1. (12 points) Modify the following VHDL model to use process(es) instead of blocks.

```vhdl
use WORK.TSL.all, WORK.SYS.all;
entity I8212 is
  generic (GDEL, FFDEL, BUFDEL: TIME);
  port (DI: in WORD;
        DO: out WORD;
        NDS1, DS2, MD, STB, NCLR: in BIT;
        NINT: out BIT := '1');
end I8212;

architecture BEHAVIOR of I8212 is
begin
  I8212_BLK: block
    signal S0, S1, S2, S3, S4: BIT;
    signal SRQ: BIT;
    signal Q: WORD;
  begin
    INT_BLK: block (S1='1' and S4='1')
      begin
        Q <= guarded DI after FFDEL;
      end INT_BLK;
    S0 <= not NDS1 and DS2 after GDEL;
    S1 <= (S0 and MD) or (STB and not MD) after (2*GDEL);
    S2 <= (S0 nor (not S4)) after GDEL;
    S3 <= (S0 or MD) after GDEL;
    S4 <= (S1 OR NCLR) after GDEL;
    SRQ <= '1' after FFDEL when (S2='0') else
      '0' after FFDEL when (S2='1') and (not STB'STABLE) and (STB='0') else
      SRQ;
    NINT <= not SRQ nor S0 after GDEL;
  end block I8212_BLK;
end BEHAVIOR;

Solution:

use WORK.TSL.all, WORK.SYS.all;
entity I8212 is
  generic (GDEL, FFDEL, BUFDEL: TIME);
  port (DI: in WORD;
        DO: out WORD;
        NDS1, DS2, MD, STB, NCLR: in BIT;
        NINT: out BIT := '1');
end I8212;
```
architecture BEHAVIOR of I8212 is
signal S0, S1, S2, S3, S4: BIT;
signal SRQ: BIT;
signal Q: WORD;
begin
process (S1, S4, DI, S3, Q)
begin
if (S1 = '1' and S4 = '1') then
Q <= DI after FFDEL;
elsif (S1 = '0' and S4 = '0') then
Q <= "00000000" after FFDEL;
end if;
if (S3 = '1') then
DO <= Q after BUFDEL;
else
DO <= "ZZZZZZZZ" after BUFDEL;
end if;
end process;
S0 <= not NDS1 and DS2 after GDEL;
S1 <= (S0 and MD) or (STB and not MD) after (2*GDEL);
S2 <= (S0 nor (not S4)) after GDEL;
S3 <= (S0 or MD) after GDEL;
S4 <= (S1 or NCLR) after GDEL;
SRQ <= '1' after FFDEL when (S2 = '0') else
'0' after FFDEL when (S2 = '1') and (not STB'STABLE) and (STB='0') else
SRQ;
NINT <= not SRQ nor S0 after GDEL;
end BEHAVIOR;

2. (5 points) What kind of hardware element will be inferred by a synthesis tool from the following model?

Solution: A latch, since the behavior is level sensitive.

library ieee;
use ieee.std_logic_1164.all;
entity WIDGET is
Port (A, B : in SIGNED (0 to 2);
CLK, RESET : in std_logic;
Z : out SIGNED(0 to 2));
end WIDGET;
architecture EXAMPLE of WIDGET is
begin
process (CLK, RESET)
begin
if (RESET = '1') then
Z <= '0';
elsif (CLK = '1') then
Z <= A nor B;
end if;
end process;
end EXAMPLE;
3. (5 points) Draw the transistor-level diagram of a CMOS inverter.

**Solution:**

![CMOS Inverter Diagram]

Consider the following VHDL code:

```
-- Entity declaration
---------------------------------------
entity SCHED2 is
    port (A, B, C, D, E, F: in INTEGER;
        CLK : in BIT;
        W, X, Y: out INTEGER);
end SCHED2;
---------------------------------------

-- Architecture declaration
---------------------------------------
architecture HIGH_LEVEL of SCHED2 is
    signal Z: INTEGER;
begin
    X <= (A - B) * Z;
    Y <= (A * B) + Z;
    Z <= (C * D) + D * (E + F);
end HIGH_LEVEL;
```

4. (18 points) The following tasks refer to the VHDL code above. Assume that there are no hardware constraints.
a. (6 points) Draw a data flow graph.

**Solution:**

b. (6 points) Derive an ASAP schedule.

**Solution:**

<table>
<thead>
<tr>
<th>Ready</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2, 3, 4}</td>
<td>{1, 2, 3, 4}</td>
</tr>
<tr>
<td>{5}</td>
<td>{5}</td>
</tr>
<tr>
<td>{6}</td>
<td>{6}</td>
</tr>
<tr>
<td>{7, 8}</td>
<td>{7, 8}</td>
</tr>
</tbody>
</table>

**Solution:**

<table>
<thead>
<tr>
<th>Ready</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>{7, 8}</td>
<td>{7, 8}</td>
</tr>
<tr>
<td>{1, 4, 6}</td>
<td>{1, 4, 6}</td>
</tr>
<tr>
<td>{2, 5}</td>
<td>{2, 5}</td>
</tr>
<tr>
<td>{3}</td>
<td>{3}</td>
</tr>
</tbody>
</table>
5. (10 points) Derive a list schedule using the critical path priority metric for the VHDL code above, using the following hardware constraint; all operations are done in an ALU module and there are two ALU modules available.

**Solution:**

<table>
<thead>
<tr>
<th>Op</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

6. (10 points) For the data lifetime chart shown, use the left edge algorithm to obtain an efficient register allocation.

**Solution:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>C</th>
<th>G</th>
<th>E</th>
<th>J</th>
<th>B</th>
<th>H</th>
<th>F</th>
<th>A</th>
<th>I</th>
<th>L</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>D</td>
<td>C</td>
<td>G</td>
<td>--</td>
</tr>
<tr>
<td>S2</td>
<td>E</td>
<td>C</td>
<td>G</td>
<td>J</td>
</tr>
<tr>
<td>S3</td>
<td>H</td>
<td>F</td>
<td>G</td>
<td>J</td>
</tr>
<tr>
<td>S4</td>
<td>A</td>
<td>F</td>
<td>I</td>
<td>L</td>
</tr>
<tr>
<td>S5</td>
<td>A</td>
<td>F</td>
<td>I</td>
<td>L</td>
</tr>
</tbody>
</table>

7. (1 point) **Synthesis** is the process of transforming a behavioral description into a structural gate-level circuit.
8. (18 points) Design a Moore state machine in VHDL which converts NRZ (non-return-to-zero) coding to Manchester coding. In NRZ coding, each bit is transmitted for one bit time without any change. For the Manchester code, a 0 is transmitted as 0 for the first half of the bit time and 1 for the second half, but a 1 is transmitted as 1 for the first half and 0 for the second half. In order to do this conversion, use a clock(CLOCK2) that is twice the frequency of the basic clock. Note that if the NRZ bit is 0, it will be 0 for two CLOCK2 periods. Similarly, if the NRZ bit is 1, it will be 1 for two CLOCK2 periods. Your design should have an active low synchronous reset and work with CLOCK2.

<table>
<thead>
<tr>
<th>NRZ</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOCK2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manchester</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Solution:**

entity NRZ2MAN is
  port (NRZ, CLOCK2, RESET : in BIT;
         MAN : out BIT);
end NRZ2MAN;

architecture BEHAVE of NRZ2MAN is
  type STATE_TYPE is (S0, S1, S2, S3);
  signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
begin
  process (CURRENT_STATE)
  begin
    case CURRENT_STATE is
    when S0 => if (NRZ = '0') then
               NEXT_STATE <= S1;
               else
               NEXT_STATE <= S3;
               end if;
    when S1 => NEXT_STATE <= S2;
    when S2 => if NRZ = '0' then
               NEXT_STATE <= S1;
               else
               NEXT_STATE <= S3;
               end if;
    when S3 => NEXT_STATE <= S0;
    when OTHERS => NEXT_STATE <= S0;
    end case;
  end process;
  process (RESET, CLOCK2)
  begin
    if (CLOCK2'EVENT and CLOCK2 = '1') then
      if (RESET = '0') then
        CURRENT_STATE <= S0;
      else
        CURRENT_STATE <= NEXT_STATE;
      end if;
    end if;
  end process;
end BEHAVE;
end if;
end process;
process (CURRENT_STATE)
begin
  case CURRENT_STATE is
    when S0 | S1 => MAN <= '0';
    when S2 | S3 => MAN <= '1';
    when OTHERS => MAN <= '0';
  end case;
end process;
end BEHAVE;

9. (12 points) Write a VHDL entity and architecture of a two-input AND gate with the generics, TPLH an

**Solution:**

entity AND2 is
generic (TPLH, TPHL : time);
port (I1, I2 : in BIT;
     O : out BIT);
end AND2;
arithmetic BEHAVE of AND2 is
begin
  process (I1, I2)
    variable O_INT, LAST : BIT;
  begin
    O_INT := I1 and I2;
    if (LAST = '0' and O_INT = '1') then
      O <= '1' after TPLH;
    elseif (LAST = '1' and O_INT = '0') then
      O <= '0' after TPHL;
    end if;
    LAST := O_INT;
  end process;
end BEHAVE;

10. (5 points) If the NRE costs for FPGA and CBIC circuits are $21,000 and $187,000, respectively, and the cost of individual parts for FPGA and CBIC circuits are $18 and $5, respectively, what is the break-even manufacturing volume for these two types of circuits?

**Solution:**

\[ 18x + 21,000 = 5x + 187,000 \]
\[ 13x = 166,000 \]
\[ x = 12769 \]

11. (1 point) An(n) ____ASIC__________________ is an integrated circuit produced for a specific application and produced in relatively small volumes.
12. (1 point) ____VHDL_________________ is an annoyingly strongly typed language.

13. (1 point) ____Flattening_________________ removes the hierarchy in a circuit, so the circuit has a single level.

14. (1 point) _____Communication________________ is the hardest problem.