Példa feladatok a kisZH4-hez

Digitális technika (VIMIAA02)

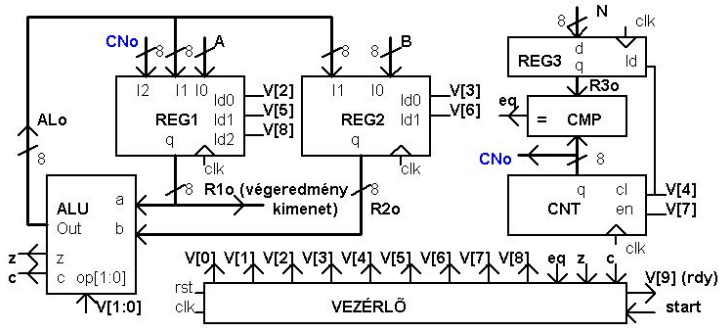
A példa kisZH feladatok célja, hogy a kisZH-ra történő felkészülés során segítséget nyújtson pár tipikus feladat ismertetésével, valamint hogy a hallgatók számára egyenlő esélyeket biztosítson azáltal, hogy a példa kisZH feladatokat a legkorábbi labor kurzus időpontja előtt pár nappal közzétesszük. (A kisZH-n nem pontosan ilyen feladatok lesznek, de némileg hasonlóak.)

Az alábbiakban ezúttal csak egy, de egy nagyobb feladat szerepel. A kisZH-n lévő feladat bevezető szövegezése, a benne szereplő adatstruktúra, az ALU vezérlési funkcióit leíró táblázat megegyezik a példa kisZH-ban szereplőkkel. Ugyanakkor az elvégzendő művelet különbözhet!

Komplex feladat: Adatstruktúra – Vezérlő

A modul általános leírása

Funkcionális blokkvázlatával (adatstruktúra + vezérlő) adott egy a bemenő adatokkal többféle művelet elvégzésére alkalmas számító modul. Az adatstruktúra **3 db 8 bites adat bemenettel** rendelkezik: **A, B, N** (előjel nélküli számok). Az aritmetikai logikai egység **(ALU**) 4 műveletet tud elvégezni az **a (R1o)** és **b (R2o)** operandusok között, az alább megadott táblázat szerint. Az **elvégzendő művelet** az ALU **op[1:0]** bementén állítható be. **Shiftelés esetén a belépő bit 0**. A **művelet eredményét** az Out **(ALo)** kimenet adja. Az ALU **z** kimenet 1 értéke jelzi, ha a **művelet eredménye 0**. A **c** kimenet **összeadás esetén a túlcsordulást, kivonás esetén a negatív eredményt, shiftelés esetén a kilépő bit** értékét jelzi. A REG1 (**R1o**) 3 forrásból, a REG2 (**R2o**) 2 forrásból tölthető **multifunkciós regiszter**. Ezek **az órajel felfutó élére az ld nevébe kódolt sorszámú In bemenet értékét töltik be** (pl. ld0 In0-at). REG3 tölthető (ld) regiszter. A CNT (**CNo**) egy törölhető (cl), engedélyezhető (en) **felfele számláló (CNo)**. Ehhez kapcsolódik a **CMP** komparátor, melynek másik adatát a Reg3 **(R3o)** tölthető (ld) regiszter adja, **eq=(R3o == CNo)**. **Az adatstruktúra kimenete az R1o, egy elvégzett számítási feladat végeredményét itt kell megjelentetni**. A vezérlőt az egy órajel hosszú **start** jel indítja. A vezérlő az adatstruktúrát a **V[8:0] vezérlő jelekkel** működteti. A működése közben figyelni képes az adatstruktúrából jövő z , c és eq feltétel jeleket. Egy számítási feladat elkészültét 1 órajel hosszú **rdy** (V[9]) jellel kell jeleznie. Az eredménynek meg kell maradnia a következő start jelig.



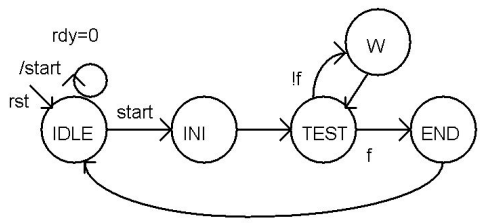
ALU funkció vezérlés: op[1:0]	ALU out (Alo)	z =1, ha	c = 1, ha
00	a + b	Alo==0	átvitel van
01	a – b	Alo==0	a–b < 0
10	a << 1	Alo==0	a[7]
11	b – a	Alo==0	b–a < 0

A megvalósítandó művelet

A számoló egységgel a bemeneti adatokon a **4A-B** műveletet kell elvégezni. Feltételezzük, hogy a bemenő adatok megadásánál biztosított, hogy nem lesz túlcsordulás.)
A képletet átalakítjuk az adatstruktúrán is elvégezhető műveletekké: **(A << 2) – B**
Az **A** 2-szeri balra shiftelséhez felhasználjuk az adatstruktúra számlálóból és komparátorból álló részét is.

Feladatok

a. Felrajzoltuk a feladat megoldására alkalmas Moore típusú HLSM gráfot. Rendelje a megadott műveleteket a gráf felsorolt állapotaihoz!
R1o=A, R2o=B, R3o=2, CNo=0, R1o=R1o<<1, R1o=R1o-R2o, CNo=CNo+1, rdy=1

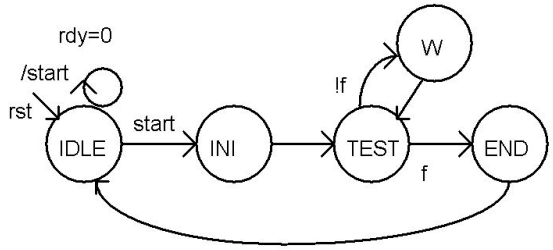


INI:
W:
END:

b. Válassza ki, a gráfban jelölt f feltétel bitet a felsoroltak közül: c,!c, z, !z, eq, !eq

f = …….

c. A fenti HLSM állapotgráf állapotaihoz rendelt műveletek alapján adja meg a vezérlő FSM Moore jellegű állapotgráfjának megadott állapotaiban kiadandó V[8:0] vezérlőjelek értékét 9 bites vektorként! (V[9] értékét megadjuk, az csak az END állapotban 1 értékű.)



INI: V[8:0] = ……

W: V[8:0] = ……

END: V[8:0] = ……