```
module UART_FPGA_TX
  #(
   parameter UART_BAUD_RATE=9600,//baud
   parameter CLOCK_FREQUENCY=50000000,//IN_CLOCK frequency
   parameter PARITY=1,
   //parameter of parity bit in package
   //PARITY==0 :  package without parity bit
   //PARITY==1 :  package contains parity bit
   //PARITY==2 :  package contains odd bit
   parameter NUM_OF_DATA_BITS_IN_PACK=8,
   //number of data bits in package
   parameter NUMBER_STOP_BITS=2,
   //number of stop bits in package
   parameter CLKS_PER_BIT_LOG_2=$clog2(CLOCK_FREQUENCY/UART_BAUD_RATE),
   //the number of bits for the register of the main counter
   parameter NUM_OF_DATA_BITS_IN_PACK_LOG_2=$clog2(NUM_OF_DATA_BITS_IN_PACK)
   //the number of bits for the register of bit counter
   )
  (
   input                                  IN_CLOCK,          //input clock
   input                                  IN_TX_LAUNCH,      //input launch port
   input [NUM_OF_DATA_BITS_IN_PACK-1:0]   IN_TX_DATA,        //input data package for transmit
   output reg                             OUT_TX_ACTIVE,     //output TX bus active
   output reg                             OUT_TX_SERIAL,     //output TX port
   output reg                             OUT_TX_DONE,       //briefly set when packet transfer ends
   output reg                             OUT_STOP_BIT_ACTIVE, //set when a start bit is transmitted
   output reg                             OUT_START_BIT_ACTIVE //set when a stop bit is transmitted
   );
  //finit state machine (FSM)
  localparam CLKS_PER_BIT = CLOCK_FREQUENCY/UART_BAUD_RATE ;
  //the number of IN_CLOCK cycles of the main generator
  //for the transmission of one data bit

  localparam STATE_WAIT         = 3'b000;//wait set IN_TX_LAUNCH
  localparam STATE_TX_START_BIT = 3'b001;//transmit start bit
  localparam STATE_TX_DATA_BITS = 3'b010;//transmit data bits
  localparam STATE_PARITY_BIT   = 3'b101;//transmit parity bit
  localparam STATE_TX_STOP_BIT  = 3'b011;//transmit stop bit

  reg [2:0]                              REG_STATE;//FSM state register
  reg [CLKS_PER_BIT_LOG_2:0]             REG_CLOCK_COUNT;//main counter register
  reg [NUM_OF_DATA_BITS_IN_PACK_LOG_2:0] REG_BIT_INDEX;//bit counter register
  reg [NUM_OF_DATA_BITS_IN_PACK-1:0]     REG_TX_DATA;//transmit data package register
  reg                                    REG_FLAG_DONE_TRANSACTION;


  initial begin
      //initial internal registers
      REG_STATE                 = STATE_WAIT;
      REG_CLOCK_COUNT           = 0;
      REG_BIT_INDEX             = 0;
      REG_TX_DATA               = 0;
      REG_FLAG_DONE_TRANSACTION = 0;

      //initial output registers
      OUT_TX_ACTIVE             = 0;
      OUT_TX_SERIAL             = 1;
      OUT_TX_DONE               = 0;
      OUT_STOP_BIT_ACTIVE       = 0;
      OUT_START_BIT_ACTIVE      = 0;
  end

  always @(posedge IN_CLOCK)
  begin
      case (REG_STATE)
        STATE_WAIT ://состояние ожидания
        begin
            /*
            формирование короткого (на один такт основного генератора)
            импульса- индикатора
            окончания элементарной транзакции
            */
            if(REG_FLAG_DONE_TRANSACTION==1)
```

```
74                begin
75                   OUT_TX_DONE                <= 1'b1;
76                   OUT_TX_ACTIVE              <= 1'b0;
77                   REG_FLAG_DONE_TRANSACTION  <= 0;
78                end
79                else
80                   OUT_TX_DONE                <= 1'b0;
81
82                OUT_STOP_BIT_ACTIVE           <= 0;
83                OUT_TX_SERIAL                 <= 1'b1;
84                if (IN_TX_LAUNCH == 1'b1) //Начало передачи данных
85                begin
86                   OUT_TX_DONE                <= 1'b0;
87                   OUT_TX_ACTIVE              <= 1'b1;
88                   REG_TX_DATA                <= IN_TX_DATA;//защелкивание данных
89                   REG_STATE                  <= STATE_TX_START_BIT;//смена состояния автомата
90                   OUT_TX_SERIAL              <= 1'b0;//линия TX притягивается к нулю в
    соответствии с протоколом
91                   OUT_START_BIT_ACTIVE       <= 1;
92                end
93             end
94             STATE_TX_START_BIT ://состояние передачи старт-бита (линия держится в нуляе на
    время одного тайм-слота)
95             begin
96                if (REG_CLOCK_COUNT < CLKS_PER_BIT-2)//классический счетчик
97                   REG_CLOCK_COUNT <= REG_CLOCK_COUNT + 1'b1;
98                else
99                begin
100                  REG_CLOCK_COUNT   <= 0;
101                  REG_STATE         <= STATE_TX_DATA_BITS; //по окончании счета смена
    состояния автомата
102               end
103            end
104            STATE_TX_DATA_BITS ://состояние передачи информационных битов
105            begin
106               OUT_START_BIT_ACTIVE <= 0;
107               OUT_TX_SERIAL <= REG_TX_DATA[REG_BIT_INDEX];//заряжается/держится конкретный
    бит на линию TX
108               if (REG_CLOCK_COUNT < CLKS_PER_BIT-1) //классический счетчик
109                   REG_CLOCK_COUNT <= REG_CLOCK_COUNT + 1'b1;
110               else
111               begin
112                  REG_CLOCK_COUNT <= 0;
113                  if (REG_BIT_INDEX < NUM_OF_DATA_BITS_IN_PACK-1'b1)
114                  /*
115                  По окончаии счета идет проверка номера бита в пакете данных.
116                  Если номер не больше, чем уговоренный объем пакета,
117                  то индекс бита увеличится
118                  */
119                     REG_BIT_INDEX <= REG_BIT_INDEX + 1'b1;
120                  else
121                  begin
122                     /*
123                     Если был отправлен последний бит,
124                     то посылка информационных бит окончена.
125                     В таком случае индекс битов обнуляется.
126                     Далее состояние автомата напрямую зависит от параметра
127                     PARITY. Автомат далее переходит либо в состояние
128                     передачи стоп-бита
129                     */
130                     REG_BIT_INDEX <= 0;
131                     if(PARITY==0)
132                        REG_STATE   <= STATE_TX_STOP_BIT;
133                     else
134                        REG_STATE   <= STATE_PARITY_BIT;
135                  end
136               end
137            end
138            STATE_PARITY_BIT://состояние отправки старт- бита
139            begin
140               case(PARITY) //на линию ставится бит в зависимости от параметра четности
141                  1: OUT_TX_SERIAL <= (sum_of_bits(REG_TX_DATA)%2==0)? 0:1'b1;
142                  2: OUT_TX_SERIAL <= (sum_of_bits(REG_TX_DATA)%2==0)? 1'b1:0;
143               endcase
144               if (REG_CLOCK_COUNT < CLKS_PER_BIT-1) //один тайм-слот
145                  REG_CLOCK_COUNT   <= REG_CLOCK_COUNT + 1'b1;
146               else
147               begin
148                   REG_CLOCK_COUNT  <= 0;
```

```
149                     REG_STATE         <= STATE_TX_STOP_BIT;
150                 end
151             end
152             STATE_TX_STOP_BIT ://состояние отправки стоп-бита
153             begin
154                 OUT_STOP_BIT_ACTIVE<=1;
155                 OUT_TX_SERIAL <= 1'b1;//На линию ставится высокий сигнал (1)
156                 if (REG_CLOCK_COUNT < CLKS_PER_BIT*NUMBER_STOP_BITS-1'b1) //классический
    счетчик
157                     REG_CLOCK_COUNT <= REG_CLOCK_COUNT + 1'b1;
158                 else
159                 begin
160                     REG_FLAG_DONE_TRANSACTION     <= 1;//транзакция считается законченной
161                     REG_CLOCK_COUNT               <= 0;//счетчик обнуляется
162                     REG_STATE                     <= STATE_WAIT;//автомат переходит в
    состояние ожидания
163                 end
164             end
165             default :
166               REG_STATE <= STATE_WAIT; //при аномальном уровне REG_STATE.
167         endcase
168       end
169       function [NUM_OF_DATA_BITS_IN_PACK_LOG_2:0] sum_of_bits;//функция суммирует компоненты
    регистра
170       //this function sums the bits in a register
171           input [NUM_OF_DATA_BITS_IN_PACK-1:0] value;
172           reg[NUM_OF_DATA_BITS_IN_PACK_LOG_2:0] sum;
173           reg[NUM_OF_DATA_BITS_IN_PACK_LOG_2:0] i;
174           begin
175              sum=0;
176              for (i=0;i<=NUM_OF_DATA_BITS_IN_PACK-1'b1;i=i+1'b1)
177                 sum=sum+value[i];
178              sum_of_bits=sum;
179           end
180       endfunction
181   endmodule
182
```