NÉV:..... neptun kód:..……

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

Olvasható aláírás:.....……

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

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!

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, 2 db 2-edes jegy	

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

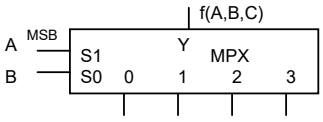
……..... .....

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

wire s, e; wire [1:0] O;

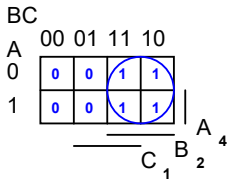
assign O[1] = s&e; assign O[0] = ~s&e; neve:.....

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);

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

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

assign F =……..

d. Adja meg a függvény legegyszerűbb SOP alakját Adja meg a függvény egyszerűsítetlen 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 hogyan kell összekötni egy USRT mastert egy USRT slave-vel! Rajzolja be a jelek irányát is nyíllal! (2p)

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

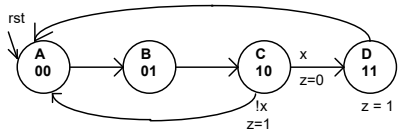
1.	Csak NOR kapukkal és a 0 logikai konstanssal minden logikai függvény megvalósítható.	
2.	Egy D flip-flopból egyetlen EXOR kapuval 1 bites engedélyezhető számláló készíthető.	
3.	Az szinkron RAM-ok az WR alatti órajel felfutó élére írják be az adatot.	
4.	Az processzorhoz a perifériától érkező interrupt 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) ……



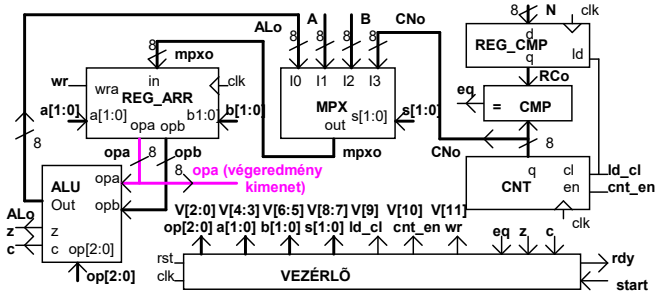
b. Adja meg az állapotregiszter Verilog leírását!
//Állapotregiszter (2p):
always@(……..)
if (rst) state <= ……..…
else state <= ……..……

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 ……

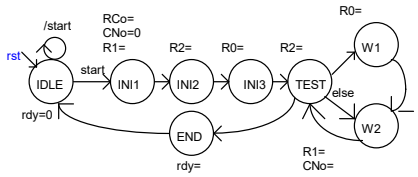
e. Az automata bemenetére x=0-át adunk folyamatosan. Milyen ismert funkcionális elemnek megfelelően viselkedik az FSM? Adja meg a nevét és tulajdonságait! (2p)
……

F2. (17p) 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**) 8 műveletet tud elvégezni az **opa** és **opb** operandusokkal az alább megadott táblázat szerint. Az **elvégzendő művelet** az ALU **op[2:0]** bemenetén 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 **REG_ARR egy 4 regiszter tartalmazó regiszter tömb.** Ennek **opa kimenetén az a[1:0]-al kiválasztott sorszámú regiszter tartalma** jelenik meg, **ha a[1:0]=i, akkor Ri**. Az **adat beírását wra=1 (wr) engedélyezi és az in bementére kiválasztott adat az a[1:0]-al kiválasztott regiszterbe** íródik az órajelre. Az **opb kimenetén a b[1:0]-al kiválasztott regiszter tartalma** jelenik meg, **ha b[1:0]=i, akkor Ri**. A **REG_ARR bemenetére kerülő adat az MPX multiplexer s[1:0] bemenetével** választható ki. A CNT (**CNo**) egy törölhető (**ld_cl**), engedélyezhető (**cnt_en**) **felfele számláló**. Ehhez kapcsolódik a **CMP** komparátor, melynek másik adatát a **REG_CMP (RCo)** tölthető (**ld_cl**) regiszter adja, **eq=(RCo = = CNo)**. Az adatstruktúra **kimenete az opa, egy elvégzett számítási feladat végeredményét itt kell megjelentetni**. A vezérlőt az egy 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 **eq** feltétel jeleket. Egy számítási feladat elkészültét 1 órajel hosszú **rdy** státus jellel kell jeleznie. Az eredménynek meg kell maradnia a következő start jelig. **Az aktuális művelet szempontjából érdektelen vezérlő jelek értékét 0-kell választani**. Így, **ha az eredményt az R0 regiszterben kérjük, akkor a vezérlő** regiszterművelet **nélküli állapotaiban az opa (eredmény kimenet) automatikusan az R0 tartalma lesz**.



a. (6p) Készítse el a fenti adatstruktúra műveleteinek felhasználásával egy 4 bites szorzó HLSM diagramját! A szorzást az LSB-vel kell kezdeni. A gráfot felrajzoltuk, adja meg az állapotokhoz tartozó műveleteket és a hiányzó állapotátmeneti feltételeket! A szorzandó az A bemeneten van, ezt az R1 regiszterben tárolja el. A szorzó a B bemeneten, ezt az R2 regiszterben kell tárolni. A ciklusváltozóról se feledkezzen meg! A végeredményt az R0 regiszterben állítsa elő! R0-ba azonban kezdetben 0-t kell írni. (Hogy lehet ezt megoldani?)

ALU funkció vezérlés: op[2:0]	ALU out (Alo)	z =1, ha	c = 1, ha
000	opa + opb	Alo==0	átvitel van
001	opa – opb	Alo==0	a–b < 0
010	opa << 1	Alo==0	a[7]
011	opb >> 1	Alo==0	b[0]
100	opa & opb	Alo==0	Soha
101	opa | opb	Alo==0	Soha
110	opa ^ opb	Alo==0	Soha
111	opb	Alo==0	Soha

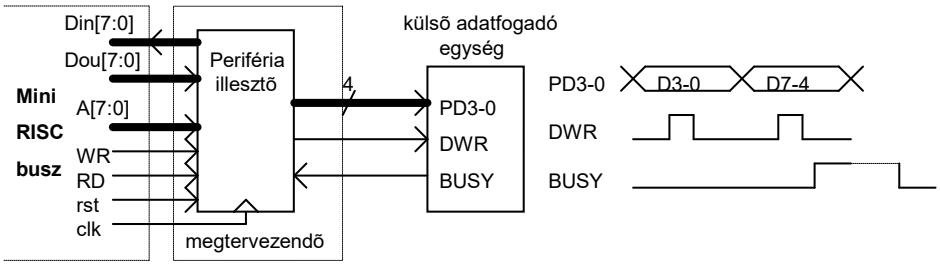


b. (6p) Készítse el a vezérlést, adja meg az egyes állapotokban a rdy és V[11:0] vezérlő jelek értékét (0, 1)!

	rdy	wr	cnt_ en	ld_ cl	s[1:0]		b[1:0]		a[1:0]		op[2:0]		
rdy,V[11:0]	xxx	11	10	9	8	7	6	5	4	3	2	1	0
IDLE													
INI1													
INI2													
INI3													
TEST													
W1													
W2													
END													

d. (6p)Valósítsa meg az ALU-t Verilog nyelven, az ALU működési táblázata alapján!.
A flag-ek megvalósításától most eltekintünk.
module ALU(input [2:0]op, input[7:0] opa, opb, output reg [7:0] Out, output z, c);

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;
```

c.
```
alwasys(*)
   case({SR_rd, DR_rd})
          2'b10:  Din[0] <= ..……
          2'b.01: Din[3:0] <=...……
        default:   Din <= 8'h..……
   endcase
```

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