**ReadMe\_Uart\_tx\_Rom**

(Entity) Inputs and Outputs have been defined (in: "resetn", "sysclk", "start\_triger" /   
out: "uart\_tx\_triger).  
  
(Architecture) First was defined the "Rom\_X" component that we created in other fille, this component contains 4 ports: (in: "address", "clock" / out: "q").   
New signals have been created: "sig\_packet" (12-bit vector),   
"sig\_byte" (8-bit vector), "sig\_rom\_data\_out" (8-bit vector), "sig\_address"   
(6-bit vector), "sig\_baud\_clk", "sig\_arising\_edge", "sig\_CLK\_C\_q\_not", "sig\_CLK\_C\_q",   
"state\_tx".  
Note: this component is not part of the "Uart\_tx\_Rom" but it is useful for system control of the data transportation in the project.  
Port map was defined when "address" was mapped to "sig\_address", "clock" was mapped to "sysclk" and "q" was mapped to "sig\_rom\_data\_out".  
This operation uses every "Rom\_X" outputs and inputs as new signals or I/O of the source. As a result, we can use the data in the rom registers for "Uart\_tx\_Rom" source.  
   
The "Uart\_tx\_Rom" source fille contains 3 processes:  
   
In the first process named " baud\_clock " the signal " sig\_baud\_clk " was created and with him output " baud\_clk " was created. If "resetn" equals to '0' then "sig\_baud\_clk" equals to '0' and variable "cnt\_baud" equals to '0', else, every rising edge of "sysclk" variable "cnt\_baud" increase by 1 and when "cnt\_baud" equals to 651 then   
"sig\_baud\_clk" <= not "sig\_baud\_clk" and "cnt\_baud" equals to 0.  
This process create a clock on 38,400 HZ and period time and therefore 13,000ns is half period time. "sysclk" period time is 20ns so that how this process created 38.4MHZ clock.  
  
In the second process named "rising\_edg" the signals " signal\_A\_q " and " signal\_A\_q \_not" were created and with them output "sig\_arising\_edge" was created   
(read "ReadMe\_Rising\_Falling\_Edge").  
  
In the third process named "transmission " the signal "sig\_bit" was created to set to output "uart\_tx\_triger ". If "resetn" equals to '0' then "sig\_address" equals to '0 0 0 0 0 0'  
(6-bits vector) because we want to start from address '0 0 0 0 0 0' and "state\_tx" start from s0, "sig\_bit" equals to '1' (not really matter if '1' or '0') and the variable "sig\_cntr" equals to 0, else, every rising edge of "sysclk" we are in state machine when

"state\_tx" 🡺 s0 then waiting to "start\_triger" equals to '1' and "state\_tx" 🡸 s1 else s0.  
"state\_tx" 🡺 s1 then waiting to "start\_triger" equals to '0' and "state\_tx" 🡸 s2 else s1.  
"state\_tx" 🡺 s2 then delay of 2 "sysclk" and "state\_tx" 🡸 s3.   
"state\_tx" 🡺 s3 then "sig\_address" increase by 1 and "state\_tx" 🡸 s4.  
"state\_tx" 🡺 s4 then "state\_tx" 🡸 s5.  
"state\_tx" 🡺 s5 then "sig\_byte" equals to "sig\_rom\_data\_out" and "state\_tx" 🡸 s6.  
"state\_tx" 🡺 s6 then delay of 5 "sysclk" and "state\_tx" 🡸 s7.  
"state\_tx" 🡺 s7 then "state\_tx" 🡸 s8.  
"state\_tx" 🡺 s8 then pocket creation to transmission and "state\_tx" 🡸 s9.  
"state\_tx" 🡺 s9 then waiting to " sig\_arising\_edge" equals to '1' then shift with '1' from left to "sig\_pocket" and "sig\_bit" equals to "sig\_packet(0)" and "sig\_cntr" increase by 1, when "sig\_cntr" equals to 11 then "state\_tx" 🡸 s10 else s9.  
"state\_tx" 🡺 s10 then if sig\_address = "000000" then "state\_tx" 🡸 s0 else s2.

**Introduction Uart**

UART (Universal Asynchronous Receiver Transmitter) is asynchronous serial communication send data bits one by one, from the least significant to the most significant.  
  
Because the computer work on serial communication and the Max 10 FPGA work on parallel communication we need the UART component to make the connection between them.  
  
We chose the UART as a RS232 protocol with band rate of 38400 bps to send every packet without collision.