1. Mi a Power-On-Reset és a Brown-Out-Reset (POR, BOR) funkciója? Mi a különbég közöttük?

Power-On-Reset.:

Ez egy olyan cella, amelyik figyeli a tápfeszültség felfutását, és annak hatására reset impulzust generál. Ilyen cella beépitésével a toknak egy lábát fel lehet szabaditani, és az alapállapot beállitását automatikusan meg lehet oldani. Ezzel azonban az a hátrány is jár, hogy a resetet csak ki-be-kapcsolással lehet végrehajtani.

Brown-Out-Reset:

Ez az áramkör Reset-jelet biztosít a mikrovezérlőnek, ha annak tápfeszültsége egy meghatározott U_{BOR} feszültségszint alá esik legalább ~100µs időtartamra. A BOR funkció kiküszöböli a hibás működést olyan alkalmazásoknál, ahol (például nagy áramfelvételű fogyasztók bekapcsolásakor) rövid idejű tápfeszültségesések fordulnak elő.
Ha mégjobban csökken a V_{DD} , akkor RESET állapotba kényszeríti a mikrovezérlőt. Innentől pedig már a POR veszi át a feladatot.
–Megakadályozza, hogy a Flash-be írni lehessen

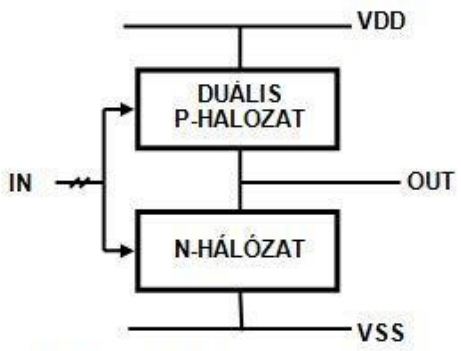
–Megakadályozza, hogy bizonytalan működést mutassanak az alulfeszültség miatt a perifériák és valami nem kívánt eseményt okozzanak

2. Milyen megoldások védik a CAN busz adatbiztonságát min. 8 db! (Fizikai és adat kapcsolat szinten)

1) Csavart érpár
2) Szimmetrikus jelvezetés
3) Busz monitorozása, az eszköz figyeli, hogy az van-e a buszon amitt kitett, ha nem: error.
4) CRC hibavédő kód
5) Bit stuffing: 5 ugyanolyan bit után egy ellentétes beszúrása
6) Konzisztencia a buszon: vagy mindenki veszi az üzenetet vagy senki sem
7) Vételi és adás hiba számlálók
8) Szinkronizációs szegmens

3. CMOS logikai alapkapuk felépítése. Multiplexer transzfer kapuval.

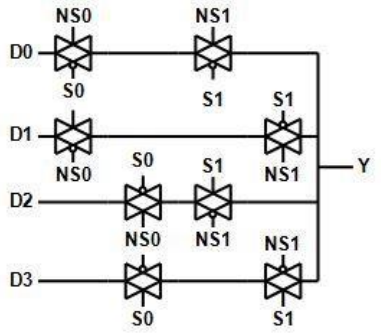
Az kapu 2 hálózatból áll, amik egymás duálisai: P-hálózat (pull up network), és N-hálózat (pull down network).



4-9. ábra: CMOS kapu általános felépítése

Az n-csatornás aktiváláskor vezet és lehúzza a kimenetet földre, a p-csatornás ezzel ellentétesen akkor húzza fel a kiemenetet tápfezsültségre, ha a másik nem vezet.
Előnyök/hátrányok:
-Egy K bemenetű kapuhoz 2*K tranzisztor kell
-Nincs statikus fogyasztása
-LOW: GND, HIGH: VDD

Multiplexer transzfer kapuval:



Négybemenetü multiplexer.
Négy bejövö jelböl egyet kiválasztunk és kiadjuk a kimenetre. Ehhez két szelektáló jelre, S0 és S1 van szükség, és természetesen ezek inverzei, NS0 és NS1 is szükségesek, de ezeket is meglévö bemenö jeleknek tekintjük.
A négy adatjel legyen D0, D1, D2 és D3.

4. A tervező feladatai az ötlettől a sorozatgyártásig, szerepe a megrendelő és a gyártó között.

Ism. az adott IC tech. lehetőségeit, Logikai tervezés, első megrendelő általi ellenőrzés, layout, verifikáció, m.m.á.ell., wafer-teszt, gyártás előtti teszt, gyártás (másik kidolgozás szerint)

1) A megrendelő közreműködésével lefekteti a specifikációt.
2) Logikai tervezés, verifikációs szimuláció bemutatása.
3) Fizikai tervezés (layout), verifikálás, tervezési szabály ellenőrzés, postlayout szimuláció bemutatása.
4) Prototípusok tesztelése, ezeket is megmutatják a megrendelőnek.

A tervező a megrendelő és a gyártó között áll mindkettővel kapcsolatot tart, ismeri az adott technológia lehetőségeit.
Tanácsokkal látja el a megrendelőt, a gyártótól kapja a tervezési szabályokat, cellakönyvtárakat, tranzisztor paramétereket.

5. Hibamodellek digitális áramkörökben. Hogyan kell egy feltételezett hibát detektálni. Mi a hibaszimulátor, hogyan működik, mire használjuk.

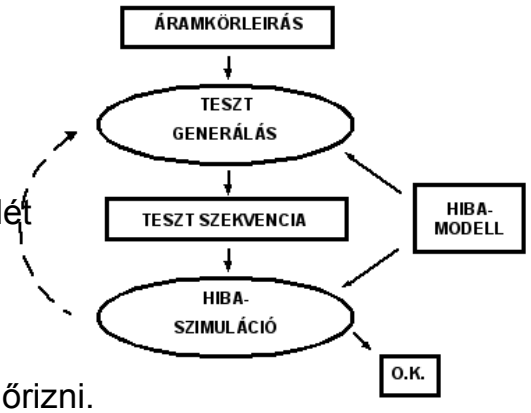
A strukturális teszt során használjuk őket.

Kiakadás (stuck-at):
Egy csomópont logikai szintje kiakad 1-1 fix értékre és nem változik. pl.: zárlat tápra vagy földre

Jelvezeték zárlat: (bridging):
Két jelvezeték között megszűnik a szigetelés, rajtuk csak azonos jelszint alakulhat ki.
Ha azonos logikai jel van a kettőn nem okoz gondot, de ha ellentétes akkor a létrejövő szint az áramköri viszonyoktól függ.

Kiakadt tranzisztor (stuck open):
A tr. nem képes a drain-jére csatlakozó vezetéket meghajtani. Az áramkört szekvenciálissá teszi.

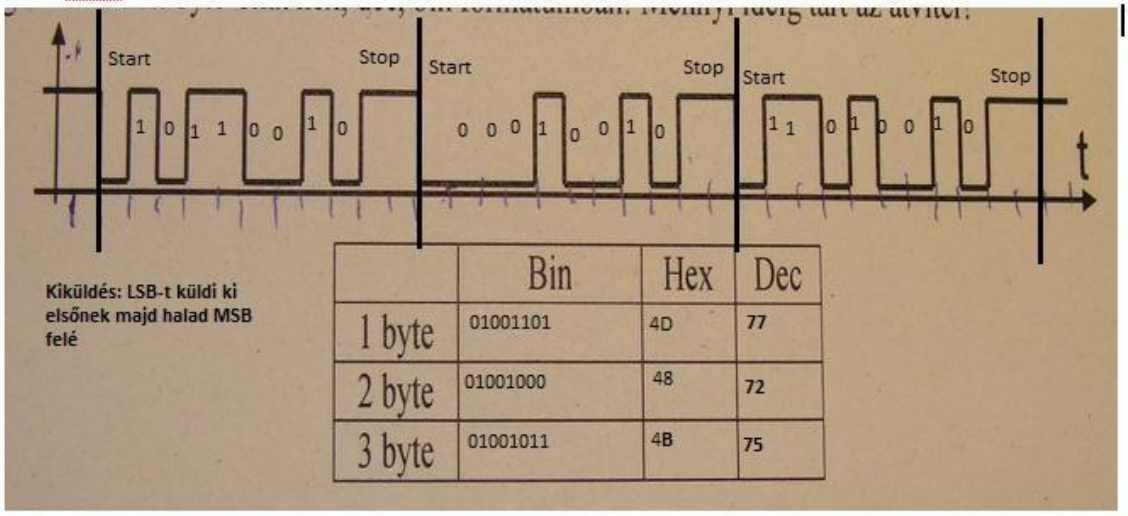
Gyakorlatban csak a stuck-at hibákra tesztelnek.
Módszer:
1) *Érzékenyítés*: megfelelő bemeneti kombináció segítségével a hibától függő csomóponti érétket hozunk létre.
2) *Megfigyelés:* el kell intézni, hogy az érzékenyített csomópont jelét egy kimeneten meg tudjuk figyelni, ehhez egy érzékenyített utat hozunk létre egy kimenetig.



2-17. Ábra: Hibaszimuláció folyamatábrája

Hibaszimulátor:
Egy tesztszekvencia hibalefedését a hibaszimulátorral tudjuk ellenőrizni.
Hibalefedés = detektálható hibák / összes lehetséges hiba.
Működés: konkurens hibaszimuláció, összeállítja a lehetséges hibák listáját, szimuláció közben figyeli érzékenyítve lett-e, ha igen és eljut a kimenetig is átteszi a detektált hibák listájába.

6. A soros porton 9600 8N2 küldés történik és a következő TTL szintű időDiagramot látja! Adja meg az átküldött byte-okat hex, dec, bin formátumban! Mennyi ideig tart az átvitel?



Kiküldés: LSB-t küldi ki elsőnek majd halad MSB felé

	Bin	Hex	Dec
1 byte	01001101	4D	77
2 byte	01001000	48	72
3 byte	01001011	4B	75

9600 bit/s

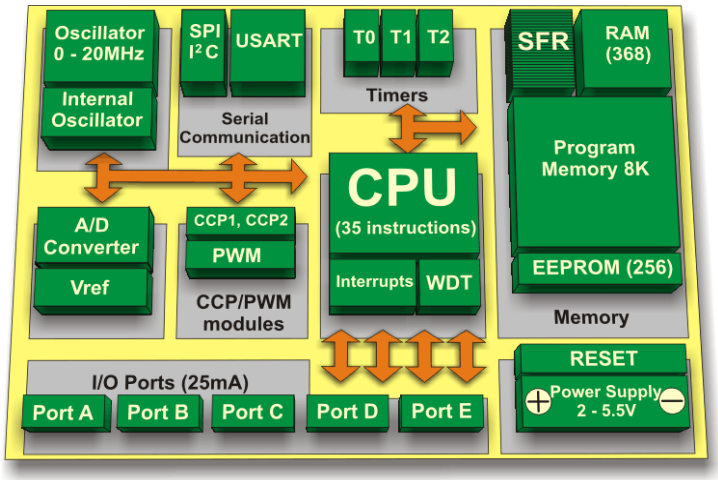
8N2:1 start, 8 adat, parítás nincs, 2 stop

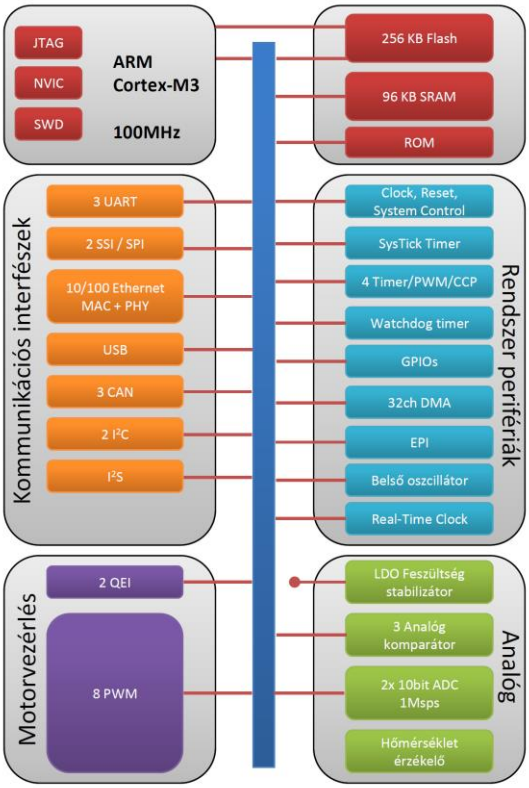
11 bit/packet.
3 packet

3*11 = 33 bit összesen.
t = 33bit / (9600 bit/s)

7. Sorolja fel a mikrokontrollerek alapvető építőelemeit (blokkvázlat).

- **Mikrovezérlő felépítése:**
  - Processzor mag
  - Memória (Flash, SRAM, (EEP)ROM)
  - Rendszer perifériák
  - Analóg perifériák
  - Speciális funkció (pl.: Motorvezérlés)
  - Kommunikációs interfészek

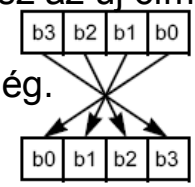




A nagyobb dobozok nevei a fontosak, és középen az őket összekötő busz szerintem.

8. Bit-reversed címzés, előnyei, mire használjuk?

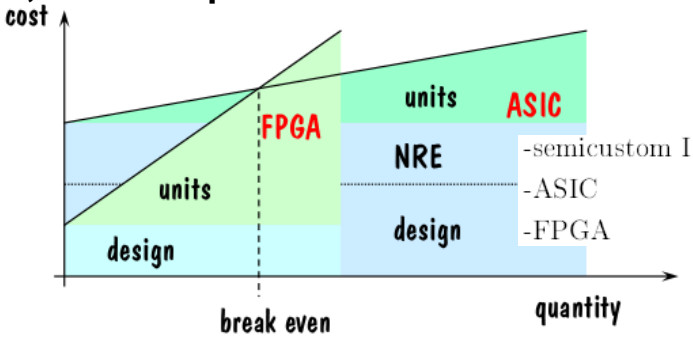
Bit Reverse, ami annyit jelent, hogy egy adott címnek a bitjeinek a tűkörképe lesz az új cím.
Bit Reverse címzés, megfordítja a bitek sorrendjét.



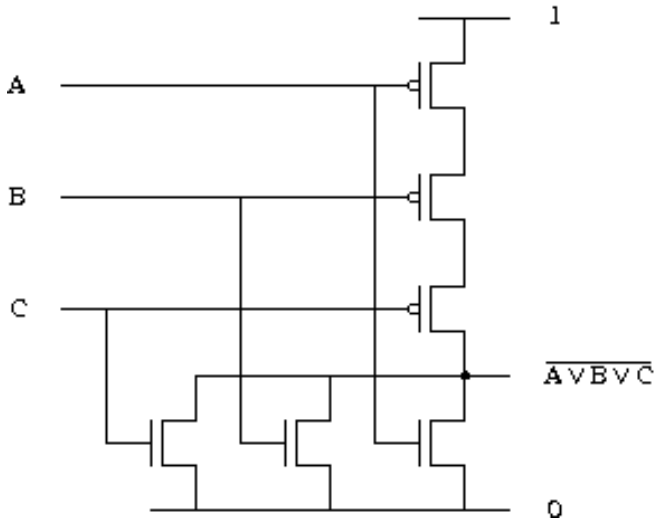
Nagyon hasznos például az FFT algoritmus megírásánál, ahol számít a sebesség.
A címzés egy adott hosszúságú memóriazónát érint. A túlcsordulást pedig jobbra viszi át, nem balra,és ezáltal a fordított címet kapjuk.

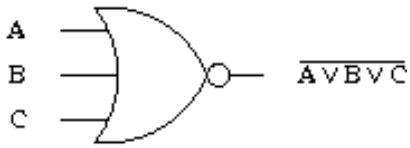
9. Hasonlítsa össze a különböző HW reprezentációkat a piacra kerülés/tervezési idő, a költségek, komplexitás, performance, fogyasztás, stb szempontokból.

	PCB	SiP	SoC (+FPGA)	ASIC	
piacra kerülési idő	→				nő
fogyasztás	←				nő
performance	→				nő
méret	←				nő
felxibility	→				nő



10. Rajzolja fel egy három bemenetű CMOS NOR kapu kapcsolási rajzát. Miért nem szokás több mint négy bemenetű CMOS kapukat alkalmazni?





4-nél több bemenet:

Bulkhatás miatt elfogy a nyitófeszültség, ha sok tranzisztor van sorba kötve és ezért nem lehet.

11. Mire jó az aszinkron reset digitális áramkörökben?

Egy szekvenciális hálózat elektromos állapota bekapcsolás után nem ismert. Ennek az állapotnak a megszüntetésére használják az aszinkron resetet, amely a hálózatot egy jól definiált kiindulási helyzetbe hozza.

Kivételt képeznek a közvetlen hozzáférésű tárolók, ezeket üzemszerűen fel lehet tölteni (memórán nincs reset).

FPGA-knál szinkron resetet használnak, mivel nem minden flip-flopjának van aszinkron reset bemenete.

12. Hasonlítsa össze a két tanult processzor-architektúra típust (blokkséma, előnyök, hátrányok)!

CISC

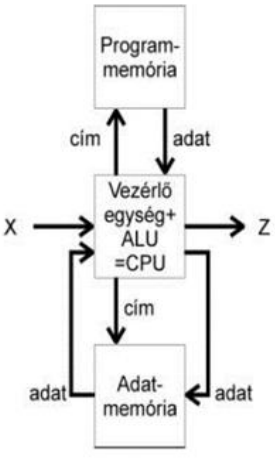
-A hardver megtervezése nehéz
-Több órajel alatti komplex utasítások
-Memory to Memory LOAD and STORE utasítások egyesültek
-Kicsi kód méretek nagy ciklusok/másodperc
-Sok tranzisztor komplex utasítások tárolására
-MULT utasítások
-Kevés RAM igény

RISC

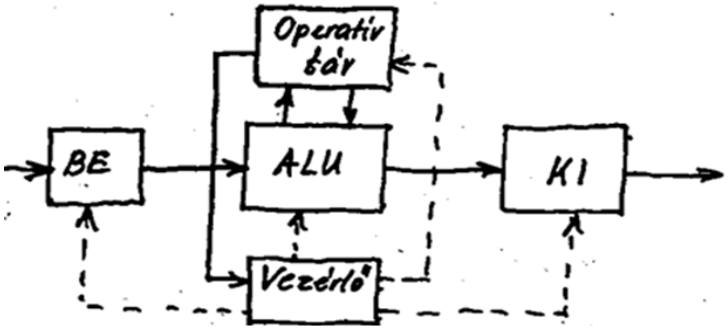
-A programozása nehéz
-Egy órajel alatti egyszerű utasítások csak
-Register to Register LOAD and STORE független
-Kicsi ciklus/másodperc nagy kód méretek
-Sok tranzisztor a memory regiszterekre
-Nagy RAM igény

Harvard architektúra:

- A program- és az adatmemória külön helyezkedik el
- Tulajdonságok, következmények, előnyök:
  - Program memória flash, adatmemória RAM
  - Egy órajel alatt elvégezhető az utasítás és az operandusok elővétele (fetch)
  - A programmemória zavarvédettebb
  - Az adat és a programmemória lehet különböző szélességű



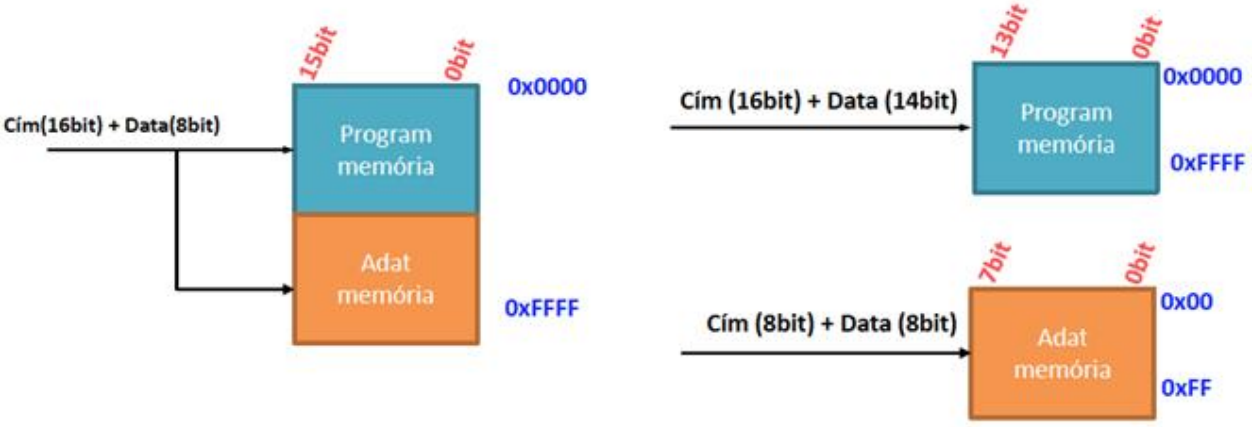
Neumann architektúra



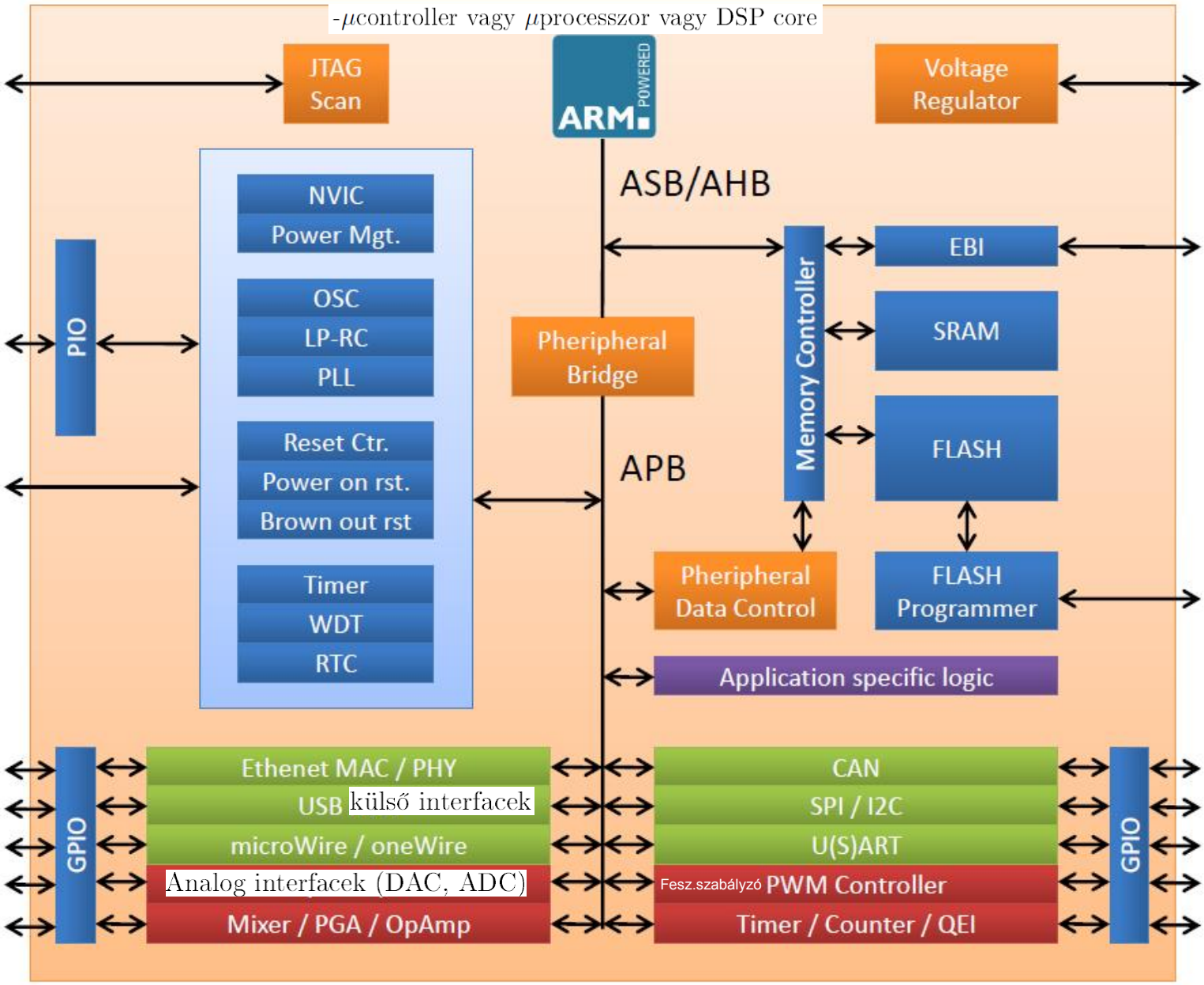
belső programtárolás, programvezérlés
utasítás és adat azonos közegen és formában
(értelmezés algoritmus ill. PC szerint)
szekvenciális ut. végrehajtás
egydim. lin. címzésű mem.
bináris számábrázolás
ÁBRÁN: szaggatott: vezérlés, folytonos: adat

Processzor architektúrák

- Neumann architektúra
  - Azonos címtartományon van a program és adat memória
- **Harvard architektúra**
  - Eltérő címtartományon van a program és az adat memória



13. Sorolja fel a SoC különböző alap építőelemeit (blokkvázlat).



14. SiP megvalósítási lehetőségek (legalább 3 féle) tulajdonságai piacra kerülés/tervezési idő, a költségek, komplexitás, performance, fogyasztás, stb szempontokból.

Megvalósítási lehetőségek: CHIP stacks, MCM , Belső huzalozás, µbumpolás FLIPCHIP. Tartalmazhat ANALOG DIGITAL RF chippeket.

-Kis méret, nagyobb teljesítmény (rövid jelutak)
-Olcsóbb teljes költség mint több IC
-Gyorsabb piacra kerülési idő
Példa felhasználásra: GPS modulok,Bluetooth modulok,Cellular RF,WIreless, Teljesítmény erősítők.
-Nagyon sok részegységet tartalmazhat! DRAM PROCESSZOR FLASH stb.
akár több SOC is lehet SiP-ben.

	2D SOC	3D Sip
Cost	−	+
Performance	+	−
Power	+	−
Function	+	−
Time to market	−	+

15. A procedurális értékadás Verilogban. Flipflopok és latchek leírása (kódolása) és a szintézis eredménye.

- Az értékadás a programozásban az a művelet, amikor egy **változó értékét megváltoztatjuk**.
- Verilogban a fizikai valóság pontosabb modellezése érdekében **háromféle értékadás létezik** – nem mindegy például, hogy vezetéknek vagy regiszternek adunk értéket:
  1. **folyamatos értékadás**: vezetéknek,
  2. **procedurális értékadás**: regiszternek
     (a) **blokkoló procedurális értékadás**,
     (b) **nem-blokkoló procedurális értékadás**.
- Procedurális értékadás **kizárólag eseményvezérelten történhet**.
- Ez azt jelenti, hogy egy **olyan blokkban kell elhelyezni, amely események hatására aktivizálódik**.
- Verilogban **két ilyen blokk létezik:**
  1. **initial**: a szimuláció elején, egyszer fut le, és többször nem,
  2. **always**: a fejében megadott feltétel teljesülésekor mindig lefut.
- Procedurális értékadás tehát **csak initial vagy érzékenyített always blokkban szerepelhet**.
- Két fajtája van:
  1. **blokkoló értékadás**: az egymás alatt lévő értékadások a megadás sorrendjében, egymás után hajtódnak végre,
  2. **nem-blokkoló értékadás**: az egymás alatt értékadások párhuzamosan, egyidőben hajtódnak végre.

Blokkoló: egy blokban mindig előbb hajtódik végre mint az őt követő utasítások. <regiszter> = <kifejezés>
Nem blokkoló :<regiszter> <= <kifejezés>

Folyamatos értékadás

assign <változnév> = <kifejezés>

- Az értékadás **jobb oldalán lévő értékek bármelyike is megváltozik, azonnal frissül a jobb oldal** is a kifejezésnek megfelelően.
- Ez a **logikai kapuk** működését modellezi.

```
wire a, b, c;

assign a = b & c;
```

Kombinációs logikát lehet vele modellezni egy logikai fgv-el. Bármikor ha a jobb oldal változik, változik a bal oldal is. Always blokkon kívül használható. Wire típushoz használható.

```
reg [7:0] t, r;
always @ (posedge clk)
begin
   t = a & b;
   r = t | c;
end
```

```
reg t, r;
always @ (posedge clk)
begin
   t <= a & b;
   r <= t | c;
end
```

Latch: egy szintérzékeny tároló elem, mely a Gate bemenetének magas állapota alatt mintavételezi, illetve a kimenetén megjeleníti az adatbemenetén található értéket (transzparensen viselkedik), míg Gate jelének alacsony állapotában a mintavételezett értéket tartja.

```
reg q;
wire d,en;
always @(en or d)
  if (en) q <= d;        // latch
```

```
always @ (posedge clk)
if ( reset) begin
q<=1'b0;
end else if en begin
q<=data;
end
```

Tipikus hibák, melyek latch alkalmazásához vezetnek, miközben a cél kombinációs logika leírása:

- nem teljesen kifejtett case szerkezet
- if…else használata esetén hiányzik a feltétel nélküli else ág

D flip-flop: A D tároló egy élérzékeny tároló típus, amely az órajel bemenetének felfutó élére mintavételezi az adat bemenetét, s azt a következő órajel felfutó élig a kimeneten tartja. Ebből adódóan leírásakor élérzékeny *always* blokk használandó.

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

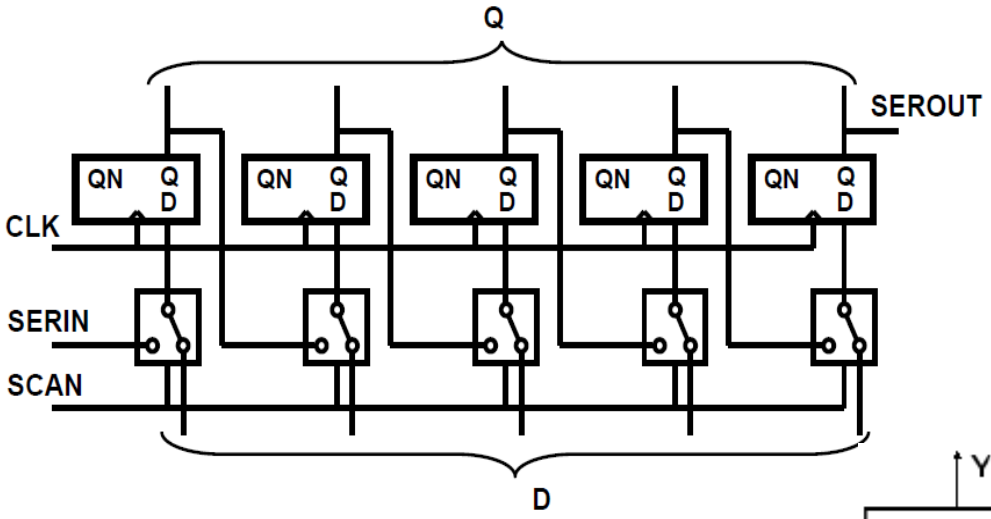
```
always @ (posedge clk)
if ( reset) begin
q<=1'b0;
end else begin
q<=data;
end
```

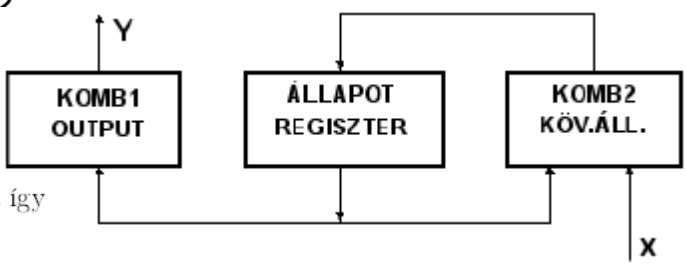
16. Digitális szimuláció jeleinek érték-készlete, hogyan értelmezzük őket?

Adott bemeneti kombinációra adott kimenet szimulációjának jelei a megadott bemeneti változók (clk, rst,set, x..) valamint a kimeneti változók. Az idő függvényében értelmezzük őket. (???)

17. Mire való a Scan-Path? Hogy néz ki az áramköri megoldás?

A belső állapotokat tartalmazó regiszter be- és kimeneteit teszi elérhetővé úgy, hogy a regiszter flip-flopjait átkonfigurálja léptető regiszterré.





A szekvenciális hálozat átkonfigurálására való kombinációs hálozattá így lehetővé teszi automatikus tesztgeneráló módszerek alkalmazását.

18. Tesztelhetőségre tervezés (DFT)

Az integrált áramkörök tervezezésésben a növekvő bonyolultság miatt már a tervezés során oda kell figyelni a leendő termék tesztelési problémáira. Könnyen tesztelhetővé kell tervezni, az erre vonatkozó módszereket és alapelveket foglalja össze a DFT(Design for Testability). A gyakorlatban a könnyebb tesztehetőséget általában meg kell fizetni (test overhead):

1. Többlet Si felület: Normál funkciót megvalósító elemeken kívül járulékos elemeket iktatunk az áramkörbe(ezek gyakran multiplexerek, amelyekkel a jelek útját a teszt céljaira módosítjuk). Ezek növelik a chip méretét.
2. Többlet pinek: Különböző teszt utak megfigyelésére járulékos kivezetéseket iktatunk be. Ez rossz, ha így is kb. 100%-os volt a lábak kihasználtsága. Nagyobb tok kell, ami drágítja az IC-t és növeli az alkatrész méretét.
3. A működési sebesség korlátozása: Növeli a futási időt.

A DFT módszerei:

1. Szisztematikus(Scan-Path, peremfigyelés)
2. Ad-hoc(teszt pinek alkalmazása, teszt üzemmód, belső részegységek elérése multiplexerrel)

20. Mi a PWM, alapfogalmak, mire használjuk, előállítási módjai (legalább kétféle)

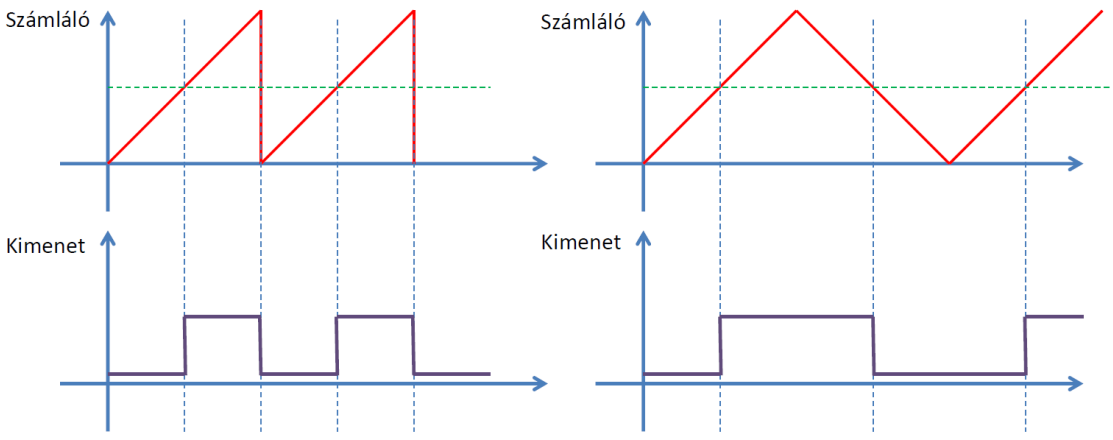
Az impulzusszélesség moduláció (*pulse-width modulation*) az inerciális elektromos eszközök szabályozására elterjedten alkalmazott technika, melyet a korszerű teljesítményelektronika tett a gyakorlatban is használhatóvá.

Az fogyasztóba táplált átlagos elektromos feszültséget és áramerősséget a táp és a fogyasztó között lévő kapcsoló gyors ütemű be- és kikapcsolásával szabályozzák. Minél hosszabb ideig van a kapcsoló a bekapcsolt állapotában a kikapcsolt állapot időtartamához képest, annál nagyobb lesz a fogyasztóba táplált teljesítmény.

A PWM kapcsolási frekvenciájának sokkal magasabbnak kell lennie, mint ami hatással lenne a fogyasztóra. A kapcsolási frekvenciának jellemzően 120 Hz-nek kell lennie egy lámpa fényerőszabályozójában, néhány kHz-től néhány száz kHz-ig terjedőnek egy motorvezérlőben, és jó néhány száz kHz-nek egy hangerősítő és egy számítógépes tápegység esetében.

- Periódusidő
- Komparálási szint
- Egyszerű DA
- Motor és teljesítményvezérlés
- LED vezérlése

- Regiszterek:
  - Value
  - Period
  - Compare
  - Control



21. Verilog: modul felépítés, tesztbench

Modulok definiálása

```
module <név>(<portlista>);
...
endmodule
```

```
module my_counter(clock, reset, q);
  input clock;
  input reset;
  output [7:0] q;
  reg [7:0] q;
  ...
endmodule;
```

- **A modulokat a `module` kulcsszóval vezetjük be, egyedi nevet adunk nekik és felsoroljuk a portjaik nevét.**
- Figyeljük meg, hogy a modulfejet megadó sort pontosvessző zárja!
- A modulok definíciójának a végét az `endmodule` kulcsszó jelzi.
- A modulok elején **fel kell sorolni a portjaikat** az `input` illetve `output` kulcsszavak segítségével.
- A portok **alapértelmezett típusa a vezeték**. Ha egy **kimenetet regiszterként szeretnénk megadni, azt külön sorban, a regiszterek megadásának szintaktikájával** kell megtenni.

Tesztbench:

- A teszkörnyezet is **egy modul, amit az különböztet meg a többitől, hogy nincsenek portjai**.
- Az ilyen modult **a szimulátorok fel szokták ismerni legmagasabb hierchia-szintű modulként**, kézzel csak olyankor kell megadni, ha több ilyen is van a rendszerben.
- A felépítése általában a következő:
  1. A tesztelendő modulok példányosítása és összekötése.
  2. A tesztszekvenciák leírása egy `initial` blokkban.
  3. A periodikus jelek előállítása `always` blokkokban.

```
module testbench;
  reg input_1, input_2 ...
  reg clk;
  //modul példányok
  initial
    begin
    clk = 0;
    //tesztjelek
    $finish;
  end
  always
  begin
    #1 clk = ~clk;
  end
endmodule
```

22. Mit ért Pad-limited valamint Core-limited design alatt?

- **Kifejtett áramkörleírás**
- **Floorplan**

core kialakítása

tappancsgyűrű kialakítása (pad limited, core limited)

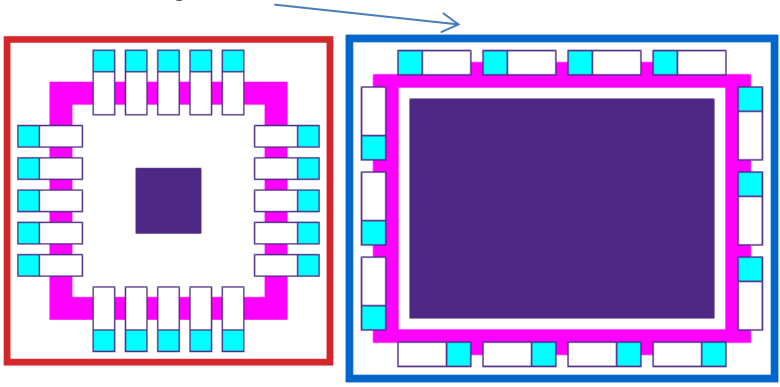
cellák elhelyezése

- **Globális huzalozás**

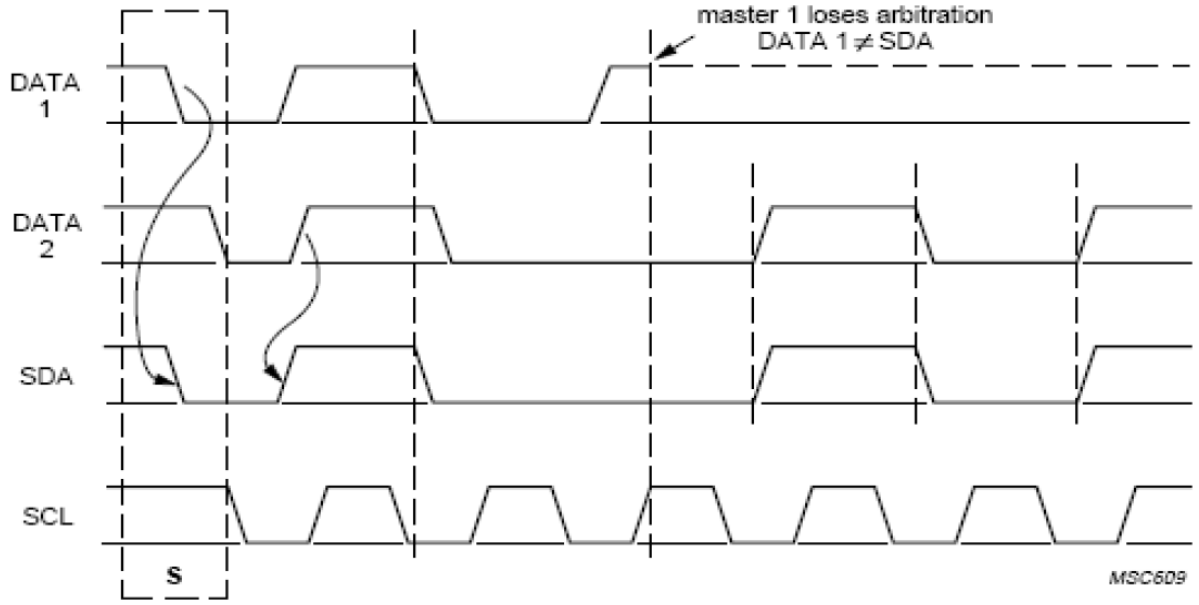
huzalozási csatornák kialakítása

föld és táp ellátás (*supply tree*)

- **Részletes huzalozás**
- **DRC**



24. Rajzolja fel az I^2C protokollokban használt arbitrációs folyamat idő diagramját.



25. Rajzoljon fel egy tipikus RS485 half-duplex busz architektúrát, a szükséges kiegészítő komponenseket is jelölje! Sorolja fel a busz néhány jellemzőjét (min. 3)!

- 52Mbit/s
- 120Ω lezárás
- Full / Half duplex üzemmód
- < 1200m
- Szimmetrikus jelvezetés
- Galvanikus leválasztás
- Fail Protected, Fail Safe, ESD protected

Input impedancia $\geq 12k\Omega$

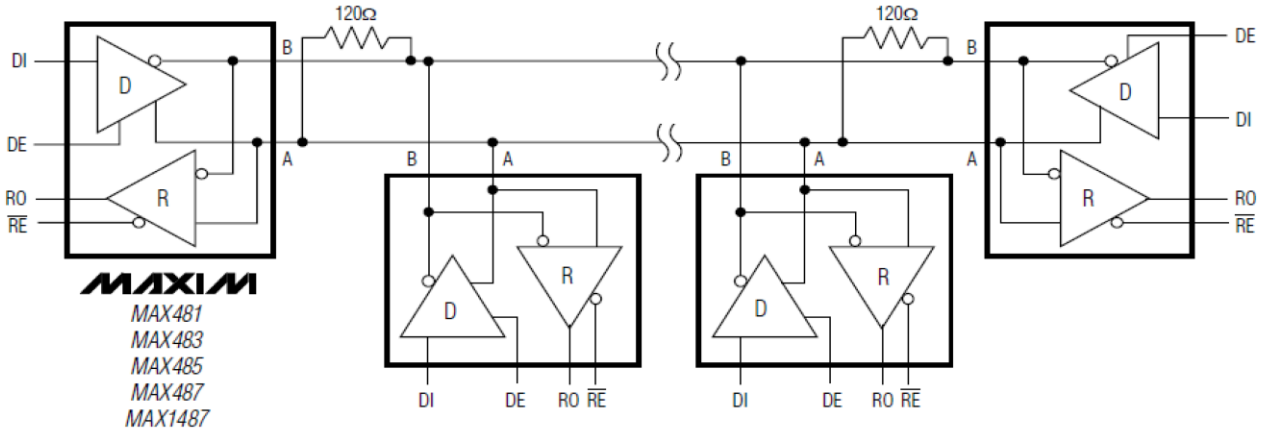
Common mode voltage $-7 \rightarrow +12V$

-Multiple drivers receivers ugyanazon a jelúton.

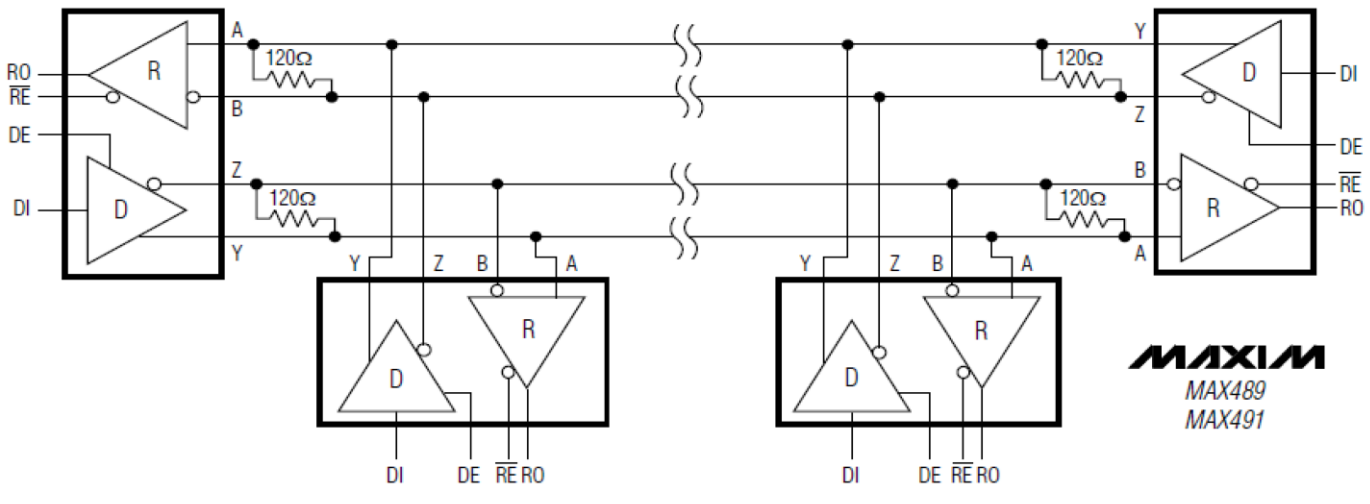
-Szükség van driver/receiver enable PIN-re

-Oda-vissza de egyidőben csak az egyik kommunikáció.

Half duplex



Full duplex

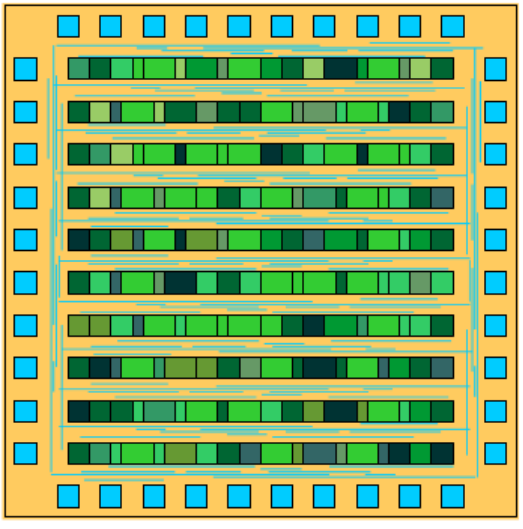


26. Ismertesse a valós idejű óra működését, mutassa be a főbb jellemzőit! Milyen órajelet szoktak használni a meghajtásra és miért?

- **Alkalmazása:**
  - Másodperc, perc, óra, hónap napja, hónap, év számlálás
  - Kalendárium funkciók
  - Alacsony fogyasztás -> alvó üzemmódban is működhet
  - 32.768kHz oszcillátorról szokták működtetni (2^{15})
  - Néhány (20) regiszter, amik értékét megőrzi
  - Külön tápáramköre van, nem a chip közös tápját használja
  - Kalibrációs lehetőség
  - Megszakítás, Alarm funkciók
- **Regiszterek:**
  - Control register
  - Interrupt control register
  - Sec, Min, Hour, Day, Week, Month, Year registers
  - Alarm register
  - Calibration value
  - GPRegister

27. A teljes chip geometriája (floorplanning). Tápfeszültség ellátás. Standard cellás tervezés. Pad- és core-limited design.

- complete fabrication process
  - predefined library of base functions
  - modular similar to TTL families
- features
  - chip size limits complexity
  - long turn around time
  - cheap at high quantities
  - standardized cell height
  - unsuitable for regular structures
  - more flexible and compact (1:4) than gate array



-Tápfesz ellátás: Horizontális tápfesz sinek be vannak építve a cellákba így a cellák egymás mellé helyezésével a táplálás automatikusan megvalósul.

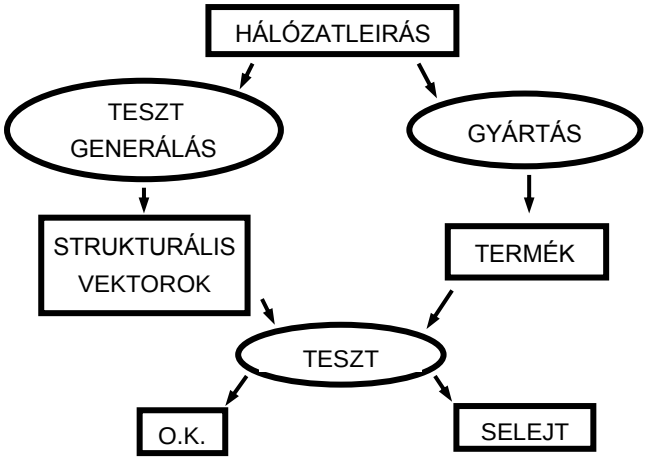
A sinek a mag két oldalán a tápfesz gyűrűkhöz csatlakoznak
(core power ring, Vdd, GND).
I/O cellák külön tápfesz.

-**Core limited**: Amikor a core mérete befolyásolja a chip méretét.

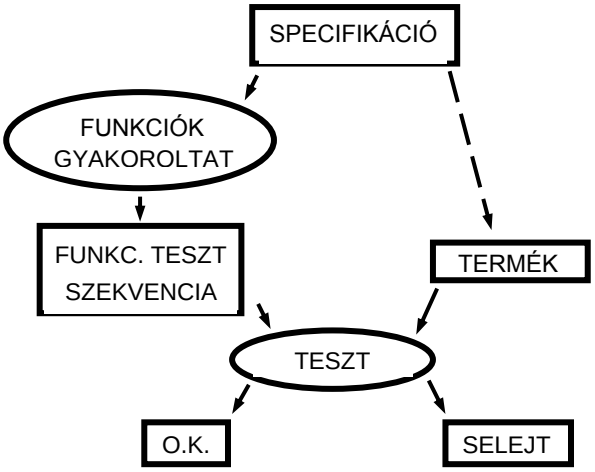
-**Pad limited**: Amikor a core túl kicsi, de sok PAD kell akkor pad limited.

28. Mi a különbség a digitális IC-k funkcionális és strukturális tesztje között? Strukturális tesztgenerálás: D-algoritmus, kritikus út érzékenyítés.

Strukturális teszt folyamatábrája



Funkcionális teszt folyamatábrája



Strukturális teszt: Csak kombinációs hálózatra létezik gyakorlatban is használható módszer.
Funkcionális teszt: Funkciókat keresünk nem hibákat.

D-algoritmus: Stuck-at hibák detektálására tesztvektor generálása, redundáns hálózatoknál nem alkalmazható.
Kritikus útérzékenyítés:Nem tesztvektort csinálunk, hanem az egész hálózatot vizsgáljuk outputokkal keresünk visszafelé. Csak egyszeres utakat tud vizsgálni.