Verilog Experiment 1 for SOC student

Silenuszhi

December 28, 2017

1 Uart Excercise

Deadline : Feb 28th 2018

In this section, you need to write a uart transmit(tx) module, a uart receive(rx) module, and a frame receive module to unpack GPS $GPGGA data, You should write the code and finish the simulation.

If you want to stay in SoC group: Always Do 'Must do'

Must do:

a)!You must use a enhanced text editor,such as notepad++,vi,sublime,emacs,etc.Choose the one you like.But at least,the editor should has the code highlight.

b)!!!You must use clk/clock as the clock signal's name, use resetn/reset_n/rst_n/rstn as the reset signal's name,all reset should be active low in our lab.Besides,if a data signal is active low, you should add _n or n or _b or b at the end of the signal, _n is preferred

c)!!!!You should never use "signal1","signal2" as your signal name

d)!!You must choose Modelsim/VCS as your verilog compiler,and use Modelsim/Verdi to watch simulation waveform, or you will not get help from others in our lab. We do only use Modelsim and Synopsys tools

e)!!!You must write a script to compile the source file and do simulation.

Should do:

a)The editor may also have auto complete and auto indent

b)Try a better font so you can distinguish 0 and O

1.1 Uart TX Module

The port definition is given below, Find out what uart is and write a uart module.
0) clk is 100MHz
1) clk_en signal is generated by a module called clk_divisor,the signal is going to be asserted one clock period according to the baudrate 115200,The actual baudrate used in system could have an at most 3% error.And you should generate the signal in your testbench
2) Datain and shoot signal means when tx_module is not busy and shoot is asserted,then send the datain serial out, shoot must be one clock valid and datain is only valid at that period
3) Dart_busy signal means whether the module is ready to accept a new shoot
4) As parameter, Verify_on and Verify_even set the verify method of the uart module
5) Write the code and testbench, do the simulation, and write a simple write up to explain what you have done

```
module uart_tx_op
  #(
    parameter VERIFY_ON = 1'b0,
    parameter VERIFY_EVEN = 1'b0
```

```
    )
   (
    input         clk_i,
    input         resetn_i,
    input         clk_en_i,
    input [7:0]   datain_i,
    input         shoot_i,
    output reg    uart_tx_o = 1'b1,
    output reg    uart_busy_o
    );
```

1.2 Uart RX Module

The port definition is given below.
1) Becareful every clk that dataout_valid_o asserts means a valid dataout_o.
2) You can use the Uart Tx Module help you to verify this one.
3) Write the code and testbench, do the simulation, and write a simple write up to explain what you have done

```
module uart_rx
  #(
    parameter VERIFY_ON = 1'b0,
    parameter VERIFY_EVEN = 1'b0
    )
   (
    input              clk_i,
    input              clk_en_i,
    input              resetn_i,
    input              uart_rx_i,
    output reg         dataout_valid_o,
    output reg [7:0]   dataout_o
    );
```

1.3 rx_package module

notation
1) start of frame (sof)
2) end of frame (eof)

scenario
1) The actual gps module output signal is mixed several types of package,like $GPGGA and $GPZDA etc, we are going to pick up a specific one.
2) When we use custom frame, there will use an fixed length of frame, we are going to pick up the fixed frame with one module as well.

Problems
The module is capable of handle a uart receive data stream use sof+eof mode or sof+framecnt mode and pickup(fixed position and length), substitute some words(fixed position and length) and output the

valid frame.
The port definition is given below:
1) If use sof+eof mode, set the EOFDETECTION to 1
2) If use sof+framecnt mode, set the FRAMELENGTHFIXED to 1
3) At first the module is coded to connect to 2 fifo, so that the toggle signal can be used to know which fifo is writen to.
4) sub_data_valid is one clock period assert
5) Other parameter is clear enough
6) Write the code and testbench, do the simulation, and write a simple write up to explain what you have done

```
module rx_package
  #(
    parameter TOGGLE = 1'b1, //use pingpong buffer
    parameter SOFLENGTH = 2, //sof length
    parameter SOFPATTERN = 16'hEB90,//sof pattern
    parameter EOFDETECTION = 1'b1,//eof detect or not
    parameter EOFLENGTH = 2,//eof length
    parameter EOFPATTERN = 16'h90EB,//eof pattern
    parameter FRAMELENGTHFIXED = 1'b1, //if 1,frame is fixed length
    parameter FRAMECNT = 64,//framecnt
    parameter SUB = 1'b1,//substitution
    parameter SUBPOS = 2,
    parameter SUBLENGTH = 8,//substitution length,
// if not used, leave unconnected, will gen a warning
    parameter PICKPOS = 8,
    parameter PICKLENGTH = 3 //pick some bytes output
    )
   (
    input                           clk,
    input                           reset,
    input                           enable,

    input                           rx_data_valid,
    input [7:0]                     rx_data,

    input                           sub_data_valid,
    input [SUBLENGTH*8-1:0]         sub_data,

    output [PICKLENGTH*4-1:0]       pick_data,
    output reg                      pick_data_valid,
    output                          FIFO_clear,

    output reg [TOGGLE:0]           frame_datavld = 0,
    output reg [7:0]                frame_data = 0,
    output reg [10:0]               frame_count = 0,
    output reg [TOGGLE:0]           frame_interrupt = 0
    );
```

2 code for clk_divider

```
module clk_divider
  #(parameter DIVISOR = 6'd0)
   (
    input       clk_i,
    input       resetn_i,
    output reg clk_en_o
    );

   reg [5:0]    clk_dividor = 0;

   // clk divider
   always @( posedge clk_i or negedge resetn_i ) begin
      if(!resetn_i)begin
         /*AUTORESET*/
         // Beginning of autoreset for uninitialized flops
         clk_dividor <= 6'h0;
         clk_en_o <= 1'h0;
         // End of automatics
      end else begin
         if (clk_dividor != DIVISOR) begin
            clk_dividor <= clk_dividor + 1'b1;
            clk_en_o <= 1'b0;
         end else begin
            clk_dividor <= 6'h0;
            clk_en_o <= 1'b1;
         end
      end
   end
endmodule
```