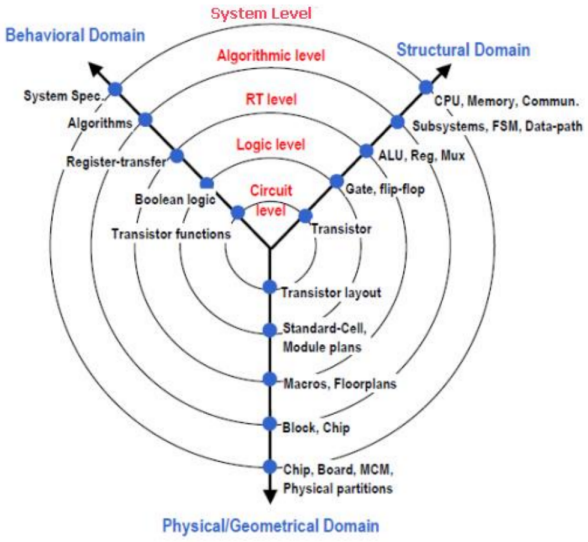
# IT eszközök technológiája - 4. előadás Digitális rendszertervezés

## Absztrakciós szintek

- digitális rendszerek bonyolultsága exponenciálisan növekszik
- tervezési metódusoknak is fejlődnie kell!
  - tervezés magasabb absztrakciós szinten
  - alacsonyabb szintek automatikusan készülnek el, az ember csak felügyel és kényszereket szab
- automatizálás negatívuma → romlik a hatékonyság

### Gajski-Kuhn Y diagram




- körökből álló a ábra
- 3 vizsgálati szempont
  - **viselkedési** szempont
  - **struktúrában** hogy jelenik meg
  - **fizikailag** hogy jelenik meg

#### Kör szintek

- **rendszerszint**
  - fő cél: részekre bontás (partícionálás)
  - hardverfüggetlen, akár több különböző megvalósítás is lehet
  - eszköz: magas szintű programnyelv
- **algoritmus**
  - fő cél: funkciók viselkedés szintű modellezése
  - eszköz: c++, SystemC, mert erőforrásigényes a szimuláció!
- **regiszter-transzfer szint (RTL)**
  - elvonatkoztatott szint

    - *regisztereket és a köztük végbemenő adatátvitelt definiáljuk, beleértve az adatátvitel összeköttetéseit és az átvitel időzítését is.*
    - mi történjen az áramkörben az órajel két aktív éle között
- **logikai szint**
    - logikai kapuk és összeköttetések → hálózatlista
    - eddig kb megvalósításfüggetlen a tervezés
- **áramköri szint**
    - kapcsolási rajz
    - fizikai terv → fizikai tervezés

## Kör szintek közti lépés

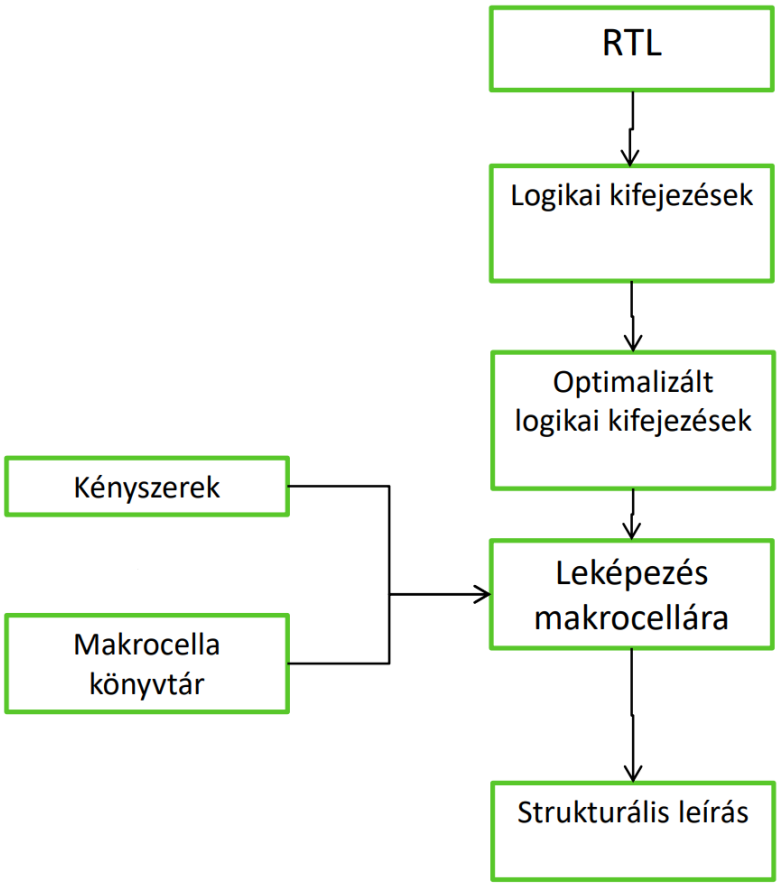
- szintézis → magasabbról alacsonyabb absztrakciós szintre lépés (beljebb lépés)
- minél beljebb vagyunk, annál inkább automatikus program váltja fel az embert
- **magas szintű szintézis (HLS)**
- **logikai szintézis**
- **elhelyezés és huzalozás**

### Magas szintű szintézis

- algoritmus szintről RTL szintre
- általában emberi közreműködésével, DE egyre több jó szintézer program
    - időzítésmentes C kódból generálnak RTL szintű leírást
- probémák:
    - **Vezérlés** jellegű funkció esetén
        - állapotgépek konstruálása, állapotgépek és kisegítő áramkörök együttműködésének megszervezése
    - **Adatfeldolgozás** jellegű funkció esetén: mikroarchitektúra választás/szintézis
        - erőforrás-allokáció, ütemezés, erőforrás-vezérlési állapot összerendelés
- automatizálás előnyei: könnyebb újrafelhasználás, kényszerek magasabb szinten való alkalmazása, több architektúrából választás lehetősége
- példa: *itt volt egy példa, ezt nem másoltam be, itt a* link

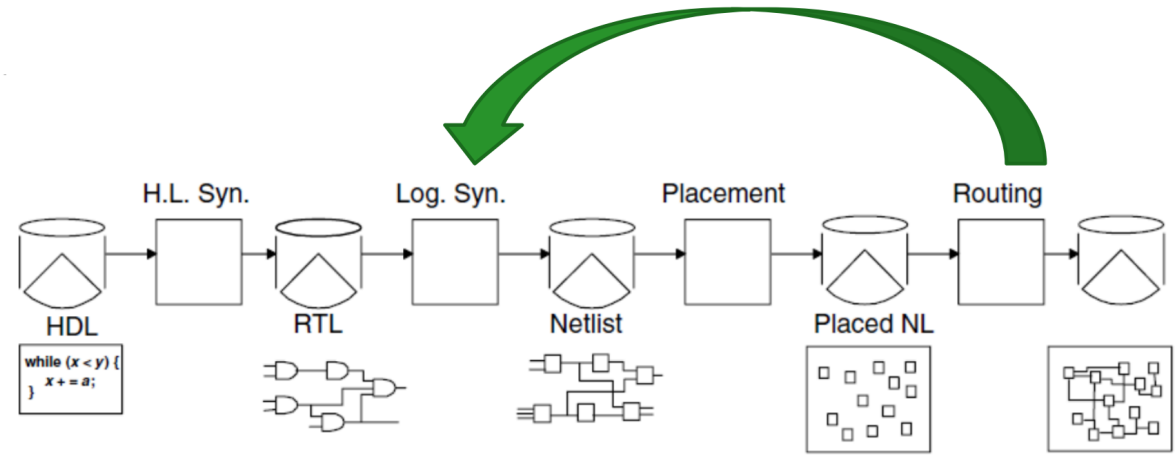
### Logikai szintézis

- RTL kódból logikai szint (strukturális, kapu szintű leírás)
- *input:* RTL kód, cellakönyvtár időzítési és teljesítményadatai
- *output:* struktúrális HDL (csak cellakönyvtárbeli elemeket tartalmaz)
- *kényszerek:* időzítés, terület, teljesítmény (ezek becsültek lesznek csak, mert még nem fizikailag elkészült)
-




- lépések:
  - HDL beolvasása, optimalizálása
  - hierarchia kifejtése, ha szükséges → flattening, mivel jobb minőségű megoldás kapható a globális optimalizálással, de sokkal erőforrásigényesebb mint modulonként optimalizálni
  - logikai kifejezések optimalizálása (generic)
    - szerkezetek (összeadó, számláló) felismerése, optimalizálja a logikai kifejezéseket
  - leképzés ASIC makrocellára
    - ha felismert template-et, azt könnyű
    - logikai kifejezést összerakja makrocellákból
      - logikai kifejezésből → fa építés, amiben csak inverter és NAND kapu van
      - megpróbálja költségfüggvények alapján lefedni

## Tervezési út → design flow



- zöld nyíl → a tervezés iteratív
    - CMOS-nál a késleltetés az összekötő hosszaktól függ → pontos késleltetés csak az elhelyezés és huzalozás után derül ki

# Hardver leíró nyelvek

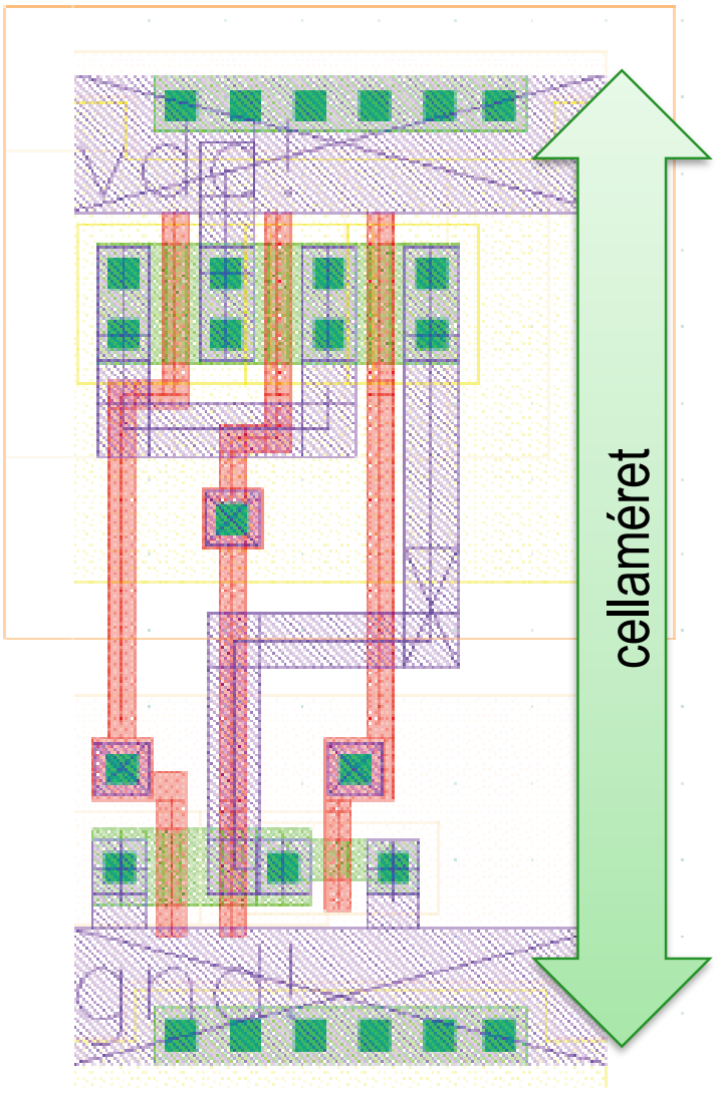
- **VHDL** → Amerikai védelmi minisztérium megrendelésére, eredetileg ASIC-ek funkciójának formális dokumentációjára találták ki
    - erősen típusos nyelv (ADA nyelven alapul)
    - nagy projekt létrehozására is alkalmas → könyvtárak, hierarchia támogatása
- **SystemVerilog** → eredetileg logikai szimulációra találták ki, eredeti őse a Verilog volt
- kód példa: link
    - utasítások párhuzamosan hajtódnak végre (sorrend mindegy)
    - kombinációs hálózat → milyen bemenetre milyen kimenetet adjunk
    - szinkron hálózat → érzékenységi lista, minek a változására mi történjen
- **SystemC**
    - C++ osztálykönyvtár digitális rendszertervezésre
    - ezeket kell megvalósítani: párhuzamosság, időzítés, késleltetés, port, biz pontos adattípusok
    - előny → alkalmas lesz lefordított szimulációra, mert szimulációs kernelt is tartalmaz (gyors)
    - fogalmak:
        - **modul** → egy adott funkcionalitás megvalósítására szolgál, más modulok vagy process-eket foglalhat magába (C++ osztály lesz)
        - **process** → funkció leírása (modult megvalósító osztály metódusa)
        - **port** → logikai jel csatlakozó pontja, iránya (bool-ok)

## Terminológia

- HDL-ek NEM programozási nyelvek
    - hasonló szerkezetek, de a jelentés sokszor eltérő (pl for cikus → egy elemet tegyünk le x-szer)
- HDL program → HDL modell
- HDL kód lefordítása → HDL modell elemzése vagy HDL modell szintézise
- HDL program futtatása → HDL szimuláció
- FPGA programozása → FPGA konfigurálása

# Fizikai tervezés (*csak kis részben ZH anyag*)

- eddig: nagyjából technológia független
- kész terv megvalósítása:
    - teljesen kézzel (csak kritikus blokk esetén)
    - előre tervezett cellakönyvtár segítségével
    - programozható logikai eszköz segítségével (pl FPGA)
- **cellakönyvtár**
    - alapvető logikai áramkörök gyűjteménye (alapkapuk, komplex kapuk, flip-flop-ok)
        - különböző meghajtóképességű kapuk → fizikai méret és fogyasztás is más lesz

    - standard cella magassága rögzített, szélessége változhat
        - azonos magasságú sorok lesznek → *cellasorok*
        - cella tetején → táp, cella alján → föld
- **layout** → egyszerre ábrázolva az összes réteg (rétegek különböző színekkel)
    - visszafejtés levezetése → VIDEÓBAN 1:09:10
    - 

- lépések
    - **Floorplan** → áramkört alkotó blokkok, be és kimenet elhelyezése
        - *Core*: áramköri mag
        - *Pad*: kivezetés
    - **Tápellátás**
        - szimulációval → statikus és dinamikus áramfelvétel becslése
        - átlagos és maximális fogyasztás ismeretében kell megtervezni
    - **Cellák elhelyezése**
    - **Huzalozás**
    - **Pad-ring elkészítése**
- post-layout szimuláció
    - fizikai tervezés után minden kapu kimeneti terhelése pontosan ismert
    - pontos késleltetési adatok keletkeznek → újra ellenőrizhető a terv
    - logikai vagy RTL újratervezhető ha nem elég jó

# Félvezető IP

- IP core vagy IP block → újrafelhasználható egység
- blokk felhasználásáért licenszdíjat kell fizetni

## Soft IP Core

- szintetizálható RTL leírás
    - titkosított forrás, vagy generikkus netlista
- tetszőleges technológiára szintetizálható
- nehézség: olyannak kell a fizikai tervezést és optimalizálást megcsinálni, aki nem ismeri pontosan a belsejét
    - nem lehet garantálni a méretet, késleltetést, fogyasztást

## Hard IP Core

- fizikai tervezés végeredménye → azaz layout
- adott félvezető gyártó adott technológiájához kötődik
    - méret, késleltetés és fogyasztás garantált!