Exercise 1 (40 min, 4 pt).

For the following digital circuit described in VHDL:

1. Complete the sensitivity lists so that simulation and logic synthesis behave correctly
2. Complete the declarations for the signals in the architecture
3. Draw the hardware resulting of P2 logic synthesis. How many flip-flops are required?
4. Draw the state diagram for the state machine described in processes P3 and P4

```
library IEEE;
use IEEE.std_logic_1164.all;

entity RS232_Tx is
  generic(
    gDataWidth    : natural := 8;  -- bits for Data
    gTotalNumBit  : natural := 11; -- WordWidth= Start(1)+Data(8)+Parity(1)+Stop(1)
    gBaudRate     : natural := 57600;      -- bps
    gClkFrequency : natural := 100000000); -- Hz
  port(
    Clk           : in  std_logic;
    ResetN        : in  std_logic;
    RSDataToSend  : in  std_logic_vector(gDataWidth - 1 downto 0);
    EnableSend    : in  std_logic;
    TxD           : out std_logic;
    TxBusy        : out std_logic
  );
end RS232_Tx;

architecture Behavioural of RS232_Tx is
  constant StartBit        : std_logic := '0';
  constant StopBit         : std_logic := '1';
  ------------------------------------------------------------------
  constant cBaudRateCycles : natural := gClkFrequency/gBaudRate;
  signal EnaBaudRate       : ____________________;
  signal ClrBaudRate       : ____________________;
  signal CountBaudRate     : ____________________;
  signal EndBaudRate       : ____________________;
  ------------------------------------------------------------------
  signal EnaNumBitsTx      : std_logic;
  signal ClrNumBitsTx      : std_logic;
  signal CountNumBitsTx    : natural range 0 to gTotalNumBit-1;
  signal EndNumBitsTx      : std_logic;
  ------------------------------------------------------------------
  type tStateTx is (Idle, Sending);
  signal CurrentState      : tStateTx;
  signal NextState         : TstateTx;
  signal RegDataTx         : std_logic_vector(gTotalNumBit-1 downto 0);
  signal ClrTx             : std_logic;
  signal ParityBit         : std_logic;
begin
  TxD          <= RegDataTx(0);
  TxBusy       <= NOT(ClrTx);
  ------------------------------------------------------------------------------
  P1: process(                    )
    variable aux_Parity: std_logic;
  begin
    aux_Parity:= '0';
       for I in RSDataToSend'range loop
      aux_Parity:= aux_Parity XOR RSDataToSend(I);
    end loop;
    ParityBit     <= aux_Parity;
  end process P1;
  ------------------------------------------------------------------------
  P2: process(                       )
  begin
```

```
  if ResetN = '0' then
    RegDataTx <= (others => '1');
  elsif Clk'event and Clk = '1'  then
    if ClrTx = '1' then
       RegDataTx <= (others => '1');
    elsif EnableSend = '1' then
      RegDataTx <= StopBit & ParityBit & RSDataToSend & StartBit;
    elsif EnaNumBitsTx = '1' then
      RegDataTx <= '1' & RegDataTx(gTotalNumBit-1 downto 1);
    end if;
  end if;
end process P2;
-------------------------------------------------------------------------------
P3: process(                              )
begin
  if ResetN = '0' then
    CurrentState <= Idle;
  elsif Clk'event and Clk = '1' then
    CurrentState <= NextState;
  end if;
end process P3;
-------------------------------------------------------------------------------
P4: process(                                      )
begin
  ClrTx        <= '0';
  ClrBaudRate  <= '0';
  ClrNumBitsTx <= '0';
  EnaBaudRate   <= '0';
  case CurrentState is
    when Idle =>
      ClrBaudRate  <= '1';
      ClrNumBitsTx <= '1';
      ClrTx         <= '1';
      if EnableSend = '1' then
        NextState <= Sending;
        ClrTx  <= '0';
      else
        NextState <= Idle;
      end if;
    when Sending =>
      EnaBaudRate  <= '1';
      if EndNumBitsTx = '1'  then
        NextState <= Idle;
      else
        NextState <= Sending;
      end if;
    when others =>
      NextState <= Idle;
  end case;
end process P4;
-------------------------------------------------------------------------------
P5: process(                                    )
begin
  if ResetN = '0' then
    CountBaudRate <= 0;
  elsif Clk'event and Clk = '1' then
    if ClrBaudRate = '1' then
        CountBaudRate <= 0;
    elsif  EnaBaudRate = '1' then
       if CountBaudRate = cBaudRateCycles - 1 then
        CountBaudRate <= 0;
       else
        CountBaudRate <= CountBaudRate + 1;
       end if;
    end if;
```

```
    end if;
  end process P5;
  EndBaudRate <= '1' when ((CountBaudRate = cBaudRateCycles - 1) AND
                           (EnaBaudRate = '1'))
                 else '0';
  ------------------------------------------------------------------------------
  P6: EnaNumBitsTx <= EndBaudRate;
  process(                                  )
  begin
    if ResetN = '0' then
      CountNumBitsTx <= 0;
    elsif Clk'event and Clk = '1' then
      if ClrNumBitsTx = '1' then
          CountNumBitsTx <= 0;
      elsif  EnaNumBitsTx = '1' then
        if CountNumBitsTx = gTotalNumBit - 1 then
          CountNumBitsTx <= 0;
         else
          CountNumBitsTx <= CountNumBitsTx + 1;
        end if;
       end if;
    end if;
  end process P6;
  EndNumBitsTx <= '1' when ((CountNumBitsTx = gTotalNumBit - 1) AND
                            (EnaNumBitsTx = '1'))
                  else '0';
end Behavioural;
```

Exercise 2 (50 min, 6 pt)

We want to design a digital circuit to calculate the exponential function (e^X). For the circuit design, we will use the function representation as a Taylor series up to the fourth element.

$$Y(X) = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} \quad \textbf{(1)}$$

For this approximation to be valid, input data (X) must be in the range -1 to +1. To implement the algorithm using integer numbers, a variable transformation is performed: Z=X*128. This way, the new input Z is an 8-bit integer (2'sC). Note that with this transformation, value X=+1 is discarded. Applying the transformation, the equation is:

$$Y(Z) = 1 + \frac{Z}{2^7} + \frac{Z^2}{2 \cdot 2^{14}} + \frac{Z^3}{6 \cdot 2^{21}} \quad \textbf{(2)}$$

With the desired input range, the function result is in the range e^{-1} to $e^{127/128}$. We want to express the result with an 10-bit integer number, so the function must be multiplied by 128 (2^7). The final equation to be implemented is:

$$W(Z) = 2^7 \left(1 + \frac{Z}{2^7} + \frac{Z^2}{2 \cdot 2^{14}} + \frac{Z^3}{6 \cdot 2^{21}}\right) = 2^7 + Z + \frac{Z^2}{2^8} + \frac{Z^3}{3 \cdot 2^{15}} \quad \textbf{(3)}$$

The circuit has a 100kHz clock (Clk) to perform the operations, an asynchronous initialization signal (Reset) and an Enable input to load input data, active during a single clock cycle (sampling frequency is 1kHz).

The circuit entity is:

```
entity EXP_23 is
  port (
    Clk        : in  std_logic;
    Reset      : in  std_logic;
    Enable     : in  std_logic;
    Data_in    : in  signed(7 downto 0);  -- Z
    Data_out   : out signed(9 downto 0)); -- W(Z)
end EXP_23;
```

Answer the following questions in a reasoned way:

1. Check the variables and functions ranges. If Z variable range is [-128, 127] (signed 8-bit integer):
   a. Specify X range according to Z range
   b. Specify Y(Z) and W(Z) according to Z range
   c. Check all of them are according to specifications (W(Z) is 10-bit signed).
2. Express W(Z) so that it can be implemented using integer numbers and using divisions by powers of 2. Required coefficients are to be implemented with 8-bit unsigned numbers.
3. We want to implement a parallel architecture.
   a. Draw the circuit at RT level (Register Transfer).
   b. Determine the number of bits required for every addend
   c. Determine the number of bits required for the final result, without dropping any. If the final result is 10-bit, indicate the bits to drop.
4. Assuming the input is captured with Enable activation and the output is registered and available the following clock cycle, write the circuit description:
   a. Write the declaration of the required signals to store the input, partial results and final result
   b. Write the sequential process to register the input data
   c. Write the intermediate operations using concurrent statements or combinational processes
   d. Write the sequential process to register the final result
5. Multipliers have a delay of 40 ns and adders a delay of 20 ns. Calculate the maximum delay in the circuit and the required clock frequency. How could you modify the architecture so that the critical path delay is reduced to 40ns?