

Доопрацювати процесорне ядро MIPS та долучити до нього наступний периферійний пристрій

№	ДК-91	Тема розрахункової графічної роботи
1		Контроллер формування керуючих сигналів зовнішнім пристроєм
2		Контроллер генератору сигналів з протоколом 1-Wire
3		Контроллер передачі атмосферних параметрів протоколом SPI
4		Контроллер передачі власних даних протоколом I2C
5		Контроллер передачі числових даних протоколом UART
6		контроллер для передачі власних даних протоколом SPI
7		Контроллер передачі синусоїдального сигналу до ЦАП шиною UART
8		контроллер формування шим-сигналу по шині SPI
9		Контроллер передачі числових даних протоколом SPI для відображення на індикаторах
10		Контроллер генератору сигналів з протоколом I2C

Алгоритм:

- Адреса `uart_mem` починається з 32 і йде до 287 (255 + 32).
- Щоб почати надсилання потрібно взвести сигнал `start`

```
uart_sender.v U X
uart_sender.v
18     integer i;
19
20     always @(posedge i_start) begin
21         o_read_addres = 0;
22         data_is_sent = 1;
23     end
24
```

- Для цього потрібно в останню комірку пам'яті контролера записати нуль

```
core.v U    uart_mem.v U X
uart_mem.v
18     reg [31:0] mem [255:0];
19
20     initial $readmemh("uart_mem_init.dat", mem);
21
22     assign o_start = ~|mem[255];
23
```

- Використовую таку команду

```

tb_sim.v U X
tb_sim.v
114 */
115
116 /*
117 | | | RGR
118 | ac00011a - Op = 43 | Rs = 0 | Rt = 0 | Imm = 287
119 */

```

- Це єдина команда яка є в пам'яті

```

tb_sim.v U   rom_init.dat U X
rom_init.dat
1  ac00011f

```

- Після того як виконалась ця команда почнеться передача по юарту
- Спершу встановиться сигнал start
- Потім sender почне починаючи з нуля збільшувати значення на адресі читання uart\_mem
- При цьому sender буде діставати по 32 біти даних, тобто по 4 байти
- Передаватиме він по юарту кадри з розміром корисної інформації 8 біт
- Гармонічний сигнал я створив за допомогою сайту [Sine Look Up Table Generator Calculator \(daycounter.com\)](http://Sine Look Up Table Generator Calculator (daycounter.com)) та записав його в пам'ять контролера

```

19
20 initial $readmemh("uart_mem_init.dat", mem);
21

```

- Сигнал юарт формується наступним чином

```

27
28 uart_message = { 1'b1, i_data[7:0], 1'b0, 1'b1, i_data[15:8], 1'b0, 1'b1, i_data[23:16], 1'b0, 1'b1, i_data[31:24], 1'b0 };
29

```

- А ось який сигнал передається юартом (зачеркнув корисні біти)

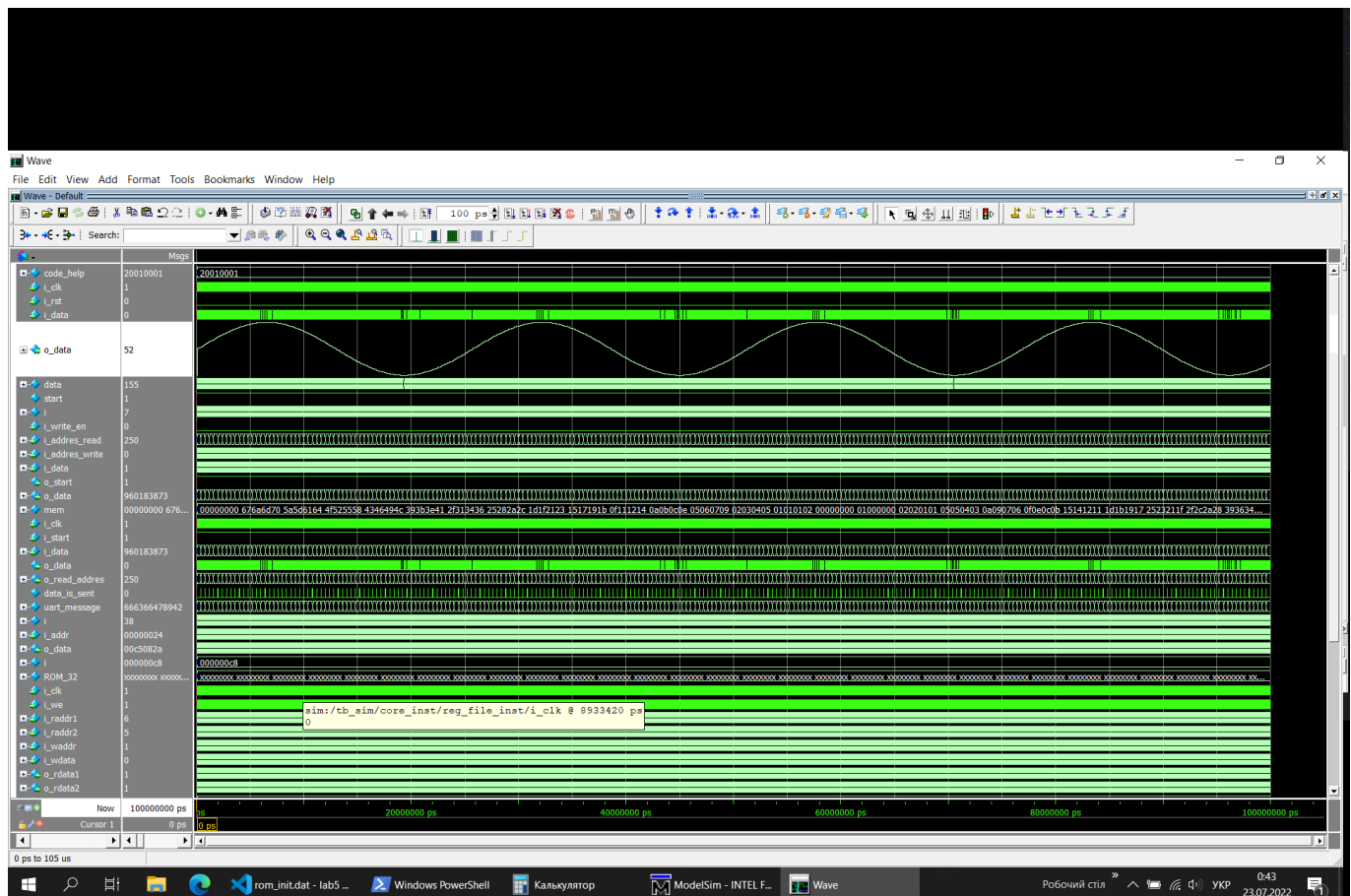
/tb_sim/core_inst/uart_sender_inst/o_read_address	250	6
/tb_sim/core_inst/uart_sender_inst/data_is_sent	0	
/tb_sim/core_inst/uart_sender_inst/uart_message	10011011001...	1110001000111000001011011111001101111000
/tb_sim/core_inst/uart_sender_inst/i	38	

- Ресівер ж приймає сигнал за рахунок чіткої синхронізації з сендером
- Негативний фронт означає старт біт і його ловить і починає прийом даних

```
uart_sender.v U    uart_receiver.v U X    core.v U
uart_receiver.v
17     integer i;
18
19     always @(i_rst) begin
20         start = 0;
21         i = 0;
22     end
23
24     always @(negedge i_data) begin
25         start = 1;
26     end
27
28     always @(posedge i_clk) begin
29         if (start && (i != 10)) begin
30             data = {data[8:0], i_data};
31             i = i + 1;
32         end
33         if (start && (i == 10)) begin
34             o_data[0] = data[8];
35             o_data[1] = data[7];
36             o_data[2] = data[6];
37             o_data[3] = data[5];
38             o_data[4] = data[4];
39             o_data[5] = data[3];
40             o_data[6] = data[2];
41             o_data[7] = data[1];
42             i = 0;
43         end
44     end
45
46 endmodule
```

- Також є ресет але для його використання слід знову оновити останню комірку пам'яті контролера
- Зауважу що сам процесор не задіяний в передачі даних і може спокійно виконувати іншу роботу
- Ось наприклад виконання сортування паралельно з ЛР 5 паралельно роботі контролера





- Для перевірки найпростіше запустити проект командою консолі `vsim -do sim.do`