For the first three problems, we practice the usage of standard interconnect components. The fourth problem covers a standard sequential component. For the last problem, we practice system design processes.

1. (Decoders) Given three four-input Boolean functions
   \[ f_1(a, b, c, d) = \sum m(0, 2, 4, 9, 15) + \sum d(3, 8, 7, 14), \]
   \[ f_2(a, b, c, d) = \sum m(2, 3, 7, 8, 11) + \sum d(0, 4, 9, 13, 15), \]
   \[ f_3(a, b, c, d) = \sum m(1, 5, 10, 12, 14, 15) + \sum d(2, 7, 9, 11). \]

1.1. Implement the functions using a minimal network of 4:16 decoders and OR gates.

1.2. Implement the functions using a minimal network of 3:8 decoders and OR gates.
1.3. Implement the functions using a minimal network of 2:4 decoders and OR gates.

2. (Multiplexers) Assume a dual-railed system, where you have access to any variable and its complement. Implement the following four-input Boolean function as indicated in each of the following subproblems.

\[ f(a, b, c, d) = \sum m(0, 2, 7, 12, 13) + \sum d(3, 5, 6, 8, 15). \]
2.1. Implement the function using a minimal network of 8:1 multiplexers.
2.2. Implement the function using a minimal network of 4:1 multiplexers.
2.3. Implement the function using a minimal network of 2:1 multiplexers.

3. Assume a dual-railed system, where you have access to any variable and its com-
implement. Given a four-input Boolean function
\[ f(a, b, c, d) = \sum m(1, 2, 4, 7, 11, 13, 14) + \sum d(3, 5). \]

3.1. Implement the function using a minimal network of 2:4 decoders and OR gates.

3.2. Implement the function using a minimal network of 4:1 multiplexers.
3.3. Implement the function using a minimal network of 2:1 multiplexers.
4. Counter: Use standard modulo counters to design various counters.
4.1 Given a modulo 16 counter, construct a 2 to 14 counter.

4.2 Given a modulo 8 counter, construct a counter that counts in a sequence 0, 3, 4, 6, 1, 7, with minimal networks of NAND gates.

4.3 Construct a modulo 64 counter using modulo 4 counters.
5. System Designs: Implement the following algorithm:

\[
\text{Alg}(X,Y,Z,\text{start},U,\text{done})
\]

Input \(X[7:0]\), \(Y[7:0]\), \(Z[7:0]\), \(\text{start}\);
Output \(U[7:0]\), \(\text{done}\);
Local-object \(A[7:0]\), \(B[7:0]\), \(C[7:0]\);
S1: If \(\text{start}'\) goto S1 \(\parallel\) \(\text{done} \leftarrow 1\);
S2: \(\text{done} \leftarrow 0 \parallel A \leftarrow X \parallel B \leftarrow Y \parallel C \leftarrow Z\);
S3: \(A \leftarrow \text{Add}(A,B) \parallel B \leftarrow \text{Inc}(B)\);
S4: If \(A'[7]\) goto S3 \(\parallel B \leftarrow \text{Inc}(B)\);
S5: If \(B'[7]\) goto S3 \(\parallel C \leftarrow \text{Inc}(C)\);
S6: If \(C'[7]\) goto S4 \(\parallel B \leftarrow \text{Add}(B,C)\);
S7: If \(B'[7]\) goto S4 \(\parallel C \leftarrow \text{Add}(A,C)\);
S8: \(U \leftarrow C \parallel\) goto S1;
End Alg

5.1 Design a data subsystem that is adequate to execute the algorithm and draw the schematic diagram.
5.2 Design a control subsystem and draw the state diagram.

5.3 Implement the control subsystem with a one-hot encoding design. Draw the logic diagram.