



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

Digitális technika (VIMIAA03)

8. gyakorlat és laboratórium

Raikovich Tamás
BME MIT

GCD (Inko) számítása

- Egy számpár legnagyobb közös osztóját keressük
- $\text{GCD}(a,b)$ Euklideszi algoritmus \rightarrow maradékos osztás
 - $a = b * q_1 + r_1,$ $r_1 = a \% b$
 - $b = r_1 * q_2 + r_2,$ $r_2 = b \% r_1$
 - $r_1 = r_2 * q_3 + r_3$ $r_3 = r_1 \% r_2$
 -, ahol $|b| > r_1 > r_2 \dots >= 0$ ahol % a Verilog mod op.
 - A GCD az utolsó nem nulla maradék
 - Jó algoritmus, viszonylag gyorsan konvergál
- **DE: A Verilog HDL % operátor nem szintetizálható**
 - Tervezzünk egy osztó modult? Lehet, de nem könnyű.

GCD (Inko) számítása

- **Egyszerűsítsük az algoritmust szintetizálható műveletre**
 - $\text{GCD}(a,b) = \text{GCD}(a-b, b)$, ha $a > b$ Művelet: $a-b \rightarrow a$
 - $\text{GCD}(a,b) = \text{GCD}(a, b-a)$, ha $b > a$ Művelet: $b-a \rightarrow b$
 - Leállás adott lépés után, ha $a_0 = b_0$, ez a $\text{GCD}(a,b)$
- **Ez már szintetizálható, egyszerűen tervezhető, de több iterációt igényel, a végrehajtás hosszabb ideig tart**
 - Különösen relatív prímeknél
- **A tervezést az adatstruktúra hálózat és az ehhez szükséges vezérlőegység szétválasztásával kezdjük**
 - Adatstruktúra: bemenet, kimenet, multifunkciós regiszterek, kivonó egység, komparátor ($>$, $=$, $<$)

GCD (Inko) számítása

Több lehetséges adatstruktúra verzió (és ennek megfelelően algoritmus végrehajtás lehetséges)

- **1. verzió:** Ha $a > b$ nem teljesül, írjuk át egymásba a regiszterek tartalmát, ezután már $a > b$ és kivonás után a különbséget írjuk az a regiszterbe. Amikor $a = b$, leállítás.
- **2. verzió:** Ha $a > b$, elvégezzük a kivonást és a különbséget írjuk az a regiszterbe. Ha $a < b$, felcseréljük az operandusokat a kivonó bemenetén (**RSUB**), és a különbséget a b regiszterbe írjuk. Amikor $a = b$, leállítás.
- **3. verzió:** Két azonos felépítésű egységet tervezünk, mindkettő képes minden ütemben kivonást végezni a saját tartalma és a másik regiszter tartalma között. Csak azt az eredményt tároljuk, ahol a nagyobb operandus volt. Amikor $a = b$, leállítás. Ezt valósítjuk meg.

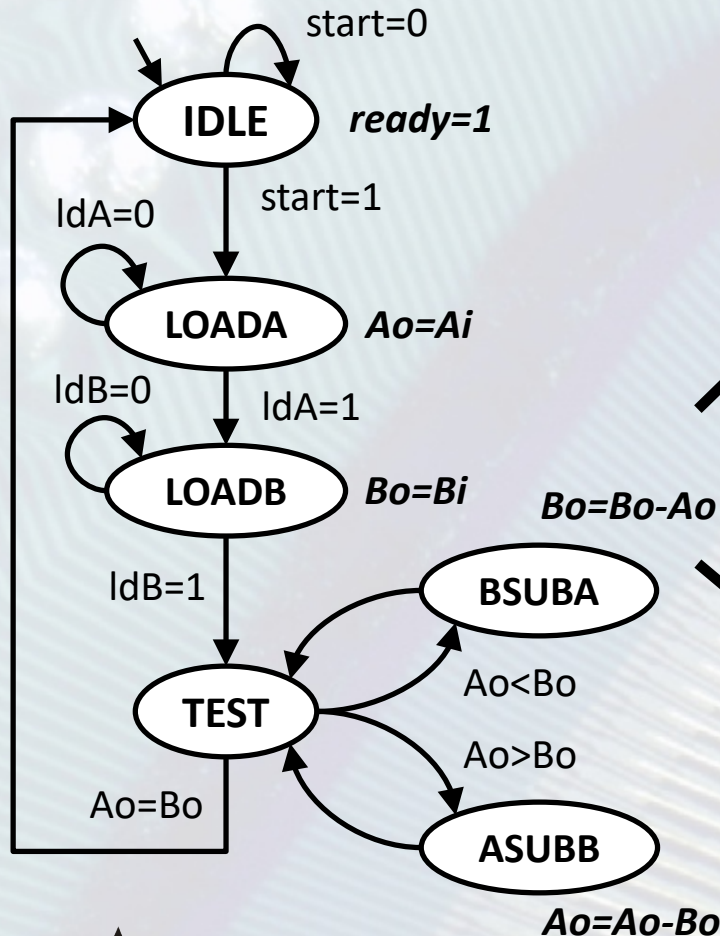
GCD (Inko) számítása

A GCD számító egység működésének specifikációja

- Bekapcsolás vagy RST után alapállapot, a kimenet 0
- Indítás a start pulzusra. Működési lépések:
 - Operandusok betöltése az IdA és IdB pulzusokra
 - Ezalatt a kimeneten megjelenik a bemenet aktuális értéke
 - Operandusok összehasonlítása
 - Amíg $a \neq b$, a szükséges művelet végrehajtása, iterációban, a kimeneten a regiszterek tartalma elérhető
- Ha $a = b$, akkor visszatérünk az alapállapotba, ahol kiadjuk a ready jelzést és leállunk

GCD (Inko) számítása

Magasszintű állapotgép



Interfész és belső regiszterek

- Bemenetek: start, IdA, IdB, Ai, Bi
- Kimenetek: ready, Ao, Bo
- Regiszterek: Ao, Bo

Adatstruktúra

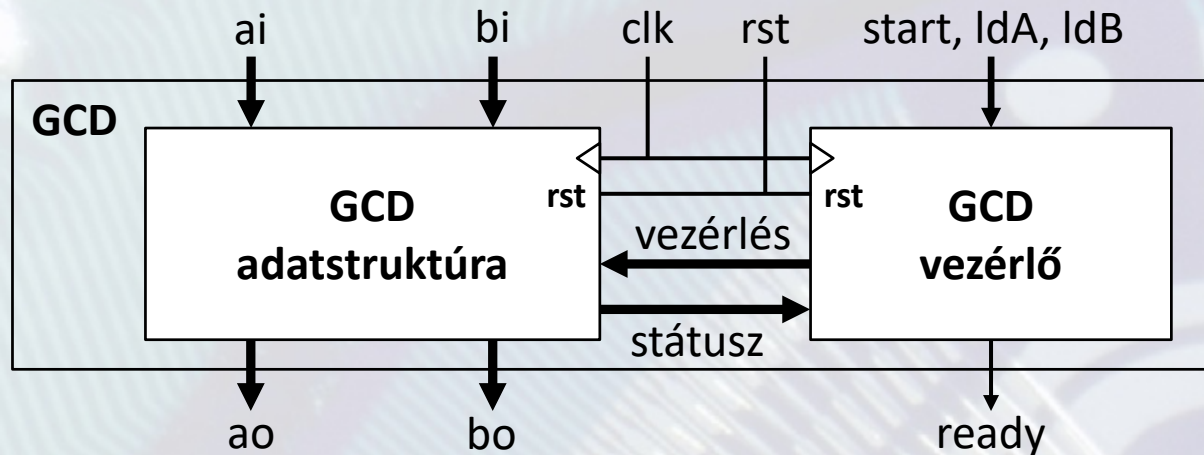
- Magasszintű műveletek megvalósítása
- Vezérlő jelek (\leftarrow FSM)
- Feltétel/státusz jelek (\rightarrow FSM)

Vezérlő

- Véges automata (FSM)
- Bitszintű műveletek
- Külső parancs és státusz jelek

GCD (Inko) számítása

- A teljes egység blokkdiagramja



- Vezérlő jelek:

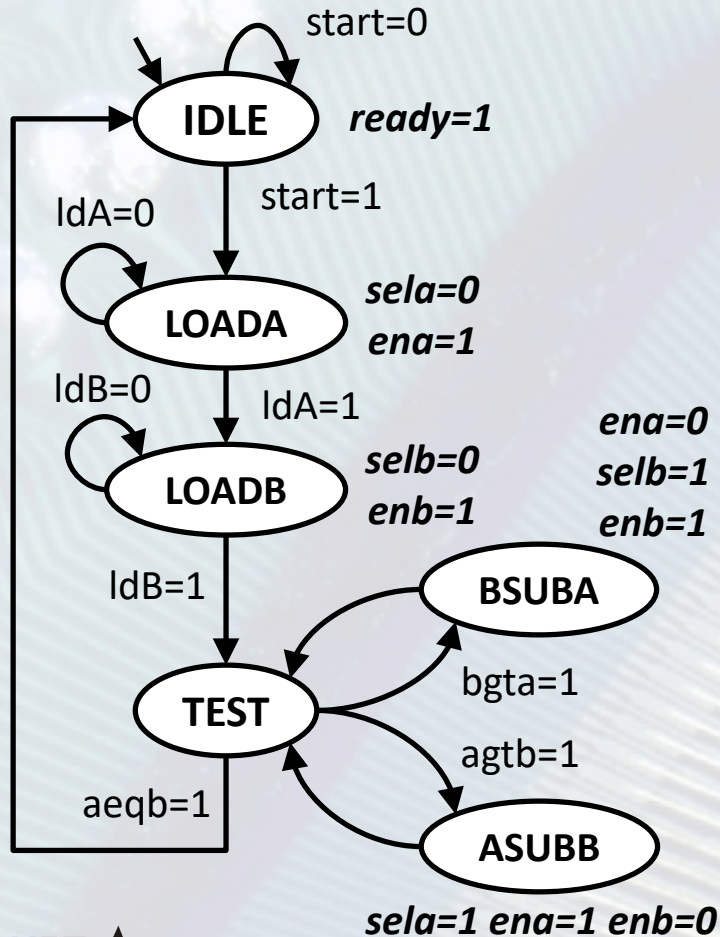
- Operandus regiszterek engedélyezése: ***ena, enb***
- Operandus regiszterek bemenet választás: ***sela, selb***

- Státusz jelek a regiszterek tartalma alapján:

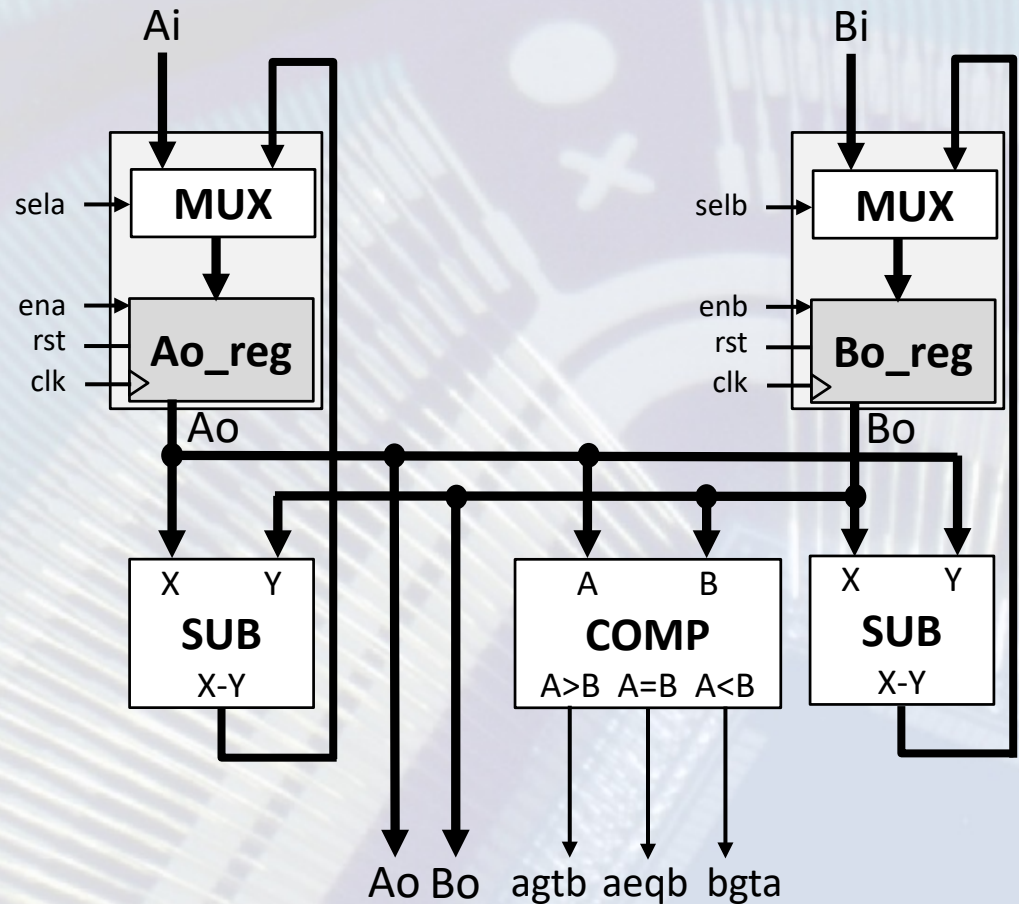
- ***agtb, bgta, aeqb***

GCD (Inko) számítása

A vezérlő állapotdiagramja



Az adatstruktúra egységei



GCD (Inko) számítása

Kérdések:

- Szükséges-e a komparátor mindhárom kimenete?
- Szükséges-e a komparátor? Tudja-e esetleg jelezni más egység is az $A < B$ és $A > B$ relációkat?

GCD (Inko) számítása

- Nyissuk meg a kiindulási projekt vázat az ISE-ben
- Interfészek
 - Rendszerórajel a letöltőkábelről: *clk*
 - Alapállapotba állítás a reset gombbal: *rstbt*
 - Bemeneti adatok: *sw[7:0]* → *Ai, Bi*
 - Vezérlés: *bt[0]* → *start*, *bt[1]* → *ldA*, *bt[2]* → *ldB*
 - Státusz: *ready* → *ld[0]*, *FSM állapot* → *ld[7:5]*
 - Kijelzés: *clk16M*, *seg_n[7:0]*, *dig_n[3:0]*, *col_n[4:0]*
 - A kijelző vezérlést a projekt váz tartalmazza
- Egészítsük ki a forrásfájlok hiányzó részeit
- Ellenőrizzük az elkészült rendszert szimulációval (opcionális)
 - A tesztkörnyezetet a projekt váz tartalmazza
- Próbáljuk ki a működést az FPGA kártyán
 - A *clk* órajelet a BitBang I/O funkcióval állítsuk elő

GCD (Inko) számítása

Néhány bemeneti adat a várt eredménnyel

- Milyen formátumban kell megadni a bemeneti adatokat?

Ai operandus	Bi operandus	Inko(Ai, Bi)
35 = 5·7 (0x23, 0010_0011)	63=3 ² ·7 (0x3f, 0011_1111)	7 (0x07, 0000_0111)
66 = 2·3·11 (0x42, 0100_0010)	84 = 2 ² ·3·7 (0x54, 0101_0100)	6 (0x06, 0000_0110)
84 = 2 ² ·3·7 (0x54, 0101_0100)	70 = 2·5·7 (0x46, 0100_0110)	14 (0x0e, 0000_1110)
65 = 5·13 (0x41, 0100_0001)	21 = 3·7 (0x15, 0001_0101)	1 (0x01, 0000_0001)

10-es számrendszer használata (opcionális)

- A bemenetek bináris (hexadecimális) számrendszerben történő megadása nem túl kényelmes
- Tudnánk-e használni a decimális számrendszert is?
 - Ha igen, akkor mit kell módosítani ehhez?

10-es számrendszer használata (opcionális)

- A kártya felépítése a 2 digités BCD bemeneti adatok (0 – 99) használatát teszi lehetővé
- A regiszterek és a komparátor BCD adatokkal is megfelelően működnek
- Egyedül a kivonással van probléma, ez esetben a normál bináris kivonó nem működik, BCD kivonó szükséges (a működését most nem részletezzük)
 - Adjuk hozzá a projekthez az előre elkészített *bcd_sub_4bit* és *bcd_sub_8bit* modulokat
 - Cseréljük le a bináris kivonót a *bcd_sub_8bit*-re
 - Próbáljuk ki az FPGA kártyán a módosítást