NÉV:..... neptun kód:.……

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

Olvasható aláírás:.....……

Kedves Kolléga! ***A kitöltést a név és aláírás rovatokkal kezdje!*** Az alábbi kérdésekre a válaszokat - ahol lehet - mindig a feladatlapon adja meg! A feladatok megoldása során a részletes kidolgozást nagyfeladatonként, ha szükséges 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 (27p)

E1. Konvertálja az alábbi számokat a kért formátumra! (3p)

kiinduló formátum	konvertálandó szám	kért formátum	konvertált szám
6 bites 2-es komplemens	101101	8 bites 2-es komplemens	
decimális	1642	BCD (16 bites)	
Fixpontos decimális	3.25	8 bites fixpontos 2-es komplemens, 4 db 2-edes jegy	

E2. Adja meg, a Boole algebra disztributív tulajdonságának 2 féle alakját 2 változóval! (2p)

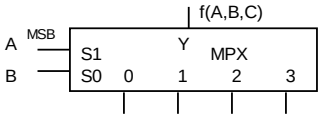
……..... .....

E3. (2p) Megadtuk egy ismert funkcionális elem Verilog leírásának egy jellemző részletét. Adja meg a funkcionális elem nevét! (2p)

wire [7:0] A,B; wire [2:0] O;

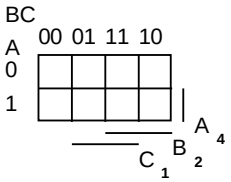
assign O = {A<B, A==B, A>B} ; neve:.....

E4. A felrajzolt multiplexerrel az alábbi 3 változós logikai függvényt kell megvalósítani. Kösse rá a multiplexer bemeneteire a megfelelő jeleket és konstansokat a következők közül: C, /C, 0, 1! (Írja oda a megfelelő jelneveket!) (2p)



f(A,B,C) = /A/B/C + /AB + A/BC

E5. Alább megadtunk egy F(A,B,C) logikai függvény Verilog leírását. Végezze el az alábbi feladatokat! (C az LSB)



assign F = ({A,B,C}>3'b001) & ({A,B,C}<3'b100) | ({A,B,C}>3'b101);

a. Töltse ki a függvény Karnaugh tábláját! (2p)

b. Adja meg a függvény egyszerűsítetlen SOP alakját Verilog logikai kifejezésként! (1p)

assign F =.....

c. Adja meg a függvény legegyszerűbb SOP alakját Verilog logikai kifejezésként! (1p)

assign F =.....

E6. a. Rajzolja le egy olyan logika blokkvázlatát tanult funkcionális elemekkel, amely a bemenetére érkező A[3:0], B[3:0] előjel nélküli 4 bites számok közül a nagyobbat teszi a 4 bites O[3:0] kimenetére! (2p)

b. Adja meg a fenti logika Verilog leírását! (2p) assign O = ...

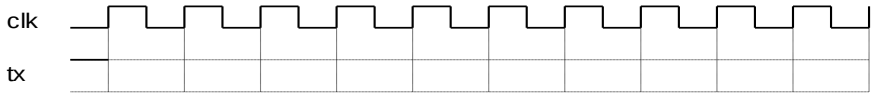
E7. Röviden írja le, mi történik egy szubrutin meghívásakor a MiniRISC CPU esetén! (2p)
1. ..... 2.....

E8. Melyik tanult kombinációs funkcionális elem Verilog leírását adtuk meg alább? (1p)

wire [3:0] In0, In1, Y; wire s;	assign Y = In0 & {4{~s}} | In1 & {4{s}};

funkcionális elem neve: .....

E9. Rajzolja le az USRT soros adatátvitel idődiagramját (start bit, 8 adat bit, 1 stop bit esetére)! (Egészítse ki az alábbi rajzot!) (2p)



E10. Mely állítások igazak és melyek hamisak? Jelölje ***+-al az igaz, --al a hamis*** állításokat! (5p)

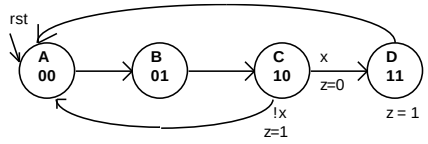
1.	Csak NOR kapukkal és a 0 logikai konstanssal minden logikai függvény megvalósítható.	
2.	Egy D flip-flopból egyetlen EXOR kapuval 1 bites engedélyezhető számláló készíthető.	
3.	Az aszinkron RAM-ok az /WR alacsony szintje alatt írják be az adatot.	
4.	A processzorhoz a perifériától érkező interrupt kérés mindig érvényre jut.	
5.	A MiniRISC processzor 3 regiszter címes architektúrával rendelkezik.	

Feladatok:

F1. (13p) Adott egy FSM az alábbi ***kódolt állapotgráfjával.*** Végezze el az alábbi feladatokat!

//A konstans és változó deklarációk
localparam [1:0] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;
reg [1:0] state;
reg [1:0] next_state;

a. Mealy vagy Moore modell szerint működik? (1p) ……



b. Adja meg az állapotregiszter Verilog leírását!
//Állapotregiszter (2p):
always@(……...)

if (rst) state <= ……

else state <= ……

c. //Next_state logika (6p):
always@(.....)
case (state)
A: next_state <=……..;

B: …….

C: if(x) ……...
.
else …….

D: ……..

default: ……;
endcase

d. Az FSM kimenete az állapotkód és a **z** változó. Adja meg a z kimenet Verilog leírását! Elkezdtük, folytassa! (2p)
wire z;

assign z =……

e. Az automata bemenetére x=0-át adunk folyamatosan. Milyen ismert funkcionális elemnek megfelelően viselkedik az FSM? Adja meg a nevét és tulajdonságait! (2p)

……

F2. (18p) A feladat egy páros reflexidő játék tervezése adatstruktúra-vezérlő felbontásban. A készülék nyomógombjai lenyomáskor 1 órajel hosszú impulzust adnak a rendszerórajellel szinkronban (nem kell megtervezni). A játékot a ***START*** nyomógombbal indítja a játékmester. Ennek hatására a készülék véletlenszerűen kigyújtja valamelyik játékos LED-jét ***(LD0*** vagy ***LD1***). (Rendelkezésre áll egy külső 1 bites jel (***RND***), melynek állapota a START impulzus megjelenése után véletlenszerű). Ugyanekkor elindul ***a másik játékoshoz tartozó*** 2 számjegyes pont számláló (BCD_cnt0 vagy BCD_cnt1) és 0.1sec-onként lép 1-et. (A léptetéshez adott egy külső 0.1 sec-onként 1 órajelig aktív ***sig_01s*** jel.) A játékosok feladata, hogy a saját LED-jük kigyulladásakor minél gyorsabban nyomják le a nyomógombjukat (PB0 ill. PB1), hogy a másik játékos pontszámlálója minél kevesebbet lépjen. Amint az aktuális játékos lenyomta a nyomógombját, leáll a másik ponszámlálója, elindul a sajátja és kigyullad a másik játékos LED-je. Most neki kell minél gyorsabban lenyomni a nyomógombját. ***A játék addig tart, amíg valamelyik játékos pont számlálója el nem éri a 99-et*** (ez a játékos nyer), ekkor ponszámlálók megállnak, LD0 és LD1 kialszik. Ezután a START újboli megnyomására kezdhető újra a játék. Az áramkör órajele egy néhány MHz-es clk jel (pontos értéke itt lényegtelen). A kijelzés megvalósításától az egyszerűsítés végett eltekintünk.

A feladat megoldását részfeladatokra bontottuk.

a. Tervezzen meg egy BCD_cnt, 10-es modulusú felfele számláló egységet! (Adja meg a Verilog leírását!) Legyen szinkron törölhető (rst), engedélyezhető (e)! A leírást alább elkezdtük, fejezze be! (5p)

b. Példányosítson az ***a***-pontbeli modulból kettőt-kettőt úgy, hogy azok ***páronként egy 2 dekádos BCD számlálót alkossanak*** (BCD_cnt0L ill. BCD_cnt1L a kisebb helyiértékű digit). Ezek a pont számlálók. A számlálókat törölje az ***rst*** és a vezérlő ***cnt_cl*** jele is, 0.1sec-onként lépjen (***sig_01s***), amelyiket a vezérlő engedélyezi a ***cnt_e0*** vagy ***cnt_e1*** jellel! (6p)

BCD_cnt BCD_cnt0L (.clk(.……), rst(….……), .e(……&…….....), .tc(tc0L), .q(q0L));

BCD_cnt BCD_cnt0H (.clk(…….), rst(…….), .e(.……), .tc(tc0H), .q(q0H));

BCD_cnt BCD_cnt1L (.clk(…….), rst(…….), .e(……&……....), .tc(tc1L), .q(q1L));

BCD_cnt BCD_cnt1H (.clk(…….), rst(…….), .e(……), .tc(tc1H), .q(q1H));

a. BCD számlálóVerilog leírása:

module BCD_cnt (input clk, rst, e, output tc, output reg [3:0] q);

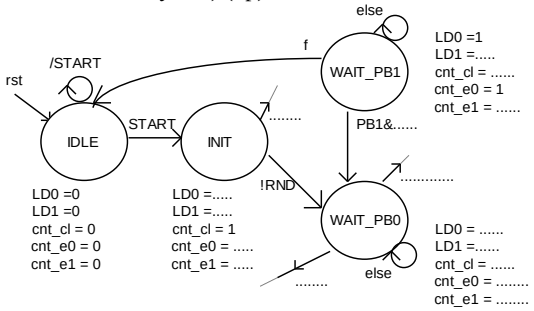
assign tc = e &……..
always @(……)
begin
if (rst) .....
else
if (……)
if(…….) .....
else ……
end
endmodule

c. Adja meg azt az f feltételt, amely jelzi, hogy két kaszkádosított számláló közül valamelyik végállapotban van! (a q0H, q0L, q1H, q1L jeleket használja) (1p)

assign f = ……

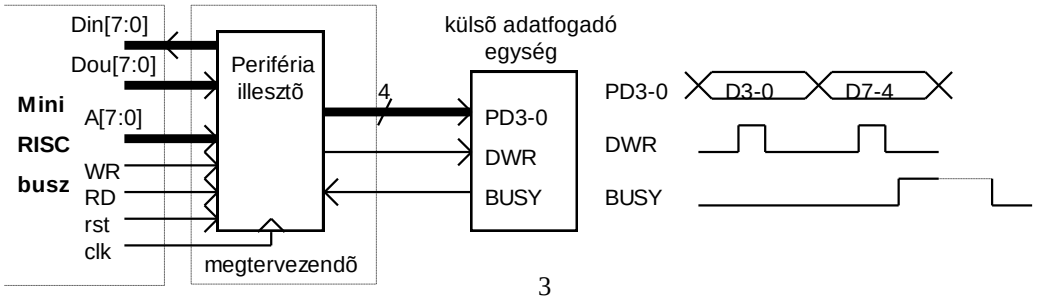
……

d. Tervezze meg a vezérlő Moore állapotgráfját és rajzolja le! A rajzot elkezdtük, fejezze be! (Hiányzó állapotátmenetek (nyilak), hiányzó állapotátmeneti feltételek, hiányzó kimenetek! Feltétel nélküli állapotátmenet esetén ne írjon semmit a nyílra!) (6p)



IMSC (5p) Külön lapon adja meg a sig01s jelet előállító modul Verilog leírását! A modul a rendszer 16MHz-es órajeléből állítja elő a 0.1sec-onként 1 órajelig aktív ***sig_01s*** jelet.

F3. (17p) Egy külső adatfogadó egységet kell kezelni a MiniRSIC processzorhoz illesztett logika segítségével. A külső egység 4 bites részletekben képes 8 bites adatokat fogadni a PD3-0 adatvonalain. Egy BUSY jellel jelzi, amikor nem képes adat fogadására. A 8 bites adat beírása két DWR impulzussal történik. Az alsó 4 bitjet az első DWR, felső 4 bitjet a második DWR impulzus lelfutó éle írja be. A 2. DWR hatására a BUSY jel 1-be áll, amíg a periféria újabb adat fogadására nem képes. A DWR impulzus minimális hossza legalább 1 MiniRISC Tclk.



Ehhez a külső egységhez kell periféria illesztő logikát tervezni a MiniRISC buszra (A[7:0], Din[7:0], Dou[7:0], RD, WR, IRQ, clk, rst). A periféria illesztő parancs regiszterrel (PR), státusregiszterrel (SR) és egy adatregiszterrel (DR) rendelkezik. A DWR impulzus előállítására a parancsregiszter beíró jelét használjuk (DWR = PR_wr, a kiírt adat értéke lényegtelen). A periféria BUSY jelének aktuális értéke a státusregiszter D0 bitjén olvasható be, a többi bit értéke a beolvasáskor bármi lehet. Az 4 bites adatot az illesztő adatregiszterének alsó 4 bitjére kell kiírni, a felső 4 bit értéke tetszőleges lehet. Az adat regiszter visszaolvasható. A periféria báziscíme 0xD0. A programozói felülete a következő:

funkció	Cím	D7 D6 D5 D4 D3 D2 D1 D0	olvasható/írható
Parancs regiszter PR	Báziscím +0	x x x x x x x x	W (***PR_wr***)
Státus regiszter SR	Báziscím + 1	x x x x x x x BUSY	R (***SR_rd***)
Adat regiszter DR	Báziscím + 2	x x x x PD3 PD2 PD1 PD0	W/R (***DR_wr, DR_rd***),

a. Tervezze meg Verilog nyelven a periféria parancs regiszterbe írást engedélyező ***PR_wr*** , a státus regiszter és az adatregiszter olvasását engedélyező ***SR_rd*** és ***DR_rd*** jeleit, továbbá a kimeneti adatregiszter írását engedélyező ***DR_wr*** jelet! (6p)

parameter base_addr = …… // a periféria kezdőcíme

assign psel = ……. … // aktív, ha a periféria címtartományához fordul a processzor

assign PR_wr = ……..

assign SR_rd = ……..

assign DR_rd = ……..

assign DR_wr = ……..

b. Adja meg Verilog nyelven az adatregiszter leírását! Az adatregiszter 4 bites, a **Dou[3:0]** íródik bele, ha a **DR_wr** jel aktív. Az **rst** jel törli. Elkezdtük, folytassa! (2p)

c. Adja meg Verilog nyelven a státus regiszter és adat regiszter visszaolvasásához szükséges logikát! Elkezdtük, folytassa! (2p)

b.
```
reg [3:0] q;]
always @(    …….    )
 if( rst)    q <= 4'b.....
  else
     if(.....)  q <= .....

assign PD = q;  // PD a periféria kimenete
```

c.
```
always(*)
   case({SR_rd, DR_rd})
         2'b10:  Din[0] <= .....
         2'b01: Din[3:0] <=.....
       default:   Din <= 8'h.....;
   endcase
```

d. Folytassa az assembly programhoz szükséges definíciókat! (1p)

```
DEF PR       0xD0
DEF SR       .....
DEF DR       .....
DEF BUSY     .....
          DATA
DatArrLen: DB 0x06
DatArr:    DB 0xaa, 0xfd, 0xbf, 0x75, 0x79,0x55
          CODE
```

e. Irjon meg egy olyan szubrutint, amely az r0 regiszterben megadott 8 bites adatot két 4 bites részletben kiírja a perifériára, ha az nem foglalt! (2p)

```
DatWR:  mov r2, .....    ;státus olvasás
        tst  .....      ;BUSY bit tesztelése
        .....……..... ;vissza, ha még foglalt
        ……..……..... ;adat alsó 4 bit kiírása DR-be
        ……; DWR generálás
        .....;felső alsó 4 bit csere
        ……..……..... ;adat felső 4 bit kiírása DR-be
        ……; DWR generálás
        …….....  ;visszatérés
```

f. Az előbbi szubrutint felhasználva írjon olyan program részletet, amely a d. pontnál definiált **DatArr** területről kiír a **DatArrLen** címen található számú adatot a perifériába! (DatArrLen > 0 és DatArrLen < 32) (4p)

```
ArrWr:  mov r10, #……...      ;tömb hossz címének betöltése
        mov r10,……          ;tömb hossz beolvasása
        mov r11, #…….        ;tömb címének betöltése
loop:   mov r0, …….          ;adat beolvasása a memóriából
        ……..                 ;adat kiírása a perifériába
        ……                   ;cím növelés
        ……                   ;adat számláló csökkentés
        ……                   ;vissza, ha van még adat
        …
```

IMSC (5p) A külső egység kezelése a fenti módon kicsit nehézkes. Használjon az illesztő eredeti 4 bites adatregisztere helyett 8 biteset, így a teljes 8 bites adat egyszerre beleírható! Tervezzen utána olyan logikát, amely a külső egység 4 bites PD[3:0] adat bemenetére az adatregiszterbe írás (DR_wr) után az adatregiszter alsó 4 bitjét kapcsolja, a DWR impulzus (PR_wr) kiadása után pedig a felső 4 bitjét.

Rendelkezésre álló idő: 100 perc