

```

1  `timescale 1ns / 1ps
2
3  module uart_top
4  #(
5  parameter clk_freq = 1000000,
6  parameter baud_rate = 9600
7  )
8  (
9      input clk,rst,
10     input rx,
11     input [7:0] dintx,
12     input send,
13     output tx,
14     output [7:0] doutrx,
15     output donetx,
16     output donerx
17     );
18
19     uarttx
20     #(clk_freq, baud_rate)
21     utx
22     (clk, rst, send, dintx, tx, donetx);
23
24     uartrx
25     #(clk_freq, baud_rate)
26     rtx
27     (clk, rst, rx, donerx, doutrx);
28
29
30 endmodule
31
32
33 ///////////////////////////////////////////////////
34
35 module uarttx
36 #(
37 parameter clk_freq = 1000000,
38 parameter baud_rate = 9600
39 )
40 (
41 input clk,rst,
42 input send,
43 input [7:0] tx_data,
44 output reg tx,
45 output reg donetx
46 );
47
48     localparam clkcount = (clk_freq/baud_rate); ///x
49
50     integer count = 0;
51     integer counts = 0;
52
53     reg uclk = 0;
54
55     enum bit[1:0] {idle = 2'b00, start = 2'b01, transfer = 2'b10, done = 2'b11} state;
56
57     ////////////uart_clock_gen
58     always@(posedge clk)
59     begin
60         if(count < clkcount/2)
61             count <= count + 1;
62         else begin
63             count <= 0;
64             uclk <= ~uclk;
65         end
66     end
67
68
69     reg [7:0] din;

```

```

70 //////////////////////////////////////////////////Reset decoder
71
72
73 always@(posedge uclk)
74     begin
75         if(rst)
76             begin
77                 state <= idle;
78             end
79         else
80             begin
81                 case(state)
82                     idle:
83                         begin
84                             counts <= 0;
85                             tx <= 1'b1;
86                             donetx <= 1'b0;
87
88                             if(send)
89                                 begin
90                                     state <= transfer;
91                                     din <= tx_data;
92                                     tx <= 1'b0;
93                                 end
94                             else
95                                 state <= idle;
96                         end
97
98
99
100         transfer: begin
101             if(counts <= 7) begin
102                 counts <= counts + 1;
103                 tx <= din[counts];
104                 state <= transfer;
105             end
106             else
107                 begin
108                     counts <= 0;
109                     tx <= 1'b1;
110                     state <= idle;
111                     donetx <= 1'b1;
112                 end
113             end
114
115
116
117
118         default : state <= idle;
119     endcase
120 end
121 end
122
123 endmodule
124
125
126
127 //////////////////////////////////////////////////
128
129
130
131
132 module uartrx
133     #(
134     parameter clk_freq = 1000000, //MHz
135     parameter baud_rate = 9600
136     )
137     (
138     input clk,

```

```

139 input rst,
140 input rx,
141 output reg done,
142 output reg [7:0] rxddata
143 );
144
145 localparam clkcount = (clk_freq/ baud_rate);
146
147 integer count = 0;
148 integer counts = 0;
149
150 reg uclk = 0;
151
152
153 enum bit[1:0] {idle = 2'b00, start = 2'b01} state;
154
155 //////////////uart_clock_gen
156 always@(posedge clk)
157     begin
158         if(count < clkcount/2)
159             count <= count + 1;
160         else begin
161             count <= 0;
162             uclk <= ~uclk;
163         end
164     end
165
166
167
168 always@(posedge uclk)
169     begin
170         if(rst)
171             begin
172                 rxddata <= 8'h00;
173                 counts <= 0;
174                 done <= 1'b0;
175             end
176         else
177             begin
178                 case(state)
179
180                     idle :
181                     begin
182                         rxddata <= 8'h00;
183                         counts <= 0;
184                         done <= 1'b0;
185
186                         if(rx == 1'b0)
187                             state <= start;
188                         else
189                             state <= idle;
190                     end
191
192                     start:
193                     begin
194                         if(counts <= 7)
195                             begin
196                                 counts <= counts + 1;
197                                 rxddata <= {rx, rxddata[7:1]};
198                             end
199                         else
200                             begin
201                                 counts <= 0;
202                                 done <= 1'b1;
203                                 state <= idle;
204                             end
205                         end
206
207

```

```
208     default : state <= idle;
209
210     endcase
211
212 end
213
214 end
215
216 endmodule
217
218
219 ///////////////////////////////////////////////////////////////////
220
221 interface uart_if;
222     logic clk;
223     logic uclktx;
224     logic uclkrx;
225     logic rst;
226     logic rx;
227     logic [7:0] dintx;
228     logic send;
229     logic tx;
230     logic [7:0] doutrx;
231     logic donetx;
232     logic donerx;
233
234 endinterface
```