Code:

```
----------------------------------------------
-- Clock Design
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Clock is port (
        CLK : inout std_logic := '0'
        );
end Clock;

architecture dataflow of Clock is
begin
        CLK <= (not CLK) after 25 ns;
end dataflow;
----------------------------------------------
-- Sender Design
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Sender is port (
        CLK : in std_logic;
        D : in std_logic_vector(15 downto 0);
        Q : out std_logic_vector(15 downto 0)
        );
end Sender;

architecture dataflow of Sender is
```

```
begin
        Q <= D when rising_edge(CLK);
end dataflow;
-----------------------------------------------
-- Receiver Design
-----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Receiver is port (
        CLK : in std_logic;
        D : in std_logic_vector(15 downto 0);
        Q : out std_logic_vector(15 downto 0)
        );
end Receiver;

architecture dataflow of Receiver is
begin
        Q <= D when rising_edge(CLK);
end dataflow;
-----------------------------------------------
-- Error Injector Design
-----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Error_Injector is port (
        EI : in std_logic_vector(15 downto 0);
        Din : in std_logic_vector(15 downto 0);
        Dout : out std_logic_vector(15 downto 0)
```

```
        );
end Error_Injector;

architecture dataflow of Error_Injector is
begin
        Dout <= EI xor Din;
end dataflow;
-----------------------------------------------
-- Parity Generator Design
-----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Parity_Generator is port (
        D : in std_logic_vector(15 downto 0);
        PAR : out std_logic_vector(4 downto 0)
        );
end Parity_Generator;

architecture dataflow of Parity_Generator is
begin
        PAR(4) <= D(15) xor D(14) xor D(13) xor D(12) xor D(11);
        PAR(3) <= D(10) xor D(9) xor D(8) xor D(7) xor D(6) xor D(5) xor D(4);
        PAR(2) <= D(15) xor D(14) xor D(10) xor D(9) xor D(8) xor D(7) xor D(3) xor D(2) xor D(1);
        PAR(1) <= D(13) xor D(12) xor D(10) xor D(9) xor D(6) xor D(5) xor D(3) xor D(2) xor D(0);
        PAR(0) <= D(15) xor D(13) xor D(11) xor D(10) xor D(8) xor D(6) xor D(4) xor D(3) xor D(1) xor
D(0);
end dataflow;
-----------------------------------------------
-- Parity Checker Design
```

```
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Parity_Checker is port (
        D : in std_logic_vector(15 downto 0);
        PAR : in std_logic_vector(4 downto 0);
        ERR : out std_logic_vector(4 downto 0)
        );
end Parity_Checker;

architecture dataflow of Parity_Checker is
begin
        ERR(4) <= PAR(4) xor D(15) xor D(14) xor D(13) xor D(12) xor D(11);
        ERR(3) <= PAR(3) xor D(10) xor D(9) xor D(8) xor D(7) xor D(6) xor D(5) xor D(4);
        ERR(2) <= PAR(2) xor D(15) xor D(14) xor D(10) xor D(9) xor D(8) xor D(7) xor D(3) xor D(2) xor
D(1);
        ERR(1) <= PAR(1) xor D(13) xor D(12) xor D(10) xor D(9) xor D(6) xor D(5) xor D(3) xor D(2) xor
D(0);
        ERR(0) <= PAR(0) xor D(15) xor D(13) xor D(11) xor D(10) xor D(8) xor D(6) xor D(4) xor D(3) xor
D(1) xor D(0);
end dataflow;
----------------------------------------------
-- Structural Design (Top Level)
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Lab5 is port (
   D : in std_logic_vector(15 downto 0);
   CLK : inout std_logic;
```

```
    EI : in std_logic_vector(15 downto 0);
    Q : out std_logic_vector(15 downto 0);
    ERROR : out std_logic_vector(4 downto 0)
    );
end Lab5;

architecture structural of Lab5 is
    signal Gen_PAR : std_logic_vector(4 downto 0);
    signal Send_Q, Inj_D : std_logic_vector(15 downto 0);
component Clock is port (
    CLK : inout std_logic
    );
end component;

component Sender is port (
    CLK : in std_logic;
    D : in std_logic_vector(15 downto 0);
    Q : inout std_logic_vector(15 downto 0)
    );
end component;

component Receiver is port (
    CLK : in std_logic;
    D : in std_logic_vector(15 downto 0);
    Q : out std_logic_vector(15 downto 0)
    );
end component;

component Error_Injector is port (
    EI : in std_logic_vector(15 downto 0);
```

```
    Din : in std_logic_vector(15 downto 0);
    Dout : out std_logic_vector(15 downto 0)
    );
end component;

component Parity_Generator is port (
    D : in std_logic_vector(15 downto 0);
    PAR : out std_logic_vector(4 downto 0)
    );
end component;

component Parity_Checker is port (
    D : in std_logic_vector(15 downto 0);
    PAR : in std_logic_vector(4 downto 0);
    ERR : out std_logic_vector(4 downto 0)
    );
end component;

begin
    C1: Clock port map (CLK => CLK);
    C2: Sender port map (CLK => CLK, D => D, Q => Send_Q);
    C3: Parity_Generator port map (D => Send_Q, PAR => Gen_PAR);
    C4: Error_Injector port map (EI => EI, Din => Send_Q, Dout => Inj_D);
    C5: Parity_Checker port map (PAR => Gen_PAR, D => Inj_D, ERR => ERROR);
    C6: Receiver port map (CLK => CLK, D => Inj_D, Q => Q);
end structural;
----------------------------------------------
-- Simulation Test Bench
----------------------------------------------
library IEEE;
```

```
library STD;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;

ENTITY sim IS
END sim;

ARCHITECTURE behavioral OF sim IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Lab5
PORT(
D : in std_logic_vector(15 downto 0);
CLK : inout std_logic;
EI : in std_logic_vector(15 downto 0);
Q : out std_logic_vector(15 downto 0);
ERROR : out std_logic_vector(4 downto 0)
);
END COMPONENT;

-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal D : std_logic_vector(15 downto 0);
signal EI : std_logic_vector(15 downto 0);
signal CLK : std_logic;
-- Test bench signals to represent the output
signal ERROR : std_logic_vector(4 downto 0);
signal Q : std_logic_vector(15 downto 0);
```

```
-- begin of test bench action
BEGIN
-- Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Lab5 PORT MAP (
D => D,
CLK => CLK,
EI => EI,
Q => Q,
ERROR => ERROR
);
-- Stimulus Process example
-- We use 100 ns pulse width for this example
process
begin
        Report "Test begins...";
        -- Stimulus inputs
-------------------------------------------------------
EI <= "0000000000000000"; D <= "1100110000111010";
wait for 100 ns;
EI <= "1000000000000000"; D <= "1100110000111010";
wait for 100 ns;
EI <= "0000100000000000"; D <= "1100110000111010";
wait for 100 ns;
EI <= "0000000010000000"; D <= "1100110000111010";
wait for 100 ns;
EI <= "0000000000001000"; D <= "1100110000111010";
wait for 100 ns;
EI <= "0000000000000110"; D <= "1100110000111010";
```

```
wait for 100 ns;
-------------------------------------------------------
        Report "Test ends...";
        Report "All test vectors passed!";
wait for 50 ns;
end process;

end;
```