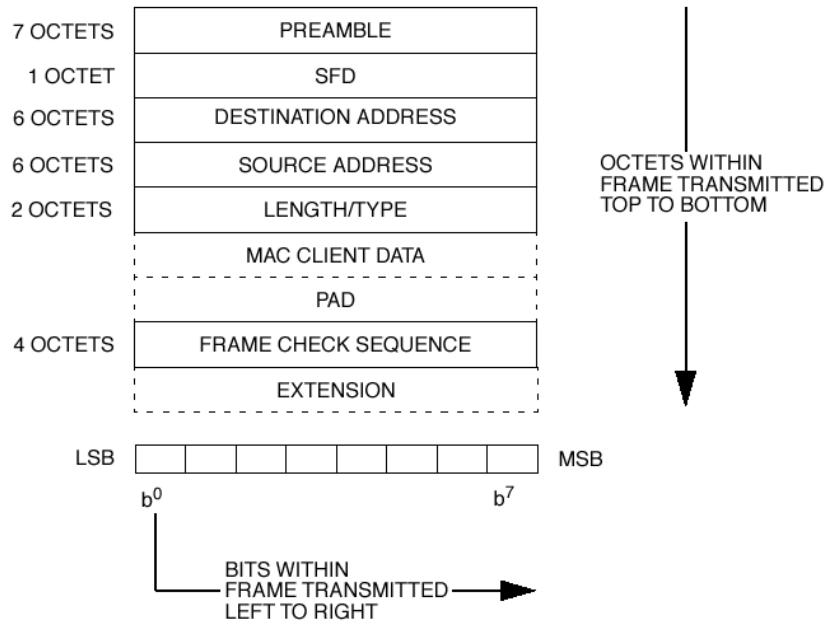
EXERCISE 1 (Ethernet Frame Check Sequence)

The purpose of this exercise is to design a bit error checker for an Ethernet frame. The last 4 octets in the frame is the Frame Check Sequence (FCS) field. See picture below.



A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source address, destination address, length, LLC data and pad (that is, all field except the preamble, SFD, FCS, and extension). The encoding is defined by the following generating polynomial.

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.

b) The bits of the frame are then considered to be the coefficients of a polynomial M(x) of degree n-1, where n is the number of bits in the frame. (The first bit of the Destination Address field corresponds to the x ^(n-1) term and the last bit of the data field corresponds to the x 0 term.)

c) M(x) is multiplied by x^32 and divided by G(x), producing a remainder R(x) of degree 31.

d) The coefficients of R(x) are considered to be a 32-bit sequence.

e) The bit sequence is complemented and the result is the CRC.

The 32 bits of the CRC value are placed in the frame check sequence field so that the x^ 31 term is the left-most bit of the first octet, and the x^0 term is the right most bit of the last octet. (The bits of the CRC are thus transmitted in the order x^31 , x^30 ,.., x^1 , x^0 .).

FCS calculation circuit:

The transmitted data stream T(X) is thus defined as the frame data plus the CRC field, which is the complemented remainder Rcomp(X).

$$T(X) = M(X)*x^{32} + Rcomp(X)$$

Thus polynomial division calculates the remainder:

$$R(X) = \text{remainder of } [M(X)*x^{32}/G(X)]$$

Thus, $M(X)*x^{32}$ can be written as:

$$M(X)*x^{32} = P(X)*G(X) + R(X).$$

By adding R(X) on both sides of this equation we obtain (remember we are using modulo 2 algebra, which means that R(X) + R(X) = 0):

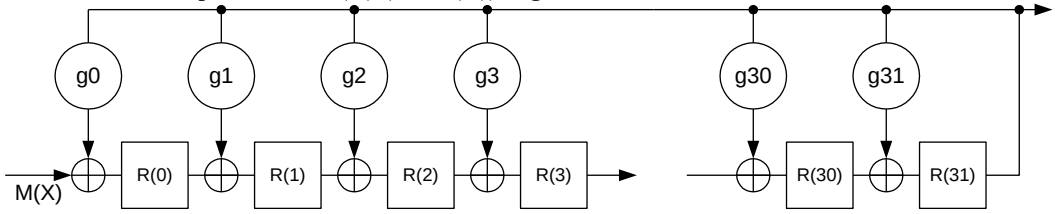
$$M(X)*x^{32} + R(X) = P(X)*G(X).$$

We notice that $M(X)*x^{32} + R(X)$ is divisible by G(X). In other words, the packet is considered error free if the remainder from dividing $M(X)*x^{32} + R(X)$ with G(X) is zero.

The remainder is stored in the 32 registers R(31)..R(0). The generating polynomial is generally defined as:

$$G(X) = g32*X^{32} + g31*X^{31} ..... g2*X^2 + g1*X^1 + g0$$

The circuit for the computation of R(X) (and P(X)) is given below.



The packet bit stream, $M(X)*x^{32} + R(X)$, is fed into the calculator. The Registers must contain "0" before the calculation is initiated. The last 32 bits that enter the circuit is the FCS (Remember to complement to convert to the CRC value). The data bits are error free if all the registers contain "0".

Example Ethernet packet with valid FCS checksum (`E6 C5 3D B2`):

```
00 10 A4 7B EA 80 00 12 34 56 78 90 08 00 45 00 00 2E B3 FE 00 00 80 11
05 40 C0 A8 00 2C C0 A8 00 04 04 00 04 00 00 1A 2D E8 00 01 02 03 04 05
06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 E6 C5 3D B2
```

Tasks

Task 1

Generate VHDL code that checks an Ethernet packet for bit errors. Assume that the packet is received as a serial bit stream. That is, the circuit above can be used directly. The module "fcs_check_serial" should contain the following IO signals:

```
entity fcs_check_serial is
  port (
    clk            : in std_logic; -- system clock
    reset          : in std_logic; -- asynchronous reset
    start_of_frame : in std_logic; -- arrival of the first bit.
    end_of_frame   : in std_logic; -- arrival of the first bit in FCS.
    data_in        : in std_logic; -- serial input data.
    fcs_error      : out std_logic -- indicates an error.
  );
end fcs_check_serial;
```

Task 2

Perform Synthesis and Place & Route of fcs_check_serial. (device: Stratix IV EP4SGX230KF40C2)
Note the achieved speed!

Task 3

Generate a testbench for "fcs_check_serial"

Task 4

Generate a parallel implementation of Task 1: Packets arrive in 8 bit octets each clockcycle. Hint: Create a small software program that determines the values of R(0)..R(31) after 8 clock cycles.

```
entity fcs_check_parallel is
  port (
    clk            : in std_logic; -- system clock
    reset          : in std_logic; -- asynchronous reset
    start_of_frame : in std_logic; -- arrival of the first byte.
    end_of_frame   : in std_logic; -- arrival of the first byte in FCS.
    data_in        : in std_logic_vector(7 downto 0); -- input data.
    fcs_error      : out std_logic -- indicates an error.
  );
end fcs_check_serial;
```

Synthesize and place & route the code and note the clock frequency. (device: Stratix IV EP4SGX230KF40C2). Verify the design in a testbench.

Report

The report must contain:

- VHDL code
- Synthesis report, Place & route report
- Logical equations + software program for Task 4