

# 电子技术基础实验第十二周实验报告

王磊 2022012972

2023 年 12 月 12 日

## 1 task1

### 1.1 模块设计

#### 1.1.1 24位正弦数据生成

使用matlab生成用于初始化ROM的mif文件，其中存储了256个24位的正弦数据，代码如图1所示。

```
% 设置参数
width = 24; % 每个数据元素的位宽为24位
depth = 256; % 存储器的深度，即数据元素的数量为256个
%phase = 0; ..... % 相位初始为0，这里被注释掉
phase = 0; % 相位设置为0，即90度

% 打开文件并写入文件头信息
fid = fopen('sin_phase0_24bit.mif', 'w');
fprintf(fid, 'WIDTH=%d;\n', width);
fprintf(fid, 'DEPTH=%d;\n', depth);
fprintf(fid, 'ADDRESS_RADIX=DEC;\n');
fprintf(fid, 'DATA_RADIX=HEX;\n');
fprintf(fid, 'CONTENT BEGIN\n');

% 循环生成24位正弦波数据并写入文件
for i = 0:depth - 1
    sin_data = floor((sin(2 * pi * i / depth + phase) + 1) * 0.5 * (2 ^ width - 1));
    fprintf(fid, '%d:%x;\n', i, sin_data);
end

% 写入文件尾部信息并关闭文件
fprintf(fid, 'END :\n');
fclose(fid);
```

图 1: matlab代码

生成的mif文件内容如图2所示。

```
WIDTH=24;
DEPTH=256;
ADDRESS_RADIX=DEC;
DATA_RADIX=HEX;
CONTENT BEGIN
0:7fffff;
1:83242a;
2:8647d8;
3:896a8f;
4:8c8bd2;
5:8fab26;
6:92c80f;
7:95e213;
8:98f8b7;
9:9c0b81;
10:9f19f8;
11:a223a4;
12:a5280b;
13:a826b8;
14:ab1f34;
```

图 2: mif文件内容

### 1.1.2 顶层文件设计

顶层文件代码如下：

```
1      'include "fre_div.v"
2      'include "addr_tx_en.v"
```

```
3      'include "sin_24_rom.v"
4      'include "uart_NbyteTran_3byteData_controller.v"
5      'include "uart_tx_byte.v"
6      module task1_top(
7          input clk,
8          input rst,
9          output sci_tx
10     );
11
12     wire clk_div_addr;
13     fre_div uut1(
14         .clk(clk),
15         .rst(rst),
16         .clk_div_addr(clk_div_addr)
17     );
18
19     wire [7:0] addr;
20     wire send_en;
21     addr_tx_en uut2(
22         .clk_origin(clk),
23         .rst(rst),
24         .clk(clk_div_addr),
25         .addr(addr),
26         .send_en(send_en)
27     );
28
29     wire tx_done;
30     wire tx_en;
31     wire [23:0] data;
32     wire [7:0] tx_d;
33     wire send_done;
34     uart_NbyteTran_3byteData_controller uut3(
35         .clk(clk),
36         .rst(rst),
37         .send_en(send_en),
38         .tx_done(tx_done),
39         .tx_en(tx_en),
40         .data(data),
41         .tx_d(tx_d),
42         .send_done(send_done)
```

```

43 );
44
45 sin_24_rom uut4(
46     .address(addr),
47     .clock(clk),
48     .q(data)
49 );
50
51 uart_tx_byte uut5(
52     .clk(clk),
53     .rst(rst),
54     .tx_en(tx_en),
55     .rx_d(rx_d),
56     .sci_tx(sci_tx),
57     .tx_done(tx_done)
58 );
59
endmodule

```

对应的RTL图如图3所示。

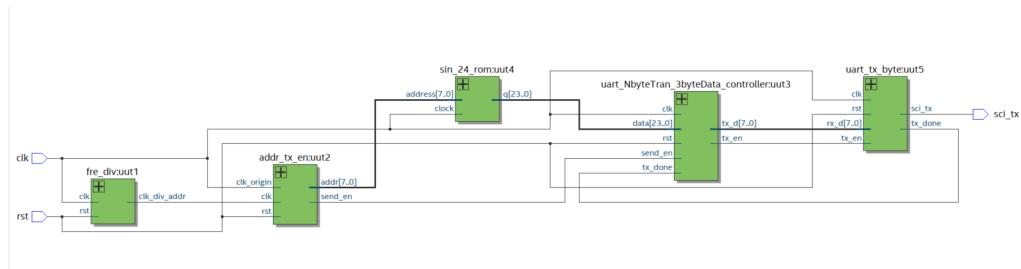


图 3: task1顶层模块RTL图

## 1.2 实验结果

按照任务要求将simulink设置如图4所示。接收结果如图5所示，可以看出数据成功发送。

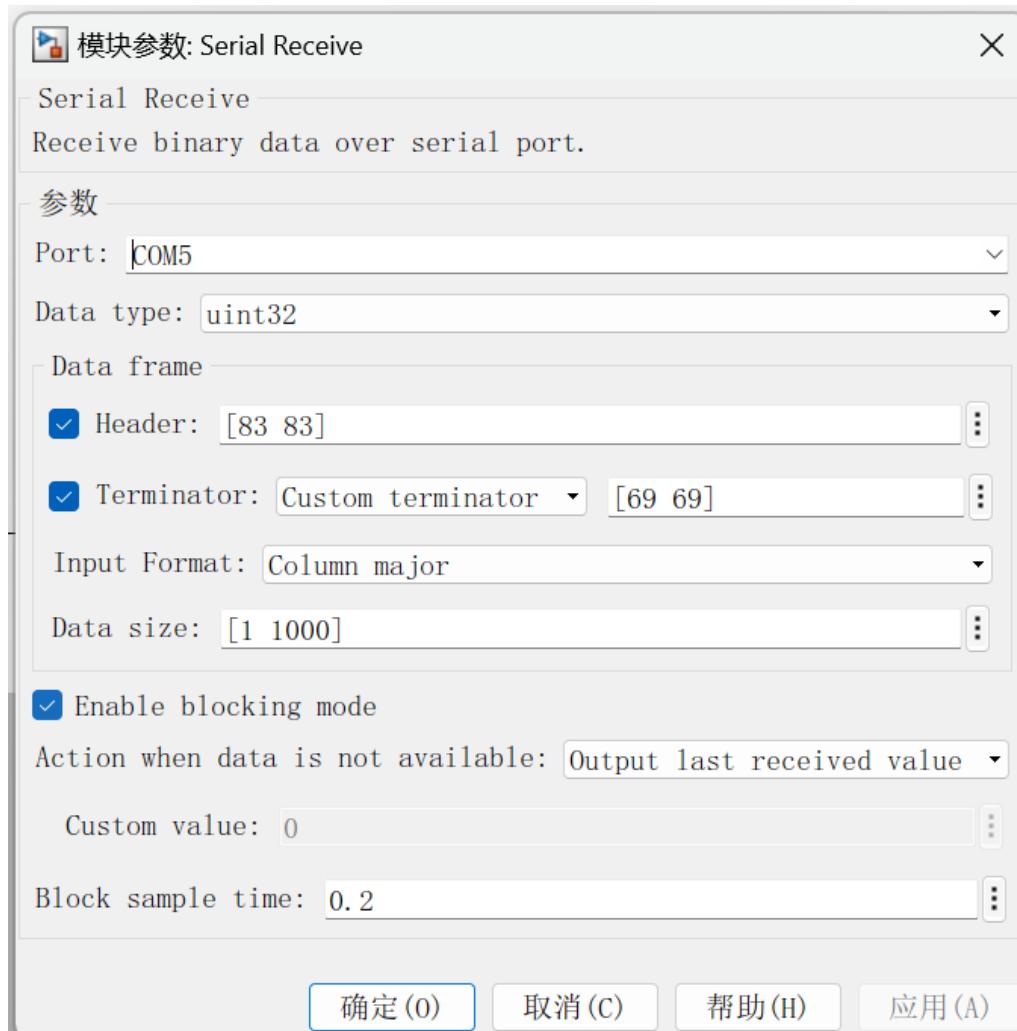


图 4: task1 simulink设置

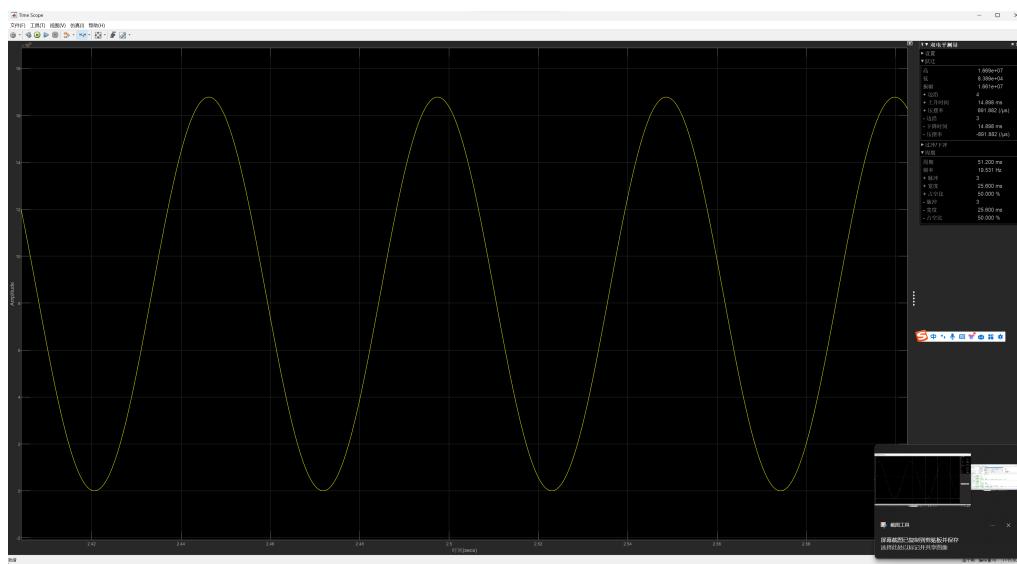


图 5: task1 接收结果

## 2 task2

### 2.1 模块设计

task2中使用的24位正弦波数据与task1中使用的24位正弦波数据相同，可以直接使用。

#### 2.1.1 地址模块

地址模块代码如下：

```
1 module rom_data_tri (
2     input lr_ch_tri_clk,
3     input rst,
4     output reg [7:0] addr_chL,
5     output reg [7:0] addr_chR
6 );
7     always @ (posedge lr_ch_tri_clk or posedge rst) begin
8         if (rst) begin
9             addr_chL <= 8'd0;
10        end else begin
11            if (addr_chL == 8'd255) begin
12                addr_chL <= 8'd0;
13            end else begin
14                addr_chL <= addr_chL + 1'b1;
15            end
16        end
17    end
18
19    always @ (negedge lr_ch_tri_clk or posedge rst) begin
20        if (rst) begin
21            addr_chR <= 8'd0;
22        end else begin
23            if (addr_chR >= 8'd255) begin
24                addr_chR <= 8'd0;
25            end else begin
26                addr_chR <= addr_chR + 2'd2;
27            end
28        end
29    end
30
```

```
31 endmodule
```

地址模块将DAC传出的lrck\_dac信号作为时钟信号，每次上升沿时，addr\_chL自增1，addr\_chR自增2。因此右通道的频率为左通道的二倍。<sup>1</sup>

### 2.1.2 顶层模块

顶层模块代码如下：

```

1 'include "rom_data_tri.v"
2 'include "rom_array_sync.v"
3 'include "dac_controller.v"
4 'include "fre_div.v"
5 module task2_top(
6     input clk,
7     input rst,
8     output lrck_dac,
9     output sclk_dac,
10    output mclk_dac,
11    output sdata_dac
12 );
13
14 wire[23:0] data_dac_chL;
15 wire[23:0] data_dac_chR;
16 wire[7:0] addr_chL;
17 wire[7:0] addr_chR;
18 wire c0; //6553600hz
19 wire c1; //13107200hz
20
21 rom_data_tri uut1(
22     .lr_ch_tri_clk(lrck_dac),
23     .rst(rst),
24     .addr_chL(addr_chL),
25     .addr_chR(addr_chR)
26 );
27
28 rom_array_sync uut2(

```

<sup>1</sup>要想实现两路信号频率不同应该有更好的方法，但我修改时钟的方法无法实现，会产生无规律的波形

```
29     .clock(clk),
30     .address(addr_chL),
31     .q(data_dac_chL)
32 );
33
34 rom_array_sync uut3(
35     .clock(clk),
36     .address(addr_chR),
37     .q(data_dac_chR)
38 );
39
40 dac_controller uut4(
41     .clk(c0),
42     .rst(rst),
43     .data_dac_chL(data_dac_chL),
44     .data_dac_chR(data_dac_chR),
45     .mclk_dac(mclk_dac),
46     .sclk_dac(sclk_dac),
47     .lrck_dac(lrck_dac),
48     .sdata_dac(sdata_dac)
49 );
50
51 fre_div uut5(
52     .inclk0(clk),
53     .c0(c0)
54 );
55 endmodule
```

其中fre\_div为使用IP核生成的频率分频器，将clk分频为6553600hz作为DAC的时钟信号，从而实现DAC输出的音频信号频率为 $6553600\text{hz}/256/256=100\text{Hz}$ 。

对应的RTL图如图6所示。

## 2.2 实验结果

示波器测量的DAC输出信号如图7所示，可以看出两路信号频率为二倍关系，一个为100Hz，一个为50Hz。

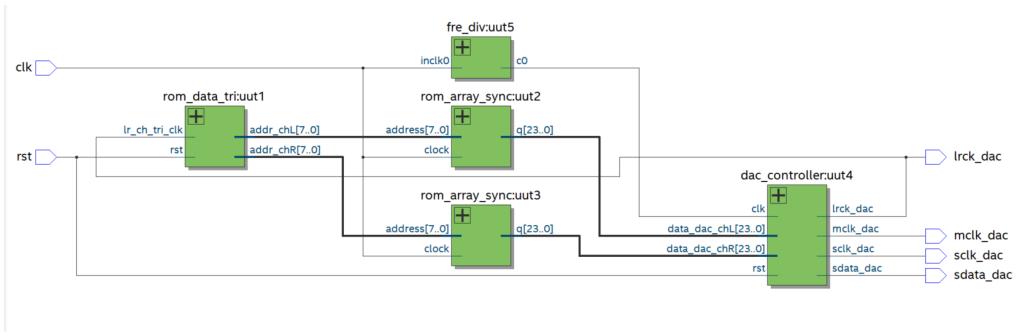


图 6: task2顶层模块RTL图

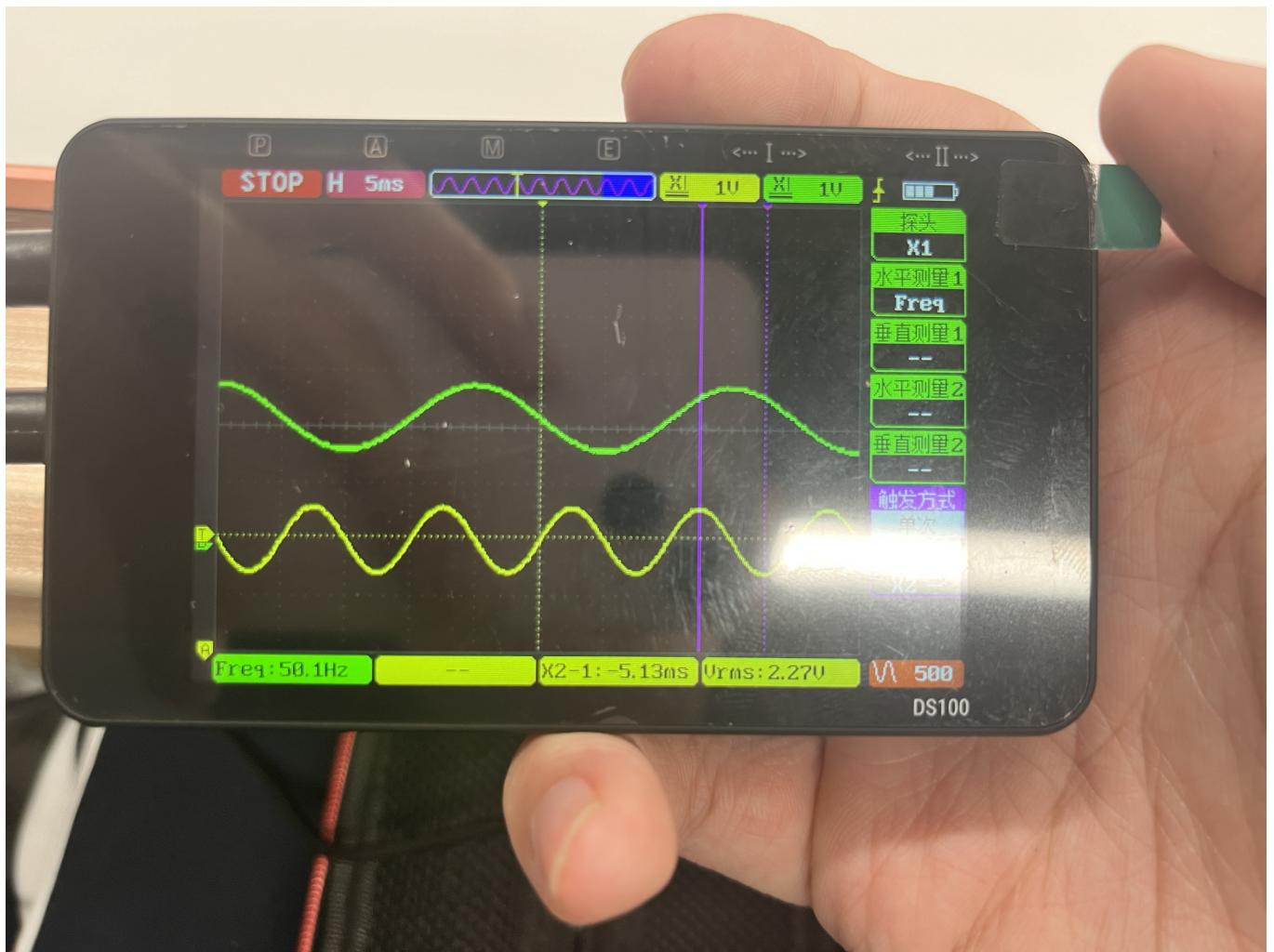


图 7: task2 DAC输出信号