VHDL project

Serial data receiver

Omar Mohamed Ahmed Shuwaikh Mahfouz	**21010881**
Farah Elsayed Elsayed Mostafa	**21010947**
Mark Zakaria Soliman	**21011022**
Raghda Maher Abudahab	**21010524**
Aya Ahmed Soliman	**21010339**

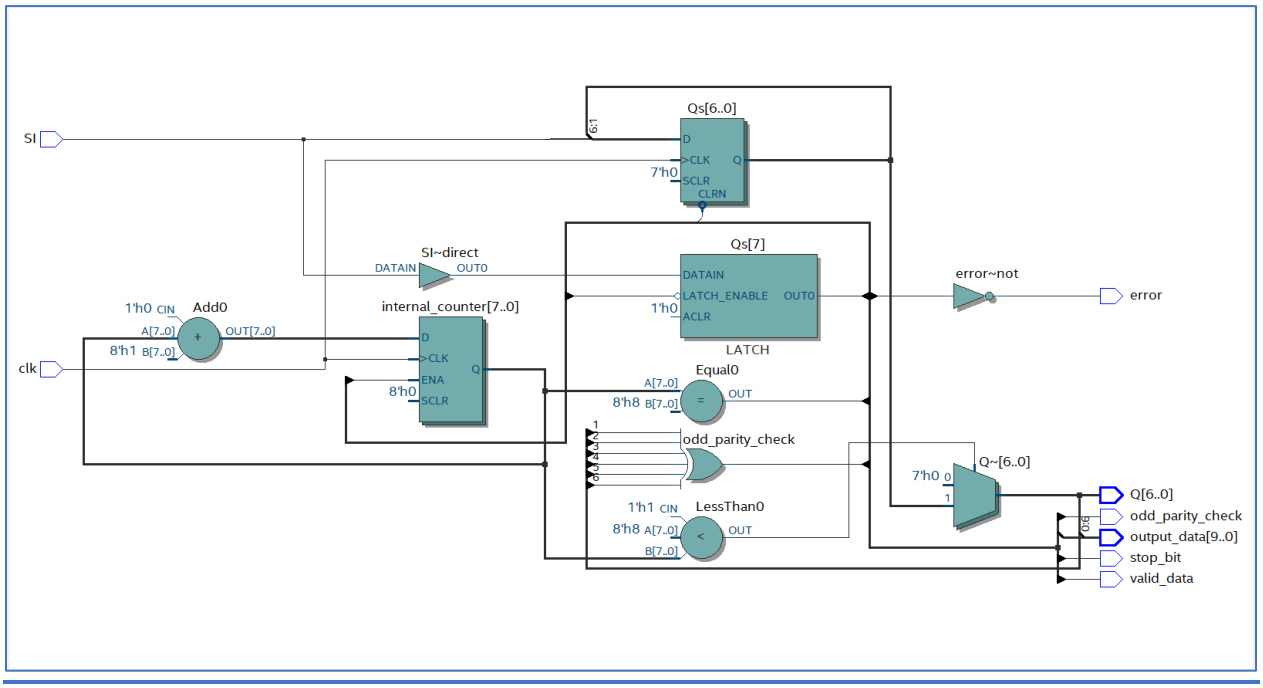
Project

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all; -- why unsigned library dont start from zero
entity code_project is
port( clk :in std_logic;
      SI:in std_logic;
      output_data:out std_logic_vector(9 downto 0);
      odd_parity_check,error:buffer std_logic ;
      valid_data: out std_logic;
      stop_bit : buffer std_logic;
      Q :buffer std_logic_vector(6 downto 0));
end code_project;
architecture beh of code_project is
signal Qs: std_logic_vector(7 downto 0);
signal internal_counter: integer range 0 to 255:=0;
begin
process(clk)
begin
if Qs(7)='1' then
        if rising_edge(clk) then  Qs(6 downto 0)<= SI& Qs(6 downto 1);
        internal_counter<= internal_counter+1;

        end if;
else Qs(6 downto 0)<="0000000"; Qs(7)<=SI;
end if;
end process;
```

```vhdl
Q <= Qs(6 downto 0) when internal_counter >= 8 else
     "UUUUUUU";
odd_parity_check<= Q(0) xor Q(1) xor Q(2) xor Q(3) xor Q(4) xor Q(5) xor Q(6);

stop_bit<= '1' when internal_counter=8 else
          '0';
error <= '1' when stop_bit='0'    else
         '0' when odd_parity_check='0' or odd_parity_check='1';
valid_data <= '1' when error='0' and internal_counter=8 else
              '0';
output_data<= Qs(7) & Q & odd_parity_check & stop_bit;
end beh;
```



- **We have two inputs SI and clk , the SI input enters the SIPO register and the first SI bit is an indicator if the SPIO will start or not .**
- **If Qs(7)='1' the SIPO register will start taking bits and shift the bits bit by bit .**
- **If the internal_counter reaches 8 the Q output will show the incoming bits by SI**
- **Odd parity bit is '1' if the number of 1s is odd else it is '0'**
- **Stop_bit is high when the 7 bits is received correctly (when internal counter =8).**
- **Error_bit is '1' if stop_bit is '0' and odd parity is'0'(not check number of 1s yet).**

Entity:

We will show the signal names and describe why we use these following modes indicate the signal direction:

Clk: is an input so it takes the mode of (in std_logic).

SI: it is an input signal , it takes the mode of(in std_logic).

output_data: it is an output signal that consists of 10 bits vector so, it takes the mode of (out std_logic_vector(9 downto 0)).

Odd_parity_check, error,stop_bit: they are outputs which are used to get other outputs so their values can be read inside the architecture (as in line no. 31,33&35) so, they take the mode of (buffer std_logic).

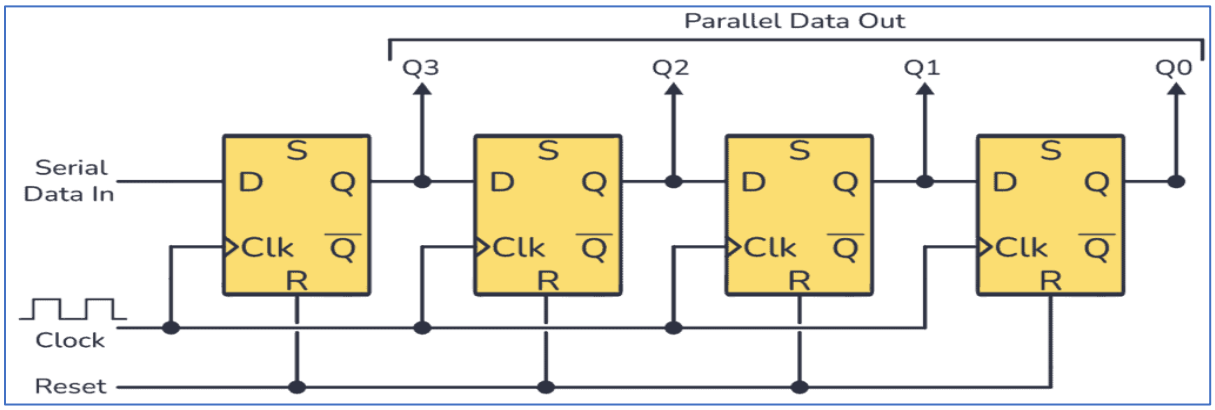
valid_data: it is an output signal so, it takes the mode of (out std_logic).

Q: it is an output vector signal consisting of 7 bits data and it is used to get other outputs so their values can be read inside the architecture (as in line no.31&39) so, it takes the mode of (buffer std_logic_vector(6 downto 0)).

```
 6  port( clk :in std_logic;
 7        SI:in std_logic;
 8        output_data:out std_logic_vector(9 downto 0);
 9        odd_parity_check,error:buffer std_logic ;
10        valid_data: out std_logic;
11        stop_bit : buffer std_logic;
12        Q :buffer std_logic_vector(6 downto 0));
13  end code_project;
```

SIPO:

Serial In- Parallel Out- Shift Register



Shift registers are essential components in digital electronic circuits, used for the storage and transfer of data. A Serial-In, Parallel-Out (SIPO) shift register is a specific type of shift register that accepts data in a serial format (train) and outputs it in a parallel format (cascaded).

It's operation is characterized by a chain of flip-flops connected in series like the figure , with each flip-flop storing a single bit of data. The data enters the register serially through a single input line, known as the Serial Input (SI) line. On each clock pulse, the data bit is shifted from one flip-flop to the next, propagating through the register.

Applications:

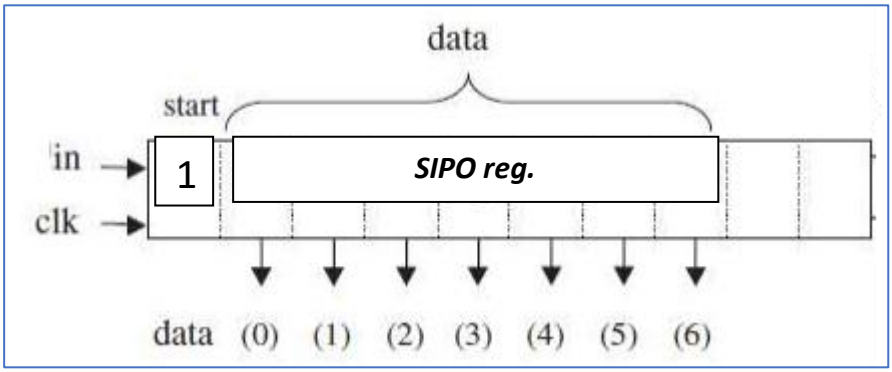
- Data storage: It can be used to store a sequence of data bits and retrieve them in parallel form.
- Data transmission: It enables the serialization of data for transmission over a single line and then parallelizes it at the receiving end.
- LED displays: SIPO shift registers can be employed to drive multiple LEDs in parallel, simplifying the control circuitry.
- Addressing: They are used for generating unique addresses in memory devices or accessing specific registers in microcontrollers.

Usage in our project:

When data is received from the user in a serial format, the register stores temporary data and at the edge of the clock any entered serial data can be loaded into the shift register to be processed and analyzed in parallel.

A control signal (Qs) has been used to state that the capacity of the input data stored in the device is 7 bits (6 downto 0):

These inputs are stored only if the first bit entered by the user were **1**



Start bit:

Qs is vector which consists of 8 bits starting from 7 and ended with 0,By taking the 7th bit (Qs(7)) and making it the start bit . We should assign the first value of SI into Qs(7) . We make if condition that indicate if Qs(7)=1, the SIPO register can start shifting the Incoming bits, If the first incoming bit (SI) = 0 the SIPO can not start until Qs(7) =1 .

```
15  signal Qs: std_logic_vector(7 downto 0);
16  signal internal_counter: integer range 0 to 255:=0;
17  begin
18  process(clk)
19  begin
20  if Qs(7)='1' then
21          if rising_edge(clk) then  Qs(6 downto 0)<= SI& Qs(6 downto 1);
22          internal_counter<= internal_counter+1;
23
24          end if;
25  else Qs(6 downto 0)<="0000000"; Qs(7)<=SI;
26  end if;
27  end process;
```

Stop bit

Stop _bit's mode is buffer because we want use this bit again in error detection . stop bit must be we high when transmission is correct , therefore if we receive 7 bits and the start bit Qs(7) , we make a counter to count how many bits have been received , if the count reaches 8 that indicates the there are 8 bits received successfully therefore the stop bit ='1'.by using concurrent when…..else statement , we can assign '1' if the counter=8 or '0' else as shown in figure .

Odd parity bit

odd parity check mode is buffer because we want use this bit again in error detection . using XOR is a technique used to verify the integrity of binary data by ensuring that the number of 1s in a binary sequence is odd, if the number of 1's is odd parity bit='1',otherwise odd parity bit='0'.

```
31  odd_parity_check<= Q(0) xor Q(1) xor Q(2) xor Q(3) xor Q(4) xor Q(5) xor Q(6);
32
33  stop_bit<= '1' when internal_counter=8 else
34          '0';
```

Error bit

The "error" signal is set to '1' when the "stop_bit" signal is '0', indicating an error condition. Otherwise, when the "stop_bit" is '1', the "error" signal is set to '0', indicating no error.

The reason for declaring "error" as a buffer is to allow bidirectional access to the signal. the "error" signal is assigned a value based on the condition, and it can be read by other parts of the design to determine the presence of an error.

Valid data:

the valid_data bit indicates whether the received data is error-free (error = '0') and if the required number of bits (8 bits) have been received (internal_counter = 8). When both conditions are met, valid_data is set to high, indicating that the received data is valid .

output data:

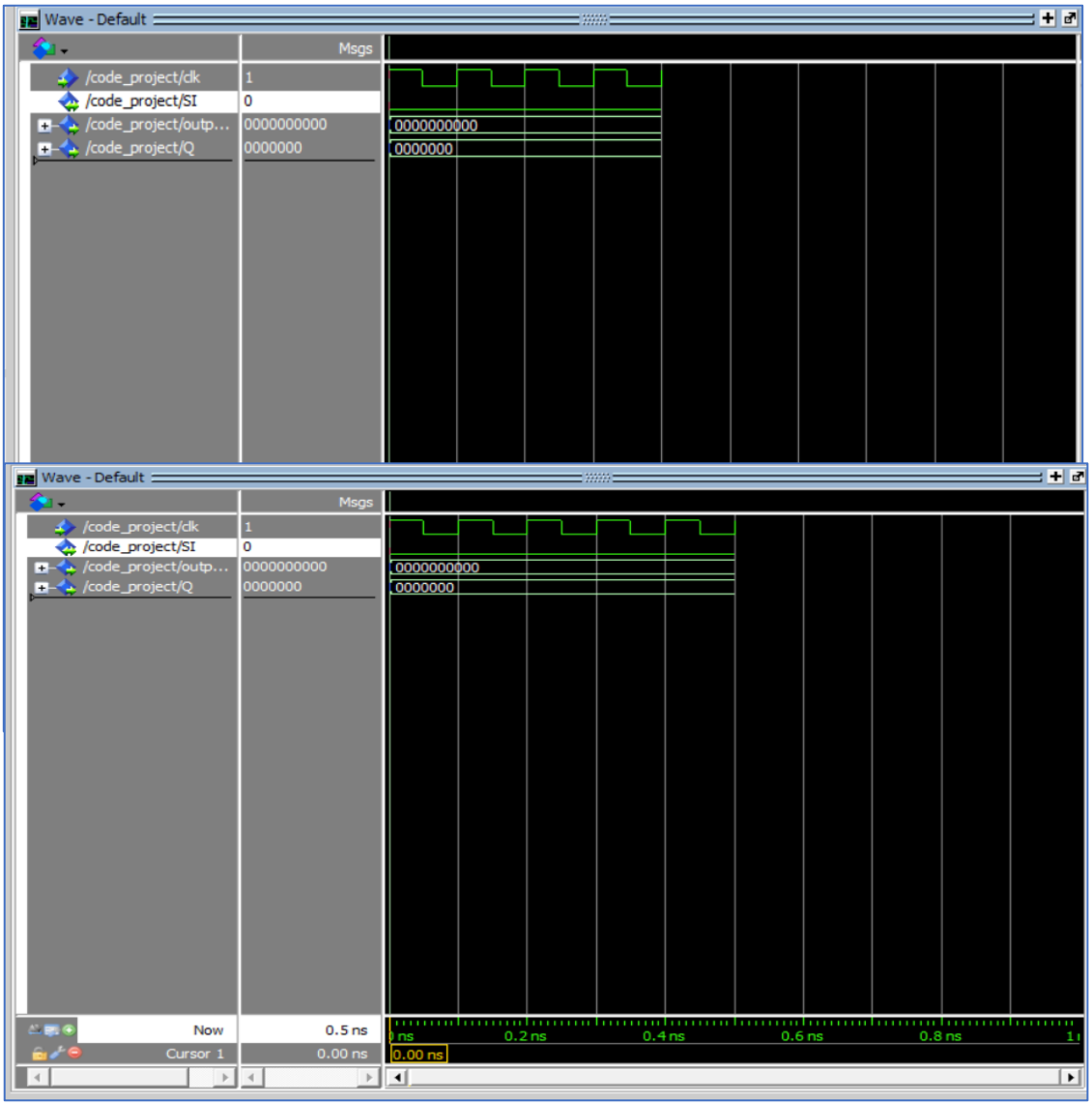
the final output data will be represented in ten bits like[start bit/received bits/parity bit/ stop bit].

Therefore, we have to concatenate all this bits in output data variable which take out mode .

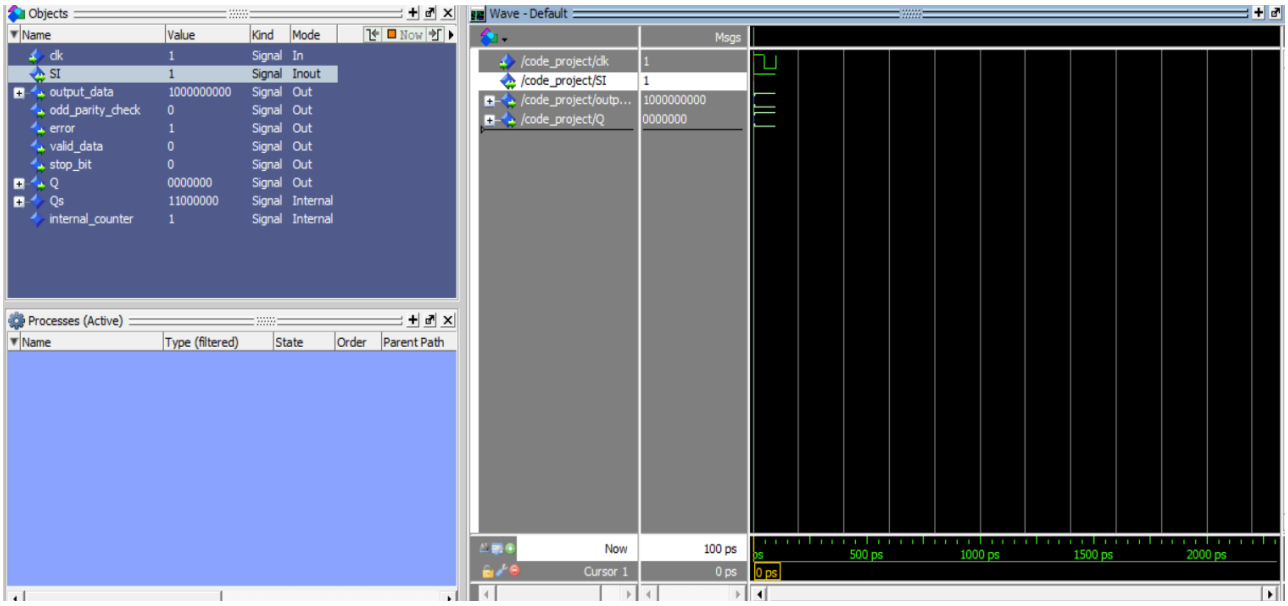
```
35  error <= '1' when stop_bit='0'    else
36           '0' when odd_parity_check='0' or odd_parity_check='1';
37  valid_data <= '1' when error='0' and internal_counter=8 else
38                '0';
39  output_data<= Qs(7) & Q & odd_parity_check & stop_bit;
40  end beh;
```

Results :

1) if SI=0 then the start bit will be zero and the SIPO can not work , as shown of the following figures output_data and output (Q) will be zeros.



2) if we insert the first 1 in SI then the start bit will be 1 as shown in the following figure , now the register can do it's work correctly .



---

3) By inserting the serial data ,we will insert 7 bits and after internal_counter=8; the output(Q) and output_data will appear in modalism.

By inserting sequence 0110011 the start bit will='1' , odd parity='0' , stop _bit='1' ,error ='0'

And output_data ="1011001101".