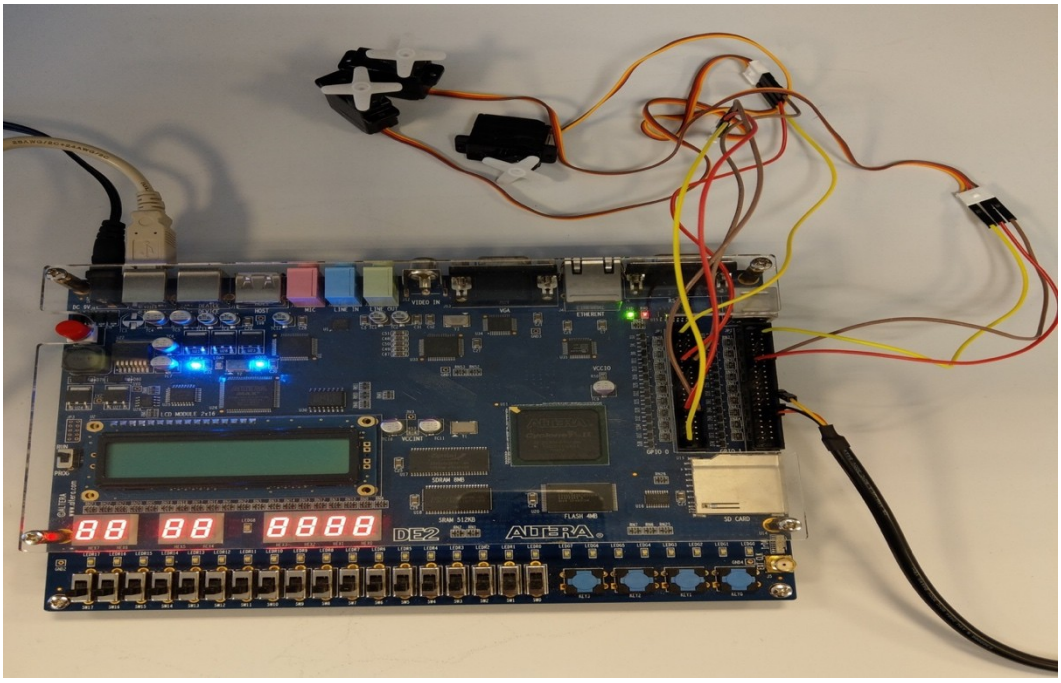
RC Servo Controller Free IP

I-Introduction :

We have decided to design this IP, in order to explain how a servo controller works.
This project includes lots of techniques that are quiet important and are used in more complex projects . For example we have got :

- Test benches and simulations scripts
- Functional blocks
- Synchronous clock and registers



On the above picture, we have got an overview picture of the project. We can see three servo , a DE2 cyclon II Altera's board.

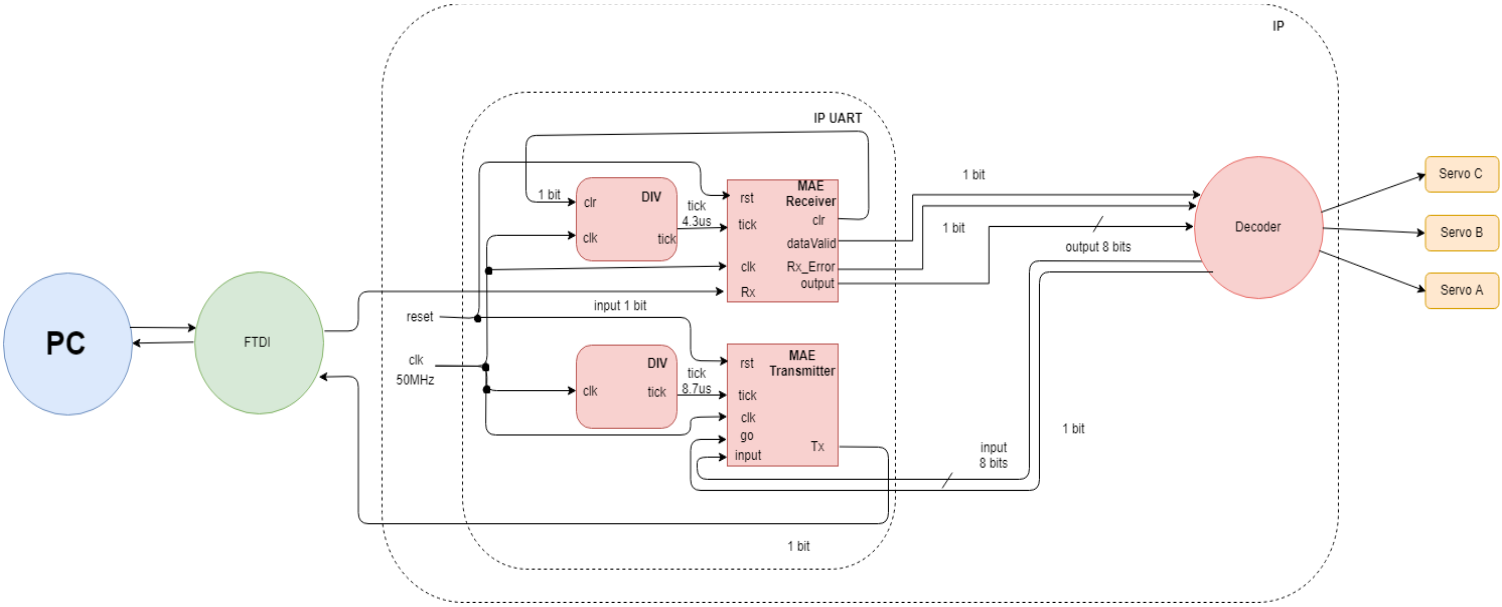
A- Principals:

The UART is driven by baudrate of 115200 bps.
The principal steps are :

- To create UART Emission and test it by a simulation.
- To create UART Reception and test it by a simulation.
- To create a test benches for all the programs
- To test IP UART on DE2 board

- To create an IP which will command a servo controller and to test by a simulation

  To complete the IP design we coded 3 different state machines(Receiver, Transmitter and the Decoder) , a frequency diviser,the emission and the reception as well ass the 3 servo.
  For each state machine , a test bench should be done to check that the state machine is doing the demanded task.

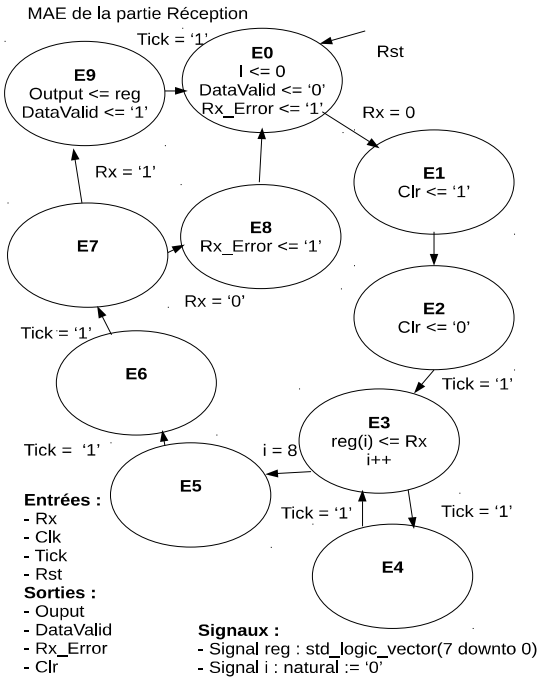


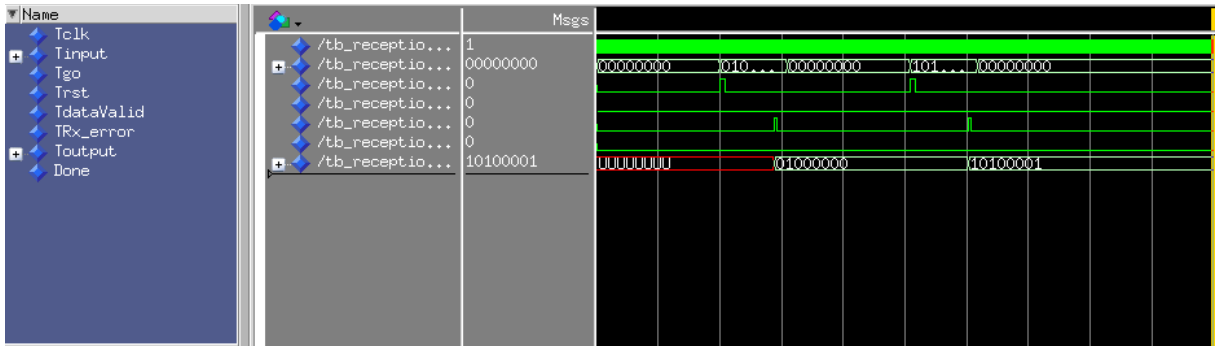
B- The UART's IP

The UART is designed to execute 2 tasks, either to receive or to transmit a serial signal. In our servo project, we only use the reception part but we will explain in this part how both reception and transmission work.
The UART's IP is made of two state machines and 2 frequency divider.
Both of them , they use the same general system's clock(50 MHz).Nevertheless,they have got respective ticks. The first state machine is the Receiver state machine which receives a 4.3 µs tick whereas the second state machine is the transmitter state machine which receives a 8.7 µs tick. We will now explain in great details each state machine.

The Receiver state machine

Rx: Serial Receiver
Clk: System's clock(50MHz)
Tick:A synchronous clock with the Rx(Receiver)
Rst: Restarts the state machine to zero
Output: the generated 8 bit data
DataValid:get the value updated to 1 when the 8 bits are out
Rx_error: indicates if there is an error in the receiver data
Clr: Synchronize the tick with the entering signal data

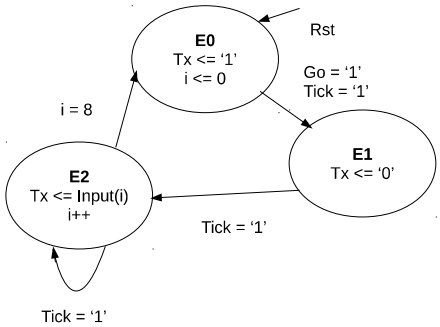




The transmitter state machine

MAE de la partie Emission

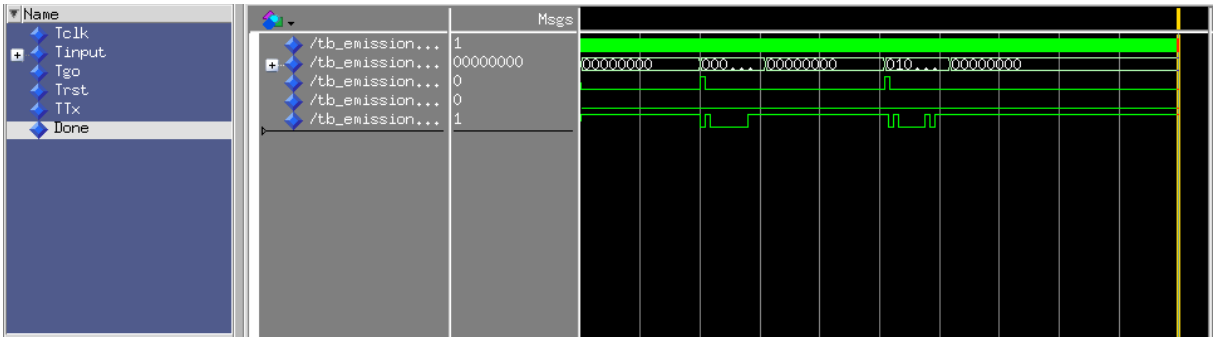
Clk: System's clock(50MHz)
Input: The entering data
Tick: A synchronous clock with the Tx(tranmistter)
Rst: Update the value of Go and Tick
Tx:Serial output



Entrées :
- Input
- Clk
- Tick
- Rst

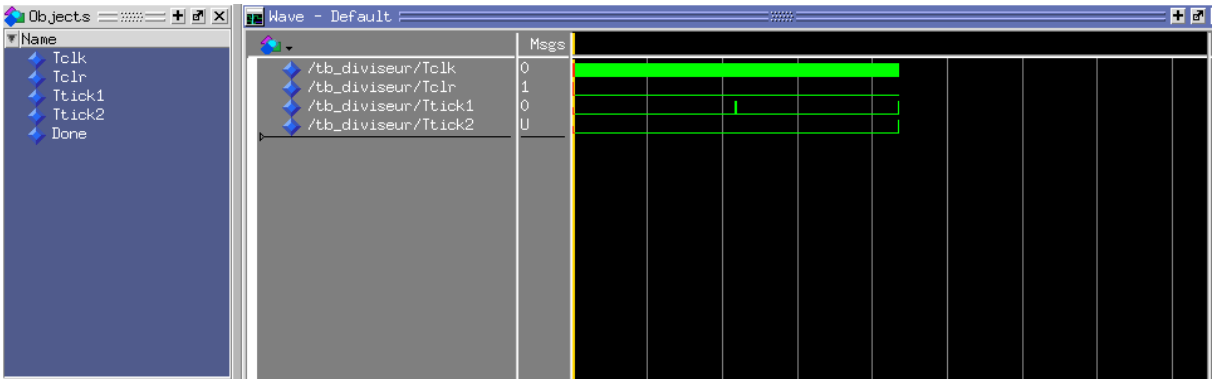
Sorties :
- Tx

Signaux :
- Signal i : natural := '0'



Frequency dividers

The frequency dividers help us to synchronize the system by creating different ticks with different values (4,3 µs and 8,7 µs) by only using the general system's clock. Currently, the ticks are synchronized with the entering signal's baudrate, allowing the concerned state to read the signal's data as it comes through



C- The servo controller IP

Servo controller interface

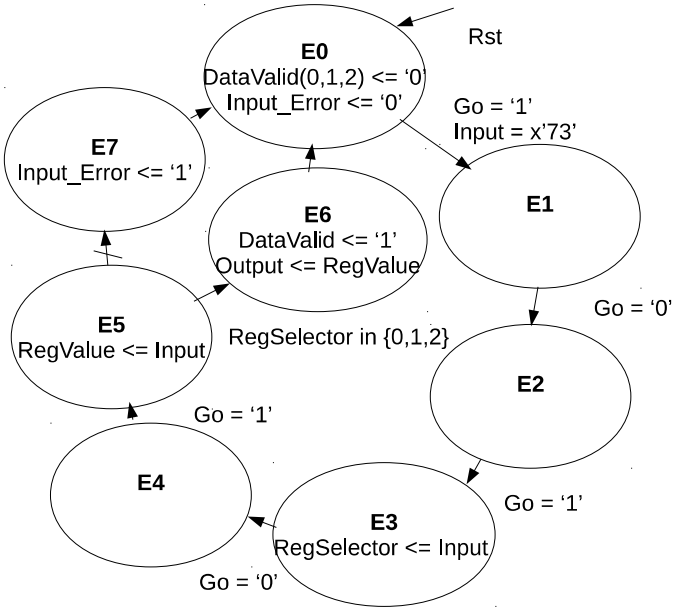
In the following part we will be talking about Servo's state machine,which it's IP includes the decoder and the 3 servos' control part.

```
entity MAE_servo is port(
        clk             : in std_logic;
        input           : in std_logic_vector (7 downto 0);
        go              : in std_logic;
        rst             : in std_logic;
        output0         : out std_logic_vector (7 downto 0);
        output1         : out std_logic_vector (7 downto 0);
        output2         : out std_logic_vector (7 downto 0);
        dataValid0      : out std_logic;
        dataValid1      : out std_logic;
        dataValid2      : out std_logic;
        input_Error     : out std_logic);
    end MAE_servo;
```

The servo controller implementation

The code is separated into two different parts: The decoder part and the servo part.
The following is the state machine that includes both of the part.

MAE de la partie Servo



Entrées :
- Input
- Clk
- Go
- Rst

Sorties :
- Output(0,1,2)
- DataValid(0,1,2)
- Input_Error

Signaux :
- Signal regSelector std_logic_vector (7 downto 0)
- Signal regValue std_logic_vector (7 downto 0)

Conclusion:

The IP demonstrates how to code functions that will do different services such as: Sequencing , delays, timing controls, project managing , loops, simulations and decoding.

How to use it :
First of all , launch “Screen” on the console. Secondly go to root to get into the right directory DEV.
The connected USB is called ttyUSB0 (or ttyUSB1 …).
This USB should be used at 230400 baudrate.
We need the double of the baudrate thus we should enter 230400.

```
screen /dev/ttyUSB0 230400
```