UART IP CORE

Universal Asynchronous Receiver Transmitter

Developed by : Ayoub SABRI, Antoine SOMSAY, Adam VRIGNAUD

EI-SE 3 | Polytech Sorbonne

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview

This IP Core is an implementation of the Universal Asynchronous Receiver/Transmitter (UART) protocol. This core is designed for FPGA and ASIC that need to interface with other devices and who share this standard.

1.2 Features

Here are the main features :

- Single channel UART
- Programmable baud rates : 9600, 19200, 38400, 57600, 115200 bps
- 8-N-1 format : 1 start bit, 8 bits data, no parity bit, 1 stop bit
- Entirely portable IP
- Fully synchronous design

1.3 Deliverables

- Synthesizable VHDL source code
- Testbenches
- Verification testbenches
- User Guide

1.4 UART 8-N-1 format

The 8-N-1 UART format consists of one start bit, eight data bits, no parity bit and one stop bit. An example with 8 data bits and one stop bit is shown in figure 1. As can be seen, the idle level of the bus is high, the start bit is low and the stop bit is high again. The data is sent with the least significant bit first.

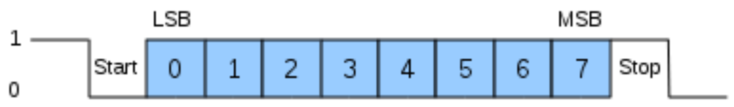


figure 1: 8-N-1 format

2. UART IP CORE DESIGN

2.1 Block Diagram

The main building block and interface signals of the UART are shown in the block diagram below :

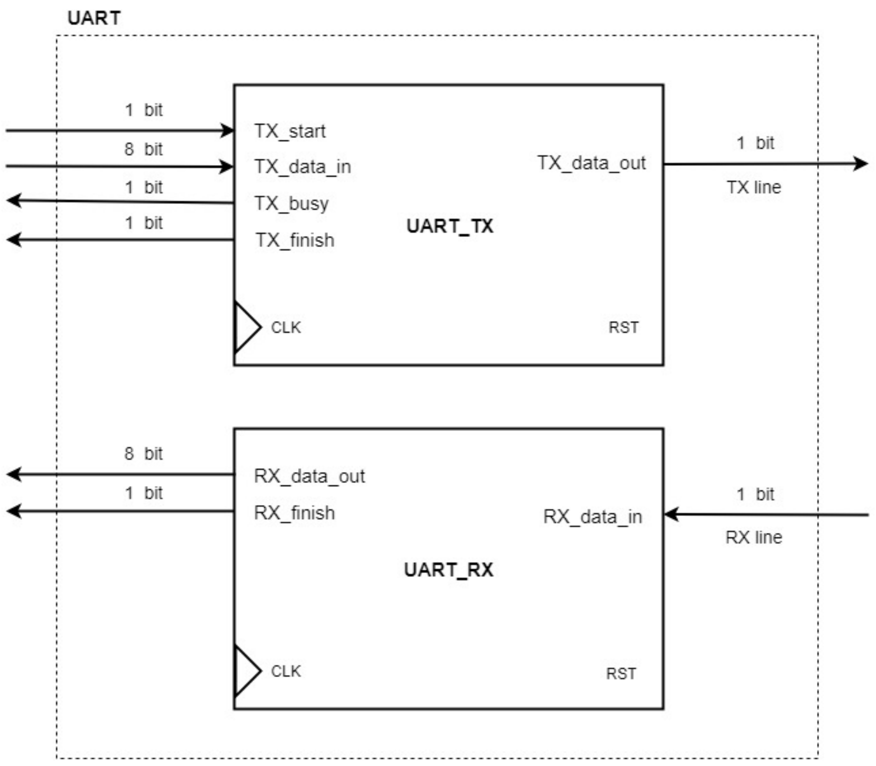


figure 2 : UART block diagram

2.2 Block Description

The output data baud rate depends on **clk_baudrate**, an integer generic parameter which is defined in the entity description of both transmitter and receiver. Its value has to be set to the result of the division between the clock frequency and the desired baud rate. For exemple if your system works using a 50 MHz clock frequency and you want to send data at 9600 bps, *clk_baudrate* has to be set to 5208. Note : the lower the baud rate, the better is the accuracy.

clk_baudrate = clk_frequency / baudrate

Both UART_TX (transmitter) and UART_RX (receiver) have been described as Finite State Machines (FSM) in the source code.

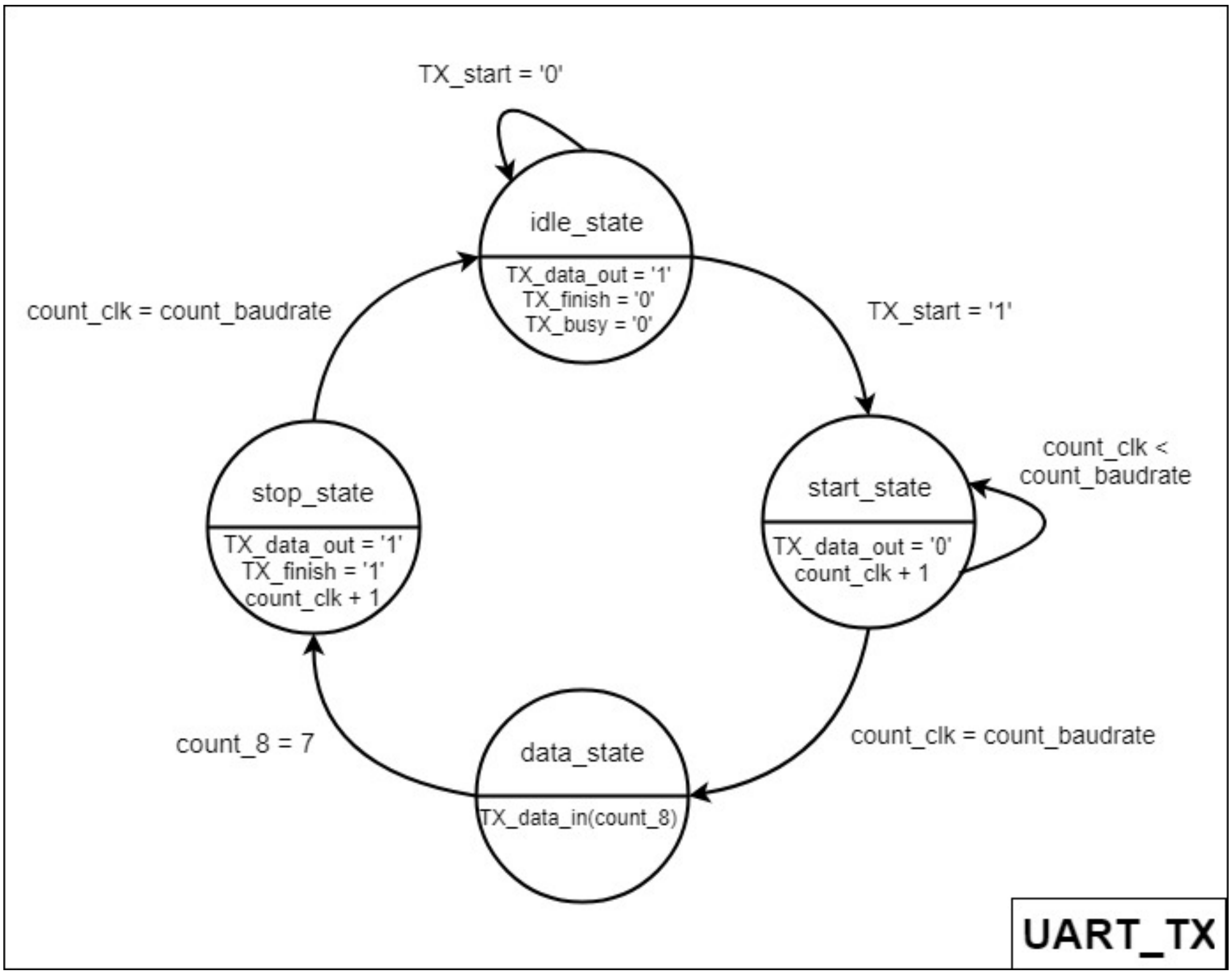


figure 3 : UART_TX FSM diagram

2.2.1 Transmitter (UART_TX) : This FSM starts in idle_state where the TX bus is set to high level. When a send data request occurs (*TX_start* = '*1*') TX_data_in is loaded to the buffer and the machine passes to start_state. Output is driven to low until *count_clk* reaches *count_baudrate* in order to obtain the chosen baud rate and then goes to data_state. In this state the machine uses *count_8* to count the number of bit sent through the line. It's important to say that *count_clk* is used in order to preserve the length of every bit transmitted. Finally, when *count_8* reaches 7, stop bit is transmitted, *TX_finish* is set to high to notify the end of the transmission and the machine goes back to the idle state where all the outputs are reset.

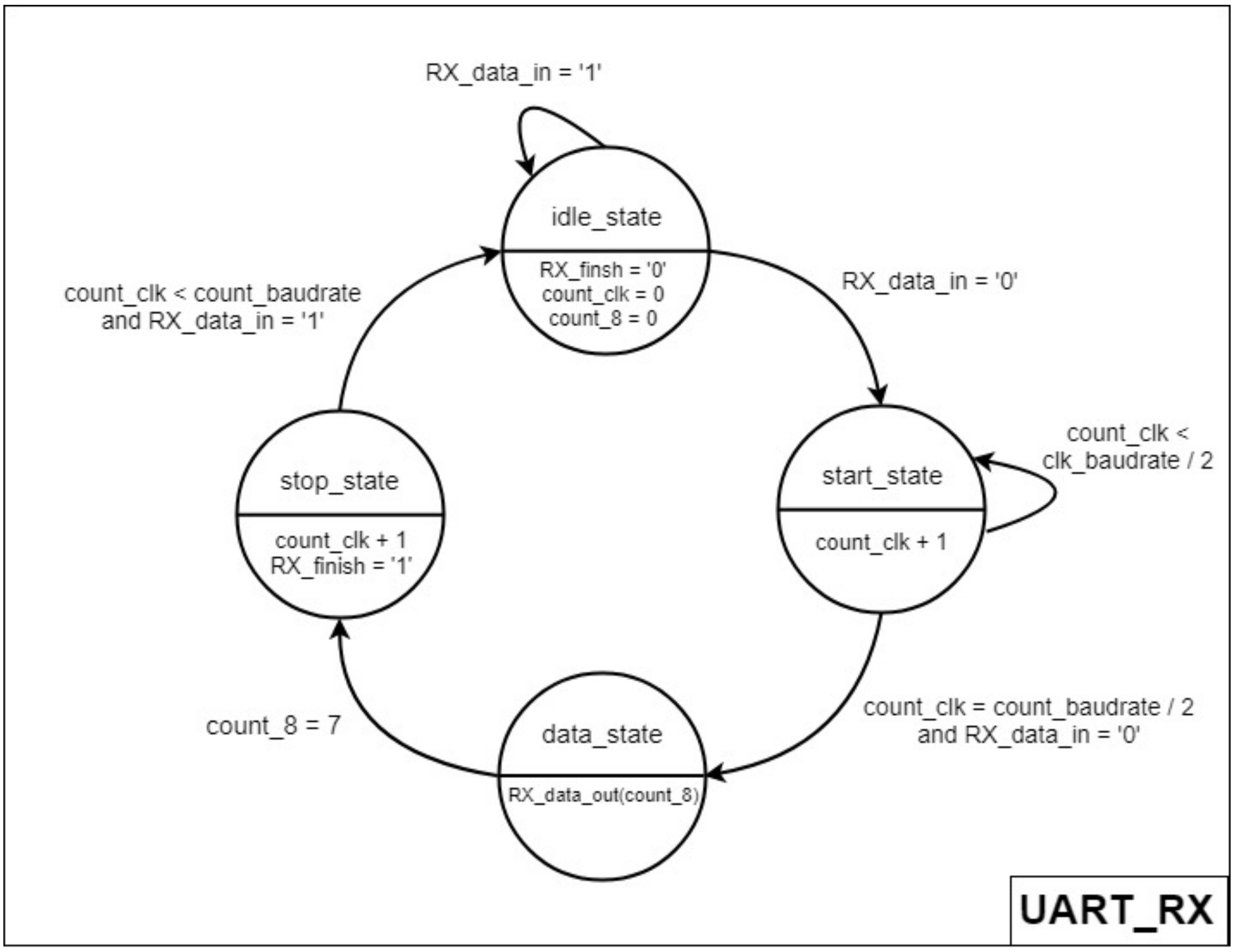


figure 4 : UART_RX FSM diagram

2.2.2 Receiver (UART_RX) : As can be seen in the diagram in figure 4, the FSM starts in idle_state until a start bit (low level) is detected on the RX line. Then the machine passes to start_state where it check if the low level detected before was a real start bit or a glitch measuring its length. Next, if the length matches with the chosen baud rate the FSM move to data_state where all data is captured (synchronization in middle of the bit in order to make sure to read the right value). This is repeated 8 times and data is written in *RX_data_out*. At last, the machine verifies if stop bit is received correctly in stop_state and set *RX_finish* to high to indicate that reception is done (*RX_data_out* can be read now). To finish, the FSM goes back to idle_state where all values are reset.

2.3 Signals Description

The following paragraphs list the inputs and outputs of the UART and provide an overview of the core's functionality.

2.3.1 Global signals

All units of this core are reset with the ***rst*** signal and clocked with ***clk***.

PIN NAME	TYPE	DESCRIPTION
rst	IN	Asynchronous reset, active high
clk	IN	System clock

2.3.2 Transmitter Interface

PIN NAME	TYPE	DESCRIPTION
TX_start	IN	When a pulse occours on this input, data is stored in the buffer and transmission is started. This must not be activated when transmitter is busy.
TX_data_in[7:0]	IN	This input contains the data loaded when TX_start is asserted.
TX_data_out	OUT	Transmit output pin (TX line).
TX_busy	OUT	When high the transmitter is busy and sending data, when low the transmitter is idle.
TX_finish	OUT	A one clock cycle pulse indicates the end of the data transmission.

2.3.3 Receiver Interface

PIN NAME	TYPE	DESCRIPTION
RX_data_in[7:0]	IN	Receive input pin (RX line)
RX_data_out[7:0]	OUT	8 bits captured data
RX_finish	OUT	This signal passes high (for one clock cycle) for signaling that a byte has arrived and RX_data_out can be read now.

2.4 Simulation Results

In this section a description of UART's simulation results obtained with Multisim is provided. In order to test the IP some changes have been brought to the source code: transmitter output has been connected to receiver input so both can be tested more easily.

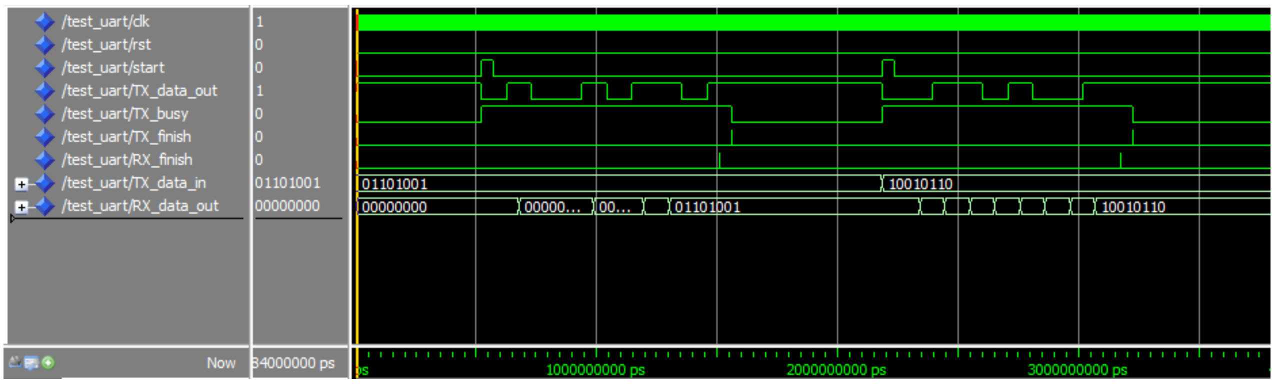


figure 5 : UART_TX and UART_RX timing diagram (Multisim screenshot)

As it can be seen in figure 5, when no data is sent both transmitter and receiver are idle. Then, when *TX_start* passes high, the transmitter sends a start bit followed by 8 bits data contained in *TX_data_in* (starting with LSB) and *TX_busy* stays active during all the transmission. When the transmission reaches its end, a stop bit is sent through the line and *TX_finish* passes high for a clock period in order to indicate the end of transmission.

At the same time, the receiver detects start of transmission when start bit is received and each bit is read at the middle of its length (synchronisation made by a counter inside the FSM). *RX_data_out* is updated every time a bit is received and at the end of transmission, when stop bit is received correctly, *RX_finish* indicates that output can be read.

3. REFERENCE DESIGN

3.1 Introduction

The following reference design has been designed to test the previously described UART. The system allows to performs distance measurements using an *HC-SR04* ultrasound sensor (please refer to annex 1 for more details) which is controlled from an external device. The measurement can be started from an external device through the UART by sending a character (single measurement) or from an on-off switch (real-time measurement). Once the measurement is done the distance (expressed in cm) is output on three seven segment displays and sent back to the device connected to the UART.

3.2 Block Diagram

The main building block and interface signals of the reference design are shown in the block diagram below. Each module's role will be described in the next paragraph.

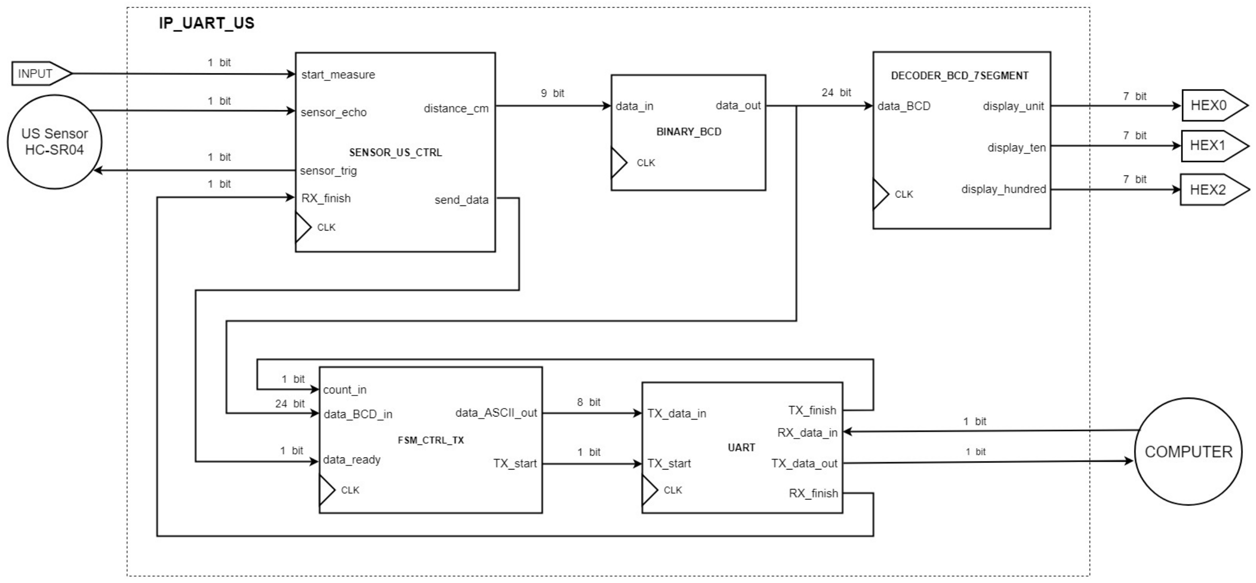


figure 7 : IP_UART_US reference design bock diagram

3.3 Block Description

3.3.1 SENSOR_US_CTRL

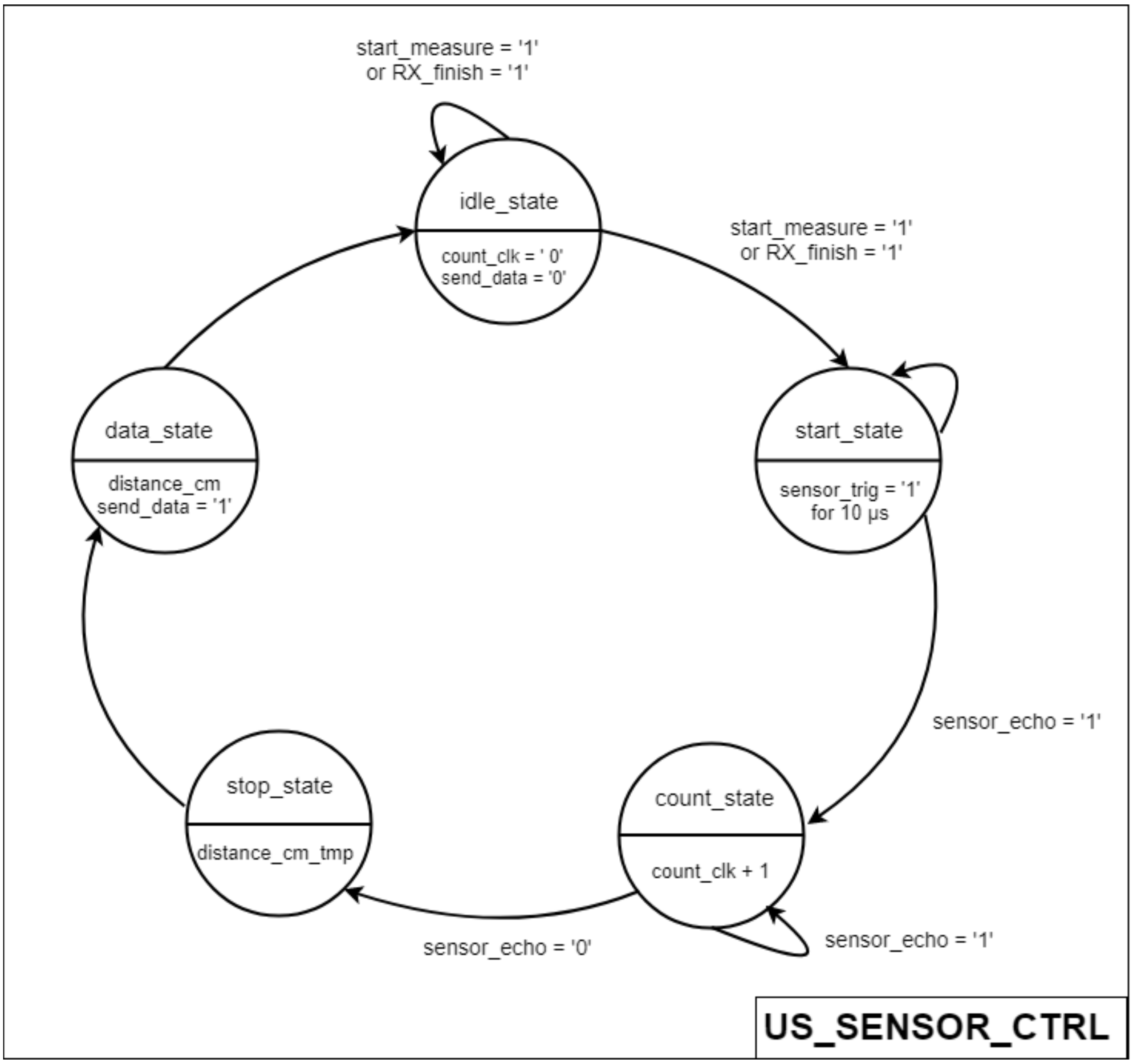


figure 8 : US_SENSOR_CTRL FSM diagram

This module is connected to the on-off switch, UART and the sensor. The FSM (described in figure 8))permits to start the measurement as well as calculate the distance value. All command signals are generated inside the FSM.

PIN NAME	TYPE	DESCRIPTION
start_measure	IN	This input is connected to an on-off switch. When high a real-time measurement is performed, when low it has no effect
RX_finish	IN	This input passes high for a clk period when a character is received. It's used in order to control the sensor for a single measurement
sensor_echo	IN	The sensor set this input high during the measurement. The signal length is directly proportional to the measured distance
sensor_trig	OUT	A 10 µs pulse is sent when a measurement is required in order to activate the sensor
send_data	OUT	This signal passes is set high by the FSM when the distance it's calculated
distance_cm[8:0]	OUT	9 bits coded distance expressed in cm and calculated from sensor_echo input

3.3.2 BINARY_BCD

This module allows to convert a data from binary to BCD. The output data contains units, tens and hundreds as shown in figure 9 :

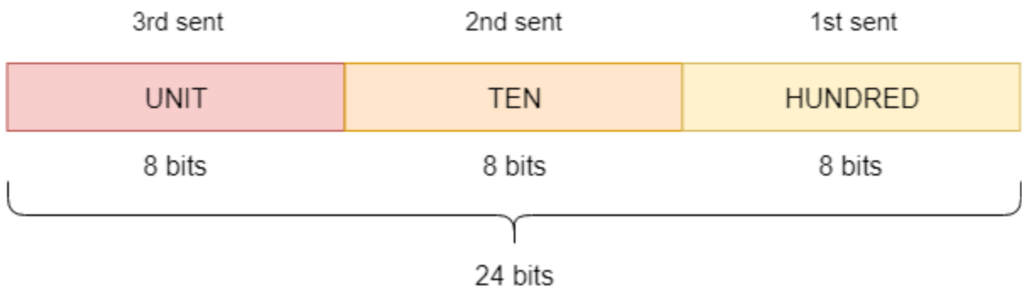


figure 9 : graphic representation of data_out vector

PIN NAME	TYPE	DESCRIPTION
data_in[8:0]	IN	9 bits binary data to convert
data_out[23:0]	OUT	24 bit BCD data converted, 8 bits for each digit

3.3.3 DECODER_BCD_7SEGMENT

Once the distance has been converted into 24 bits BCD array (*figure 8*), the signal is sent to this module in order to display each digit. Please refer to annexe 2 for the BCD to 7 segment conversion table used to write the program.

PIN NAME	TYPE	DESCRIPTION
data_BCD[23:0]	IN	24 bit BCD data, 8 bits for each digit
display_unit[6:0]	OUT	7 bits seven segments coded units
display_ten[6:0]	OUT	7 bits seven segments coded tens
display_hundred[6:0]	OUT	7 bits seven segments coded hundreds

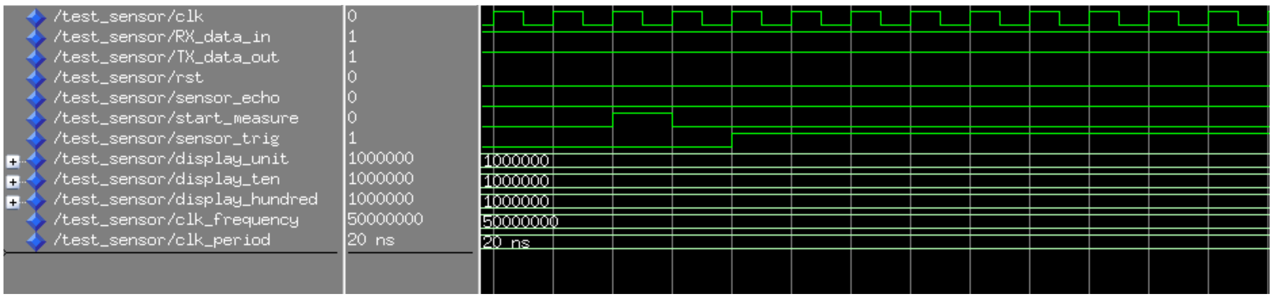
3.3.4 CTRL_TX

As seen before the UART allows to send 8 bits for a single transmission request. This module employs a FSM in order to convert in ASCII code and send separately units, tens and hundreds contained in the 24 bits BCD array (*data_BCD_in*).

PIN NAME	TYPE	DESCRIPTION
data_BCD_in[23:0]	IN	24 bit BCD data, 8 bits for each digit
count_in	IN	This signal is connected to TX_finish of UART_TX. When the UART sends a character a pulse is received on this input
data_ready	IN	This signal passes high (for one clock cycle) to indicate that data_BCD_in has been received and RX_data_out can be read now
data_ASCII_out[7:0]	OUT	8 bit data that need to be sent through the TX line. The data is sent by the UART when TX_start passes high
TX_start	OUT	Pulse generated by the FSM to start UART transmission

3.4 Reference Design Test Results

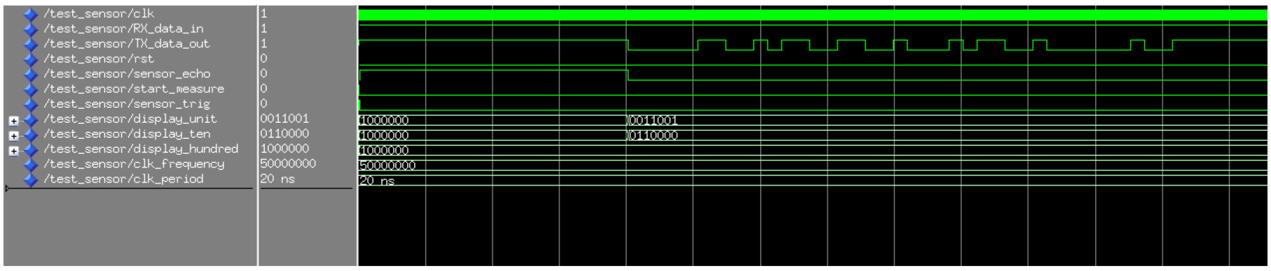
The whole system is in an idle state until *start_measure* will pass to 1 (this has been simulated using a physical switch on the board). Then the finite state machine sends a 10 µs pulse (which corresponds to 500 * *clk_period* for a 50 MHz clock) in order to activate the sensor and to start new measurement, as you can see in *figure 10* here below.



```
/test_sensor/clk               0
/test_sensor/RX_data_in        1
/test_sensor/TX_data_out       1
/test_sensor/rst               0
/test_sensor/sensor_echo       0
/test_sensor/start_measure     0
/test_sensor/sensor_trig       1
/test_sensor/display_unit      1000000     1000000
/test_sensor/display_ten       1000000     1000000
/test_sensor/display_hundred   1000000     1000000
/test_sensor/clk_frequency     50000000    50000000
/test_sensor/clk_period        20 ns       20 ns
```

figure 10 : start-measure and sensor_trig timing diagram (Multisim screenshot)

When the measurement is done, the sensor send through the *sensor_echo* input a pulse whose length is directly proportional to the measured distance (*sensor_echo* length has been defined in the test bench). As long as the input is high the finite state machine will count the number of *clk_period* and then when it passes to zero the distance (expressed in cm) is calculated (the mathematician formula will be explained in annex 1). As you can see in *figure 11* the distance value is displayed just after being calculated and all the three digits (hundreds, tens and units) are converted to ASCII code and sent to an external device through the UART TX line. Finally, when all data is sent the system goes back to the idle state.



```
/test_sensor/clk               1
/test_sensor/RX_data_in        1
/test_sensor/TX_data_out       1
/test_sensor/rst               0
/test_sensor/sensor_echo       0
/test_sensor/start_measure     0
/test_sensor/sensor_trig       0
/test_sensor/display_unit      0011001     1000000    0011001
/test_sensor/display_ten       0110000     1000000    0110000
/test_sensor/display_hundred   1000000     1000000
/test_sensor/clk_frequency     50000000    50000000
/test_sensor/clk_period        20 ns       20 ns
```

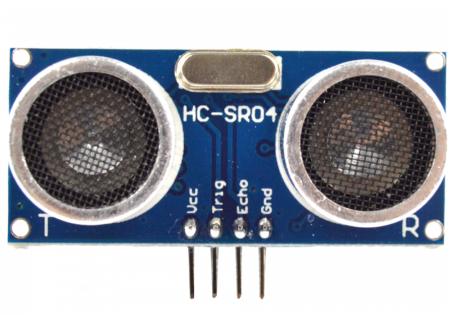
figure 11 : full reference design timing diagram (Multisim screenshot)

This test has been made with an echo length of 2 ms, which corresponds to a distance of 34 cm. Please refer to annex 2 for the BCD to 7 segment conversion table.

4. ANNEXES

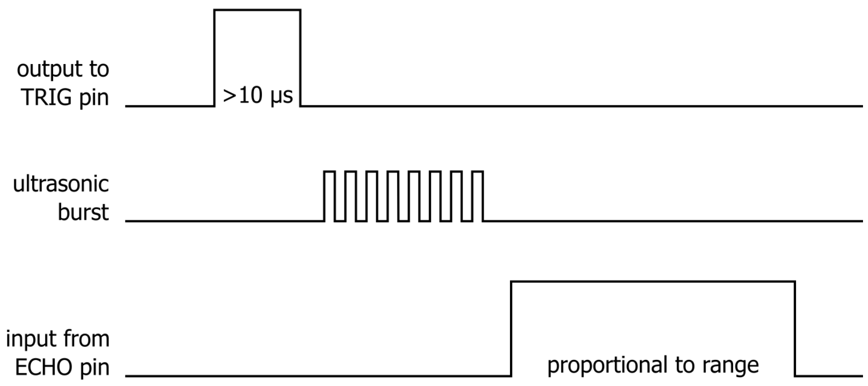
4.1 HC-SR04 Ultrasound Sensor

4.1.1 FEATURES



- Dimensions : 45 mm x 20 mm x 15 mm
- Measurement range : 2 cm to 400 cm
- Measurement resolution : 0.3 cm
- Trigger Input Pulse Width : 10 µs

4.1.2 TIMING DIAGRAM



4.1.2 DISTANCE CALCULATION

The period of the internal clock is equal to 20 ns, and the speed of the sound in the air is equal to 340 m/s. The time spent when *sensor_trig* is high is equal to the clock period multiplicated by the number of clock's pulse (*count_clk*). The distance has to be in cm, and is equal to the speed multiplicated by the time divided by two (because the signal travels one time, touch the object and travels back to the sensor). The distance is the equal to :

$$d = \frac{v \times t}{2} = \frac{340 \times 10^2 \times 20 \times 10^{-9} \times countclk}{2} \quad d = \frac{v \times t}{2} = 34 \times 10^{-5} \times countclk$$

4.2 BCD to 7 Segment conversion table

DIGIT	BCD (4 bits)	7 SEGMENT
0	0000	1000000
1	0001	1111001
2	0010	0100100
3	0011	0110000
4	0100	0011001
5	0101	0010010
6	0110	0000010
7	0111	1111000
8	1000	0000000
9	1001	0000110

Please note that a zero in the seven segment column represents a turned off segment.

4.2 Guide : How to test the IP and the Reference Design

Both UART IP and reference design have been tested on a FPGA development board, precisely the DE2-115 which uses an *Altera Cyclone IV* core.

4.2.1 UART IP TEST CONFIGURATION

If you want to test the UART in simulation you can for exemple connect UART's output with its input in order to verify the behavior. In order to do this, first of all you need to replace the line 79 of *UART.vhd* file with the following one :

```
RX_data_in => TX_out,
```

and then run *testbench_UART.vhd* file, where you will be able to configure the data that will be sent.

4.2.2 REFERENCE DESIGN CONFIGURATION

All GPIO connections are already set in Quartus project in fit directory. You will for exemple a TTL to USB cable in order to display the measure on a terminal. Then connect UART's TX to cable's RX and UART's RX to cable's TX.

The reference design can work in two modes :

- Real-time measurement (when SW0 is active)
- Single measurement (when SW0 is not active and a keyboard key is has been pressed)