

# Sprawozdanie

## Język opisu sprzętu (HDL)

### Ćwiczenie 3: UART

Wykonane przez	
Imię i Nazwisko	Indeks
Miłosz Stasiak	240471
Filip Grzymski	240410

# 1 Zasada działania układu i wyjaśnienie sygnałów

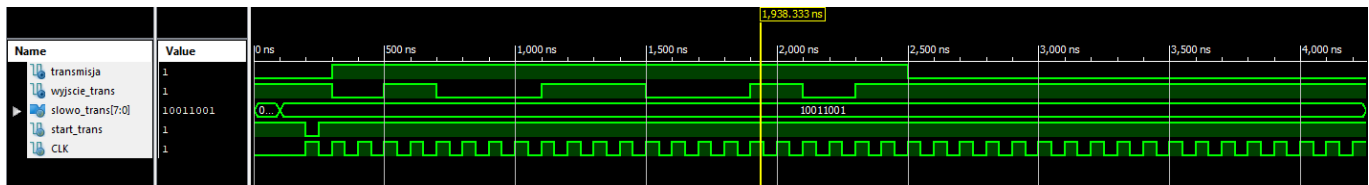
---

Zaprojektowany układ jest asynchroniczny z prędkością transmisji 2400Hz. Po rozpoczęciu transmisji nie można jej przerwać. Transmitowany jest ciąg 11 bitów w tym i każdy bit jest przesyłany co cykl zegarowy.

Na ciąg przesyłanych bitów ("zatrzasniete") składa się:

- jeden bit rozpoczęcia i jeden bit zakończenia transmisji
- osiem bitów treści (słowo-trans)
- jeden bit parzystości, sprawdzający czy transmisja była poprawna (w programie jest możliwość zmienienia bitu na "Even" lub "Odd")

Po zakończeniu transmisji zmienna "transmisja" spowrotem przyjmuje wartość 0 i układ jest gotowy na przesłanie kolejnego ciągu znaków.



Test układu przy pomocy symulacji

## 2 Schemat blokowy

---

Wstaw zdjęcie

## 3 Kod VerilogMain

---

### 3.1 | Całość kodu

---

```
'timescale 1ns / 1ps

module main(
    input [7:0] slowo_trans,
    input start_trans, input CLK,
    input czy_parz, input jaki_parz,
    output reg transmisja, output wyjscie_trans
);

reg CLK2380;
```

```

reg wyjscie;
reg [3:0] stan;
reg [10:0] zatrzasniete;
reg [5:0] stan_licznika;

initial
begin
    stan <= 0;
    stan_licznika <= 0;
end

always @(posedge CLK)
begin
    stan_licznika <= stan_licznika + 1;

    //if (stan_licznika == 6'b000001) //1
    if (stan_licznika == 6'b101010) //42
    begin
        stan_licznika <= 0;
        CLK2380 = 1'b1;
    end
    else
        CLK2380 = 1'b0;
    end
end

always @(slowo_trans or start_trans)
begin
    if(start_trans == 0)
    begin
        zatrzasniete[0] <= 0;
        zatrzasniete[8:1] <= slowo_trans[7:0];

        if (jaki_parz == 0)
            zatrzasniete[9] <= (^slowo_trans[7:0]);
        else
            zatrzasniete[9] <= (~^slowo_trans[7:0]);

        zatrzasniete[10] <= 1;
    end
end

always @(posedge CLK2380 or negedge start_trans)
begin
    if(start_trans == 0)
        stan <= 0;
    else if (stan == 4'b1001 && czy_parz == 0)
        stan <= stan + 2;
    else if (stan < 12)

```

```

        stan <= stan + 1;
end

always @(stan or zatrzasniete)
begin
    case (stan)
        4'b0000 : wyjscie = 1;
        4'b0001 : wyjscie = zatrzasniete[0];
        4'b0010 : wyjscie = zatrzasniete[1];
        4'b0011 : wyjscie = zatrzasniete[2];
        4'b0100 : wyjscie = zatrzasniete[3];
        4'b0101 : wyjscie = zatrzasniete[4];
        4'b0110 : wyjscie = zatrzasniete[5];
        4'b0111 : wyjscie = zatrzasniete[6];
        4'b1000 : wyjscie = zatrzasniete[7];
        4'b1001 : wyjscie = zatrzasniete[8];
        4'b1010 : wyjscie = zatrzasniete[9];
        default : wyjscie = zatrzasniete[10];
    endcase
end

always @(stan)
begin
    if(stan>0 && stan <12)
        transmisja = 1;
    else
        transmisja = 0;
    end
assign wyjscie_trans = wyjscie;
endmodule

```

---

## 3.2 | Wyjasnienie poszczególnych elementów kodu

Pierwszym elementem jest dzielnik sygnału zegarowego. Zmiana w "CLK2380" następuje co 42 cykle sygnału zegarowego, dzięki czemu wewnętrzny sygnał zegarowy wynosi dokładnie 2380 Hz.

---

```

always @(posedge CLK)
begin
    stan_licznika <= stan_licznika + 1;
    begin

        //if (stan_licznika == 6'b0000001) //1
        if (stan_licznika == 6'b101010) //42
        begin
            stan_licznika <= 0;
            CLK2380 = 1'b1;
        end
        else
            CLK2380 = 1'b0;
    end
end

```

```
end
end
```

---

Drugim elementem jest przypisanie odpowiednich bitów do później transmitowanej sekwencji (z możliwością wyboru czy posługujemy się bitem parzystości "Odd", czy "Even")

---

```
always @(slowo_trans or start_trans)
begin
    if(start_trans == 0)
    begin
        zatrzasniete[0] <= 0;
        zatrzasniete[8:1] <= slowo_trans[7:0];

        if (jaki_parz == 0)
            zatrzasniete[9] <= (^slowo_trans[7:0]);
        else
            zatrzasniete[9] <= (~^slowo_trans[7:0]);

        zatrzasniete[10] <= 1;
    end
end
```

---

kolejnym elementem jest przesyłanie bit po bicie transmisji, po zmianie stanu. Stan jest zależny od CLK2380. Możliwe jest również określenie czy zostanie przesłany bit parzystości czy nie.

---

```
always @(posedge CLK2380 or negedge start_trans)
begin
    if(start_trans == 0)
        stan <= 0;
    else if (stan == 4'b1001 && czy_parz == 0)
        stan <= stan + 2;
    else if (stan < 12)
        stan <= stan + 1;
end

always @(stan or zatrzasniete)
begin
    case (stan)
        4'b0000 : wyjscie = 1;
        4'b0001 : wyjscie = zatrzasniete[0];
        4'b0010 : wyjscie = zatrzasniete[1];
        4'b0011 : wyjscie = zatrzasniete[2];
        4'b0100 : wyjscie = zatrzasniete[3];
        4'b0101 : wyjscie = zatrzasniete[4];
        4'b0110 : wyjscie = zatrzasniete[5];
        4'b0111 : wyjscie = zatrzasniete[6];
        4'b1000 : wyjscie = zatrzasniete[7];
        4'b1001 : wyjscie = zatrzasniete[8];
        4'b1010 : wyjscie = zatrzasniete[9];
        default : wyjscie = zatrzasniete[10];
    end
end
```

```
    endcase  
end
```

---

## 4 Test układu

---

### 4.1 | symulacja w programie

---

```
'timescale 1ns / 1ps  
  
module symulacja;  
    // Inputs  
    reg [7:0] slowo_trans;  
    reg start_trans;  
    reg CLK;  
  
    // Outputs  
    wire transmisja;  
    wire wyjscie_trans;  
  
    // Instantiate the Unit Under Test (UUT)  
    main uut (  
        .slowo_trans(slowo_trans),  
        .start_trans(start_trans),  
        .CLK(CLK),  
        .transmisja(transmisja),  
        .wyjscie_trans(wyjscie_trans)  
    );  
  
    initial begin  
        // Initialize Inputs  
        slowo_trans = 0;  
        start_trans = 1;  
        CLK = 0;  
  
        // Wait 100 ns for global reset to finish  
        #100;  
  
        slowo_trans = 8'b10011001;  
  
        #100;  
  
        start_trans = 0;  
        CLK = 1;  
  
        #50;
```

```

start_trans = 1;
CLK = 0;

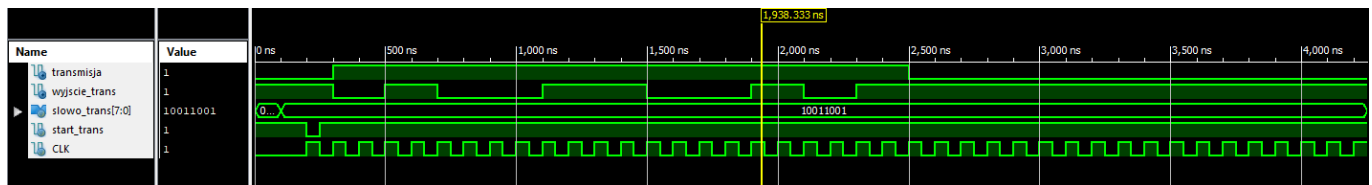
#50
CLK = 1;
#50
CLK = 0;
#50
CLK = 1;
#50
CLK = 0;
#50
CLK = 1;

...

#50
CLK = 0;

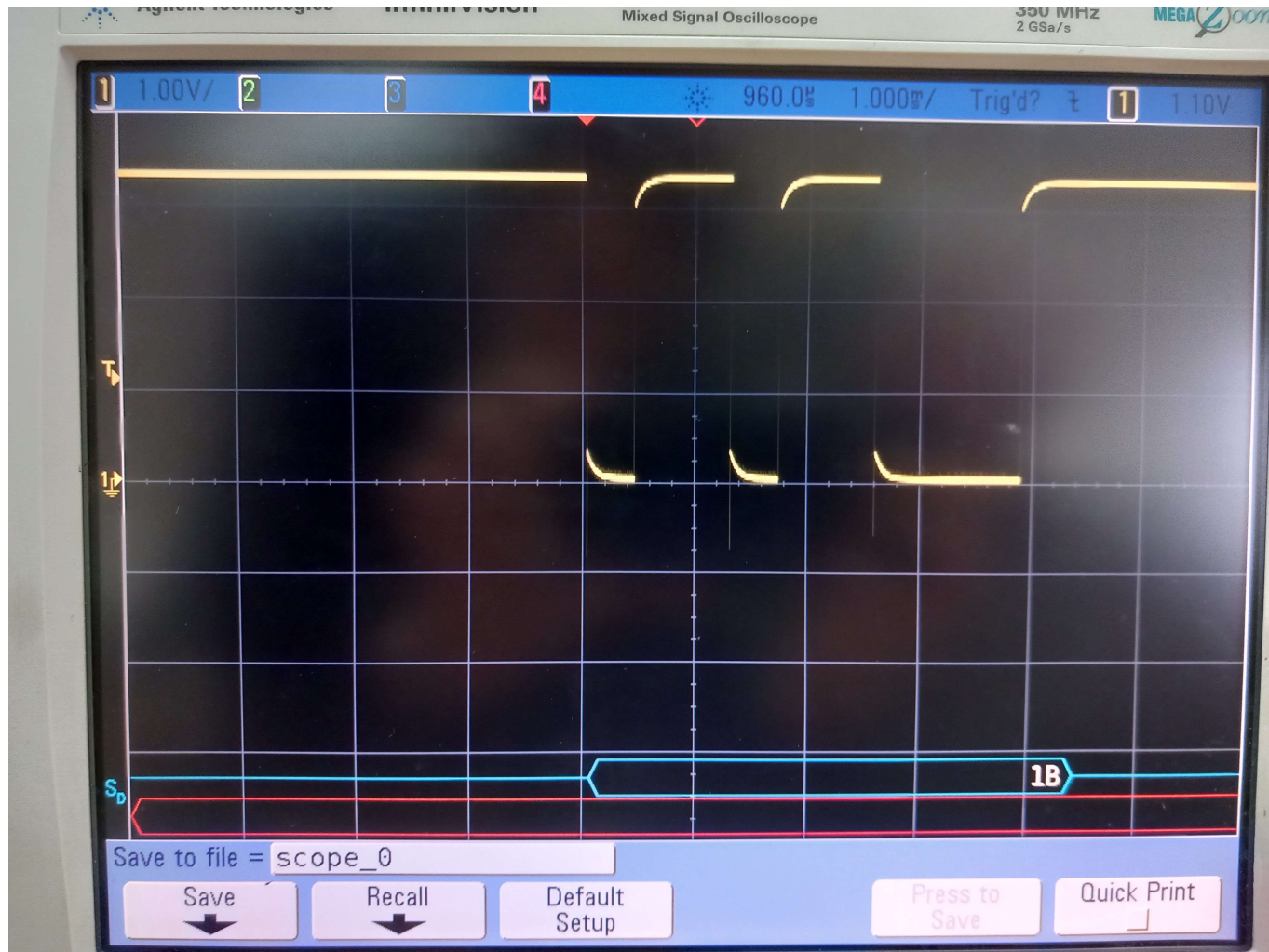
end
endmodule

```



Test układu przy pomocy symulacji

## 4.2 | Zdjęcia z testu na oscyloskopie



Test układu przy pomocy oscyloskopu