20…..év …hó …nap

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

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

Olvasható aláírás:.....……

E: (25p)
F1: (15p)
F2: (25p)
$\sum$: (65p)
IMSC: (7p)

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!

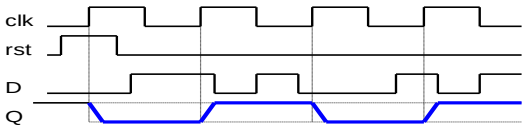
Ellenőrző kérdések (25p)

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

always@(**posedge clk**)

if(**rst**) .....**Q <= 1'b0;**.....

..**else**.... .... **Q <= D;**.....



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.

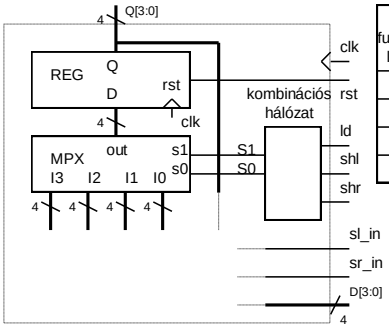
always@(**posedge clk**)

if(...**ld0**.....) .....**q <= In0;**.....

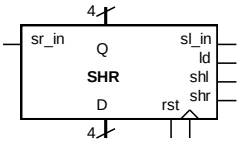
..**else**.. ..**if(ld1)**.. .... **q <= In1;**.....

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 bementeire! Ügyeljen arra, hogy a rajzon látható jel neveket használja!



dekódolt funkcióválasztás					
ld	shl	shr	S1	S0	funkció
0	0	0	0	0	tart
0	0	1	1	0	jobbra shift
0	1	..x	1	1	balra shift
..1	..x	..x	0	1	tölt



I0:.....**Q[3:0]**.....

I1:.....**D[3:0]**.....

I2:.....**sr_in, Q[3:1]**.....

I3:.....**Q[2:0], sl_in**.....

b. Töltse ki a táblázat hiányzó részeit, ha a prioritás sorrendje a legerősebb jellel kezdve a következő: betöltés (ld), balra shift (shl), jobbra shift (shr)!

E4. (5p) Adja meg egy 4 bites 10-es modulusú fel le számláló Verilog kódját! A számláló rst-re törlődik, ld-ra betölti d[3:0]-at, cnt_up=1 esetén felfele számol, cnt_down=1 esetén lefele számol, egyébként nem változik a q kimenete. A vezérlőjelek prioritása a legerősebb jellel kezdve a következő: rst, ld, cnt_up, cnt_down. A jelek definiálásától most eltekintünk.

```
assign tc =cnt_up&(q==4'b9) |
                  ~ cnt_up &cnt_down&(q==4'b0);
always@(posedge clk)
if(...rst....)  q <= .....4'd0;.....
else if(.....ld...) q <= .....d;.....
    else if(...cnt_up...)
        begin
        if(tc)..... q <= 4'd0;.....
        else..... .....q <= q + 4'd1;.....
        end
    else if(.....cnt_down.....)
        begin
        if(tc)...  q <= 4'd9;.....
        else….  q <= q - 4'd1;.....
        end
```

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. 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(..clk...), .rst(..rst....),.en(...EN...), .q(q10), .tc(...tc_i[0]....));

cnt3 cnt3_1(.clk(..clk...), .rst(..rst....),.en(..tc_i[0].), .q(q10), .tc(...tc_i[1]....));

cnt5 cnt5_1(.clk(..clk....), .rst(..rst....),.en(..tc_i[1].), .q(q10), .tc(OUT));
```

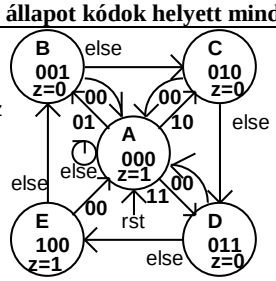
b. Hány órajelenként jelenik meg impulzus az ***OUT*** kimeneten? ....**7*3*5=105**.....

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

1.	Egy sorrendi hálózat (FSM) állapotgráfjának állapotszáma biztosan nem csökkenthető úgy, hogy a működése az állpotcsökkentés előtti változatával azonos maradjon.	-
2.	Ha egy sorrendi hálózat kimenetét az állapotregszter valamely bitjei adják, akkor az biztosan Moore modell szerint működik.	+
3.	A **FIFO**-ba bármikor lehet adatot írni.	-
4.	Egy FSM vezérlő állapotgráfjába írt állapotátmeneti feltétel lehet A == B.	-
5.	Az **ASM** (algoritmikus állapotvezérlő) leírás egyetlen feltétel doboza csak 2-felé ágazást tesz lehetővé.	+
6.	3 regiszter címes processzor architektúra esetén egy művelet eredménye nem rontja el szükségszerűen az egyik operandust.	+

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

bemenet: x[1:0]
kimenet:s[2:0], z



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 @ ( …posedge clk…)

   if (rst)  s <= …A…;

   else      s <= …next_s…;
```

b. (9p) Következő állapot logika

```
//Next_state logika
always @ (.......*.......)
   case (s)
    A:  case(…x…)
        2'b01:   next_s <= ……..B……..;

        2'b10:   next_s <= ……..C……..;

        2'b11:   next_s <= ……..D……..;

        default: next_s <= ……..A……..;
        endcase
    B: if(x == 2'b00) next_s <= ……..A……..;

       else           next_s <= C;

    C: if(x == 2'b00) next_s <= ……A……;

       else           next_s <= ……..D……..;

    D: if(x == 2'b00) next_s <= ……A……;

       else           next_s <= ……..E……..;

    E: if(x == 2'b00) next_s <= ……A……;

       else           next_s <= ……..B……..;

default:              next_s <= ……A……;
    endcase
```

c. (1p) Milyen modell szerint működik? (Húzza alá a megfelelőt) Mealy Moore

d. (2p) Adja meg a z kimenet logikai függvényét Verilog-ban!

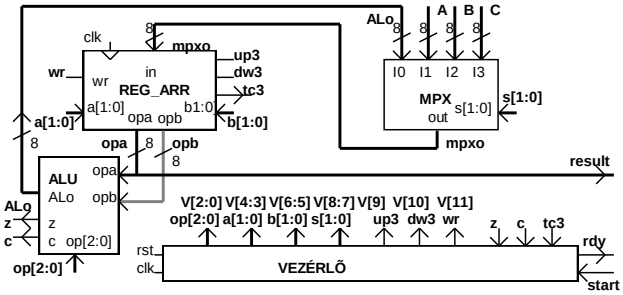
Ha csak az s-t, az állapot neveket, az == operátort és | műveletet használhatja:

assign z = ……(s == E);……..

A legegyszerűbb alakban: **assign z =** ……s[2];……..…

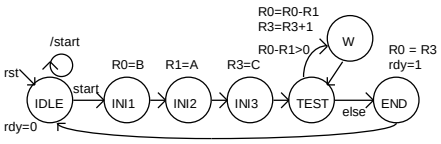
F2. (25p) Funkcionális blokkvázlatával (adatstruktúra + vezérlő) adott egy a bemenő adatokkal többféle művelet elvégzésére alkalmas számító modul. Az adatstruktúra **3 db 8 bites adat bemenettel** rendelkezik: **A, B, N** (előjel nélküli számok). Az aritmetikai logikai egység **(ALU**) 8 műveletet tud 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 **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 regisztert tartalmazó regiszter tömb.** Ennek **opa 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 in bementére kiválasztott adat (mpxo) **az a[1:0]-val kiválasztott regiszterbe** íródik az órajel felfutó élére. Az **opb kimenetén a b[1:0]-val kiválasztott regiszter tartalma** jelenik meg, **ha b[1:0]=i, akkor Ri**. A **REG_ARR bemenetére kerülő adat** (mpxo) **az MPX multiplexer s[1:0] bemenetével** választható ki. **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 MPX-el kiválasztott érték) és normál regiszterként olvasható**. Azonban, ha éppen nem írják, akkor dw3=1 esetén lefele számol (R3=R3-1), up3 = 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**. Az adatstruktúra **kimenete az opa (result), egy elvégzett számítási feladat végeredményét itt kell megjelentetni**. 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 **tc** 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-nak kell választani**. Így, **ha az eredmény az R0 regiszterben van, akkor a vezérlő** regiszterművelet **nélküli állapotaiban az opa (eredmény kimenet) automatikusan az R0 tartalma lesz**.



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	opa[7]
011	opb >> 1	Alo==0	opb[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)

a. (10p) Az alábbi Moore jellegű HLSM állapot diagram által leírt feladatot a fenti adatstruktúrával kell megvalósítani.

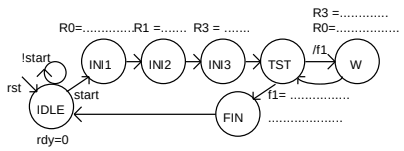


(A feladat a B/A egész részét állítja elő, C=0.) Adja meg a ***vezérlő állapotgráfjának megadott állapotaiban kiadandó vezérlőjeleket a V[11:0] bitek értékének (0, 1) alábbi táblázatba való beírásával!*** (Segítségképp néhány értéket megadtunk. A rdy-t most nem kérjük.)

	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**
INI1	1	0	0	1	0	0	0	0	0	0	0	0
INI2	1	0	0	0	1	0	0	0	1	0	0	0
INI3	1	0	0	1	1	0	0	1	1	0	0	0
TEST	0	0	0	0	0	0	1	0	0	0	0	1
W	1	0	1	0	0	0	1	0	0	0	0	1
END	1	0	0	0	0	1	1	0	0	1	1	1

b. (1p) A vezérlő blokkvázlat mely feltétel jele képes jelezni az R0-R1>0 feltételt? ……**c**…….. (Segítő kérdés: Melyik műveletet kell beállítani ehhez a TEST állapotban?)

c. (7p) A számító modullal a C*B szorzatot kell kiszámítani (Az A bemenetet most nem használjuk).
A megvalósítás: ***A 0 kezdőértékű R0 regiszterhez C-szer hozzáadjuk B értékét. A hozzáadások számát a speciálisan számlálóként is használható R3 regiszterrel számoljuk. Csak az R0, R1 és R3 regisztereket használhatja. Rendelje a számítás elvégzéséhez szükséges műveleteket a 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!***



1: R0 = R0+R1 **2:** R0 = A **3:** R0 = R0 $\oplus$ R0 **4:** R0 = C
5: R1 = C **6:** R1 = A **7:** R1 = B **8:** R1 = R0+R1
9: R3 = R3+1 **10:** R3 = C **11:** R0 = R0+R1, R3 = R3-1
12: R0 = R0+R1, R3 = R3+1 **13:** rdy = 1

IDLE:..**x**…. INI1: ….**3**… INI2:……**7**….. INI3:.…**10**.….

TST: …..**x**.…. W:……**11**…… FIN: ….**13**…..

d. (2p) ***Adja meg a kért feltétel(eke)t magaszintű feltétel megadással!***

f1: ….**R3==0**……

Adja meg a kért feltéte(lek)et blokkvázlaton szereplő azon jel(ek) megnevezésével, amely(ek) jelzi(k) a feltétel teljesülését. A parancs és feltétel bitek negáltját is fel lehet használni.

f1: …**tc**…

e. (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!

```
module REG_ARR (input clk, wr, input [1:0] a, b,
input [7:0] in; output reg [7:0] opa, opb);

// beírás megvalósítása
reg [7:0] arr[3:0];
always@(.posedge clk)

if(wr) arr[a] <= in;
```

always@(...*...) // **opa elő-** // **állítása**	always@(..*...) // **opb elő-** // **állítása**
opa <= **arr[a]**;	**opb** <= **arr[b]**;

```
endmodule
```

IMSC1 (3p)

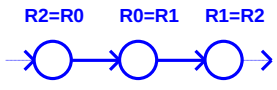
Tervezzen ***programozható modulusú 4 bites felfele számlálót***! A számláló modulusa 2-től 16-ig legyen változtatható a 4 bites d bemenetére adott szám függvényében. Addja meg az egység Verilog kódját! A számláló regiszter neve legyen q.

```
reg [3:0] q;

wire [3:0] d;

always@(posedge clk)
if(rst) q <= 4'h0;
else if(q == d) q <= 4'h0;
     else       q <= q + 4'h1;
```

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?

Művelet: op[2:0]=111 (ALo = opb)

HLSM diagram részlet:



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