CPE 322-01: Digital Hardware Design Fundamentals

Homework 2: Design & Implementation of Sequential FSM Logic

Submitted by: Hannah Kelley

Deadline: February 13, 2026

Part 1: Mealy Implementation of the FSM State Graph

Table 1 shows the truth table for the M-Bit 2's Complement Addition Boolean Network. To turn the truth table into a state graph, the carry was assumed to be the state. C_{in} was assumed to be the present state and C_{out} was assumed to be the next state. Figure 1 shows the Mealy state graph with states S0 and S1 where S0 is the initial state. The inputs and outputs of the design are given by AB/S where A and B are inputs and S is the output.

A	B	C_{in}/Present State	S	C_{out}/Next State
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: M-Bit 2's Complement Addition Boolean Network Truth Table

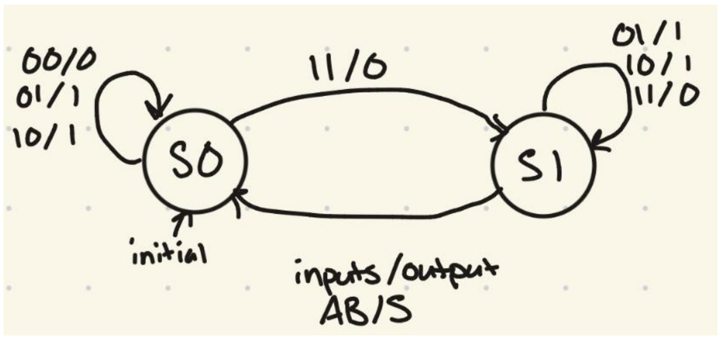


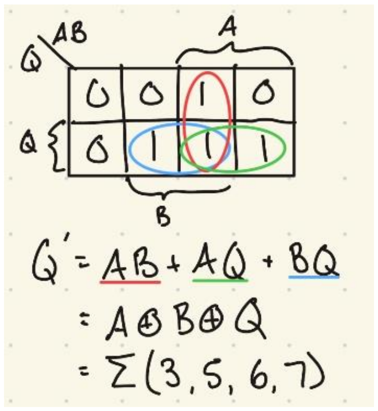
Figure 1: Mealy State Graph

Part 2: Mealy minimum 2-level AND/OR/NOT logic & D-flip-flop Implementation

Next the truth table was turned into a state table. Then the next state Q was minimized using a K-Map to obtain the simplified logical expression for Q'. Table 2 shows the state table and Figure 2 shows the K-map simplification. Figure 3 shows the implementation of the simplified logical expression using on AND, OR, and NOT gates along with a single D flip-flop to represent the states Q and Q'. Table 3 evaluates the correctness of the design by comparing the expected outputs based on the state graph to the actual outputs. As Table 3 shows, the

A	B	Q	S	Q'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2: M-Bit 2's Complement Addition Boolean Network State Table



$$Q' = \underline{AB} + \underline{AQ} + \underline{BQ}$$
$$= A \oplus B \oplus Q$$
$$= \Sigma(3, 5, 6, 7)$$

Figure 2: K-Map Simplification

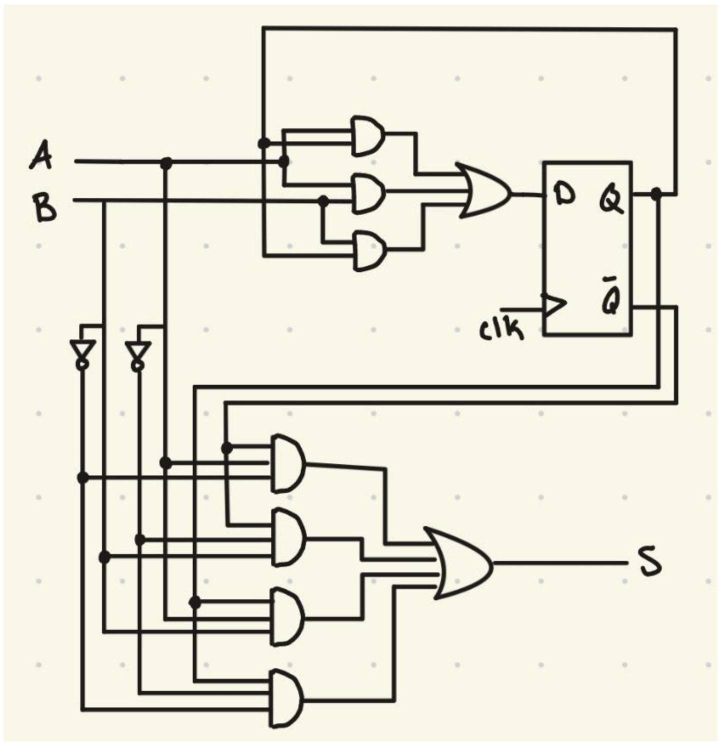


Figure 3: Mealy Minimum Logic and Flip-Flop Representation

Inputs		Current State	State Graph (Expected)		Design (Actual)	
			Output	Next State	Output	Next State
A	B	Q	S	Q'	S	Q'
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	1	1

Table 3: Expected vs. Actual Outputs

Part 3: Mealy ROM logic realization & D-flip-flop Implementation

Next, the combination implementation was swapped for a ROM implementation. A single D flip-flop was still necessary to represent the states Q and Q'. The minimum size of the ROM was determined to be 8 words (3 bits each) by 2 bits. Table 4 shows the full ROM table. The inputs QAB are represented by the address bus $A_2A_1A_0$ and the outputs Q'S are represented by the output bus O_1O_0. Figure 4 shows the full ROM implementation.

A_2	A_1	A_0	O_1	O_0
Q	A	B	Q'	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 4: ROM Address Table

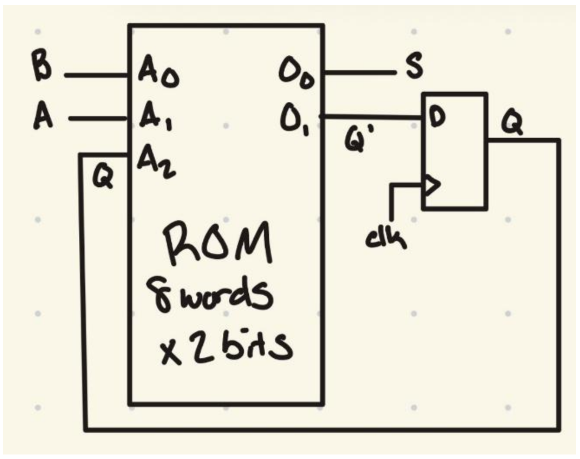


Figure 4: Mealy ROM Implementation

Part 4: Mealy 4-to-1 MUX logic realization & D-flip-flop Implementation

Next, a 4-1 multiplexer implementation of the same design was implemented. One D flip-flop will be used to store the state Q. The select lines were chosen to be A and Q. Next, the outputs S and Q' were expressed in relation to the third input B which is present on the input lines.

Q	A	B	S		Q'	
0	0	0	0	B	0	0
0	0	1	1		0	
0	1	0	1	B'	0	B
0	1	1	0		1	
1	0	0	1	B'	0	B
1	0	1	0		1	
1	1	0	0	B	1	1
1	1	1	1		1	

Table 5: Multiplexer Table

Based on this representation of S and Q', the multiplexer implementation was created as shown in Figure 5.

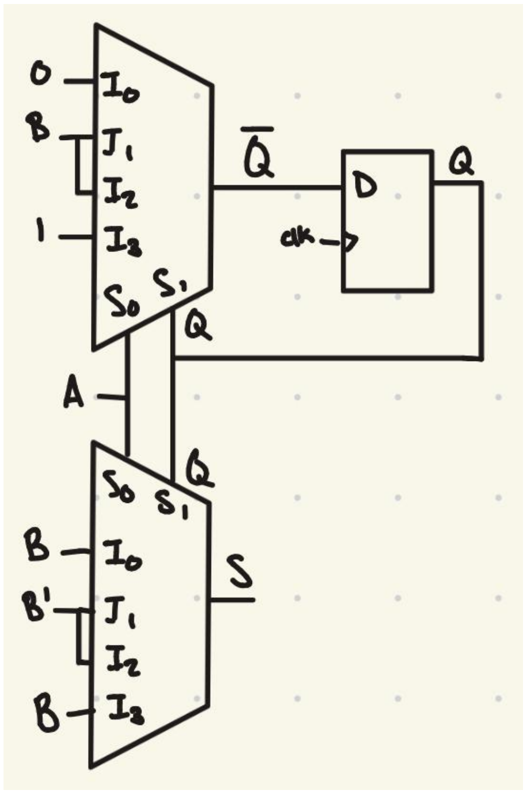


Figure 5: Multiplexer Implementation

Part 5: Moore Implementation of the FSM State Graph

Table 6 shows the state definitions for the Moore state machine according to the inputs A and B. Figure 6 shows the Moore state graph based on the complete transition table in Table 7 with state S0 as the initial state.

State	Carry	Sum
S0	0	0
S1	0	1
S2	1	0
S3	1	1

Table 6: Moore State Definitions

Current State	A	B	Next State	Output (S)
S0	0	0	S0	0
	0	1	S1	
	1	0	S1	
	1	1	S2	
S1	0	0	S0	1
	0	1	S1	
	1	0	S1	
	1	1	S2	
S2	0	0	S1	0
	0	1	S2	
	1	0	S2	
	1	1	S3	
S3	0	0	S1	1
	0	1	S2	
	1	0	S2	
	1	1	S3	

Table 7: Moore State Transition Table

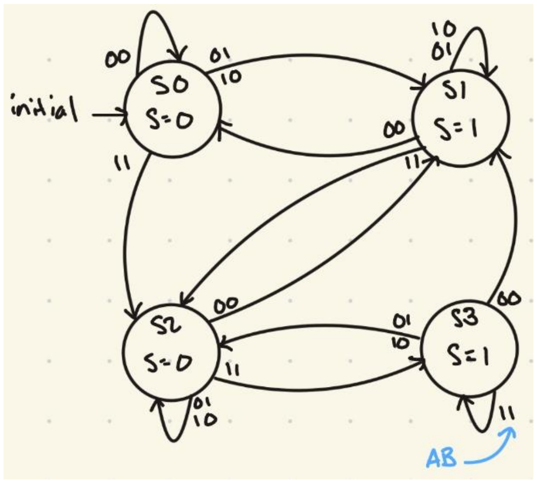


Figure 6: Moore State Graph

Part 6: Moore One-hot encoded logic realization & D-flip-flop Implementation

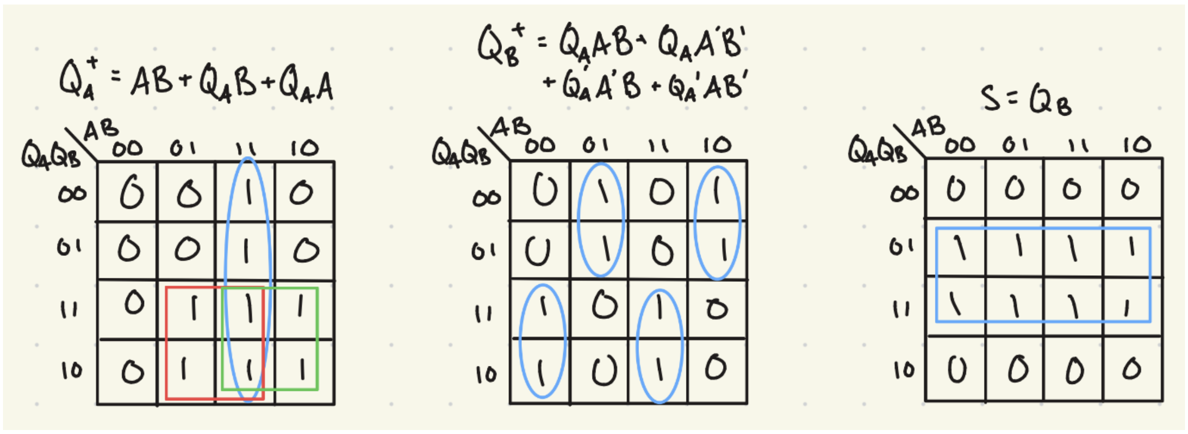
Next, the Moore state graph was implemented using the minimum number of AND, OR, and NOT gates and D flip-flops. Table 8 shows the state table. Table 9 shows the complete state transition table. Figure 7 shows the K-Maps used to obtain simplified expressions for Q_A^+, Q_B^+, and S. Figure 8 shows the one-hot encoding implementation of the design. Table 10 evaluates the correctness of the design by comparing expected outputs based on the state graph to the actual design outputs. As Table 10 shows, the expected and actual outputs match.

Current State	**Next State**				**Output S**
	AB=00	**01**	**10**	**11**	
S0	S0	S1	S1	S2	0
S1	S0	S1	S1	S2	1
S2	S1	S2	S2	S3	0
S3	S1	S2	S2	S3	1

Table 8: State Table

Current State		Inputs		Next State		Output
Q_A	Q_B	A	B	Q_A^+	Q_B^+	S
0	0	0	0	0	0	0
0	0	0	1	0	1	
0	0	1	0	0	1	
0	0	1	1	1	1	
0	1	0	0	0	0	1
0	1	0	1	0	1	
0	1	1	0	0	1	
0	1	1	1	1	1	
1	0	0	0	0	1	0
1	0	0	1	1	0	
1	0	1	0	1	0	
1	0	1	1	1	1	
1	1	0	0	0	1	1
1	1	0	1	1	0	
1	1	1	0	1	0	
1	1	1	1	1	1	

Table 9: State Transition Table



$Q_A^+ = AB + Q_AB + Q_AA$
$Q_B^+ = Q_AAB + Q_AA'B' + Q_A'A'B + Q_A'AB'$
$S = Q_B$
Q_AQ_B
AB
00
01
11
10

Figure 7: K-Maps

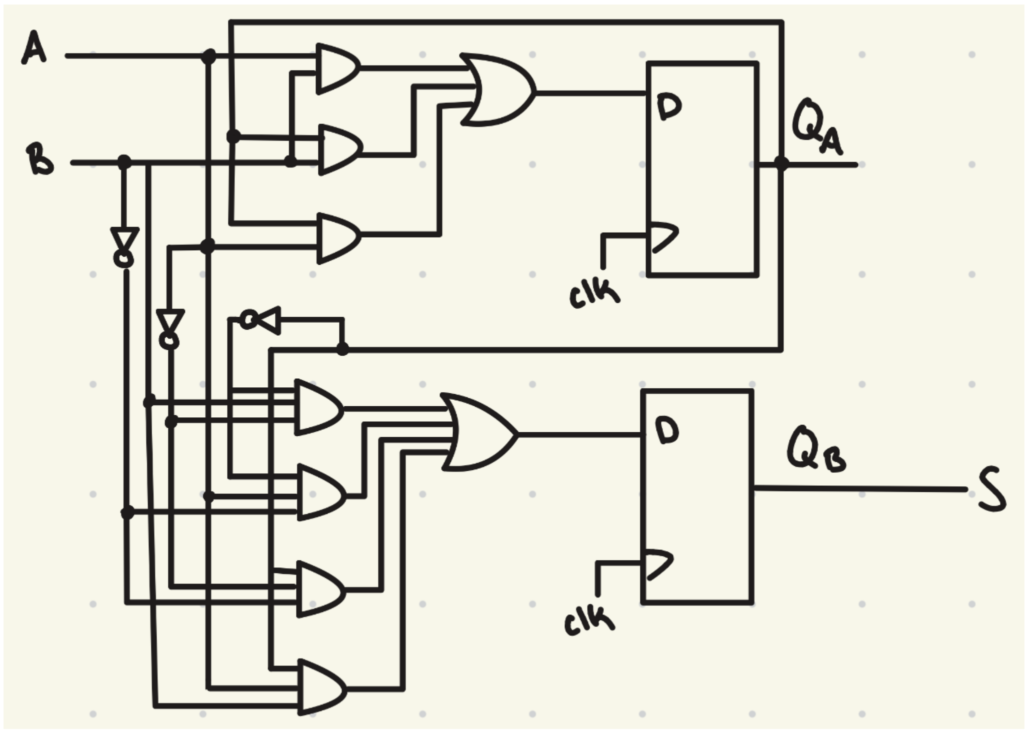


Figure 8: One-Hot Encoding Implementation

Current State		Inputs		State Graph (Expected)			Design (Actual)		
				Next State		Output	Next State		Output
Q_A	Q_B	A	B	Q_A^+	Q_B^+	S	Q_A^+	Q_B^+	S
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1		0	1	
0	0	1	0	0	1		0	1	
0	0	1	1	1	1		1	1	
0	1	0	0	0	0	1	0	0	1
0	1	0	1	0	1		0	1	
0	1	1	0	0	1		0	1	
0	1	1	1	1	1		1	1	
1	0	0	0	0	1	0	0	1	0
1	0	0	1	1	0		1	0	
1	0	1	0	1	0		1	0	
1	0	1	1	1	1		1	1	
1	1	0	0	0	1	1	0	1	1
1	1	0	1	1	0		1	0	
1	1	1	0	1	0		1	0	
1	1	1	1	1	1		1	1	

Table 10: Expected vs. Actual Results