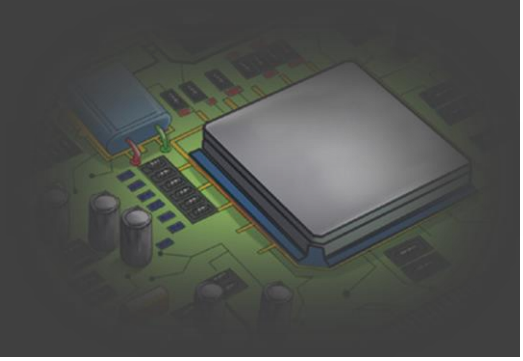
Serial Data Receiver



VHDL Final Project

Logic Circuits Design | EEC 242

Group 27

Team Members:

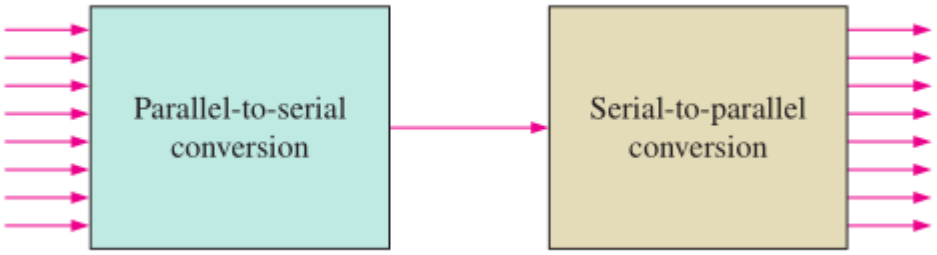
Islam Ibrahim Mohamed	21010247
Ahmed Walid Hassan	21010205
Omar Mohamed AbdelMawgoud	21010887
Mohamed Elsayed Mohamed	21011097
Ahmed Mahmoud Farag	21010184

Background

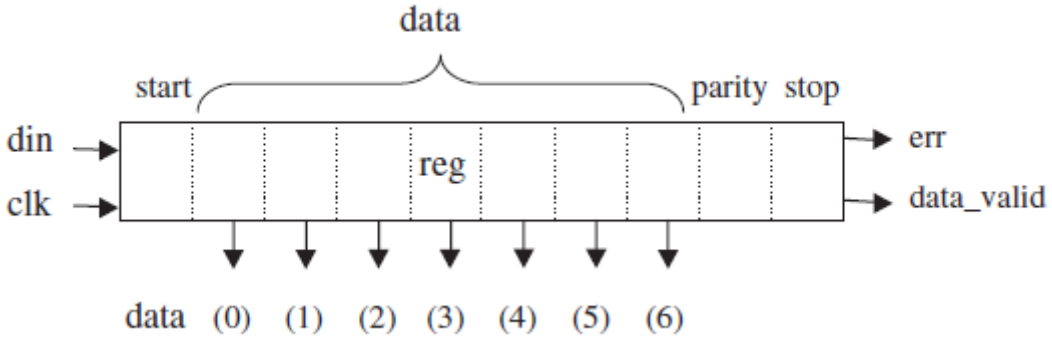
Serial data transmission is a method of sending data one bit at a time over a communication channel or a wire. This stands in contrast to parallel data transmission, where multiple bits are sent simultaneously over separate channels. Serial transmission has several advantages, including simplicity, reduced cabling, and the ability to transmit data over longer distances.

Serial data transmission is a widely used method for transmitting data. Various standards and interfaces define how serial communication is implemented in different applications.

The operation of serial transmission works through conversion of parallel data to serial data and vice-versa.



In this project we'll be focusing on building a serial data receiver specifically a 7-bit one with a single bit for parity.



The design specifications are as follows:

Input Signals:

- **din** (serial data input): Receives a serial data stream.
- **clock**: A clock signal [new bits are received at the rising edge of the clock].

Data Format:

- Data train consists of ten bits.
- Start Bit: First bit in the data train; when high, it signals the circuit to start receiving data.
- Data Bits: Seven bits following the start bit.
- Parity Bit: Ninth bit; its status must be '0' if the number of ones in the data is even or '1' otherwise.
- Stop Bit: Tenth bit; must be high if the transmission is correct.

Supervision Signals:

- **start_detected**: Set HIGH when the first bit in the register is set to 1.
- **reception_done**: Set HIGH when the last bit in the register is set to 1.
- **parity_xor**: Set to '0' if the number of ones in the data is even otherwise it's set to '1'.

Error Detection:

- An error is detected if either the parity check fails or the stop bit is not '1'.

Data Transfer:

- When reception is finished and no errors are detected, the data stored in internal registers is transferred to the output **data(6:0)**.

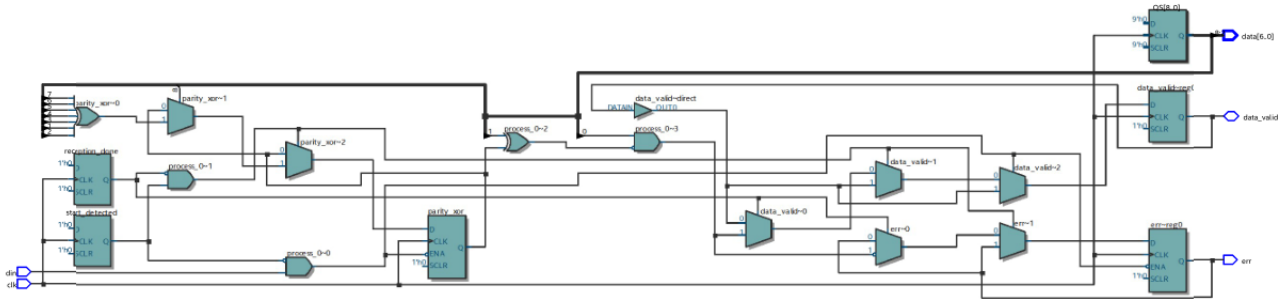
Output Signals:

- **data(6:0)**: Parallel data output.
- **data_valid**: Output signal indicating that the data is valid.
- **Err**: generated when an error occurs either because the parity is invalid, or the last bit is not 1.

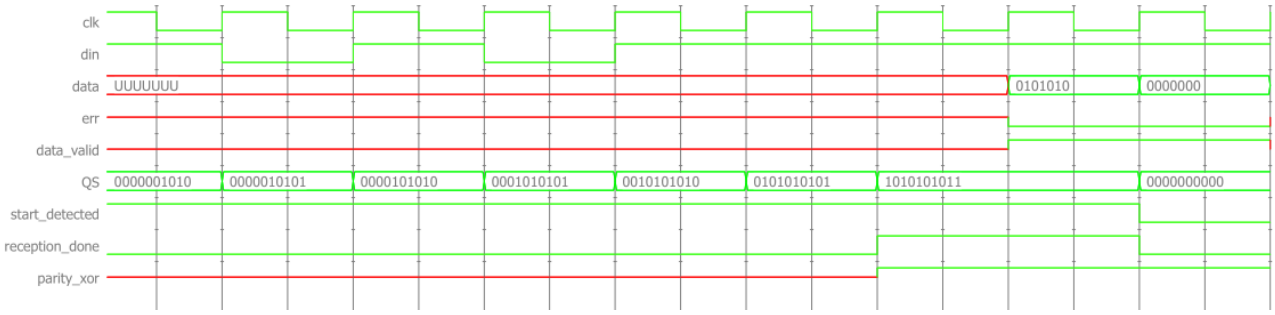
VHDL Code

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity SerialDataReceiver is
    port (
        din, clk: in std_logic;
        data: out std_logic_vector (6 downto 0);
        err: out std_logic;
        data_valid: inout std_logic
    );
end SerialDataReceiver;

architecture beh of SerialDataReceiver is
    signal QS: std_logic_vector(9 downto 0) := (others => '0');
    signal start_detected: std_logic := '0';
    signal reception_done: std_logic := '0';
    signal parity_xor: std_logic;
begin
    process(clk)
    begin
        if rising_edge(clk) then
            if start_detected = '0' and din = '1' then -- Detecting start bit
                start_detected <= '1';
                QS(0) <= din;
            elsif start_detected = '1' and reception_done = '0' then -- Data reception
                for i in 9 downto 1 loop
                    QS(i) <= QS(i-1);
                end loop;
                QS(0) <= din;

                if QS(8) = '1' then -- Reception completed
                    reception_done <= '1';

                    parity_xor <= QS(1) xor QS(2) xor QS(3) xor QS(4) xor QS(5) xor QS(6) xor QS(7);

                end if;



                elsif reception_done = '1'then
                        if (QS(0)='1' and (parity_xor= QS(1))) then

                        err <= '0';
                        data_valid <= '1';
                        else
                            err <= '1';
                     data_valid <= '0';
                  end if;

                end if;

                if data_valid = '1' or data_valid = '0' then
                        start_detected <= '0';
                        reception_done <= '0';
                        qs<= (others =>'0');
                end if;


                if start_detected = '0' and reception_done <= '0' then
                    err <= 'U';
                    data_valid <= 'U';
                end if ;
           end if;
    end process;

    data <= QS(8 downto 2) when data_valid = '1' ELSE -- Output the received data
     "UUUUUUU";
end beh;
```

Schematics [RTL netlist]

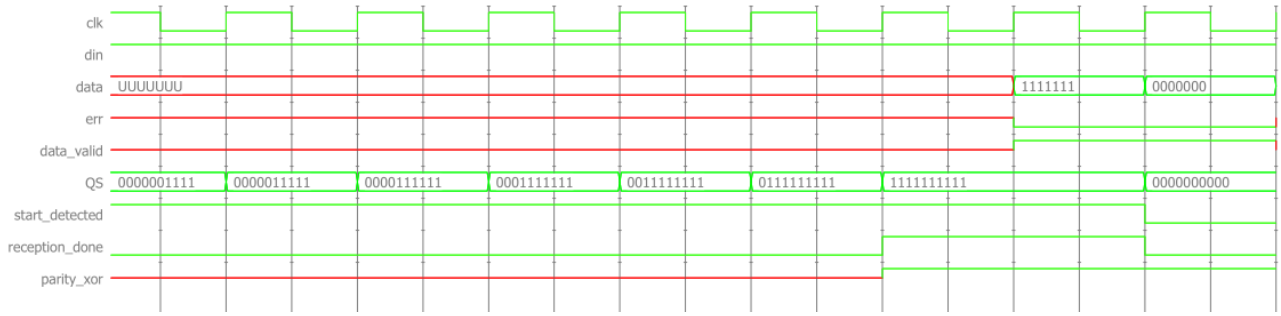


Simulation Results



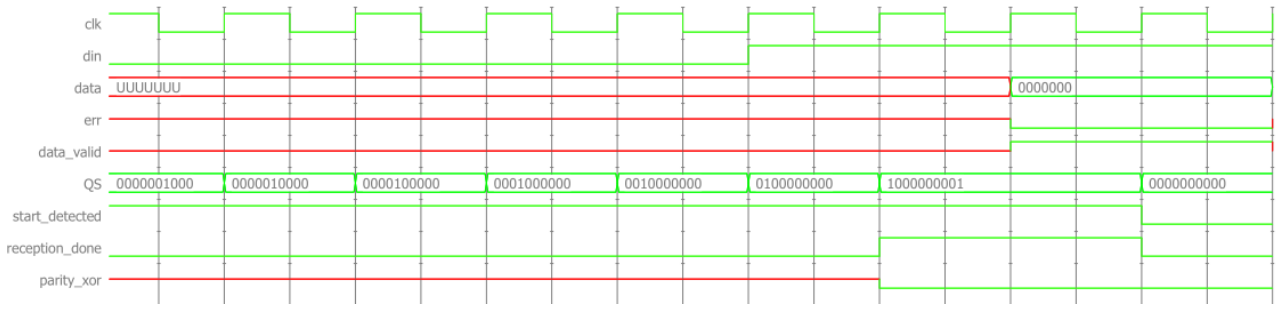
Data Sequence: 0101010

Parity Check: Valid



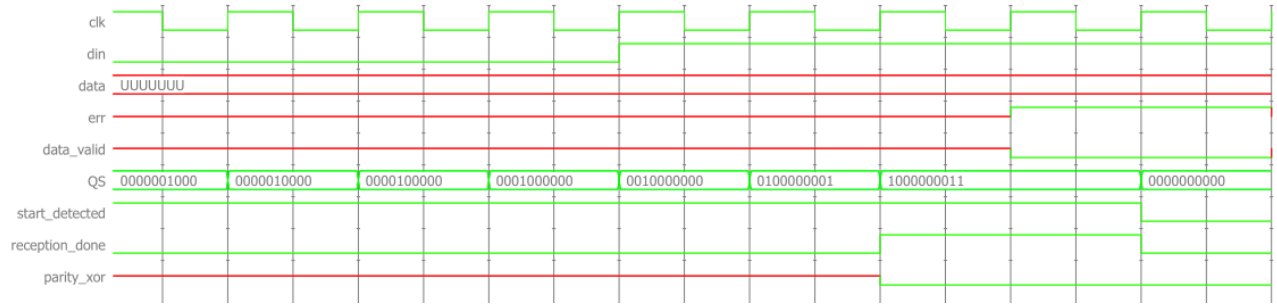
Data Sequence: 0000000

Parity Check: Valid



Data Sequence: 1111111

Parity Check: Valid



Data Sequence: 0000000

Parity Check: Invalid

Comments

- As observed in the simulation waveforms the data is transferred from the internal registers into the output signals only if the **data_valid** signal goes HIGH.
- When there's an error as in the 4th waveform (due to the parity check being invalid) the output data signals stay in an undetermined state and the **err** signal goes HIGH.
- When the reception is done all the internal registers are set LOW.