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.

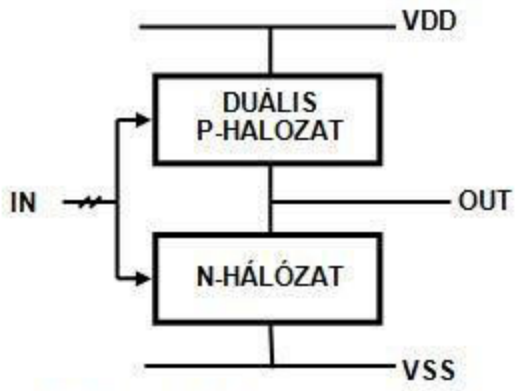
Brown-Out-Reset:
Ez az áramkör Reset-jelet biztosít a mikrovezérlőnek, ha annak tápfeszültsége egy meghatározott UBOR feszültségszint alá esik legalább ~100µs időtartamra. A BOR funkció kiküszöböli a hibás működést olyan alkalmazásoknál, ahol (például nagy áramfelvételű fogyasztók bekapcsolásakor) rövid idejű tápfeszültségesések fordulnak elő.
Ha még jobban csökken a VDD , 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) Bit stuffing: 5 ugyanolyan bit után egy ellentétes beszúrása
2) Busz monitorozása, az eszköz figyeli, hogy az van-e a buszon amitt kitett, ha nem: error.
3) Csavart érpár
4) CRC hibavédő kód
5) Szimmetrikus jelvezetés
6) Szinkronizációs szegmens
7) Konzisztencia a buszon: vagy mindenki veszi az üzenetet vagy senki sem
8) Vételi és adás hibaszámlálók

3. CMOS logikai alapkapuk felépítése. Multiplexer transzfer kapuval.

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

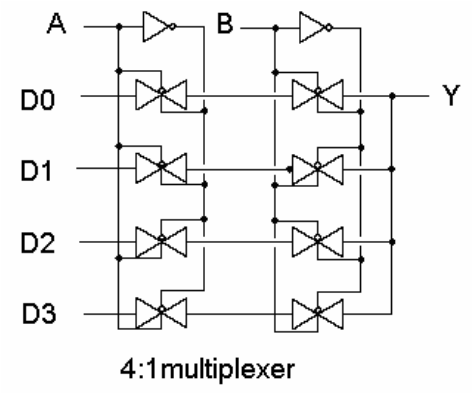


4-9. ábra: CMOS kapu általános felépítése

Az n-csatornás aktiváláskor vezet és lehúzza a kimenetet földre, a p-csatornás ezzel ellentétesen akkor húzza fel a kimenetet tápfeszültségre, ha a másik nem vezet.

Előnyök/hátrányok:
- Egy K bemenetű kapuhoz 2*K tranzisztor kell
- Nincs statikus fogyasztása

Négybemenetű multiplexer:



Négy bejövő jelből egyet kiválasztunk és kiadjuk a kimenetre.

Ehhez két szelektáló jelre, A és B –re van szükség, és természetesen ezek inverzei is szükségesek, de ezeket is meglévő bemenő 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.

- 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.

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.

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 egy 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.

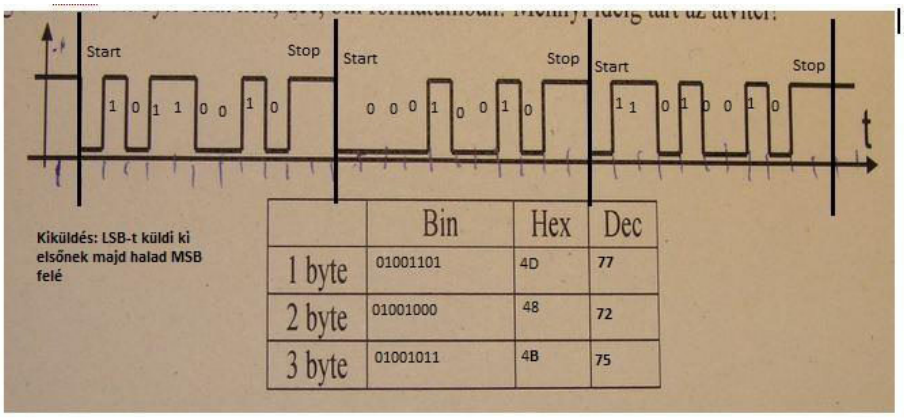
Kiakadt tranzisztor (stuck open):
A tr. nem képes a drain-jére csatlakozó vezetéket meghajtani. Az áramkört szekvenciálissá teszi.

Gyakorlatban csak a stuck-at hibákra tesztelnek.

Módszer:
1) Érzékenyítés: megfelelő bemeneti kombináció segítségével a hibától függő csomóponti értéket 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.

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?

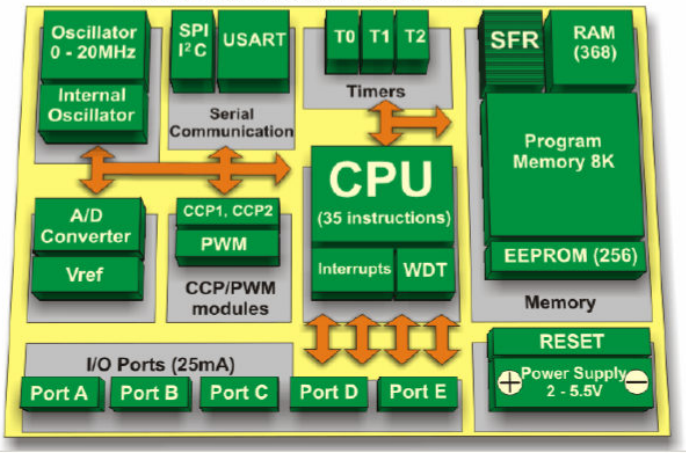


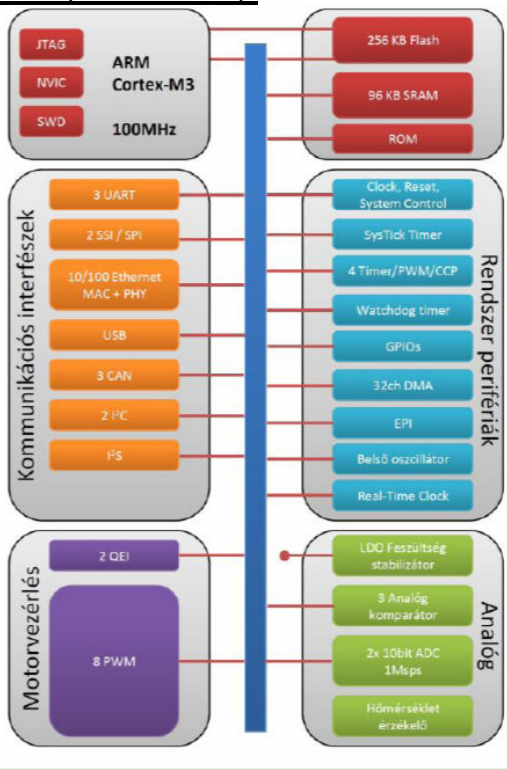
	Bin	Hex	Dec
1 byte	01001101	4D	77
2 byte	01001000	48	72
3 byte	01001011	4B	75

9600 bit/s
8N2: 1 start, 8 adat, 0 parítás, 2 stop
11 bit/packet.
3 packet
3*11 = 33 bit összesen.
t = 33bit / (9600 bit/s)

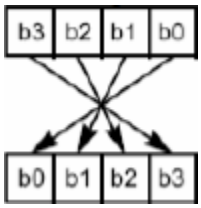
7. Sorolja fel a mikrokontrollerek alapvető építőelemeit (blokkvázlat)!

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





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



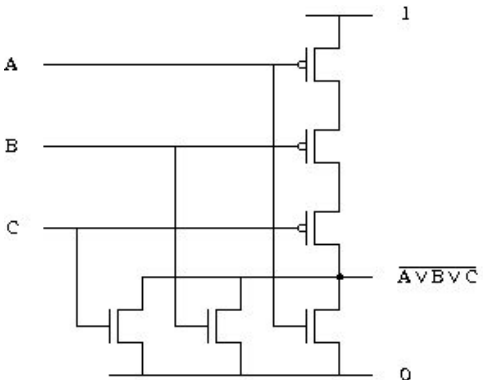
Bit Reverse, ami annyit jelent, hogy egy adott címnek a bitjeinek a tűkörképe lesz az új cím. Bit Reverse címzés, megfordítja a bitek sorrendjét. Nagyon hasznos például az FFT algoritmus megírásánál, ahol számít a sebesség. A címzés egy adott hosszúságú memóriazónát érint. A túlcsordulást pedig jobbra viszi át, nem balra, és ezáltal a fordított címet kapjuk.

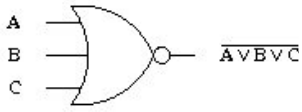
9. Hasonlítsa össze a különböző HW reprezentációkat a piacra kerülés/tervezési idő, a költségek, komplexitás, performance, fogyasztás, stb szempontokból.

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

	Cost	Performance	Power	Function	Time to market
2D SoC	–	+	+	+	–
3D SiP	+	–	–	–	+

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





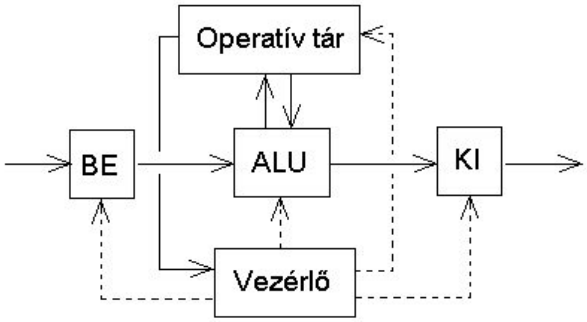
4-nél több bemenet:
Bulkhatás miatt elfogy a nyitófeszültség, ha sok tranzisztor van sorba kötve és ezért nem lehet.

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)!

Neumann architektúra:



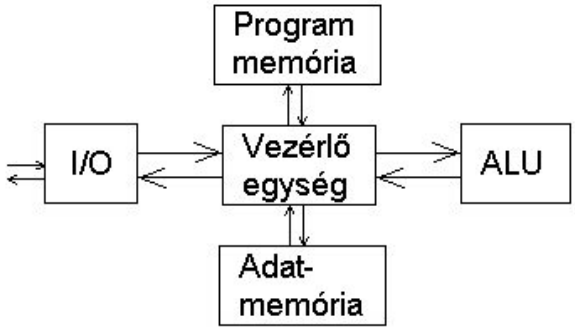
Szaggatott vonal: vezérlés | Folytonos: adat

Tulajdonságok:
- belső programtárolás, programvezérlés
- utasítás és adat azonos közegen és formában
- szekvenciális utasítás végrehajtás
- egydimenziós, lineáris címzésű memória
- bináris számábrázolás

Előnyök:
- egyszerű megvalósítás
- programok módosíthatják saját magukat
- minimális hardverigény
- minimális memóriaigény

Hátrányok:
- nem lehet egyszerre hozzáférni az adathoz és programkódhoz, ezért nehezebb pipeline-osítani a folyamatokat
- adatok és utasítások ugyanazon a csatornán mennek (szűk keresztmetszet)
- kényelmetlen a többfelhasználóra (multi-user / multi-threading)

Harvard architektúra:



Tulajdonságok:
- a programkód és az adatok külön, fizikailag elkülönített útvonalon
- Szóhossz, időzítés, technológia, memória címzés kialakítása lehet különböző is.
- Utasításokat általában a ROM-ban tárolják, míg a RAM-ban.
- A különálló buszrendszer segítségével egyidőben akár egy utasítás beolvasását, illetve adat írását/olvasását is el lehet végezni.

Előnyök:
- nagyobb memória sávszélesség
- sávszélesség pontosabb jóslása
- utasítás és adat egyszerre olvasható / írható
- az adatok és a programok fizikailag el vannak választva, így hozzáférési jogok és memóriavédelem könnyebben megvalósítható
- programhiba esetén programkódot nem ír felül véletlen

Hátrányok:
- a nem használt adatmemóriát nem lehet programmemóriaként felhasználni
- Az olyan egychipes rendszereknél (pl. SoC), ahol egyetlen chipen van implementálva minden funkció, nehézkes lehet a különbözőmemória technológiák használata az utasítások és adatok kezelésénél. Ezekben az esetekben a Neumann architektúra alkalmazása megfelelőbb.
- A magas szintű nyelveket sem közvetlenül támogatja (nyelvi konstrukció hiánya az utasítás adatként való elérésére) – assembler szükséges

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

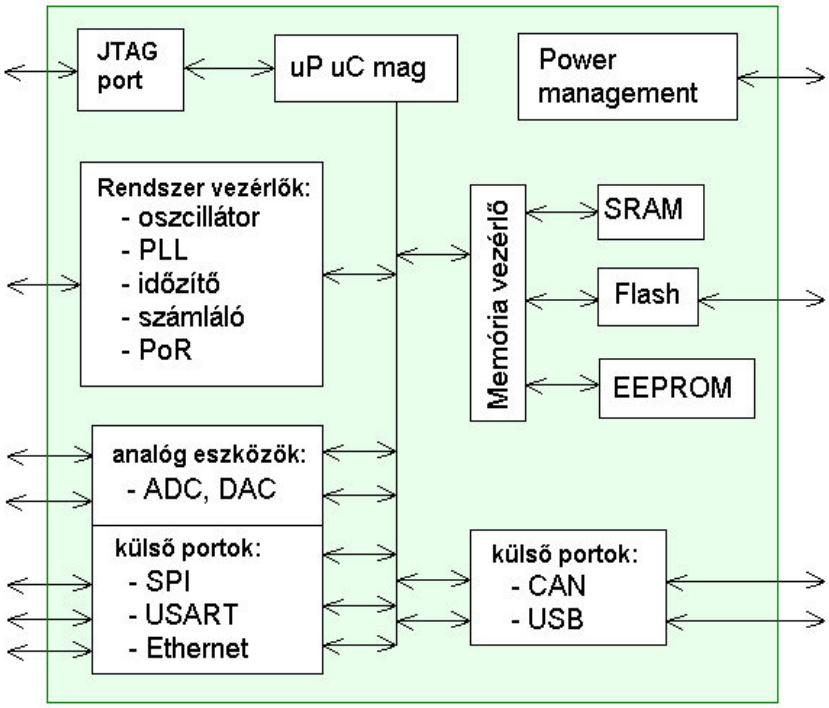
Fő részek:
- processzor (ALU + regiszterek)
- grafikus processzor
- hangchip
- memória (RAM, ROM, memória- és DMA-vezérlő)

Belső egységek:
- busz
- órajel, számláló
- megszakítás kezelő
- WDT (Watchdog Timer) biztonsági időzítő áramkör
- Debug-portok, pl. JTAG
- speciális matematikai egységek (pl. DSP-SoC-ban)
- kriptográfia – kódolástechnika

Periféria-egységek és portok:
- billentyűzetvezérlő
- grafikus portok (monitor)
- soros portok (USB, CAN-busz)
- párhuzamos portok
- impulzus-szélesség moduláló áramkör (pl. motorvezérlő)
- egyéb modulátorok (GSM-kódolás, QAM, fázismodulálás)
- egyéb portok (ethernet, MAC vagy USB)
- AD konverter

DIÁBÓL:

- uC, uP, DSP mag
- Memória:
  - ROM,
  - RAM,
  - EEPROM,
  - Flash
- Oszcillátor, PLL
- Időzítő, számláló, PoR
- Külső interfészek:
  - SPI
  - USART,
  - CAN,
  - USB,
  - …
- Analóg interface: ADC, DAC
- Power management
- Összeköttetés, hálózat



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

Megvalósítási lehetőségek: Chip-Stack, MCM (multi-chip module), belső huzalozás, microbump-olás, FlipChip. Tartalmazhat analog-digital RF chipeket.

Tulajdonságok:
- 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.

	Cost	Performance	Power	Function	Time to market
2D SoC	–	+	+	+	–
3D SiP	+	–	–	–	+

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. A Verilogban a fizikai valőság pontosabb modellezése érdekében 3-féle értékadás létezik - nem mindegy például, hogy vezetéknek vagy regiszternek adunk értéket:

- folyamatos értékadás: vezetéknek. Az értékadás jobb oldalán lévő értékek bármelyike is megváltozik, azonnal frissül a bal oldal is a kifejezésnek megfelelően. Ez a logikai kapuk működését modellezi. Kombinációs logikát lehet vele megvalósítani egy logikai függvénnyel. Always blokkon kívül használható és kizárólag wire típusú változónak lehet vele értéket adni.
- procedurális értékadás (always, initial): regiszternek. Procedurális értékadás kizárólag eseményvezérelten történhet. Ez azt jelenti, hog egy olyan blokkban kell elhelyezni, amely események hatásásra aktiválódik. Verilogban két ilyen blokk létezik:
  a) initial: a szimuláció elején, egyszer fut le, és többször nem,
  b) always: a fejében megadott feltétel teljesülésekor mindig lefut.

  Procedurális értékadás tehát csak initial vagy érzékenyített always blokkban szeretepelhet. Két fajtája van:
  a) blokkoló értékadás: az egymás alatt lévő értékadások a megadás sorrendjében, egymás után hajtódnak végre. Egy blokkban mindig előbb hajtódik végre, mint az őt követő utasítások.
  b) nem-blokkoló értékadás: az egymás alatti értékadások párhuzamosan, egyidőben hajtódnak végre

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
```

Tipikus hibák, melyek latch alkalmazásához vezetnek, miközben a cél kombinációs logika leírása:
- nem teljesen kifejtett case szerkezet (nincs default)
- 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;
```

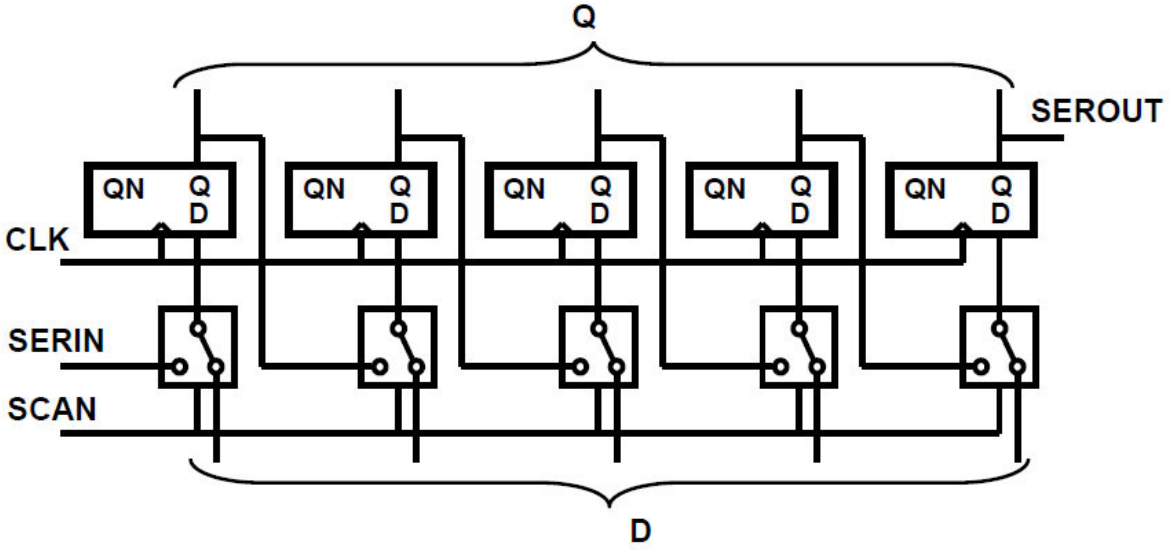
16. Digitális szimuláció jeleinek érték-készlete, hogyan értelmezzük őket?

LOGIKAI JELSZINTEK:

L → 0, U < 0.5 V
H → 1, U > 4.5 V
X → HATÁROZATLAN, 0 < U < 5 V
Z → NAGYIMPEDANCIA (FLOATING)
U → ISMERETLEN (NINCS INICIALIZÁLVA)

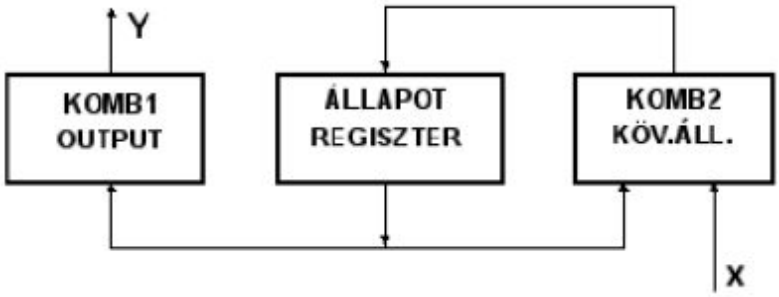
17. Mire való a Scan-Path? Hogy néz ki az áramköri megoldás?

A belső állapotokat tartalmazó regiszter be- és kimeneteit teszi elérhetővé úgy, hogy a



regiszter flip-flopjait átkonfigurálja léptető regiszterré.

A szekvenciális hálózat átkonfigurálására való kombinációs hálózattá, 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:

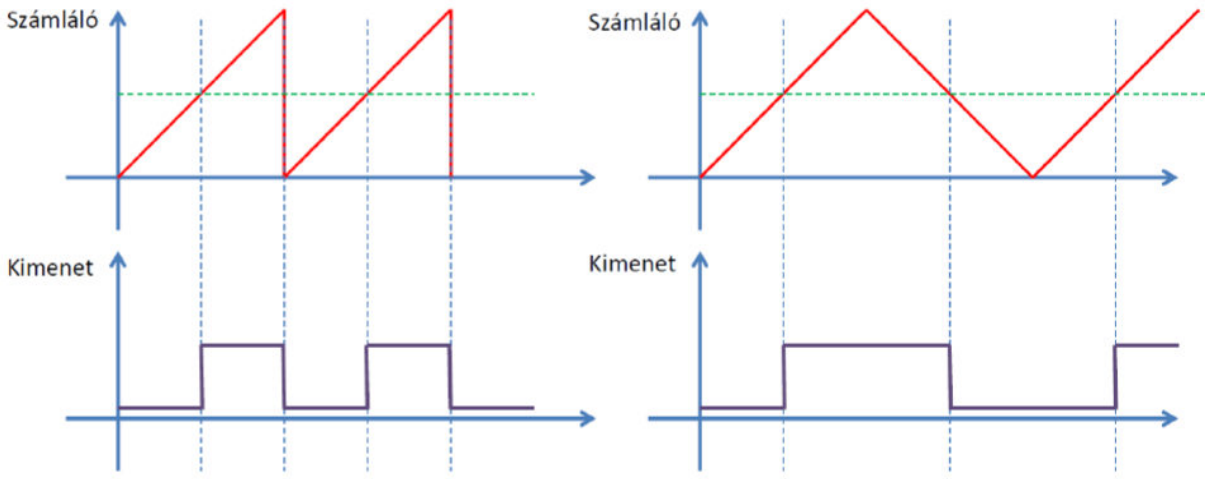
1) Szisztematikus (Scan-Path, peremfigyelés)
2) Ad-hoc (teszt pinek alkalmazása, teszt üzemmód, belső részegységek elérése multiplexerrel)

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.
- A 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 vezérlés

- Regiszterek:
  - Value
  - Period
  - Compare
  - Control



21. Verilog: modul felépítés, tesztbench

- 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 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.

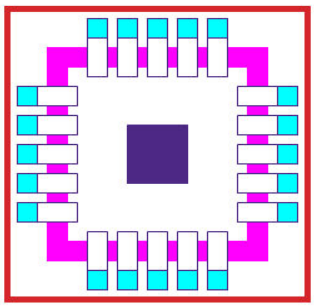
Tesztbench:
A tesztkörnyezet is egy modul, amit az különböztet meg a többitől, hogy nincsenek portjai.
Az ilyen modult a szimulátorok fel szokták ismerni legmagasabb hierarchia-szintű modulként.
Kézzel csak olyankor kell megadni, ha több ilyen is van a rendszerben.
A felépítése általában a következő:

1) A tesztelendő modulok példányosítása és összekötése.
2) A tesztszekvenciák leírása egy initial blokkban.
3) A periodikus jelek előállítása always blokkokban.

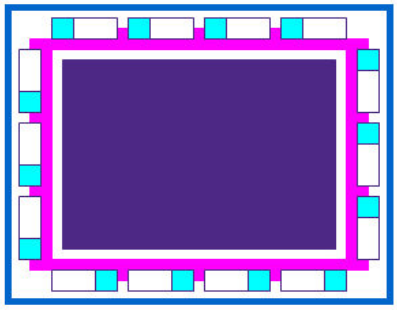
22. Mit ért Pad-limited valamint Core-limited design alatt?

pad limited: túl nagyok a forrszemek (pad-ek). Több be- kimenet van, mint ami elférne a "die"-on.

core limited: túl kevés forrszem van, és a forrszemek között üres hely van. Túl sok logikai elem van, és nem fér el a "die"-on.

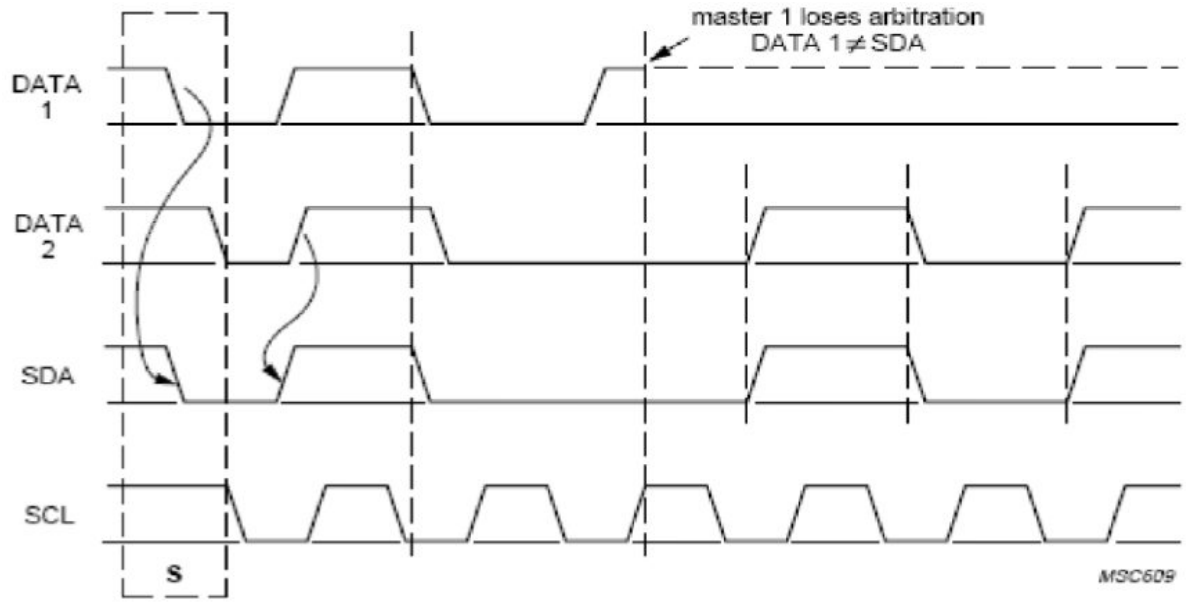


pad limited



core limited

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

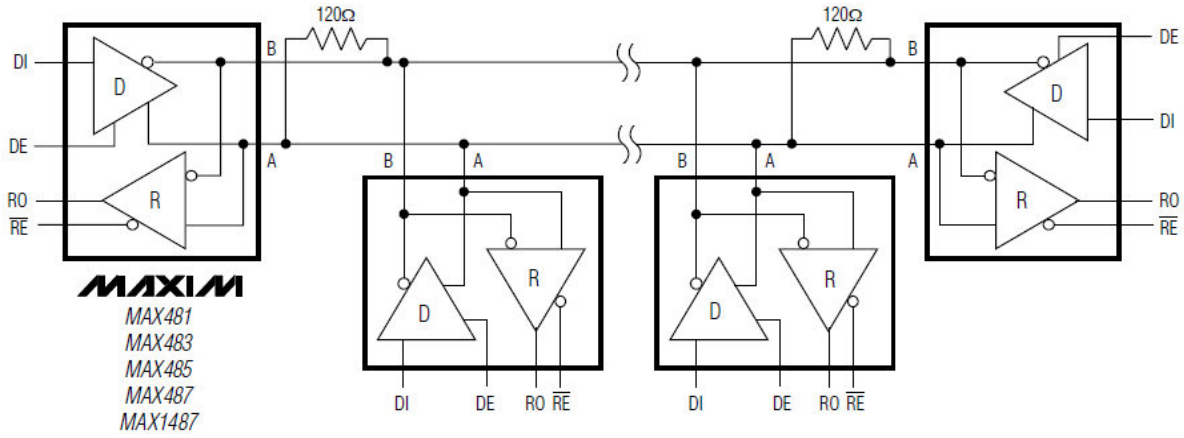


25. Rajzoljon fel egy tipikus RS485 half-duplex busz architektúrát, jelölje a szükséges kiegészítő komponenseket is! Sorolja fel a busz néhány jellemzőjét (min. 3)!

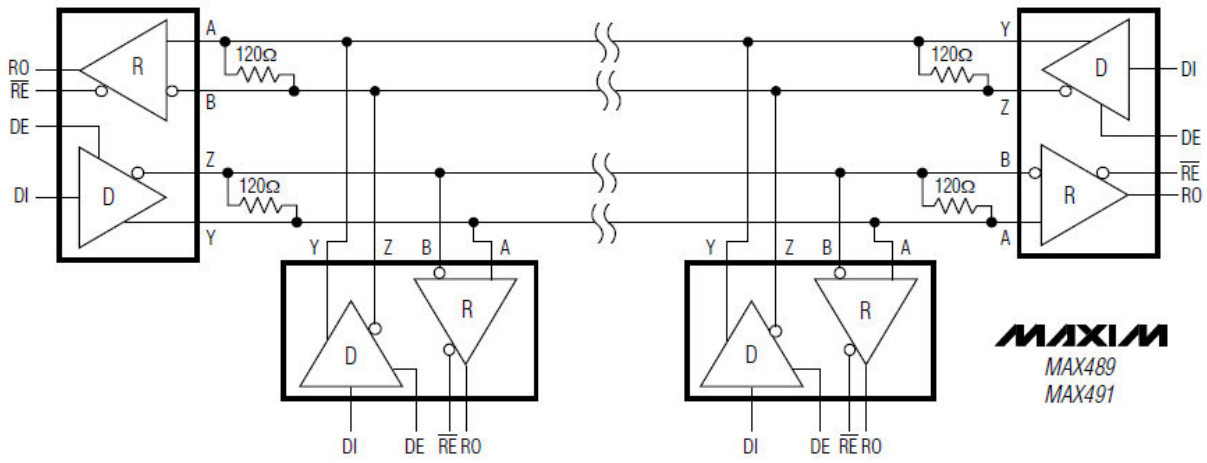
Busz jellemzői:

- 52Mbit/s
- 120Ω lezárás
- Full / Half duplex üzemmód
- < 1200m
- Szimmetrikus jelvezetés
- Galvanikus leválasztás
- Fail Protected, Fail Safe, ESD protected
- Input impedancia: ≥ 12kΩ
- Common mode voltage: -7 → +12V
- ugyanazon a jelúton több meghajtó és fogadó (multiple drivers / receivers)
- szükség van meghajtó / fogadó engedélyező PIN-re (driver / receiver enable PIN
- oda-vissza kommunikáció, de egyidőben csak az egyik irányban

Half duplex:



Full duplex:

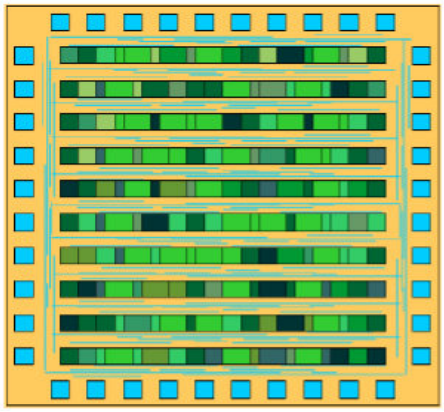


26. Ismertesse a valós idejű óra működését, mutassa be a főbb jellemzőit! Milyen órajelet szoktak használni a meghajtásra és miért?

Alkalmazása:
- másodperc, perc, óra, hónap napja, hónap, év számlálás
- kalendárium funkciók
- alacsony fogyasztás --> alvó üzemmódban is működhet
- 32.78 kHz-es oszcillátorról szokták működtetni (2^15)
- néhány (20) regiszter, ami megőrzi az értékét
- 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 registers (Min, Hour, Day, Month)
- calibration value

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

Tápfeszültség ellátás:
- horizontális tápfeszültség sínek vannak beépítve a cellákba, így a cellák egymás mellé helyezésével a táplálás automatikusan megvalósul.
- A sínek a mag két oldalán a tápfeszültség gyűrűkhöz csatlakoznak (core power ring, Vdd, GND).
- I/O cellák külön tápfeszültséggel rendelkeznek

Core limited: amikor a core mérete befolyásolja a chip méretét.
Pad limited: amikor a core túl kicsi, de sok I/O PAD kell, akkor pad limited.

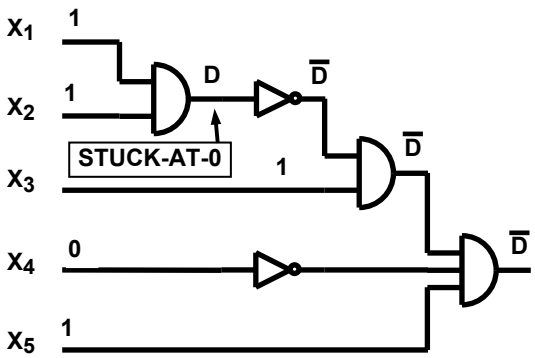
28. Mi a különbség a digitális IC-k funkcionális és strukturális tesztje között? Strukturális tesztgenerálás: D-algoritmus, kritikus út érzékenyítés.

Strukturális teszt: csak kombinációs hálózatra létezik gyakorlatban is használható módszer. A strukturális teszthez hibamodellt definiálunk, a hálózatleirás alapján képezzük a lehetséges hibák listáját, majd ezekre sorban a vezérlés/megfigyelés elv alapján tesztvektorokat generálunk. Ennek a módszernek lényeges megkötöttsége, hogy csak kombinációs hálózat

esetére léteznek a gyakorlatban is használható módszerek, amelyek a későbbiekben ismertetésre is kerülnek. A strukturális tesztszekvencia jellemzöje a százalékos hibalefedés, amit hibaszimulációval lehet megállapitani.

Funkcionális teszt: funkciókat keresünk nem hibákat. Hibamodellt nem definiálunk, viszont a stimulusokkal gyakoroltatjuk a funkciókat. A helyzet hasonló a logikai verifikációhoz, de itt a gyakoroltatásnak sokkal alaposabbnak kell(ene) lennie, amit nehéz kivitelezni.

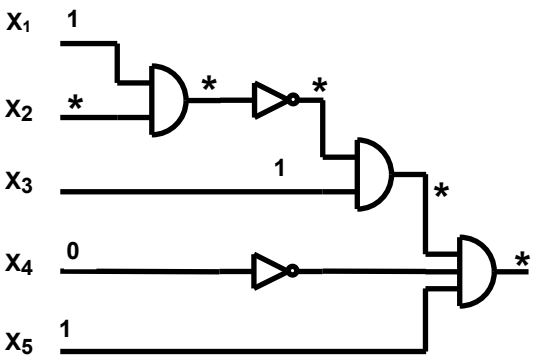
D-algoritmus: stuck-at hibák detektálására tesztvektor generálása, redundáns hálózatoknál nem alkalmazható.



Kombinációs hálózatokban stuck-at hibák detektálására a D-algoritmussal lehet teszt vektorokat generálni. A D-algoritmus exakt és komplett: amennyiben az adott hiba detektálható, a tesztvektort elöállitja.

Mind a feltételezetten hibás kapu, mind pedig az érzékenyitett ut mentén fekvö kapuk bemenetére normál logikai jeleket (nulla és egy) kell adni ugy, hogy az érzékenyités létrejöjjön. Ezután ezeket a jeleket “visszafelé kell terjeszteni” (*backward propagation*) a hálózat primer bemeneteihez, azaz ott olyan értékeket kell elöirni, amelyek mellett az elöbbi kijelölések teljesülnek.

Kritikus útérzékenyítés: nem tesztvektort csinálunk, hanem az egész hálózatot vizsgáljuk outputokkal keresünk visszafelé. Csak egyszeres utakat tud vizsgálni.



egy érzékenyitett ut mentén egy jelváltással az összes stuck-at hibát detektálni lehet. Itt nem keresünk tesztvektorokat egyes kiválasztott hibákhoz. Ehelyett az áramkört egészében kezeljük. Primer outputoktól indulva keresünk visszafelé jel-utat a bemenetek felé, és ezeket érzékenyitjük.
Ezekkel az érzékenyitett – itt kritikusnak nevezett – utakkal az egész áramkört lefedjük, és ezzel (majdnem) minden kiakadási hibát tudunk detektálni.
A módszer gyengéje, hogy csak egyszeres utakat tud kiépiteni. Ha a hálózatban egy jel-ut elágazik, majd ujból egyesül (*rekonvergens fanout*), akkor ott hibák kimaradhatnak a detektálásból. Ilyenkor ezeknél a hibáknál vissza kell nyulni a D-algoritmushoz.

29. Miért jó a szimmetrikus jelvezetés?

A szimmetrikus jelvezetés a professzionális hangtechnikai berendezéseknél elterjedt jelvezetési eljárás. Előnye, hogy a kábel környezetében előforduló elektromágneses zajok nem kerülnek rá a jelre, függetlenül a kábel hosszától. Ennek oka, hogy a szimmetrikus jelvezetéshez három vezetékre van szükség, melyek közül egyiken a föld, másikon a jel pozitív fázisa (aszimmetrikus jel) harmadikon annak negatív fázisa - ellentettje - halad. A hasznos jelet a pozitív és a negatív fázis közti különbségből kapjuk. Így ha a kábel mentén zaj kerül a jelhez, az mindkét fázisra egyformán rákerül. Ennek köszönhetően a fázisokon levő zajok közti különbség nulla lesz, vagyis a kábel menti zajok szimmetrikus jelvezetéssel kizárhatóak.

30. Elhelyezési és huzalozási algoritmusok

Cellák elhelyezése:

A program hétköznapi elnevezését (P&R, Place & Route) is adó két fö fázisból ez az elsö. Ez többnyire a nehezebb, és hosszabb futási idöt is igényel. A cellák elhelyezésének egyik célfüggvénye az, hogy a program becsüli a (következö fázisban esedékes) huzalozás össz vezetékhosszát, és minimumot próbál keresni. A jó elhelyezés azért fontos, mert huzalozás közben már nem lehet rajta változtatni. Lehet az Encountertól "elözetes elhelyezést" is kérni, amely kevésbé részletes/optimális és ezért gyorsan elkészül. Ezt azután ki lehet iratni az Encounternél szokásos DEF fájlba, és át lehet adni az RTL Compilernek (Layout tudatos szintézis!), hogy az felhasználja a vezeték kapacitások pontosabb becsléséhez.

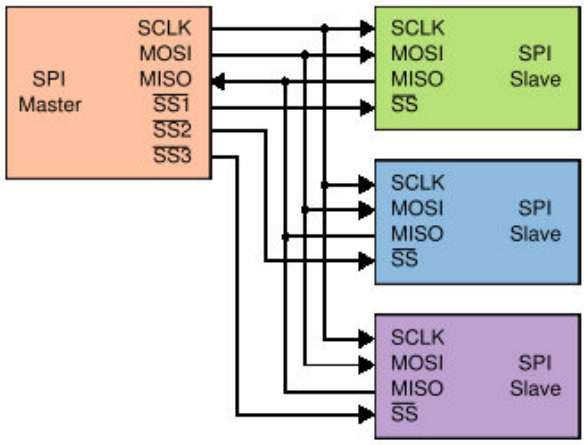
Huzalozás:

Ez az Encounter második fő, névadó fázisa. A kivitelezés két lépésben történik, global és final r*outing*. A global routing feladata a vezetékek nagyjából elrendezése Manhattan (derékszögű) stílusban. A tervező ezután, ha szükségesnek látja, kérhet a vezetékek sűrűségéről egy színkódos ábrázolást és kiértékelheti, hogy van-e vezeték torlódás (congestion). Ha van, akkor fontolóra veheti, hogy csináltat egy újabb elhelyezést szigorúbb feltételekkel, vagy esetleg egy kissé megnövelt floorplannel.

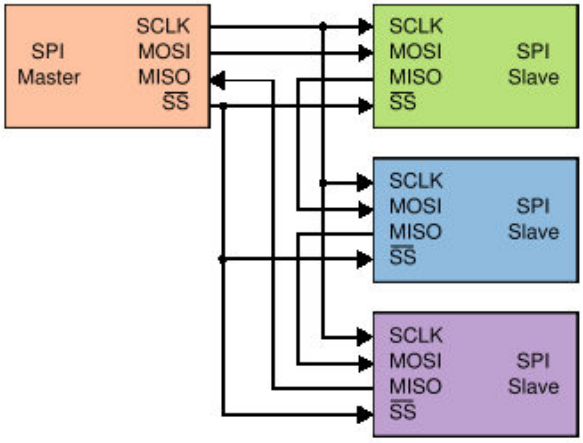
A final routingnál a program a teljes chip felületét felosztja kisebb részekre, "dobozokra" (kb. 5x5 és 15x15 között) és ezeknél egyenként, egymás után végzi el az összeköttetések végleges elhelyezését. A siker itt sincs garantálva, de a rutinos tervező a floorplan alkalmas kialakításával jó esélyeket teremthet rá.

31. Összekötős feladat, hogy független slave-ek legyenek (+ ábra, nem-Gärtner)

független slave-ek:



láncolt slave-ek:



32. A digitális szimuláció időkezelése. Késleltetések, hazárdok.

a) Késleltetések:

A Boole-algebra az idő dimenzióját nem foglalja magában. Ezen a téren csak többé-kevésbé közelítő, lényegében heurisztikus módszereket lehet alkalmazni.

(Tisztán) logikai szimuláció:

Nulla késleltetésű (Zero-delay)

Egységnyi késleltetésű (Unit-delay)

Időzítő (timing) szimuláció:

Névleges késleltetésű (Nominal-delay)

Kevert módusú (Mixed-mode)

A zero-delay szimuláció nem más, mint a hálózat Boole-egyenleteinek statikus megoldása. Nagyon egyszerű, de csak a stacionárius megoldást szolgáltatja. Az időbeli viszonyokra semmi felvilágosítást sem ad. Az egyes kapuk terjedési idejének a hatása a működésre ismeretlen marad. Elegendő a hálózatnak csak azokat az elemeit szimulálni, amelyek bemeneti állapota változik → *eseményvezérelt* szimuláció.

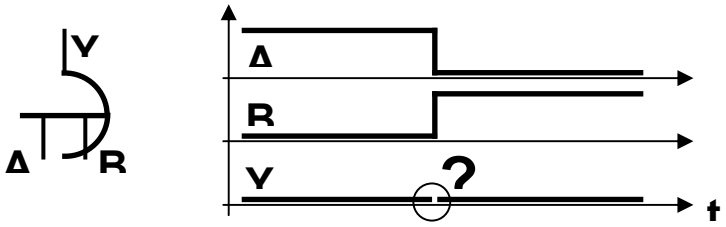
A *unit-delay szimuláció* nem más, mint a zero-delay szimuláció lébonyolítása az esemény-vezérelt elv alkalmazásával úgy, hogy minden kiértékelési fordulóhoz egy t_d idő-inkrementet rendelünk hozzá. Ez úgy értelmezhető, hogy a jelek minden kapun való áthaladáskor egységnyi t_d késleltetést szenvednek. A teljes késleltetés meghatározásához nem kell mást tenni, csak megszámolni a fordulókat.

A *nominal-delay szimuláció* szintén az eseményvezérelt elvet alkalmazza, de úgy, hogy közben az időt is figyeli. Minden kapunak megvan a saját „névleges" késleltetése.

A kevert módusú (mixed mode) szimuláció úgy jött létre, hogy – legalábbis az elméleti megfontolások szintjén – vissza kellett nyúlni az áramköri szintre. A CMOS áramköröknél ugyanis az időzítés erősen terhelésfüggő.

b) hazárdok:

egy AND kapu esetén ha A és B egyszerre változnak, bizonytalan helyzet áll elő.



A helyzet kétszeresen nehéz:

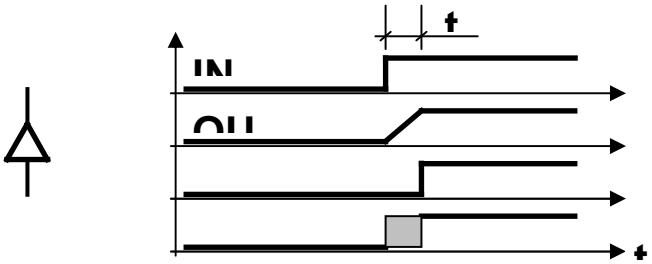
1. A két változás soha nem igazán „egyidejű". Ez az ideális eset nem létezik. Az egyik jel valamivel előbb, a másik valamivel később jön. Ebben az esetben a Boole-algebra már tudna érvényes eredményt adni, a baj csak az, hogy nem tudjuk, melyik változás jön hamarabb.
2. A gyakorlatban nincs ideális jelváltás. A jelváltás véges idő alatt zajlik le, és még ehhez jön az esetleges időbeli eltolódás.

c) időzítési viszonyok modellezése:

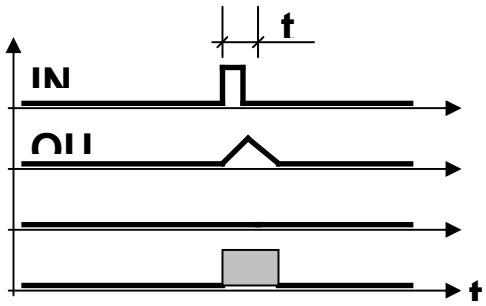
A kimeneti jel (OUT) három különböző esetben::

1. A mixed mode elvét alkalmazva t_p hosszúságú lineáris átmenetet hozunk létre.
2. A változást t_p idő elteltével áttesszük a kimenetre.
3. A váltást t_p hosszúságú X (határozatlan) szint előzi meg.

egyszerű jelátmenet:



rövid impulzus / tüske:



A bemenő impulzus hossza rövidebb a terjedési időnél.

Egyéb (kidolgozott vizsgatételek 2007 - kézzel írott):

1. IC design-flow (szimulátor, reprezentáció, absztrakciós szint)
2. standard cella IC
3. Multi project wafer
4. Rendszertervezés: hierarchikus kézi bevitel (specifikáció -> layout)
5. Rendszertervezés: HDL (top-down)
6. Tervezési szempontok: Szinkron és aszinkron hálózatok. (előnyök, hátrányok) Reset, órajel (előállítása, órafa) tápfeszültség.
7. Tervezési szempontok: disszipáció és fogyasztás. Energiacsökkentés lehetőségei. A disszipáció forrásai.
8. Tesztelhetőségre tervezés (DFT). Ad-hoc módszerek.
9. D-algoritmus, kritikus út érzékenyítés, interaktív teszt (kézi szekvencia / véletlen szekvenciák), kombinált stratégiák.
10. Funkcionális és struktúrális teszt. (folyamatábrák) Tesztgenerálás módszerei. Teszter.
12. Verilog lexikális elemek, adattípusok, kifejezések, értékadás, viselkedési modellek.
13. Verilog: procedúrák (értékadás, feltételek), modul deklarálás és felépítés, hierarchikus struktúra és tesztbench.
14. Verilog szerkezetek szintézise (kombinációs logika, regiszter (D-FF, latch), reset, case, zero-delay időzítés problémája
15. Cadence PKS szintézis program: felépítés, generikus szintézis
16. PKS: mapping, optiomalizálás, futásidő analízis, futásidő csökkentése
18. Áramkör szimulációi, felépítés, működés, algoritmusok, szolgáltatások. Alkatrészkészlet, modellek.
19. Analízis fajtás
21. Áramtükrök, feszültség és áram referencia
22. Bandgap-referencia
23. Differenciál erősítők
25. Kaszkád erősítők
26. NMOS minimál inverter, CMOS kapuáramkörök, duális szerkezet
27. Transzfer kapuk és alkalmazásaik
29. Layout tervezés, tervezési szabályok, lambda (véletlen/rendszeres hibák, reprodukálhatóság, egyenszilárdság, lambda tervezési szabályok), kihozatal
30. Layout tervezés: pálcika diagrammok (topológiai tervezés, méretbecslés, egyszerű ROM vagy PLA tervezés)
31. Layout reprezentáció számítógépekben (leírásmódok, leírás rendszere, hierarchia, belső leírás, belső tárolás, műveletek, GDS II)
32. Layout ellenőrzés: DRC, extruct, LVS (+tipikus hibák)
33. Layout tervezési szempontok: arányok, szimmetria, common centroid struktúra
34. A teljes chip geometriája
36. Semicustom IC-k: Gate forest rendszer
37. Mixed signal ASIC-ek (analóg cellakönyvtár, AD + DA konverzió)
37b. Komparátor működése, nominal delay, mixed mode, logikai értékek, szimuláció
39. A szimuláció szintjei. Logikai szimuláció. Értékkészlet, időkerék, CMOS modell.