Laboratórium 1 felkészülési feladat

Hallgató:
Mérés sorszáma: 11

A mérési útmutatóban található segédlet alapján tervezzen egy UART adót!
A részletes specifikációt és a megvalósításhoz kapcsolódó javaslatokat mindenki figyelmesen olvassa el a Mérési útmutató 'A házi feladat elkészítésének szempontjai' című alfejezetben!
A soros átvitel paraméterei az alábbiak legyenek:
19200 bps. átviteli sebesség, 8 adatbit, Páratlan(O) paritás, 1 stop bit.
Ügyeljen arra, hogy az "Idle" állapot, ahonnan a kommunikáció indul, logikai 1 értékű legyen!
Kötelező kritériumok a beadandó anyagra: Ezek be nem tartása esetén a feladtot nem fogadjuk el!
Hozza el a teljes működő ISE projektet! Készítsen egy word (vagy pdf) dokumentumot, amiben szövegesen összefoglalja a kód felépítését, működését! Nem elég csupán a forráskódot bemásolni!
E dokumentumnak kötelezően tartalmaznia kell a működést igazoló szimulációs hullámformát és annak szöveges értelmezését. Az értelmezés nélküli képeket nem értékeljük, nem fogadjuk el.
A fájl kinyomtatása nem szükséges, ennél a mérésnél elektronikus dokumentációt kérünk.

A beadás tudnivalói:

- **Az önállóan kidolgozott feladatot a következő mérési gyakorlat elején a mérésvezetőnek kell bemutatni, - a mérési útmutatóban előírtak szerint - írott vagy elektronikus formában.**
- A felkészülési feladat utólag már nem adható be. Pótlására a szorgalmi időszak végén egy alkalommal, az adott mérési gyakorlat pótlásával egy időben van lehetőség.

A feladatokat önállóan, meg nem engedett segítség igénybevétele nélkül oldottam meg:

.....
aláírás

uart_module.v

```
`timescale 1ns / 1ps

module uart_module(
   input clk,
   input rst,
   input [3:0] bcd0,
   input [3:0] bcd1,
   output tx_out
   );

reg [5:0] tx_cntr;   // melyik fázisban tart a ciklus (hexadecimálisan)
reg [32:0] tx_shr;  // shiftregister, amibe az egy ciklus során küldendő három karaterhez tartozó összes bitet egy lépésben betölti
reg tx_en;       //órajel engedélyező bemenet
reg tx_reg;

reg [9:0] bd;    // az 19200 bit/sec sebesség előállításához(16MHz/19200)

assign tx_out = tx_reg;

always@(posedge clk) begin
          if(rst)
          begin
                    // reset jel (értékek alaphelyzetbe)
                    tx_en <= 0;
                    tx_shr <= 0;
                    bd <= 0;
                    tx_cntr <= 0;
                    tx_reg <= 1;

          end

          else
          begin
          // UART 'órajelének' előállítása
                    if(bd!=833) begin       // [16E6/19200-->833]
                              bd <= bd + 1;
                              tx_en <= 0;
                    end
                    else begin
                              bd <= 0;
                              tx_en <= 1;
                    end
          end
```

```
        if(tx_en) begin

                if(tx_cntr==0) begin
                //először a kocsivissza
                        tx_shr[32] <= 1;        //stopbit
                        tx_shr[31]      <= 0;   //paritásbit
                        tx_shr[30:23] <= 11'b00001101;  // kocsi vissza( decimálisan 13)
                        tx_shr[22] <= 0;        //kezdőbit


                //bcd0 (kétjegyű szám alsó helyiértéke)
                        tx_shr[21] <= 1;        //stopbit
                        tx_shr[20] <= bcd0[0]+bcd0[1]+bcd0[2]+bcd0[3]+1; //paritásbit
                        tx_shr[19:12] <= {4'b0011,bcd0[3:0]}; //hexa 30 ami ASCII ben a 0
                        tx_shr[11] <= 0;        //kezdő_bit


                //bcd1 (kétjegyű szám felső helyiértéke)
                        tx_shr[10] <= 1;        //stopbit
                        tx_shr[9] <= bcd1[0]+bcd1[1]+bcd1[2]+bcd1[3]+1;  //paritásbit
                        tx_shr[8:1] <= {4'b0011,bcd1[3:0]}; //hexa 30 ami ASCII ben a 0
                        tx_shr[0] <= 1'b0;      //kezdő_bit


                        tx_cntr <= 32; // számolja, hogy hány bit lett kiküldve
                        tx_reg <= 1;    //kimeneten 1
                end

                else
                begin
                        tx_reg <= tx_shr[0]; //0. bit a kimenetre
                        tx_shr <= tx_shr >> 1; // jobbra shift
                        tx_cntr <= tx_cntr - 1; // a hátralévő bitek száma csökken 1-gyel
                end
        end
end


endmodule
```

uart_test.v

```
`timescale 1ns / 1ps

module uart_test;

	// Inputs
	reg clk;
	reg rst;
	reg [3:0] bcd0;
	reg [3:0] bcd1;

	// Outputs
	wire tx_out;

	// Instantiate the Unit Under Test (UUT)
	uart_module uut (
		.clk(clk),
		.rst(rst),
		.bcd0(bcd0),
		.bcd1(bcd1),
		.tx_out(tx_out)
	);

	initial begin
		clk = 1;
		rst = 1;
		bcd0 = 0;
		bcd1 = 0;

		#100;
		rst=0;
		// Add stimulus here
		#1

		// születési dátum 11.22  hh/nn
		bcd1 = 2;
		bcd0 = 2;
		#500
		bcd1 = 1;
		bcd0 = 1;

	end
		always #1 clk=~clk;


endmodule
```

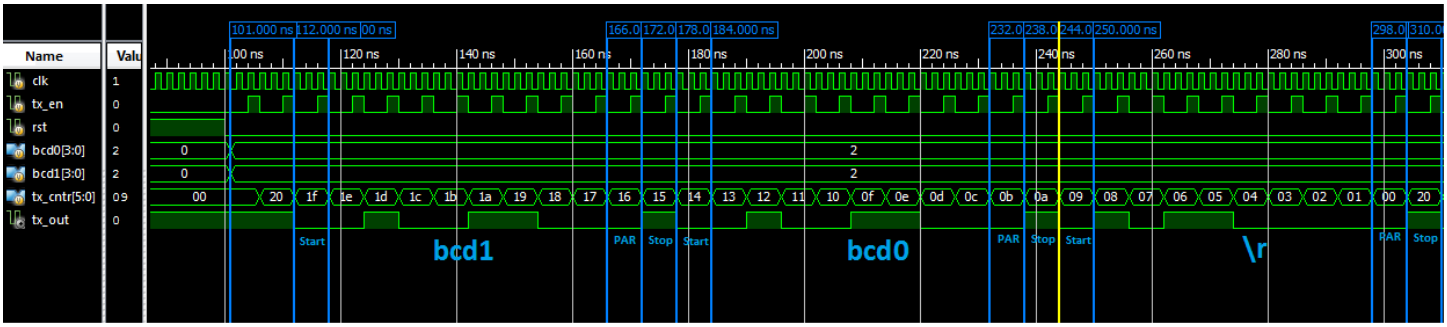
Rövid összefoglaló:

Az órajel felfutó élére ütemezünk.

Először előállítunk egy ütemezőt, ami engedélyezi minden 833. órajel ciklus után az UART-ot.
Reset jellel alaphelyzetbe állítjuk az értékeinket.
Számláló értékét vizsgáljuk:

Ha a belső számlálom 0-nál tart, akkor beolvassa a bcd0 és bcd1 bemeneteket és egy regiszterben tárolja a kimeneten előállítandó logikai értékeket. Egyébként pedig egyesével kiküldi. (LSB→MSB)

A működést igazoló szimulációs hullámforma:



Értelmezés:

A hullámforma létrehozásakor a baud-ot 833 helyett 2-nek választottam, hogy jól látható legyen, mit is csinál az UART-modul.

Az engedélyező jel után mindig kiküld egy bitet az UART, amit a shift-regiszterben tárol.

Az ábrán látható, hogy a tx_cntr számláló tárolja nekünk ,hogy épp melyik fázisban tart a ciklus.

Az adó folyamatosan 3 karaktert küld, a bemeneten BCD formában kapott két számjegy ASCII kódját és a kocsi-vissza karaktert.

Mivel én 11.22-én születtem, ezért a ’22\r’-t küldöm ki, aztán pedig a ’11\r’-t
22/r:

0|**0100 1100**|0|1||0|**0100 1100**|0|1||0|**1011 0000**|1|1		
bcd1	bcd0	\r
32h	32h	0Dh

Zöld: Start bit Kék: Paritás bit Piros: Stop bit

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?**

   1 startbit + 8 adatbit + 1 paritásbit +1 stopbit = 11 bit

   0|LSB … MSB|paritás|1

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 a 0 a 30h. A számokat pedig úgy kapjuk, hogy hozzáadjuk ezt.
   Pl.: 9 → 30h+9 → 0011 1001 (ahol az alsó négy bit változik csak)
   A paritásbit hibaészlelésre alkalmas. A számot egy bittel egészítjük ki, úgy hogy páros v. páratlan darab 1-es legyen benne.
   A legegyszerűbben XOR-kapuval számítható(, vagy akár egy olyan számlálóval, aminek a kimenete 1 bites).
   ABh → 1010 1011 mivel páratlan db 1-es van benne, ezért a paritásbit 1 páros paritás esetén.

3. **Ha a rendszer órajele 16MHz, mekkora a házi feladatként létrehozott UART modul sebességének hibája százalékosan kifejezve, a specifikált értékhez képest? Okozhat-e problémát?**

   Az UART-modul rate generátorjának előállításához számlálót használunk. A számláló működési elve, hogy ismerjük mind a referencia órajelet, mind pedig a kívánt órajelet, ezek hányadosának megfelelő méretű regisztert hozunk létre.

   Egy számláló folyton számlál, és ha eléri a hányados értékét, akkor logikai 1-es engedélyező jelet generál.

   Mivel a hányados nem minden esetben egész szám, így az órajel csúszik, vagyis hibája van. Ennek a hibának a mértéke:

   16Mhz/19200 = 833,33 →833

   16Mhz/833=19207,68307Hz

   A kettő közötti bitidő eltérés: $\frac{1}{19200} - \frac{1}{19207,68307} = 20,83332ns$

   Ahhoz, hogy a mintavétel ténylegesen hibátlan legyen, a stop bit 16mintavételi pontjából a 8-as és 9-es mintavételi időpontok még ténylegesen a stop bitből kell, hogy mintát vegyenek.

   A maximális hiba/keret: $\frac{1}{19207,68307\text{Hz}} * \frac{9}{16} = 29,2851us$

   A maximális hiba/ bitidő (11 bit esetén): $\frac{29,2851us}{11} = 2,66228us$

$$\frac{számláló\ kerekítéséből\ keletkező\ hiba}{maximálisan\ megengedett} = \frac{20,83332ns}{2,662281us} = 0,782\%$$