3-as mérés

1. Adja meg az alábbi portokkal rendelkező Verilog modul modul deklarációját. Bemenetek: skalár clk, skalár en; kimenet: 8 bites dout.

```
module example(clk,en,dout);
	input clk;
	input en;
	output [7:0] dout;

//modul funkcionalitásának leírása

endmodule
```

2. Deklaráljon egy 4 bites data nevű, wire típusú jelet, és folyamatosan adja neki a hexadecimális 0x42 értéket bináris formában megadva.

```
wire [3:0] data;
assign data =

//0x42=b01000010 ezt hogy adjam értékül egy 4 bites cuccnak?
```

3. Deklaráljon három darab 8 bites, előjeles wire típusú változót (res, op_a, op_b). A res értékének adja át op_a és op_b összegét.

```
wire signed [7:0] res,op_a,op_b;
assign res = op_a + op_b;
```

4. Deklaráljon egy 16 bites és egy 8 bites wire típusú változót (d16, d8). A 16 bites változónak adja át a 8 bites változó előjel kiterjesztett értékét (a 8 bites változó kettes komplemens értékeket tartalmazhat).

```
wire [15:0] d16;
wire signed [7:0] d8;

assign d16={{8{d8[7]}},d8};
```

5. Deklaráljon egy 8 bites, reg típusú (res) és két darab 8 bites wire típusú változót (op_a, op_b). Adja meg azt a Verilog kódot, ami olyan kombinációs logikát valósít meg, amelyben res értéke op_a és op_b összege.

```
wire [7:0] op_a, op_b;
reg [7:0] res;
always @ (op_a, op_b)
res <= op_a + op_b;
```

6. Adja meg egy 4:1 multiplexer Verilog kódját. Bemeneti jelek: kiválasztó jel (sel), adat bemenetek (in0, in1, in2, in3); kimenet: r. A bemeneti jeleket deklarálja wireként (az adatok 1 bitesek), a kimenetet pedig a leírás módjának megfelelően.

```
module mux_41 (input in0, in1, in2, in3, input [1:0] sel, output reg r);
always @ (*)
        case(sel)
                2'b00: r <= in0;
                2'b01: r <= in1;
                2'b10: r <= in2;
                2'b11: r <= in3;
        endcase
endmodule
```

7. Adja meg egy aszinkron reset-elhető (rst), set-elhető (set), engedélyezhető (en) 1 bites D FF Verilog kódját. Deklarálja a kimeneti és bemeneti jeleket is (utóbbiakat 1 bites wire-ként).

```
        module dff(clk,rst,en,d,q);
input clk,rst,en,d;
output q;
        always @ (posedge clk, posedge rst, posedge set)
        if (rst)
                q <= 1'b0;
        else if (set)
                q <= 1'b1;
        else if (en)
                q <= d;
endmodule
```

8. Adja meg azon testbench kódját, ami 10 MHz-es órajelet generál. Az órajel neve legyen clk, ezt deklarálja is.

```
‘timescale 1ns/1ps
    //T=1/10 MHz=100ns
    wire clk;
    always #50 clk <= ~clk;
```

9. Adja meg egy 8 bites, reset-elhető, felfelé számláló számláló modul Verilog kódját. Bemenetek: órajel (clk), reset (rst); kimenet: a számláló értéke (cntr).

```
    module counter(
    input clk, rst,
    output [7:0] cntr
    );
    reg [7:0] tmp;
    always @(posedge clk, posedge rst)
    if (rst==1)
            tmp <= 8’b0; //itt szerintem nem kell kiírni a sok 0-t
    else
            tmp <= tmp + 1;
    assign cntr=tmp;
    endmodule
```

10. Adja meg egy 8 bites, balra léptető shift regiszter modul Verilog kódját. Bemenetek: órajel (clk), soros adatbemenet (din); kimenet: a shift regiszter értéke (shr).

```
    module shift_reg(
    input clk, din,
    output [7:0] shr
    );
    reg [7:0] dout;
    always @(posedge clk)
    dout <= {dout[6:0],din};
    assign shr = dout;
    endmodule
```

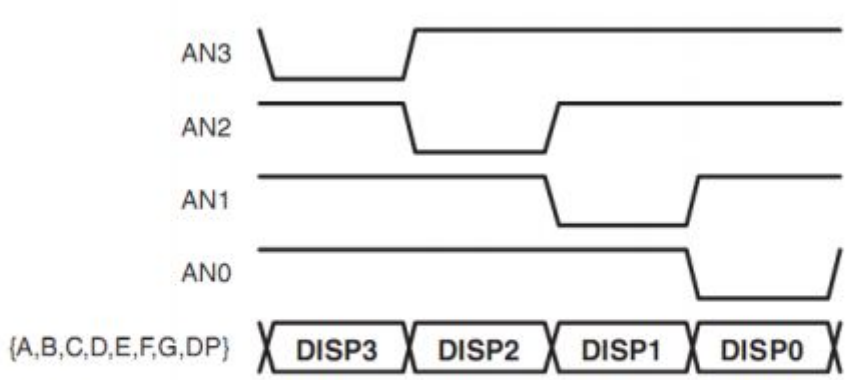
|
|
|

|
|
|
|
| :D
|
|
|
|

11 Verilog doge

10-es mérés

1.Adja meg egy időmultiplexált, 4 digites 7-szegmenses kijelző vezérlésének hullámformáját (AN és DIG jelek). Milyen gyakorisággal kell változtatni az AN jelet? Mi a hatása ha nagyon lassan (~1 sec) változik ez a jel?



Ha a fenti megjelenítési ciklus túl gyors, akkor a 7-szegmenses kijelző LED-jeinek „nincs idejük kigyulladni", ha túl lassú, akkor pedig folyamatosnak tűnő megjelenítés helyett villogást tapasztalunk. **A megfelelő frekvencia kHz nagyságrendű.**

Talán ez

2.Mire kell figyelni Verilogban, ha if vagy case szerkezettel írunk le egy kombinációs logikát?

Legyen default érték, vagy ha nincs akkor térjünk ki a "don't care" esetre (???)

3.Készítsen 3 bemenetű és/vagy kaput 2 bemenetű és/vagy kapuk (and2/or2) példányosításával!

Ez most 2 feladat?

```
module orgate2(a,b,y);

input a,b;
output y;

assign y=a|b;

endmodule
```

```
module orgate3(A,B,C,Y);

input A,B,C;
output Y;
wire orwire;

orgate2 kuk1(.a(A), .b(B), .y(orwire));
orgate2 kuk2(.a(C), .b(orwire), .y(Y));


endmodule
```

A maradékot aki akarja írja be de gondolom mind ez csak más kapuval, már ha ez volt a feladat…

4. Adja meg egy 4 bites, visszacsatolt shift regiszter Verilog kódját. A regiszter kezdeti értéke legyen 4'b1000, a shiftelés jobbra történjen. A modul bemenetei: órajel (clk), reset (rst); a kimenete a shift regiszter aktuális állapota (shr).

```
module ring_count(shr,clk,clr);
                                    input clk,rst;
                                    output [3:0]shr;
                                    reg [3:0]shr;
                                    always @(posedge clk)
                                        if(rst==1)
                                            shr<=4'b1000;
                                         else
                                            begin
                                               shr[3]<=shr[0];
                                               shr[2]<=shr[3];
                                               shr[1]<=shr[2];
                                               shr[0]<=shr[1];
                                            end
                                endmodule
```

Wikipedián találtam ez elvileg az úgynevezett Jhonson számláló vagy ring counter, nekem az az ötletem volt, hogy mezei D flip flopokat példányosítok. Ezeken kívül más nem jut eszembe.

Szerintem jó csak lehetne egyszerűbben meg rövidebben úgy ahogy a nyolcasban csinálja: shr<={shr[0],shr[3:1]}; Ami sztem úgyanaz csak így elegánsabb meg rövidebb.

5. Adja meg egy engedélyező jel generátor modul (rategen) Verilog forráskódját, amely 16 MHz-es rendszer órajel használatával 800 kHz frekvenciájú, egy órajel hosszúságú engedélyező impulzus sorozatot állít elő. A modul bemenetei: órajel (clk), reset (rst); kimenete: engedélyező jel (en).

```
module rategen800(clk,rst,en);
  input clk,rst;
  output en;
  reg [23:0]q;

  always @(posedge clk)
   if(rst|en)
          q <= 0;
          else
          q<=q+1;

          assign en = (q==799999);

endmodule
```

Igazából nemtom, hogy kell a 800kHz-t kiszámolni ez tipp, valaki majd jelezze ha tudja pls. Mer én most azt csináltam, hogy elvileg 800khz enként jön egy órajel, ez a feladat egyáltalán?

Megoldásom:
Tclk=1/16Mhz=62.5 ns
Trate=1/800khz=1.25 us
Trate/Tclk=20 20-ig kell számolni
Egyszerűbben is kijön:fclk/frate=1/20
reg[5:0]q;
q==20

Szerintem is 20 de nem csak 19.ig kell számolni hogy a 20.ra már az legyen a kimenet.és akkor a q=19? Mert hogy mindig eggyel kevesebbet szoktunk számolni asszem?
Igazad van

6.Adja meg egy 1 digites felfelé számláló BCD számláló Verilog kódját. A számláló reset állapota legyen 0. A modul bemenetei: órajel (clk), reset (rst); a kimenete a számláló aktuális állapota (cnt).

```
module BCD(clk,rst,cnt);

  input clk,rst;
  output [3:0]cnt;
  reg [3:0]cnt;
  wire cnt9;

  always @(posedge clk)
   if(rst|cnt9)  // ha elértük a 9-et akkor túlcsordul és vissza 0 vagy reset
          cnt <= 0;
          else
          cnt <=cnt+1;

          assign cnt9 = (cnt==9);

endmodule
```

7. Adja meg azt a Verilog Test Fixture kódrészletet, ami 10 MHz frekvenciájú órajelet biztosít a vizsgálandó modul számára!

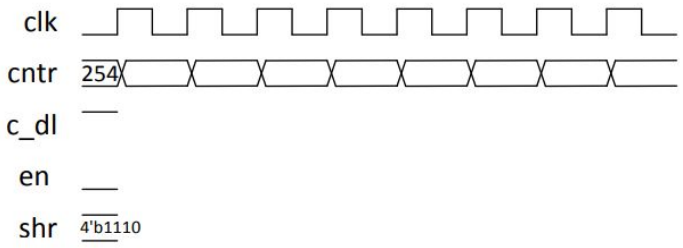
```
`timescale 1ns / 1ps
always #50
          clk <= ~clk;
```

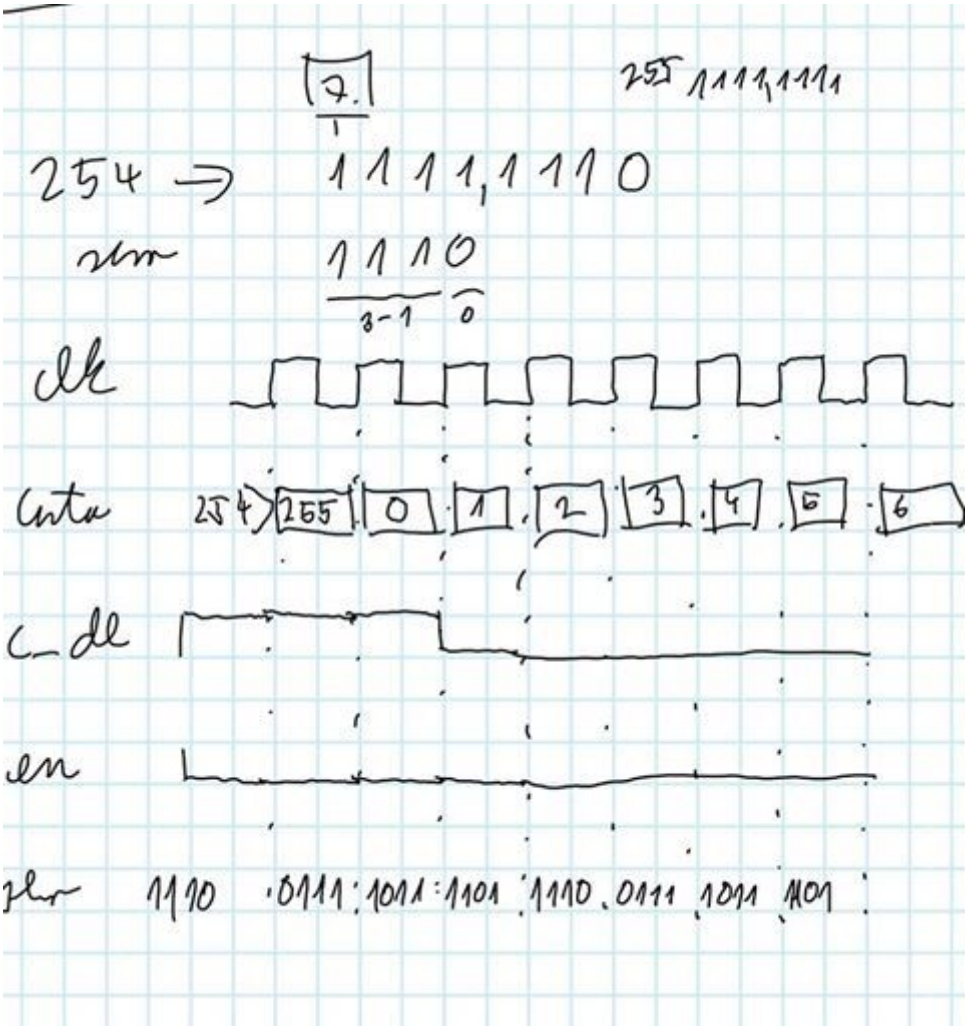
50ns akkor a periódusidő 100ns (50 alacsony 50 magas él) f=1/T → 1/10^-7=10MHz
asszem, ennyi?
Ennyi

8.Adott az alábbi Verilog kód. Egészítse ki az időzítési diagramot.

```
reg [7:0] cntr=0;
reg en; reg c_dl;
always @ (posedge clk)
begin
  cntr <= cntr + 1;
  c_dl <= cntr[7];
  en <= ~cntr[7] & c_dl;
end

reg [3:0] shr=4'b1110;
always @ (posedge clk)
  shr <= {shr[0], shr[3:1]};
```



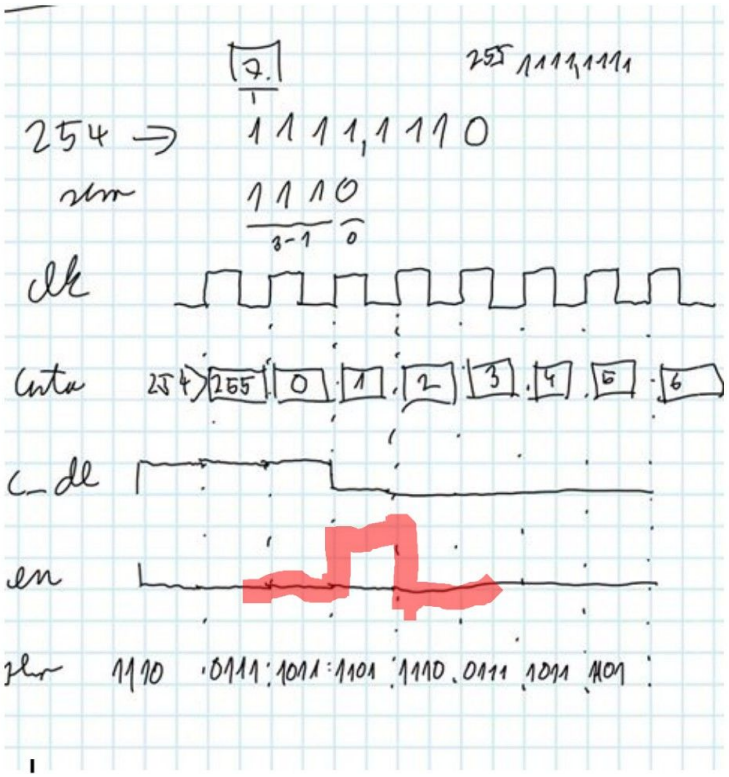


Szerintem azért mert a begin-end ben nem blokkoló érték adások vannak tehát egyszerre értékelődnek ki ,így amikor azt mondjuk cntr+1 akkor a következő sor még az inkrementálás előtti értékkel dolgozik, meg a többi is és csak kövi ciklusban dolgozik az új értékkel

Jónak tűnik csak az en szerintem felugrik mert a ~cntr[7] már 1 és a c_dl még 1

Szerintem is átvált 1re a kimenet a második ábrának megfelelően.

Valóban en felfut!



Az

…

11-es mérés

1. Egyetlen ASCII karakter UART átvitele hány bit kiküldését jelenti a házi feladatban adott paraméterek esetén? Pontosan hogy épül fel ez a keret, milyen sorrendben kell kiküldeni a biteket?
   7N1 tehát 9 bit, start, adat, stop
   Sorrend: LSB először

   Azt hiszem ez 2-szer 9 bit mert mert az ascii karakter 8 bites úgyhogy két keretbe fér bele
   lol 3 hónap múlva kiderült hogy az eredeti ascii 7 bites

2. Hogy kapjuk meg egy 4 bites BCD számjegy 8 bites ASCII kódját? Mit mutat a paritásbit, milyen hardver elemmel számítható a legegyszerűbben, ha az összes adatbit rendelkezésre áll? Páros paritás esetén milyen paritás érték tartozik az ABh adathoz?

Az ASCII szabvány szerint egy „x” számjegy ASCII kódja 3xh alakban áll elő, pl. a 9-es számjegynek az ASCII kódja 39h = 0011.1001b.

A paritásbitet arra használjuk, hogy egy bináris számot úgy egészítsünk ki vele, hogy az vagy páros darab, vagy páratlan darab 1-est tartalmazzon. A gyakorlatban ezt hibaészlelésre használjuk.

A paritás legegyszerűbben XOR kapuk segítségével határozható meg.

ABh = 1010.1011b, mivel a bináris alak páratlan darab 1-est tartalmaz, így a paritásbit értéke 1-es lesz.

3. Ha a rendszer órajele 16 MHz, mekkora a házi feladatként létre hozott UART modul sebességének hibája százalékosan kifejezve, a specifikált értékhez képest? Okozhat e ez problémát? (Válaszát indokolja!)

   Az UART modul rate generátorjának előállításához számlálós osztót alkalmazunk. A számlálós osztó működési elve, hogy ismerve a referencia órajelet, és ismerve a kívánt órajelet, a kettő hányadosának egészre kerekített nagyságával egyező méretű (vagy nagyobb) regisztert hozunk létre, melynek értékét minden egyes felfutó élkor növeljük eggyel. Ha a regiszter eléri a kerekített hányados értékét, akkor 1-es logikai szintű jelet adunk ki. Ez lesz a leosztott rate generátor felfutó éle. Lássuk, mekkora hibát okoz a fent említett hányados kerekítése! A számláló optimális mérete: 16Mhz/(57600Hz=277,77…) Mivel csak egész számokat tudunk a számlálóval számolni, így a kerekített érték 278. Ezt visszaváltva frekvenciába: 16Mhz/(278=57553,3957Hz) A két frekvencia közötti bitidő eltérés: 1/57553,3957-1/57600=14.058ns

Ahhoz, hogy az adatkeret átvitele hibátlan legyen, a stop bit 16 mintavételi pontjából a 8-as és 9-es mintavételi időpontok még ténylegesen a stop bitből kell, hogy mintát vegyenek. Ebből és a bitidőből kiszámolhatjuk, hogy mekkora lehet egy keret esetén a maximális hibánk: A maximális hiba / keret: Bitidő×9/16=1/57553,3957×9/16=9,7735µs

Ebből a maximális hiba / bitidő, 11 bit átvitele esetén: 9.7735µs/11=888,503ns

A számlálónál tett kerekítésből adódó frekvenciahibából adódó bithiba, és a maximálisan megengedett bithiba aránya százalékosan kifejezve:

(számláló kerekítése miatti bithiba)/(maximálisan megengedett bithibas)=14,058ns/888,503ns×100=1.5822%

Ez, ha a vevő fél hibája 0%, akkor a specifikáció által ajánlott 2%-os hibahatáron belül van. (A specifikáció az adó és a vevő, nem pedig az adó és a clk közötti frekvenciahibához köti a hibahatárt)

Első 3 feladat személyre szabott!

4. Egy 115200 b/s, 8N1 beállítású UART átvitel esetén mekkora eltérés engedhető meg az adó és a vevő bitidejében, ha sikeres átvitelt szeretnénk?

   A vevőnek elvileg bitidőnként, a bitidő közepén kellene mintavételezni az adatokat. Mivel az adó és a vevő frekvenciája kicsit eltér, ezért ez nem teljesül. Ha a hiba 1 bitidő alatt dt (delay time), akkor ez az N-edik bit mintavételezésekor N*dt-re nő, ami akkor okoz hibát,

ha a legutolsó bitnél eléri a bitidő határát. Mivel a mintavételezés a bitidő közepén kezdődik, ezért N*dt < T/2-nek teljesülni kell (minden START bitnél újra szinkronizálódik a vevő). Mindez számokkal: 8N1 beállításból tudjuk, hogy 1 Start bit + 8 Adat bit + 1 Stop bit == 10 bitet kell átküldeni. (N = 10) 115200 b/s : 115200 bit megy át másodpercenként -> 1 bit: 1/115200 => 8.68 us. Képletbe behelyettesítve:

10 * dt < 8.68/2
dt < 0.434 us
(0.434/8.86)*100 = 5% eltérés.

5. A házi feladatként megvalósított UART adóval minimum mennyi időbe telne a neptun kódjának elküldése?

   (6*9)/(9600)
   x2 esetleg ASCII miatt? nope, már benne van
Startbit is benne van?

6. Egy 100 byte méretű fájl átvitele aszinkron soros csatornán 115200 bps sebesség mellett 8N1 (8 bit adat, nincs paritás, 1 STOP bit) formátumban minimum mennyi ideig tart?

   100 byte = 800 bit, byte-onként 1 stopbit ÉS EGY START BIT = 200 bit, összesen 1000 bit, 115200 bit/sec sebességgel ez 1000bit/(115200bit/s)=8,68 ms alatt megy át

7. Mi a 0001_0110 bináris szám paritásbitje az UART keretben 8E1 formátum esetén?

   1 (mert párosra kell, hogy kiegészítse a paritás bit, így 4 db 1-es lesz ami páros)

8. Egy UART adás esetén az átviendő adat: 0x42, formátuma: 8o1. Írja fel az UART keretben kiküldött biteket, balról az első kiküldött bittel kezdve.

   0(start),0,1,0,0,0,0,1,0,1(odd),1(stop)
   Ez pont szimmetrikus, de fontos hogy az lsb-t küldi először

9. A megismert soros adatátviteli módszerek körül melyiknél nem szükséges az órajelet is továbbítani?

   Az UART-nál, ott az adategységek azonosítása a lokális rendszerórák használatán alapul.

10. Melyik soros adatátviteli mód nem támogatja busz kialakítását?

    Az UART, mivel ott a kapcsolat pont-pont típusú. (pl. számítógép-modem, számítógép-terminál)

11. Mi alapján osztályozódik az SPI 4 átviteli módja?

Az adatok kiléptetése és a beérkezett adatok mintavételezése az órajel egymással ellentétes éleire történik. Ennek megfelelően, továbbá figyelembe véve az órajel kiindulási nyugalmi értékét is, alapvetően 4 fajta SPI átviteli ciklus definiálható.

SPI Mód	SCK periódus kezdő él	SCK periódus lezáró él
0	Mintavétel felfutó élre	Új adat kiadás lefutó élre
1	Új adat kiadás felfutó élre	Mintavétel lefutó élre
2	Mintavétel lefutó élre	Új adat kiadás felfutó élre
3	Új adat kiadás lefutó élre	Mintavétel felfutó élre

12. Hány vezeték kell 4 periféria SPI buszon történő használatához?

3+4 = 7 vezeték, mivel SPI esetén 3 + n kell, ahol n a perifériák száma. MISO, MOSI, SCK, SSn (n=0,1,2,3)

13. Írja fel a -42h értéket 2-es komplemens bináris formában 8 biten ábrázolva!

42h: 0100 0010
Negáljuk: 1011 1101
Hozzáadunk 1-t: 1011 1110 -42h
Vagy jobbról az első egyesig békénhagyjuk és utána negáljuk

14. Milyen formátumban küldi a TMP121 a hőmérséklet adatokat? Milyen hőmérsékletnek felel meg a 0x0C88 hexadecimális érték?

A TMP121 a **mért hőmérsékletet 16 biten, kettes komplemens formában** továbbítja úgy, hogy **az alsó 3 bit nem tartalmaz érdemi információt** (az alábbi táblázatban ezen bitek értéke 3'b000). Az adatformátumot az alábbi táblázat mutatja.

TEMPERATURE (°C)	DIGITAL OUTPUT(1) (BINARY)	HEX
150	0100 1011 0000 0000	4B00
125	0011 1110 1000 0000	3E80
25	0000 1100 1000 0000	0C80
0.0625	0000 0000 0000 1000	0008
0	0000 0000 0000 0000	0000
−0.0625	1111 1111 1111 1000	FFF8
−25	1111 0011 1000 0000	F380
−55	1110 0100 1000 0000	E480

0C80+0008 =25+0.0625=25.0625

15. Adja meg egy 16 bites, engedélyezhető balra léptető shift regiszter Verilog kódját. A modul bemenetei: órajel (clk), engedélyezés (en); a kimenete a shift regiszter aktuális állapota (shr).

```
module shiftboi(clk,rst,en,shr);

input clk,rst,en;
output reg [15:0]shr;


always @(posedge clk)
if(rst)
shr<=16'b0000000000000001;
else if(en)
shr<={shr[14:0],0};


endmodule
```

//Ez egy visszacsatolt shift regiszter, simát kértek
Rst szerintem nem kell, viszont egy adat bemenet kéne csak nem írja a bemeneteknél smh

Javítva

Dude ez mi?

Szerintem így kéne:

```
module shift_reg(
input clk, din,en
output [15:0] shr
);
reg [15:0] dout;
always @(posedge clk)
If(en)
dout <= {dout[14:0],din};
assign shr = dout;
endmodule
```

Ugyanaz mint az enyém, csak az enyém követi a specifikációt :P
Szerintem a felső a jó és kell az rst mert Daninak azért is szólták le a háziját hogy nincs rst pedig ugyan így volt mint itt hogy nem volt felsorolva de a srác azt mondta az kell

16. Adja meg egy 20 bites felfele számláló számláló Verilog kódját. A modul bemenetei: órajel (clk), reset (rst); a kimenete a számláló aktuális állapota (cntr).

```
module cntrboi(clk,rst,cnt);

input clk,rst;
output reg [19:0]cnt;


always @(posedge clk)
if(rst)
cnt<=0;
else
cnt<=cnt+1;


endmodule
```

Túlcsordulás meg ilyesmit nem kell nézni? ez csak számol és kész?
Neem, ez így tökéletes

17. Adja meg (assign értékadással) azon Vrilog kódsort, amely képes egy 14 bites kettes komplemens érték abszolút értékének kiszámítására. A bemeneti jel neve legyen

data_2comp, az abszolútérték jel neve pedig data_abs. Ez utóbbi jelet deklarálja is.

signed wire data_abs[13:0];
assign data_abs=~data_2comp+1;
Ez ennyi vagy elkúrtam?
El, nem írtál assign-t(már igen) és akkor is megnegálja ha pozitív a szám, nem tudom hogy kell mert assign blokkban ha jól emlékszem nem lehet if

signed wire data_abs[13:0];
always @ (posedge clk)
if(data_2comp[13])
data_2comp<=~data_2comp + 1;
assign data_abs=data_2comp;

így viszont nem egy sor :(

wire data_abs[13:0];
assign data_abs= (data_2comp[13]) ? (~data_2comp+1) : data_2comp;

utolsó ötlet, gg
Ez az utolsó a jó balázs is azt mondta