GENERAL INSTRUCTIONS

1. Do not open this exam until instructed to do so.

2. This is a closed book, closed notes exam. Use a pencil to complete your work. Follow the instructions as given in the problem. Calculators are allowed for the exam.

3. Please do all of your work on the exam itself. You may use the backs of the pages, if necessary. Clearly indicate the continuation of your work on other pages.

4. Show work for maximum partial credit.

5. Please be as neat and as well-organized as possible.

6. Clearly indicate your answers.

<table>
<thead>
<tr>
<th></th>
<th>MAXIMUM</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
**PROBLEM 1:** (20 pts)

We discussed the differences between ASICs and FPGAs for system implementation.

*Part A (5 pts)*

Give two reasons why an ASIC implementation might be desirable over an FPGA implementation.

- An ASIC implementation is faster (in terms of clock rate) than an FPGA implementation.
- An ASIC implementation uses less chip area (higher density) than an FPGA implementation.

*Part B (5 pts)*

Give two reasons why an FPGA implementation might be desirable over an ASIC implementation (these cannot be the Boolean negation of the reasons given in Part A).

- An FPGA implementation has a shorter time to market than an ASIC implementation (you would have to wait for the ASIC to be fabricated).
- An FPGA implementation allows designs to be corrected/modified in the field (via reprogramming or reconfiguration) while an ASIC implementation is permanent.

We discussed the differences between SRAM and antifuse programming technologies for FPGAs.

*Part C (5 pts)*

If I need to select an FPGA for implementation in a satellite, which programming technology would I select and why?

**Antifuse FPGAs** are radiation-hardened while an SRAM FPGA generally would be vulnerable to upsets in configuration bits. If the satellite application required reconfiguration, then you would need to use an SRAM FPGA with other techniques to mitigate the single-event effects.

*Part D (5 pts)*

If I need to select an FPGA for rapid prototyping of an embedded system, which programming technology would I select and why?

**SRAM FPGAs** can be reprogrammed and are ideal to test and correct design prototypes. Antifuse FPGAs are one-time programmable devices.
PROBLEM 2: (20 pts)

You are designing the architecture for a new FPGA. Assume that you will start with the following logic element (LE):

![LE Diagram]

Your FPGA will contain 256 of these logic elements. Each horizontal and vertical routing channel contains 4 wires, and the intersections of the channels allow all possible connections among the wires. Each wire in the routing channel can be connected to every input of the LE on its right (excluding the clock which has its own separate routing network) and every output of the LE on its left. How many configuration bits are required for this FPGA architecture?

The 256 logic elements are arranged in a square pattern (16 x 16) for efficiency. The number of horizontal channels is one more than the number of rows of LEs (17), and the number of vertical channels is one more than the number of columns of LEs (17). This creates a total of (17 x 17), or 289 intersections of the horizontal and vertical channels.

Each LUT needs 16 SRAM bits. Therefore: 256 x 16 = 4096

Each intersection needs 16 SRAM bits. Therefore: 289 x 16 = 4624

Each of the 7 LE Inputs/Outputs needs 4 SRAM bits. Therefore: 256 x 4 x 7 = 7168

The total number of configuration bits equals 15,888
PROBLEM 3: (35 pts)

Systems are described by their interface and their behavior. Hardware description languages can be used to both simulate and synthesize those systems.

Part A (10 pts)

Give the VHDL entity declaration for the following megafuction from Altera’s Library of Parameterized Modules (LPM).

```
ENTITY lpm_add_sub IS
  PORT
    (   add_sub      : IN STD_LOGIC ;
      dataa        : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      datab        : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      result       : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
      cout         : OUT STD_LOGIC ;
      overflow     : OUT STD_LOGIC
    );
END lpm_add_sub;
```
**Part B (15 pts)**

Complete the output waveform for the following behavioral description. Do not assume any initialization of the outputs.

```vhdl
ARCHITECTURE LogicFunction OF fulladder IS
BEGIN
    sum <= a XOR b XOR Cin after 5ns;
    Cout <= (a AND b) OR (Cin AND a) OR (Cin AND b) after 5ns;
END LogicFunction;
```

**Part C (10 pts)**

Draw the GATE-LEVEL logic generated for the VHDL code shown below (A, B, Y, and S are all single-bit signals). Your schematic must be composed of GATES (i.e., NAND, NOR, AND, OR, NOT, XOR, etc).

\[ Y \leftarrow A \text{ when } (S = '1') \text{ else } B; \]
**PROBLEM 4:** (25 pts)

Given the following function:

\[
f(A, B, C, D) = (B \cdot \overline{C} \cdot \overline{D}) + (\overline{A} \cdot B \cdot D) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot C)
\]

a straightforward implementation would use exactly one four-input LUT. Unfortunately, your FPGA architecture contains two-input LUTs! Show how this function can be realized using only **three** (3) two-input LUTs. Complements of input signals are not available. Give the truth table implemented in each LUT. You will probably need to perform some minimization of the function. Use the next sheet of paper if necessary.

The function above corresponds to the following truth table (in the form of a Karnaugh map)

Using the Karnaugh map, you can reduce the function to the following:

\[
f(A, B, C, D) = (\overline{A} \cdot B) + (B \cdot C) + (B \cdot \overline{D})
\]
PROBLEM 4: (continued)

You will need to factor the function to map it into a daisy-chain of three 2-input LUTs:

\[ f(A, B, C, D) = B \cdot (A + C + D) \]

Within the parentheses, the order of the Logical OR does not matter. Using DeMorgan’s Theorem, we determine the following:

\[ \overline{A + D} \Rightarrow \text{NAND Function} \]

Here is a mapping into the three 2-input LUTs: