáshoz kötődik dolgok.

A D-flop arról változik, ha az órajel alacsonyról magasra megy.

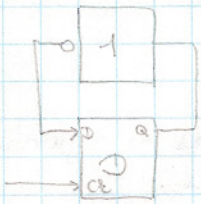
A Q addig 0, addig 1 volt, addig nem változik, amíg a Cl alacsonyról magasra nem vált. Ha ez teljesül, a Q felveszi a D értékét.



A D ezzel ide-oda változott.

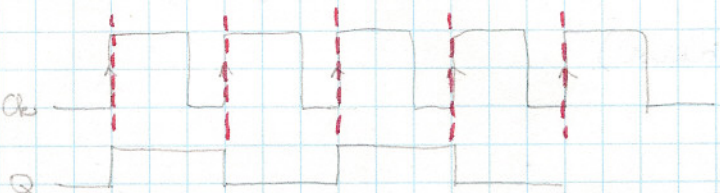
→ felveszi a D értékét. Ha az a változás pillanatában magas volt, a Q is az lesz, ha alacsony volt, az lesz.

-

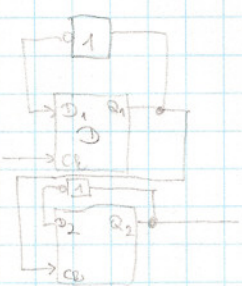


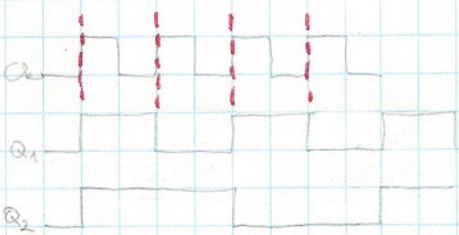
A Q kimenetet egy inverterrel visszatöltik a bemenetre.

A változás megint a Cl-től függően történik.



Ha Q magas (1) volt, a D alacsony (0) lesz, mert az inverter megfordította.



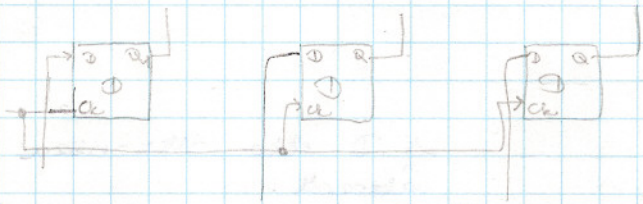


Ha az első frekvenciája F, akkor az alsó a fele, ha Q_3-t venném, a nyolcada lenne.

1 Hertz → a kvarcóra időalapja így telhet. (12... → kvarc belső)

Soreadi áramkörök felhasználása a gyakorlatban:

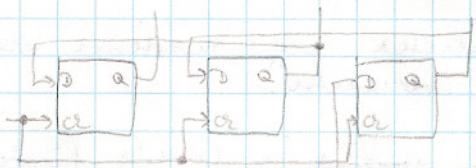
(*)



Olyan rendszert, amely D-flopsz sorozata közös órajellel. ⇒ ez egy tároló. Konkrét számítástechnikai eszköz: a SOROSÍTÓ TÁROLÓ.

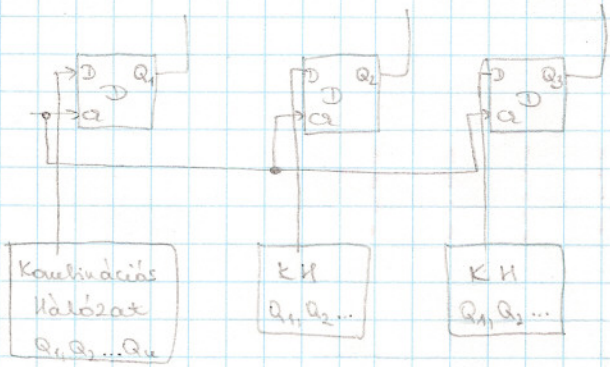
- Ha így tároljuk az adatot, hogy az első D-nél van bemenet, és a rákövetkezőnél a Q-t összekötjük a követő első D-jével ⇒ minden adat eggyel jobbra lép ⇒ 1-gyel magasabb helyiértékre lép ⇒ az ilyen a LÉPTETŐ REGISZTER.

(a 1-gyel jobbra lépés ⇒ a 2-vel való szorzás művelete)



2-vel való osztás ⇒ a másik irányba léptetés.

Kérdés: Lehet-e olyan szerkezetet csinálni, amely egyszerű műveletet végez? (pl. számol)



A Q az órajel váltásaitól függ.

A első KH-nál „0,0,0...0"-nál legyen D=0, a másodiknál 000...1-nél legyen más. De így változott a bemenet,

így a KH-ot esetben a bemeneten is. Változik a címeket, így megint más a bemenet. Ilyen KH-váltással elvesd számolni a szorzást. Ha kell, tud visszafelé is számolni.

Ennél a szorzásnál minden tag 1 időpillanatban vált ⇒ SZINKRONVÁLTÁS, míg az előzőnél ASSZINKRON (≠) váltás zajlik.

- A KH-val adnak plusz bemeneteket. Csak arra számol előre, ha pl.: a szám szorzás 000. Ha 001, visszafelé számol ... stb. Így végre tudja hajtani a lépéseket ⇒ számol. Ilyen szorzásra mondjuk, hogy a számítógép BELSŐ REGISZTEREI. (pl.: 32 bites regiszter)
- A D-flopot is fel lehet kapukból építeni.
- Szorzó már lehet (lépték)

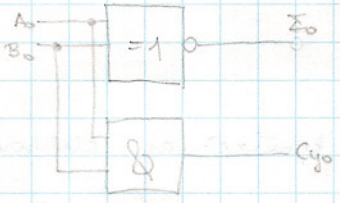
Összeadás:

A	0	0	1	0	1	1
B	+ 1	0	0	0	1	1
Σ				←	←	

átvitel a 0-s helyre

A_0	B_0	Σ_0	Cy_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

↑ XOR ↑ AND



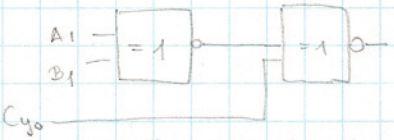
Cy_0	A_1	B_1	Σ_1	Cy_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A_1 XOR B_1

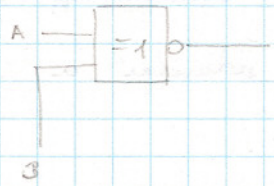
A_1 XOR B_1

↳ dontjük a sz. alapjával, lesznek a maradékot, marad a hányados.

A_1 XOR B_1



XOR



- ha B fix 0 ⇒ $Y = A$
- ha B fix 1 ⇒ $Y = \bar{A}$

- ha $C_{y_0} = 0$ ⇒ a elvégzett művelet XOR, és ez akkor is, ha $C_{y_0} = 1$ ⇒ elvégzi az A_1 XOR B_1, csak ennek a negáltja fog kijönni.

- Ez a szerepe a FÉLÖSSZEADÓ. (A előző fokozat nincs benne, így olyan „fél munkát" végez.)
- A szavazógépet A_1, B_1, C_{y_0} - bemenettel, és a félösszeadót összekapcsoljuk TELJES ÖSSZEADÓT kapunk. Logikai műveleteket végzünk, aritmetikai eredményt kapunk.