20…..év …hó …nap

NÉV:.....neptun kód:..…… 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.*** Jó munkát!

E:	
F1:	
F2:	
F3:	
$\sum$	:

Ellenőrző kérdések (25p)

E1. Konvertálja az alábbi 4 bites 2-es komplemensű számot 8 bites 2-es komplemensűre! (1p)

1001 = ...**11111001**.....

E2. Írja fel a De' Morgan azonosságokat 2 változó esetére! (1p) ..**/(A.B)= /A +/B, /(A+B) = /A/B**.....

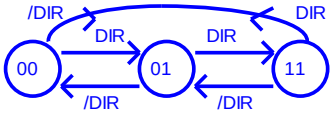
E3. Írja le az SOP alakból kiinduló kombinációs hálózat egyszerűsítés alapjául szolgáló azonosságot 2 változóval (A, B)! (1p)

.....**AB + /AB = A**.....

E4. Adja meg 2db 4 bites adat (input wire [3:0]I0, input wire [3:0] I1) multiplexerésére képes hálózat (wire [3:0] Y) Verilog leírását assign-al! (1p)

assign Y= ..**S ? I1 : I0;**.....

E5. Rajzolja le egy 3-as modulusú fel/le számláló számlálás üzemmódjára jellemző állapotgráfját! (2p)



E6. Adja meg egy törölhető, jobbra shiftelő 4 bites shiftregiszter Verilog leírását! A jobboldali belépő bit neve SIR (2p)

always@(**posedge clk**)
if(rst) q <= 4'b0000;
else q <= {SIR, q[3:1]};

E7. Adott egy 10-es modulusú számláló, cnt10(input clk, rst,en, ld, input [3:0] d, output tc, output [3:0] q). Törölhető (rst), engedélyezhető (en), tölthető (ld), végérték jelzéssel rendelkezik (tc). Készítsen egy példányából 6-os modulusú számlálót (0,1,2,3,4,5,0,...)! Adja meg a szükséges bekötéseket assign-al! (2p)

assign d = ...**4'h0**.....; assign ld =....**en & (q == 4'h5)**..**vagy** ... **en & (q == 4'h5) | load**....

E8. A 3 regiszteres processzor architektúra esetén hogyan néz ki egy összeadó utasítás mnemonikja? A megadását elkezdtük, egészítse ki a regiszeterek megadásával! (1p)

add**...R0, R1, R2.....**

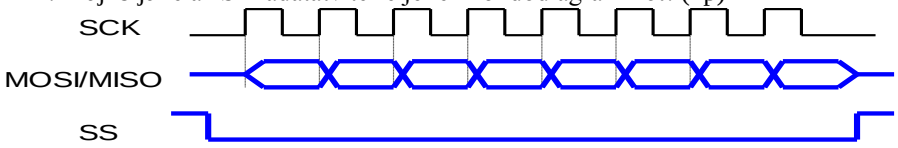
E9. Írja le az utasítás végrehajtás 3 fázisának elnevezését! (2p) .....**fetch, decode, execute**.....

E10. A MiniRisc processzor adatmozgató utasításai megváltoztatják-e a flag regiszter tartalmát? Húzza alá a megfelelő választ! (1p) igen nem

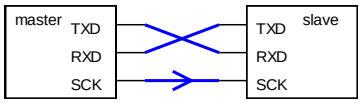
E11. Sorolja fel a LOAD/STORE architektúra 2 fontos tulajdonságát! (2p)

.....**adat regiszterekbe, műveletek regiszterek között és eredmény vissza egy regiszterbe**.....

E12. Rajzolja le az SPI adatátvitelre jellemző idődiagrammot! (2p)



E13. Rajzolja le, hogyan kell összekötni egymással egy master és egy slave USRT egységet (szintkonverter nélkül). Melyik adja az órajelet? (2p)



E14. Mely állítások igazak és melyek hamisak? Jelölje **+**-al az igaz, **-**-al a hamis állításokat! (5p)

1.	Az always blokkban nem adható meg kombinációs hálózat leírás.	**-**
2.	A Barrel shiftert megvalósítható kombinációs hálózattal.	**+**
3.	A kaszkádosított számlálók modulusai összeszorzódnak.	**+**
4.	A vektoros IT rendszerben minden interrupthoz ugyanaz a belépési cím tartozik.	**-**
5.	A DMA vezérlő a buszon slave és master is lehet.	**+**

Feladatok:

F1. (12p) Adott egy FSM-el megvalósított ismert funkcionális elem, az alábbi ***kódolt állapottáblával*** (a rubrikákban s[1:0]/z). Állapotok: A, B,C,D. Bemenetek: clk, rst (hatására az A-ba megy), x. Kimenet: z. Az állapotkódolást megadtuk.

kódolt állapottábla:

s1s0	x=0	x=1
A:00	00/0	01/0
B:01	01/0	10/0
C:10	10/0	11/0
D:11	11/1	00/1

Készítse el a Verilog leírását külön az ***állapotregiszter***, külön a ***next_state logika*** és külön a ***kimeneti logika*** megadásával! A leírást elkezdtük, fejezze be!

a. Milyen ismert funkcionális elemet valósít meg, mi az X és a Z szerepe? (3p)

funkcionális elem neve:...**bináris számláló**..... X szerepe:...**engedélyezés**.....

Z: szerepe:....**TC (végérték jelzés)**.....

b. Verilog leírás.

//A konstans és változó deklarációk, mindhárom egységhez tartozó //deklarációkat ide írja! (1p)
localparam [1:0] A = 2'b00, B =.**01**.. , C = ..**10**.., D = ..**11**....;
reg [1:0] s;
reg [1:0] next_state;
//**Állapotregiszter** (2p):
always@(**posedge clk**)
if(rst) s <= A;
else s <= next_state;

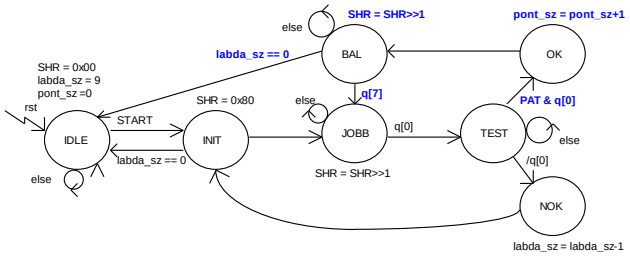
//**Next_state logika** (4p):
always@(*****)
case (s)
A: if(!x) s <=A;
else s <=B;
B: if(!x) s <=B;
else s <=C;
C: if(!x) s <=C;
else s <=D;
D: if(!x) s <=D;
else s <=A;
endcase

//**Kimeneti logika** (2p):
assign z = **s == D;**

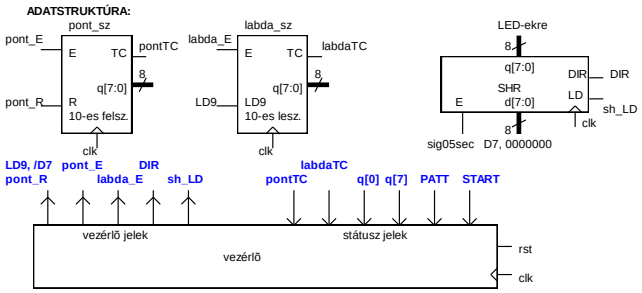
F2. (13p) A feladat egy labda pattogtató LED játék (LPJ) tervezése. Az LPJ a START nyomógomb megnyomására indul. 8db LED-ből álló sorban jobbra elindul egy LED 0.5s-onként lépve újabb pozícióba és a labda számláló 10-ről indulva dekrementálódik. A játékos feladata, hogy amikor a LED felgyullad az utolós pozícióban, nyomja meg a PAT gombot (se előbb, se később). Ha ez sikerült, akkor a pontszámláló növekszik 1-el és a LED vissza indul, majd a másik végén irányt vált, s a játékosnak újra vissza kell pattintania a labdát. Ha nem sikerül, akkor újabb labda indul bal oldalról és a labda számláló dekrementálódik. A játék akkor ér véget, ha a labda számláló eléri a 0-át. Ekkor a pontszámláló mutatja az elért pontot. Újabb játék a START gombbal indítható. A feladathoz rendelkezésre áll egy 8 bites tölthető 2 irányú shiftregiszter (DIR = 1 jobbra, DIR = 0 balra shiftel, LD = 1 tölt), 1/2 sec-onként érkező ***sig_felsec*** jel és egy néhány MHz-es órajel (clk). A feladat megoldását részfeladatokra bontottuk.

Az algoritmust megadjuk. Feltételezzük, hogy a shiftegiszter engedélyező jele 0.5sec-onként jövő, 1 clk órajelig tartó sig_05sec jel:
1. Bekapcsoláskor töröljük a pontszámlálót, 9-be írjuk a labdaszámlálót és töröljük a shiftregisztert.
2. START után 1000 0000-at írunk a shiftregiszterbe (indul egy új labda).
3. A shiftregisztert jobbra shiftelésre állítjuk.
4. Megvárjuk, míg a az 1-es a jobb szélső pozícióba ér.
5. Amíg az 1 a jobb szélső pozícióban van, leteszteljük, hogy a PAT meg van-e nyomva. Ha igen, megyünk 6.-ba. Ha az 1 eltűnik a jobb szélső pozícióból, megyünk 8.-ba
6. Növeljük a ponszámlálót, csökkentjük a labdaszámlálót
7. A shiftregisztert balra shiftelés állásba állítjuk (visszapattan a labda). Ha a labda számláló 0, akkor a kezdő állapotba (1.) megyünk (vége a játéknak), egyébként, ha az 1-es a jobb szélső előtti pozícióba ért (visszaért a labda), a 3.-nál folytatjuk (újra jön a labda).
8. Csökkentjük a labdaszámlálót és a 2.-nél folytatjuk.

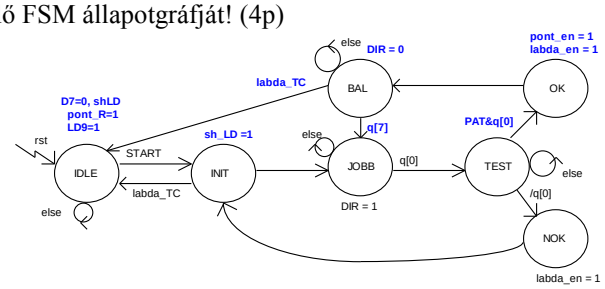
a. ***Egészítse ki az adatstruktúrát vezérlő HLSM Moore jellegű állapotdiagramját!*** A rajzolást elkezdtük, adja meg a hiányzó állapotátmeneteket és az elvégzendő magas szintű műveleteket **labda_sz**, **pont_sza**, **shr** értékadások az egyes állapotokban! Minden műveletet külön állapotban hajtunk végre. Az **labda_sz** végértékét jelölje **TL**-el. Ha egy állapotból az összes többi feltétel nem teljesülése esetén megy egy másikba, a feltételt ***else***-el jelöltük) (5p)



b. ***Rajzolja le a feladat adatstruktúrájának a blokkvázlatát***. Az adatstruktúra felrajzolását elkezdtük. Fejezze be az adatkapcsolatok, adatutak felrajzolását! Rajzolja be, hogy a vezérlő milyen kimeneti és bemeneti (feltétel) jelekkel kapcsolódik az adatstruktúrához! (LD9 hatására 9 töltődik a labda_sz-ba) (4p)



c. Egészítse ki a vezérlő FSM állapotgráfját! (4p)



F3. (15p) Egy MIniRISC buszra (A[7:0], Din[7:0], Dou[7:0], RD, WR, IRQ, clk, rst) illesztett párhuzamos bemeneti port a parancsregiszter EN bitjével engedélyezhető. A státusregiszterének RDY bitjében jelzi, adat érkezett. Ekkor az adatregiszterből kiolvasható az adat. A státusregiszter olvasása után a státus automatikusan törlődik. A periféria báziscíme 0xA0.
A programozói felülete a következő:

funkció	Cím	D7D6D5D4D3D2D1D0	olvasható/írható
Parancs regiszter	Báziscím +0	x x x x x x x EN	W
státus regiszter	báziscím + 0	x x x x x x x RDY	R
adat regiszter	báziscím + 1	D7D6D5D4D3D2D1D0	R

a. Tervezze meg Verilog nyelven a periféria parancs regiszterbe írást engedélyező parancs_wr, a státus regiszter és az adatregiszter olvasását engedélyező status_rd és data_rd jeleit! (5p)

```
parameter base_addr = 8'hA0;
        assign psel = (base_addr >> 1) == A[7:1];
        assign parancs_wr = psel & ~A[0] & WR;
        assign status_rd = psel & ~A[0] & RD;
        assign adat_rd = psel & A[0] & RD;
```

b. Írjon olyan program részletet, mely engedélyezi a perifériát, majd a státusát figyelve beolvas belőle 16 adatot és leteszi az adatokat a PERDAT memória kezdőcímtől kezdve. (5p)

```
DEF par   0xa0
DEF stat  0xA0
DEF adat  0xa1
DEF EN    0x01
DEF RDY 0x01

        DATA
PERDAT:
        DB 00

        CODE
start:
        mov r0, #EN
        mov PAR, r0             ;periféria engedélyezés
        mov r1, #16             ;cikluszámláló init
        mov r2, #PERDAT         ;adat pointer
stat_loop:
        mov r0, stat
        tst   r0, #RDY
        jz  stat_loop           ;adatra várakozás
        mov (r2), r0            ;adat a memóriába
        add r2, #1              ;adat pointer incr.
        sub r1, #1              ;ciklus sz. decr.
        jnz stat_loop           ;vissza, ha nincs kész
ready:
```

c. Írjon szubrutint, mely a PERDAT címen lévő adatok közül megkeresi az első olyat, mely meghívása előtt R0 regiszterbe tett adattal megegyezik. Ha talált ilyen adatot akkor a visszatérésekor a Z flag legyen egy és az adat címe az R1 regiszterben legyen. Ha nem talált ilyen adatot, a Z flag legyen 0. (5p)

```
keres:  mov r2, #15             ;cikluszámláló init (N-1)
        mov r1, #PERDAT         ;memória kezdőcím
loop:   mov (r3), r1
        cmp r3, r0              ;hasonlítás
        jz end_keres            ;találat esetén Z=1
        add r1, #1              ;pointer inkrementálás
        sub r2, #1              ciklus számláló csökkentés
        jnc loop                ;ha átfordul vége a ciklusnak, C=1, Z=0
end_keres:
        rts
```

Maximális pontszám: 60 pont
Rendelkezésre álló idő: 100 perc