

NÉV:.....**MINTA ZH2**..... Neptun kód:..... gyak/lab kurzus:

A feladatokat önállóan, meg nem engedett segédeszközök használata nélkül oldottam meg:

Olvasható aláírás:.....

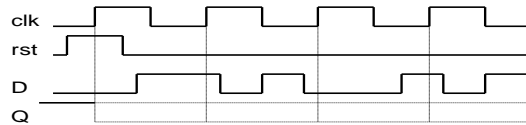
Kedves Kolléga! **A kitöltést a dátum, név és aláírás rovatokkal kezdje!** Az alábbi kérdésekre a válaszokat - ahol lehet - mindig a feladatlapon oldja meg! A feladatok megoldása során a részletes kidolgozást nagyfeladatonként külön papíron végezze, (egyértelműen jelölje, hogy melyik lap melyik feladathoz tartozik, a papírra már a kezdetkor írja rá a nevét és Neptun kódját) és ezeket a papírokat is adja be a dolgozatával! A kérdésekre a táblázatok vagy a pontozott vonalak értelemszerű kitöltésével válaszoljon, hacsak külön másként nem kérjük. **Mindenütt a legegyszerűbb megoldás éri a legtöbb pontot.** Ha több a feladat, mint a pont, a feladat pontszámát csökkentjük minden hiba esetén (azonban negatív pontot nem adunk). Jó munkát!

E: (25p)
 F1: (15p)
 F2: (25p)
 Σ : (65p)
 IMSC: (7p)

Ellenőrző kérdések (25p)

E1. (4p) a. Egy felfutó él érzékeny szinkron törölhető (rst) D flip-flop az alábbi jeleket kapja. Rajzolja le a Q kimenet idődiagramját! **b.** Adja meg egy törölő (rst) bemenettel rendelkező D flip-flop Verilog viselkedési leírását! Elkezdtük, folytassa!

```
always@(.....)
if(.....)
.....
```



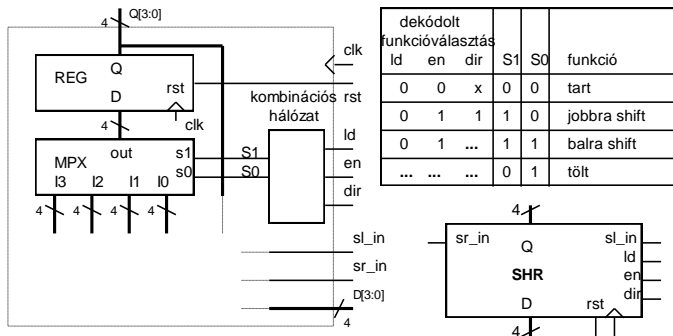
E2. (3p) Adja meg egy 2 db 8 bites forrásból betölteni képes **multifunkciós regiszter** Verilog leírását! **Ha ld0=1, q-ba töltse be In0-at, ha ld1=1, q-ba töltse be In1-et!** Az ld0 legyen nagyobb prioritású! (Most nem kell alaphelyzetbe állítás rst-re.) Elkezdtük, folytassa!

A hiányzó változó definícióktól eltekintünk.

```
always@(.....)
if(.....)
.....
```

E3. (4p) Adott egy 2 irányú shiftregiszter funkcionális blokkvázlata és működési táblázata.

a. Adja meg, bitsorrend helyesen, hogy milyen jelek kapcsolódnak az MPX multiplexer egyes bemeleteire! Ügyeljen arra, hogy a rajzon látható jel neveket használja!



I0:.....
 I1:.....
 I2:.....
 I3:.....

b. Töltse ki a táblázat hiányzó részeit, ha a prioritás sorrendje a legerősebb jellel kezdve a következő: betöltés (ld), engedélyezés (en)!

E4. (5p) Adja meg egy 4 bites 12-es modulusú fel le számláló Verilog kódját! A számláló ld-ra betölti d[3:0]-at, en=1 esetén számol, dir=1 esetén felfele, dir=0 esetén lefele, egyébként nem változik a q kimenete. A vezérlőjelek prioritása a legerősebb jellel kezdve a következő: ld, en. A jelek definiálásától most eltekintünk.

```
..... tc = .....
always@(.....)
if(.....) q <= .....
else if(.....)
    if(.....)
        if(.....) q <=.....
        .....
```

E5. (3p) Adott 3 db számláló modul példányosítva, azonban az összekötésük hiányzik. Az első 7-es modulusú, a második 3-os modulusú, a harmadik 5-es modulusú. A számláló kimenetek definiálásától eltekintünk.

a. Kaszkádosítsa a felsorolás szerinti sorrendben őket megadott wire típusú jelek megfelelő felhasználásával úgy, hogy a teljes számláncot a külső *EN* jel engedélyezze!

wire EN, OUT;
wire [1:0] tc_i;

cnt7 cnt7_1(.clk(.....), .rst(.....), .en(.....), .q(q7), .tc(.....));

cnt3 cnt3_1(.clk(.....), .rst(.....), .en(.....), .q(q3), .tc(.....));

cnt5 cnt5_1(.clk(.....), .rst(.....), .en(.....), .q(q5), .tc(OUT));

b. Hány órajelenként jelenik meg impulzus az *OUT* kimeneten?

E6. (6p) Mely állítások igazak és melyek hamisak? Jelölje **+ -al az igaz, -al a hamis** állításokat!

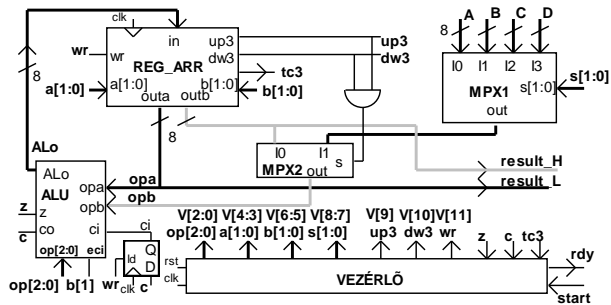
1.	Egy sorrendi hálózat (FSM) kódolt állapotábrája tartalmazza a next_state logika és a kimeneti logika igazságtábláit.	
2.	Ha egy sorrendi hálózat kimenetét az állapotregiszter valamely bitjei adják, akkor az biztosan Moore modell szerint működik.	
3.	A FIFO-ba bármikor lehet adatot írni.	
4.	Egy FSM vezérlő állapotgráfiájába írt állapotátmeneti feltétel lehet A == B.	
5.	Az ASM (algoritmikus állapotvezérlő) leírás egyetlen feltétel doboza csak 2-felé ágazást tesz lehetővé.	
6.	3 regiszter című processzor architektúra esetén egy művelet eredménye nem rontja el szükségyszerűen az egyik operandust.	

F1. (15p) Adott egy FSM az alábbi *kódolt állapotgráfiával*. A kimenete az állapotkód és *z*. Végezze el az alábbi feladatokat! **Az állapot kódok helyett mindenütt állapot neveket használjon!**

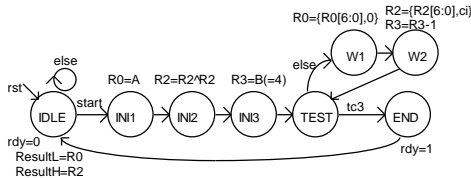
<p>bemenet: x[1:0] kimenet: s[2:0], z</p> <p>a. (3p) Adja meg az állapotregiszter Verilog leírását! //A konstans és változó deklarációk localparam [2:0] A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011, E = 3'b100; reg [2:0] s, next_s; wire clk, rst; wire [1:0] x; //Állapotregiszter always @ (.....) if (rst) s <=; else s <=</p>	<p>b. (9p) Következő állapot logika //Next_state logika always @ (.....) case (s) A: case(.....) 2'b01: next_s <=; 2'b10: next_s <=; 2'b11: next_s <=; default: next_s <=; endcase B: if(x == 2'b00) next_s <=; else next_s <= C; C: if(.....) next_s <=; else next_s <=; D: if(.....) next_s <=; else next_s <=; E: if(.....) next_s <=; else next_s <=; default: next_s <=; endcase</p>
<p>c. (1p) Milyen modell szerint működik? (Húzza alá a megfelelőt) Mealy Moore</p> <p>d. (2p) Adja meg a z kimenet logikai függvényét Verilog-ban! Ha csak az s-t, az állapot neveket, az == operátort és műveletet használhatja: assign z = A legegyszerűbb Boole algebrai alakban: assign z =</p>	

F2. (25p) Funkcionális blokkvázlatával (adatstruktúra + vezérlő) adott egy a bemenő adatokkal többféle művelet elvégzésére alkalmas számító modul. Az adatstruktúra **4 db 8 bites külső adat bemenettel** rendelkezik: **A, B, C, D**. Az aritmetikai logikai egység (ALU) 8 műveletet tud elvégezni az *opa* és *opb* operandusokkal az alább megadott táblázat szerint. Az **elvégezendő művelet** az ALU *op[2:0]* bemenetén állítható be. **Shiftelés esetén a belépő bit 0, ha eci = 0, ci, ha eci = 1. Összeadás/kivonás esetén a ci-t csak akkor veszi figyelembe, ha eci = 1.** A **művelet eredményét** az *ALo* kimenet adja. Az ALU *z* kimenet 1 értéke jelzi, ha a **művelet eredménye 0**. A *c* kimenet **összeadás esetén a túlszordulást, kivonás esetén a negatív eredményt, shiftelés esetén a kilépő bit értékét** jelzi. A *REG_ARR* egy **4 regisztert tartalmazó regiszter tömb**. Ennek *outa* kimenetén az *a[1:0]*-val kiválasztott sorszámú regiszter tartalma jelenik meg, ha *a[1:0]=i*, akkor *Ri*. Az adat beírását *wr=1* engedélyezi és az ALU-ból az *in* bementére jövő adat (*ALo*) az *a[1:0]*-val kiválasztott regiszterbe íródik az órajel felfutó élére. A *wr* jel ezen kívül az ALU *c* kimenetét beírja az ALU *ci* bementére kapcsolt 1 bites

regiszterbe. A regiszter tömb **outb** kimenetén a **b[1:0]**-val kiválasztott regiszter tartalma jelenik meg, ha **b[1:0]=i**, akkor **Ri**. Az ALU **eci** bemenetére **b[1]** van kötve, ezért a **ci** bemenetét csak akkor veszi figyelembe az ALU, ha **opb-n R2** vagy **R3** van kiválasztva. Az ALU **opb** bemenetére kerülő adat az MPX2 multiplexerről jön, mely alapesetben (**dw3,up3 != 11**) a regiszter **outb** kimenetét választja ki. Ha **dw3,up3=11**, akkor az **s[1:0]**-al kiválasztott külső adat lesz az **opb**. A regisztertömb **R3** regisztere speciális, mert fel/le számlálóként is funkcionál. Normál regiszterként írható (**a[1:0]=2'b11** és **wr = 1** esetén beíródik **R3**-ba az **Alo**-ról jövő érték) és normál regiszterként olvasható. Azonban, ha éppen nem írják, akkor **dw3,up3 = 1,0** esetén lefele számol (**R3=R3-1**), **dw3,up3 = 0,1** esetén pedig felfele számol (**R3=R3+1**), ha megjön a **clk** aktív éle. Az írás erősebb, mint a számlálás engedélyezés. Az **R3** regiszter fel vagy le számláltatásával egyidőben bármely más regiszterbe lehetséges írni. A **tc3** kimenet akkor jelez, ha **R3==0**. A vezérlőt egy kívülről jövő, 1 órajel hosszú **start** parancs indítja. A vezérlő az adatstruktúrát a **V[11:0]** vezérlő jelekkel működteti. A működése közben figyelni képes az adatstruktúrából jövő **z**, **c** és **tc3** feltétel jeleket. Egy számítási feladat elkészültét 1 órajel hosszú **rdy** státusz jellel kell jeleznie. Az eredménynek meg kell maradnia a következő **start** jelig. Az aktuális művelet szempontjából érdektelen vezérlő jelek értékét **0**-nak kell választani. Egy maximum 8 bites végeredményt a regiszter tömb **outa** kimenetén (**result_L**) kell megjeleníteni. Ha a végeredmény megjelenítéséhez 8-nál több bit kell, akkor a többi bitet az **outb** kimeneten (**result_H**) kell megjeleníteni.



a. (10p) Az alábbi Moore jellegű HLSM állapot diagram által leírt feladatot a fenti adatstruktúrával kell megvalósítani. (**R0**-at szorozza 16-al, eredmény:**R2R0**-



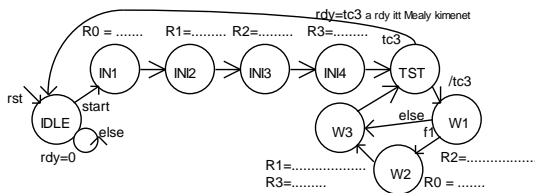
ban)

Adja meg a **vezérlő állapotgráfiájának megadott állapotokban kiadandó vezérlőjeleket** a **V[11:0]** bitek értékének (**0, 1**) alábbi táblázatba való beírásával! (Segítségképp néhány értéket megadtunk. A **rdy**-t most nem kérjük.)

b. (1p) Milyen értékű lesz **R2** az **INI3** állapotban?

c. (7p) A számító modullal két 4 bites szám szorzatát kell kiszámítani, **ResultL = A*B**.

A megvalósítást: **Kezdéskor a szorzandót R1, a szorzót, R2 regiszterbe tesszük, az eredmény regisztert 0-ázzuk. A szorzást ciklusba szervezzük.**



elvégzéséhez szükséges műveleteket az (egy kimenet kivételével) Moore jellegű HLSM állapotaihoz. Az alább felsorolt műveletekből válasszon (szükségtelenek is vannak közöttük). Írja az elvégzendő művelet(ek) sorszámát a megfelelő állapot neve mellé! Nem biztos, hogy van elvégzendő művelet, ilyenkor írjon x-et! (A HLSM gráf segítő információkat tartalmaz. A gráf pontozott részeit nem kell kitölteni, de az is segítő információ.) A többlet és hiányzó művelet sorszámokért pontlevonás jár! Egy állapothoz beírtuk, folytassa a többivel!

- 1: R0 = {R0[6:0],0} 2: R0 = R0^R0 3: R0 = {0, R0[7:1]}
 - 4: R0 = R0+R1 5: R1 = A 6: R1 = B 7: R1 = C
 - 8: R1 = R0+R1 9: R1 = {R1[6:0],0} 10: R2 = B
 - 11: R2 = C 12: R2 = {0, R2[7:1]} 13: R3 = R3+1
 - 14: R3 = R3-1, 15: R3 = R3+R1 16: R3=C 17: rdy = tc3
- IDLE:..... INI1: INI2: INI3:
- INI4: TST: ...17..... W1:..... W2:
- W3:.....

ALU funkció vezérlé s: op[2:0]	ALU out (Alo)	z=1, ha	c=1, ha
000	opa + opb + ci&eci	Alo==0	átvitel van
001	opa - opb - ci&eci	Alo==0	a-b < 0
010	{opa[6:0], ci&eci}	Alo==0	opa[7]
011	{ci&eci, opa[7:1]}	Alo==0	opa[0]
100	opa & opb	Alo==0	Soha (c=0)
101	opa opb	Alo==0	Soha (c=0)
110	opa ^ opb	Alo==0	Soha (c=0)
111	opb	Alo==0	Soha (c=0)

	wr	dw3	up3	s[1:0]	b[1:0]	a[1:0]	op[2:0]
V[11:0]	11	10	9	8 7	6 5	4 3	2 1 0
IDLE				0 0			0 0 0
INI1					0 0		
INI2					0 0		
INI3				0 0	0 0	0 0	0 0 0
TEST				0 0	0 0	0 0	0 0 0
W1					0		
W2					0		
END				0 0	0 0	0 0	0 0 0

Ciklusszámlálóként a speciálisan számlálóként is használható **R3** regisztert használjuk (kezdőértéke **C**). A szorzást **LSB**-vel kezdjük. A szorzó **LSB**-jét megjelenítjük **c**-ben. Ha **c=1**, akkor az eredményként használt regiszterhez hozzáadjuk a szorzandó aktuális 2 hatványát, ha **c=0**, nem adjuk hozzá. Mindkét eset után előállítjuk a szorzó következő 2 hatványát és ezzel egyidőben csökkentjük a ciklusszámlálót. Ezután visszamegyünk a ciklus elejére, ahol ellenőrizzük a ciklus számlálót. Rendelje a számítás

d. (1p) Adja meg a kért feltételeket a blokkvázlaton szereplő feltétel jel(ek) felhasználásával. (A parancs és feltétel bitek negáltját is fel lehet használni.)

f1:

e. (1p) Adja meg az adatstruktúra **C** bemenetének (ciklus számláló kezdőértéke) értékét! (Az algoritmusban fel kell használni.)
C =

f. (5p) Adja meg az adatstruktúra REG_ARR regiszter tömbjének Verilog leírását a bevezetőben megadott információk alapján! Az R3 speciális számláló funkciójának megvalósításától most eltekintünk. A leírást elkezdtük, egészítse ki a hiányzó részekkel! (Az **arr** 4 db 8 bit szélességű tároló regisztert tartalmazó tömb.)

<pre>module REG_ARR (input clk, wr, input [1:0] a, b, input [7:0] in; output reg [7:0] outa, outb);</pre>	
<pre>// beírás megvalósítása reg [7:0] arr[3:0]; always@(.....) if(.....)<=</pre>	<pre>always@(.....) // opa elő- // állítása</pre>
<pre>..... <=</pre>	<pre>always@(.....) // opb elő- // állítása <=</pre>
<pre>endmodule</pre>	

IMSC1 (3p)

Tervezzen *programozható modulusú 4 bites felfele számlálót!* A számláló modulusa 2-től 16-ig legyen változtatható a 4 bites d bemenetére adott szám függvényében. Adja meg az egység Verilog kódját! A számláló regiszter neve legyen q.

IMSC2. (4p)

Készítsen az F2 feladat adatstruktúrájához egy olyan HLSM diagram részletet, amely **a regisztertömb R0 és R1 regiszterének értékét cseréli fel, a legkevesebb művelettel.** Milyen műveletet kell beállítani az op[2:0]-on és mi legyen ez alatt dw3, up3 értéke?

Maximális pontszám: 65 pont (IMSC: 7p) Rendelkezésre álló idő: 100 perc