20…..év …hó …nap

NÉV:……..... Neptun kód:..…….. GYAK 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. (3p) Végezze el az előírt átalakításokat!

Kiinduló adat:	Kért talakítás:	Átalakított adat:
2-es komplemens): 1010	decimális	**-6**
BCD: 1001 0100	decimális	**94**
Hexadecimális: AE	bináris	**1010 1110**

E2. (2p) Írja le a Boole algebra De'Morgan tételének 2 féle alakját 2 változóra (A, B)!

…**/(A + B) = /A/B**……. …**/(AB) = /A + /B**……

E3. (1p) Egyszerűsítse az alábbi Boole kifejezést!

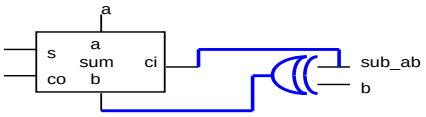
A + /AB = ...**A + B** .....

E4. (2p) Az alábbi Boole algebrai kifejezés egy ismert funkcionális elemet ír le. Adja meg a funkcionális elem pontos nevét és jeleinek funkcióját!

assign y = (A[0]& B[0] | ~A[0]& ~B[0]) & (A[1]& B[1] | ~A[1]& ~B[1]);

neve:..**egyenlőség komparátor**.... A[1:0], B[1:0]:…**a két adat** y:…**kimenet**

E5. (2p) Egészítse ki az alábbi összeadót, hogy az egy sub_ab bemenettel vezérelhető 1 bites összeadó/kivonót valósítson meg (sub_ab = 0: a + b, sub_ab = 1: a-b)!



E6. (2p) Megadtuk az ***f*** függvény definícióját igazságtáblával.

a. Írja le a függvényt megvalósító kombinációs hálózatot Verilog nyelven, ***minimalizálatlan SOP alakban***, közvetlenül a mintermeket specifikálva! (1p)

A(4)	B(2)	C(1)	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

assign f =.....**~A&~B&~C | ~A&~B&C | ~A&B&~C | A&B&C;**.....

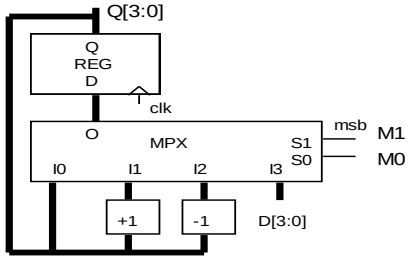
b. Az ABC bemenetre 3 bites előjel nélküli bináris számokat adunk {A, B, C}. Adja meg a függvényt a ***legegyszerűbb*** Verilog formában, ha csak a <, >, | operátorokat használhatja ! (1p)

assign f =.....**({A,B,C} < 3) | ({A,B,C} > 6);**.....

E7. (2p) Adja meg egy szinkron törölhető (r), és szinkron 1-be írható (s) D flip-flop Verilog leírását! (Legyen r a nagyobb prioritású.) A leírást elkezdtük, folytassa!

```
wire r, s, d;
reg q;
always@(...posedge clk....)
if(r…)   q <=…1'b0;……
else if(…s…) q<= …1'b1;……
     else    q<= …d;…….
```

E8. (3p) a. Milyen funkcionális elem blokkvázlata látható az alábbi rajzon?



Funkcionális elem pontos neve:

……**fel-le számláló**…….

b. Irja be az alábbi táblázatba a megfelelő M1M0 kombináció alá, hogy annak hatására mit csinál a funkcionális elem! Az elemnél megszokott szavakat használjon!

M1M0:	00	01	10	11
Funkció	……**tart**……	**fefele számol**…	**lefele számol**…	…**tölt**……

E9. (1p) Egészítse ki az alábbi Verilog leírást, hogy az egy 2/1-es busz multiplexert valósítson meg!

```
wire [7:0] I0, I1;
wire s;
reg[7:0] out;
```

```
always@(..*.)
  case(…s…)
  0: …out <= I0;……
  1:… out <= I1;…….
  endcase
```

E10. (2p) Válaszoljon az alábbi ***STACK***-re vonatkozó kérdésekre ill. húzza alá a megfelelő választ!

a. Hova kerül a bele írt adat? a tetejére az aljára máshova

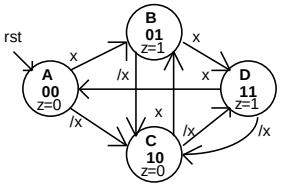
b. Olvasáskor honnan származik az olvasott adat? a tetejéről az aljáról máshonnan

E11. Mely állítások igazak és melyek hamisak? Jelölje **+*-al az igaz, -al a hamis*** állításokat! (5p)

1.	A diszjunktív normál alakban (DNF) annyi szorzat tag van, ahány 1-es az igazságtáblában.	+
2.	Egy 3 bites kiválasztó bemenetű multiplexerrel és a logikai konstansokkal tetszőleges 3 változós logikai függvény megvalósítható.	+
3.	Többkimenetű logikai hálózat legegyszerűbb megvalósítását mindig a kimenetenkénti minimalizálás adja.	-
4.	Egy Moore modell szerint működő szinkron sorrendi hálózat kimenete mindig csak az órajel hatására változhat.	+
5.	Az előjel nélküli számok összeadására képes összeadó a 2-es komplemensű számokat is helyesen adja össze.	+

Feladatok:

F1. (11p) Adott egy FSM az alábbi ***kódolt állapotgráffal.*** Állapotok: A,B,C,D. Bemenetek: clk, rst (hatására az A-ba megy), x. Kimenet: z. Az állapotkódolást és z-t megadtuk. ***Készítse el az FSM Verilog leírását*** külön az ***állapotregiszter***, külön a ***next_state logika*** és külön a ***kimeneti logika*** megadásával! A leírást elkezdtük, fejezze be!




```
reg [1:0] s, next_state;
wire z;
```

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

Verilog leírás:

A konstans és változó deklarációk. (2p)

```
localparam [1:0] A = 2'b00,  B =..2'b01.. , C = .... 2'b10...., D = .. 2'b11.....;
```

//**Állapotregiszter** (2p): always@(**posedge clk**..) if (rst) s <= …**A;**…… else …**s**…… <= …**next_state;**….. //**Kimeneti logika** (2p): Adja meg a legegyszerűbb Boole algebrai alakot! (1p) assign z =…**s[0];**…….. Adja meg az = = operátort használó legegyszerűbb alakban (1p) assign z =……**(s == B) | (s == D);** ……	//**Next_state logika** (4p): always@(…*……) begin case (s) A: if(…**x**.) next_state <= …**B**……; else …… **next_state <= C;** …… B: if(…**x**.) next_state <= ……**D**……; else …… **next_state <= C;**…… C: if(…**x**.) next_state <= …**B;**……; else …… **next_state <= D;** …… D: if(…**x**.) next_state <= …**A**……; else …… **next_state <= C;**… default: … **next_state <= A;**….. endcase end

F2. (12p) Tervezze meg egy impulzushossz mérő **adatstruktúráját**! (A vezérlőt nem kell megtervezni. A rendszerórajel 16 MHz frekvenciájú.) Az impulzushossz mérő egy külső **start** jelre (egy órajel hosszú órajellel szinkron impulzus) kezdi a mérést. Megméri az **in_pulse** bemeneten érkező jel felfutó éle és lefutó éle között eltelt időt, 1 usec felbontással. Ha kész a méréssel, azt az **rdy** kimenetén 1-el jelzi. Az eredmény (**result[15:0]**) ekkor a **REG16** regiszterben található az újabb mérés kezdetéig. Újabb start jelre rdy = 0 lesz és újra kezdődik a mérés. Bekapcsolás után az eredmény 0-át mutat.

A megvalósítás: Az impulzus hosszát egy 1us-onként 1 órajel hosszú impulzus (**us1**) hatására lépő 16 bites törölhető bináris felfele számlálóval (**BIN_CNT16**) mérjük. A us1 jelet a **PS** előosztó modul állítja elő. Az estleges túlcsordulással nem foglalkozunk. A számláló adat kimenetére egy 16 bites tölthető regisztert (**REG16**) csatlakoztatunk, ennek kimenetén jelenik majd meg az eredmény. A fel és lefutó él figyeléséhez egy éldetektort (**edge_detect**) használunk. Ennek **rise** kimenete 1 órajel hosszú impulzussal jelzi, a ha fefutó élet észlelt, a **fall** kimenete pedig ugyanígy viselkedik lefutó él esetén. Az éldetektor engedélyezhető (**en_detect**). A start jel hatására a vezérlő engedélyezi az éldetektort és törli rdy-t. Az éldetector rise jele törli PS-t és BIN_CNT16-ot, mely ezután elkezd felfele számolni. Az éldetektor fall jele áttölti a számláló aktuális értékét a REG16 regiszerbe, melynek kimenete szogáltatja az eredményt (result[15:0]), a vezérlő letiltja az éldetektort, a rdy jelet 1-be állítja és alapállapotba kerül.

A feladat megoldását részfeladatokra bontottuk.

a. Tervezze meg a BIN_CNT16 16 bites bináris felfele számlálót. A számláló törölhető (rst), engedélyezhető (en), végérték jelző kimenettel rendelkezik (tc). Adja meg a Verilog leírását! A leírást alább elkezdtük, fejezze be! A tc leírását úgy adja meg, hogy ne szerepeljen benne konstans! (4p)

b. Állítsa elő az **us1** jelet lefele számlálóval, ha az órajel fekvenciája 16 MHz! (1us-onként 1 órajel hosszú impulzus.) Adja meg a Verilog leírását! (4p)

c. Kösse össze az adatstruktúra elemeit egymással az alábbi példányok interfészének megfelelő kitöltésével! (4p)

```
edge_detect  edge_dtetect(.clk(clk.), .rst(rst..), .en(…en_detect.), .in(in_pulse),.rise(rise), .fall(fall));
                    PS PS (.clk(clk..), .rst(rst | rise.), .us1(us1));
BIN_CNT16  BIN_CNT16 ( .clk( clk…), .rst(…rst | rise….),  .en(u1s….), ..tc(), .q(q));
          REG16 REG16 ( .clk( clk….), .rst(...rst....), .ld(....fall……), .d(…q..),.q(…result…))
```

a. // 16 bites bináris számláló Verilog leírása(4p):
module _BIN_CNT16(input clk, rst, en, output tc, output reg [15:0] q);
 	
assign tc = …**(&q)&en;**……;
 	
always @(posedge clk)
begin
if (rst) q < = **16'h00;**……..
else
…**if(en) q <= q + 16'h01;**……..;
end
endmodule
	b. Állítsa elő az **us1** jelet lefele számlálóval, ha az órajel fekvenciája 16 MHz! (1us-onként 1 órajel hosszú (62.5ns) impulzus.) Adja meg a Verilog leírását! (4p)
	
	module PS(input clk, rst, output us1)
	reg [3:0] q
	wire us1;
	assign us1 = …**&~q;**……; //~|q, q==4'h0
	
	always@(...**posedge clk**...)
	begin
	if(rst) q <= …**4'h0;**….
	else
	……**q <= q – 4'h1**……..;
	end

d. IMSC (5p) Tervezze meg külön lapon az edge_detect logikát! (Adja meg a Verilog leírását!)

d. IMSC (5p) Tervezze meg külön lapon az edge_detect logikát! (Adja meg a Verilog leírását!)

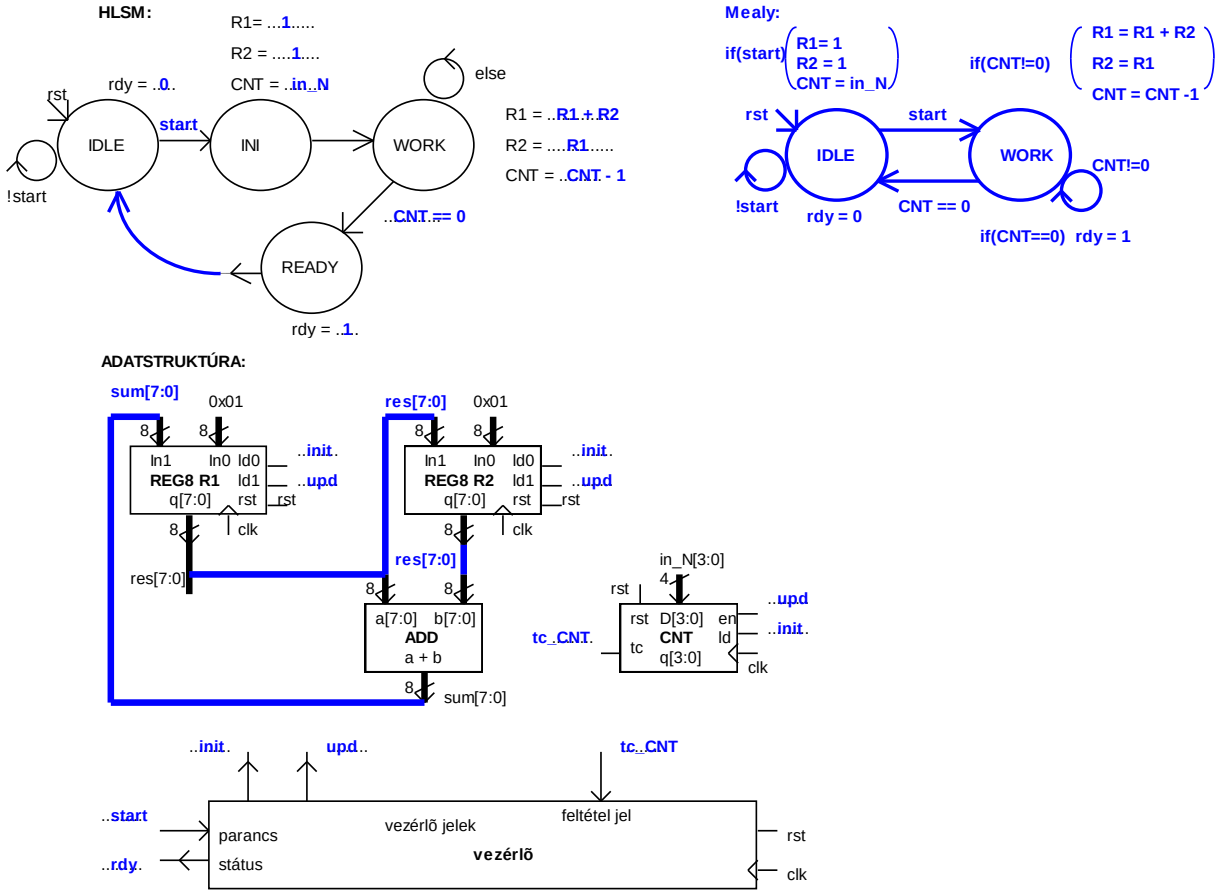
```
module edge_detect(input clk,  rst, en,  in, output rise, fall)
reg [1:0] q;
always@(posedge clk)
begin
   if(rst) q <= 2'b11;
    else
          if(en) q <= {q[0], in};  // vagy másik irányba shiftelve
end

assign rise = (q == 2'b01);  // shiftelési irányt figyelembe véve
assign fall = (q == 2'b10);  //   -  „" –
endmodule
```

F3. (12p) A feladatban az **első n (2<n<14) fibonacci szám előállítását** kell kell megvalósítani. Az áramkör bemenetén N = n-3 számot kell beállítani (0…11). Az **in_N[3:0]** beállítása után, a számítás indítását az 1 órajel hosszú szinkronizált **start** jel jelzi. Ennek hatására kell megkezdeni a számolást. Az eredményt a **res[7:0]** kimeneten kell előállítani és az elkészültét egy 1 órajel hosszú **rdy** jellel kell jelezni. Ezután a hardver újabb számításra kész. Az újraindításig az előző számolás eredményét mutatja.

a. Röviden fogalmazza meg az algoritmust! Egészítse ki, ill. folytassa az elkezdett leírást! (2p)
A számoláshoz 2 db 8 bites regisztert (**R1** és **R2**) és egy (**CNT**) 4 bites bináris lefele számlálót használunk. A **start** jel megérkezése után elvégezzük az inicializálást, tehát az R1 és R2 regiszterbe 1-et töltünk, a számlálóba pedig betöltjük az …**in_N[3:0]**...értékét. Ezután elvégezzük a következő érték számítását, tehát egy ütemben R1-be beírjuk R1 és R2 összegét, R2-be beírjuk az R1-et és CNT-t **..csökkentjük**... A számolásnak akkor van vége, ha a CNT lefelészámláló eléri a …**0**..-t (**tc_CNT**). Az eredmény R1 kimenetén, **res[7:0]**-ban található. Ekkor az **rdy = ..1..**, majd visszatérünk az újabb adatra váró állapotba, ahol rdy = **..0**...
b. Egészítse ki az adatstruktúrát vezérlő HLSM Moore jellegű állapotdiagramját! A rajzolást elkezdtük, adja meg a hiányzó állapotátmeneteket (irányított gráf élek) az átmeneti feltételekkel együtt, és az elvégzendő magas szintű műveleteket az egyes állapotokban (Verilog-szerűen)! Ha egy állapotból az összes többi feltétel nem teljesülése esetén megy valahova, akkor a feltételt ***else***-el jelöltük. (4p)

c. ***Rajzolja le a feladat adatstruktúrájának a blokkvázlatát.*** Az adatstruktúra felrajzolását elkezdtük. Fejezze be az adatkapcsolatok, adat utak felrajzolását! Rajzolja be, hogy a vezérlő milyen kimeneti (vezérlő és státus) jelekkel és bemeneti (feltétel, parancs) jelekkel kapcsolódik az adatstruktúrához és a külvilághoz! A vezérlő és feltétel jelek elnevezése: **init** (inicializálások), **upd** (részeredmény frissítése) **tc_CNT**. (6p)



e. IMSC (5p) Rajzolja le (külön lapon) a HLSM állapotgráf Mealy változatát!

Maximális pontszám: 60 pont Rendelkezésre álló idő: 100 perc