Automated Theatre Lighting and Spotlight Control System

Final Project Report

Digital System Design — ECE033 Course Project

Author: Omar Aly

December 1, 2025

Contents

Executive Summary **1**

1 Introduction **4**
1.1 Project Context and Aim 4
1.2 System Requirements 4

2 Design Approach and Solution **6**
2.1 Inputs, Outputs, Constraints, and Internal Signals 6
2.2 Divide-and-Conquer FSM Architecture 7
2.3 Mode FSM 8
2.3.1 Mode Encoding 8
2.3.2 State Diagram 9
2.3.3 State Table, State Assignment, and Transition Table 10
2.3.4 Transition Equations and K-maps 11
2.4 Spotlight FSM 13
2.4.1 Sensor Encoding 13
2.4.2 State Diagram 14
2.4.3 State Table, State Assignment, and Transition Table 14
2.4.4 Transition Equations and K-maps 15
2.5 Output Equations and Logic Diagram 17
2.5.1 K-Maps Used to Derive Output Equations 17
2.5.2 Output Logic Diagrams 19

3 Implementation, Testing, and Results **20**
3.1 SystemVerilog Implementation Overview 20
3.2 Test Case Descriptions 20
3.3 Waveforms, Logs, and synthesis 22

Conclusion **31**

Executive Summary

System Overview

This project implements a synchronous control system for a multi-purpose theatre. The system is responsible for coordinating the house lights, visualization display, and the theatre's spotlight based on the selected mode of operation and three active-low position sensors. The design is implemented using two interacting finite-state machines (FSM): a *Mode FSM* that manages theater modes and a *Spotlight FSM* that controls spotlight movement, and on/off behavior.

The system is required to:

- Support four mutually exclusive modes selected by switches: House, Music, Mode, Display
- Use an overall enable signal (`EN`) so that the theatre can be shut down and safely transitioned to an "all off" state with House lights on, or be turned on and the system be active.
- Ensure that the default behavior when the system is enabled is to have the house lights on.
- Control a spotlight that tracks the performer using three active-low sensors: left (`TL`), center (`TC`), and right (`TR`).
- Enforce that spotlight movement between left and right passes through the center position.
- Ensure all mode changes pass through an intermediate state where all lights are off.

Design Approach

The system was designed with the following approach:

1. Decompose the system into two FSMs: a Mode FSM that uses three flip-flops (Q_2, Q_1, Q_0), and a Spotlight FSM that uses two flip-flops (R_1, R_0).

2. Derive state diagrams, state tables, state assignments, transition tables, and K-map-based next state and output equations consistent with the project constraints (binary encoding, asynchronous reset, and a maximum of 5 flip-flops).

3. Implement the design structurally in SystemVerilog using D flip-flops, combinational next-state modules, encoders for modes and sensors, and an output module.

1 Introduction

1.1 Project Context and Aim

Given a goal to design a system for a theatre that often host a mixture of events: plays, concerts, guest speakers, movies, community events, and musical performances. Each event type demands different lighting, visualization, and spotlights behavior. This project focuses on designing a digital hardware controller that automates:

- House lights (for safe entry/exit and between shows),
- Visualization display (for music and plays),
- spotlights (for speakers and plays).

The goal is to create a finite-state-machine-based theatre controller, using binary state assignments, asynchronous reset, and flip-flops that can be synthesized and tested in SystemVerilog.

1.2 System Requirements

The provided product requirements document (PRD) specifies the following key behaviors:

- The theatre supports four modes: House, Music, Speaker, Play Each mode is controlled by a dedicated switch; only one mode switch is on at a time.
- A global enable signal (`EN`) turns the system on or off.

- When the system is enabled and no mode is yet fully selected, the default behavior is house lights on.
- During music, a visualization is Displayed (VD = 1).
- During a play, both the visualization and the spotlight are active.
- During speakers, only the spotlight is needed.
- The spotlight is connected at port ”S1” through three wires: one light control bit and two position bits forming a 2-bit encoded position.
- The spotlight can be set to left, center, or right: 00 = left, 01 = center, 11 = right. The spotlight always turns on and off from the center position.
- For the spotlight to switch from left to right, it must transition through center.
- Spotlight aiming is controlled by three, active-low sensors: TL, TC, TR. only 1 can be active at a time
- All mode switches turn off before the next mode is turned on, and transitions between modes should pass through a state with all lights off.

2 Design Approach and Solution

2.1 Inputs, Outputs, Constraints, and Internal Signals

The system will use the following signals:

Inputs

- `clk`: System clock.
- `reset`: Asynchronous active-high reset.
- `EN`: Overall enable.
- `House`, `Music`, `Speaker`, `Play`: Mode switches (Active high, only one high at a time).
- `TL`, `TC`, `TR`: Active-low spotlight position sensors.

Outputs

- `HL`: House lights.
- `VD`: Visualization display.
- `S1[2:0]`: Spotlight control:
  - `S1[2]`: Spotlight light on/off (1 = on, 0 = off).
  - `S1[1:0]`: Spotlight position (00 = left, 01 = center, 11 = right).

Constraints

- Only 1 mode can be active at a time
- Default state is that house lights are on
- All lights must turn off between mode changes
- spotlight follows actor's position
- if enable is off, everything shuts

Internal Signals

- Mode state bits: Q_2, Q_1, Q_0 (and their next-states $Q2_n, Q1_n, Q0_n$).
- Spotlight state bits: R_1, R_0 (and their next-states $R1_n, R0_n$).
- Mode encoder outputs: `M[1:0]`. M0 = House [00], M1 = Music [01], M2 = Speaker [10] , M3 = Play [11]. Only one mode is active at a time. When a new mode is selected, the system passes through an all lights off transition state before activating the new mode outputs.
- Sensor encoder outputs: `T[1:0]`. Consists of three sensors TL, TC, TR. Active Low. controls the spotlight location, only 1 active at a time (it tracks the main character movement). TL: Turn left, 00. TC: turn center, 01. TR: turn right, 11. Movement from left to right or from right to left must logically pass through the center.
- Spotlight activation internal signal: `SP_A` (used to activate spotlight FSM).

2.2 Divide-and-Conquer FSM Architecture

The design uses two FSMs connected together:

- **Mode FSM**: Encodes the mode of the theatre using Q_2, Q_1, Q_0. It controls when the house lights and visualization are on, and generates `SP_A` to indicate when the spotlight subsystem should be active (Speaker and Play modes only).

- **Spotlight FSM**: Encodes the horizontal position of the spotlight using R_1, R_0, moving between left, center, and right based on sensor encoder outputs (`T1,T0`) and only when `EN = 1` and `SP_A = 1`.

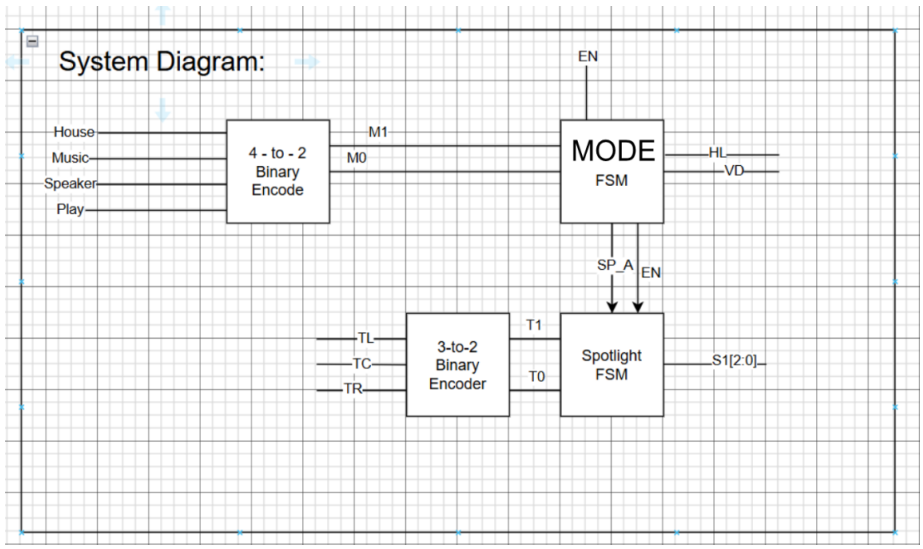

Figure 2.1: state diagram connecting the Mode FSM and Spotlight FSM.

The Mode FSM outputs determine which mode the system is in and whether the spotlight subsystem is active (`SP_A`). The Spotlight FSM uses the sensor encoder inputs to update its internal position state.

2.3 Mode FSM

2.3.1 Mode Encoding

The system have got four Active high Mode switches: House, Music, Speaker, and play. They pass through a 4-to-2 Binary encoder so that only 2 bits will pass to Mode FSM:

- `House = 1` $\Rightarrow M_1M_0 = 00$ (House Mode).
- `Music = 1` $\Rightarrow M_1M_0 = 01$ (Music Mode).
- `Speaker = 1` $\Rightarrow M_1M_0 = 10$ (Speaker Mode).

- `Play = 1` $\Rightarrow M_1 M_0 = 11$ (Play Mode).

2.3.2 State Diagram

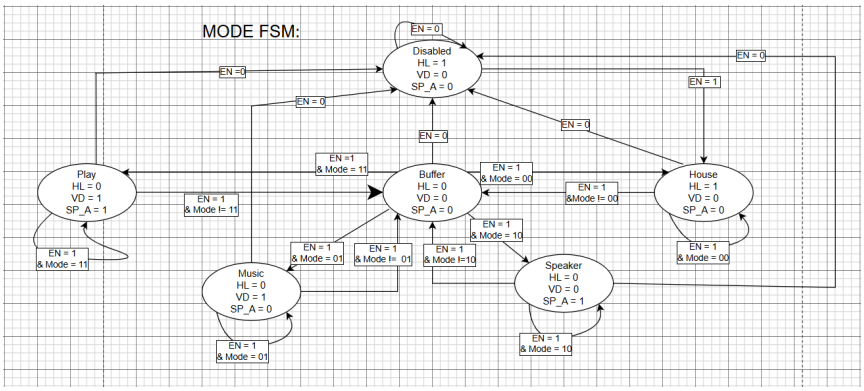

Figure 2.2: Mode FSM state diagram using Q_2, Q_1, Q_0.

the Mode FSM starts off at a disabled state where all outputs are off except for House lights, then takes the M1, and M0 from the Binary encoder, along with the Enable (EN) switch as inputs, then uses it to determine the Mode state we're currently at. each state provides a different output of HL (House lights), VD (Visulaization display), and SP_A (Internal output). Through Mode changes, the system goes through an all lights off stage (buffer state) where all lights are off.

2.3.3 State Table, State Assignment, and Transition Table

State Table (Next State)

PS	EN = 0	EN = 1, M=00	EN = 1, M=01	EN = 1, M=10	EN = 1, M=11
Disabled	Disabled	House	House	House	House
House	Disabled	House	Buffer	Buffer	Buffer
Music	Disabled	Buffer	Music	Buffer	Buffer
Speaker	Disabled	Buffer	Buffer	Speaker	Buffer
Play	Disabled	Buffer	Buffer	Buffer	Play
Buffer	Disabled	House	Music	Speaker	Play

Table 2.1: Mode FSM state Table (Next State)

State Table (Output)

State	HL	VD	SP_A (Internal Output)
Disabled	1	0	0
House	1	0	0
Music	0	1	0
Speaker	0	0	1
Play	0	1	1
Buffer	0	0	0

Table 2.2: Mode FSM State Table (Output)

State Assignment

State	Q_2 Q_1 Q_0
Disabled	000
House	001
Music	010
Speaker	011
Play	100
Buffer	101

Table 2.3: Binary Encoding for Mode FSM States

Transition Table

PS (Q2Q1Q0)	EN=0	EN=1,M=00	EN=1,M=01	EN=1,M=10	EN=1,M=11	HL,VD,SP_A
000	000	001	001	001	001	1 0 0
001	000	001	101	101	101	1 0 0
010	000	101	010	101	101	0 1 0
011	000	101	101	011	101	0 0 1
100	000	101	101	101	100	0 1 1
101	000	001	010	011	100	0 0 0

Table 2.4: Mode FSM Transition table

2.3.4 Transition Equations and K-maps

The next-state logic for the Mode FSM was derived from the transition table using 5-variable K-maps over Q_2, Q_1, Q_0, M_1, M_0, with `EN` factored out as a common enable signal. The simplified next-state equations are:

$$Q_2^* = EN\big(Q_2\overline{Q_0} + M_0M_1Q_2 + M_0Q_0\overline{Q_2} + M_1Q_1\overline{Q_0} + Q_1\overline{M_0}\,\overline{M_1} + M_1Q_0\overline{Q_1}\,\overline{Q_2}\big),$$

$$Q_1^* = EN\big(M_0Q_0Q_2\overline{M_1} + M_0\overline{M_1}Q_1\overline{Q_0} + M_1Q_0Q_1\overline{M_0} + M_1Q_0Q_2\overline{M_0}\big),$$

$$Q_0^* = EN\big(\overline{M_0} + M_1\overline{Q_2} + Q_0\overline{Q_2} + \overline{M_1}\,\overline{Q_0}\,\overline{Q_1}\big).$$

K-map Derivations

The equations above were obtained using K maps for each next-state variable. The completed K-maps are included below for reference.

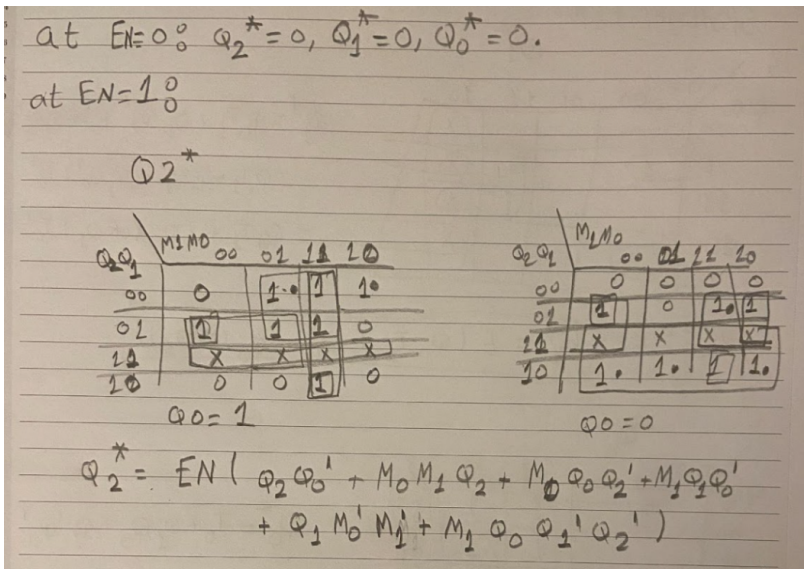

Figure 2.3: K-map used to derive Q_2^*.

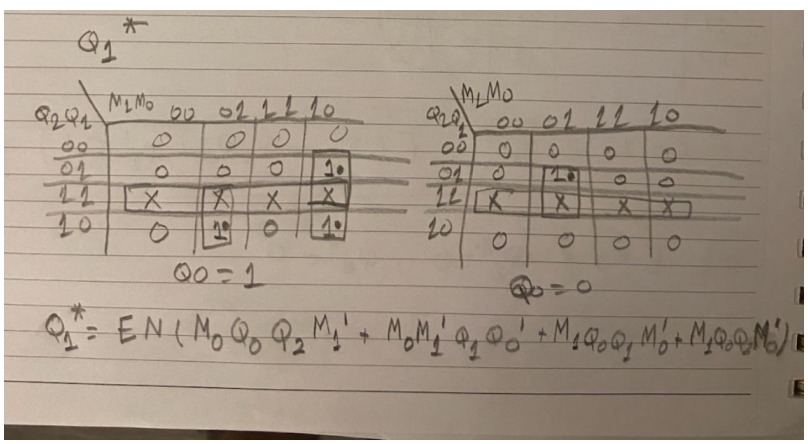

Figure 2.4: K-map used to derive Q_1^*.

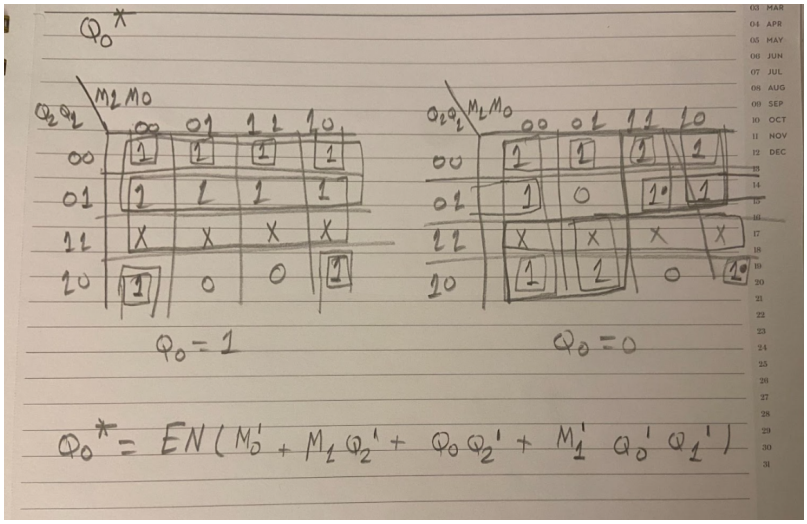

Figure 2.5: K-map used to derive Q_0^*.

2.4 Spotlight FSM

2.4.1 Sensor Encoding

The three active-low sensors are encoded into a 2-bit code (T_1, T_0) using a 3-to-2 encoder so that only two bits are passed into the Spotlight FSM:

- `TL` = 0 $\Rightarrow T_1T_0 = 00$ (left).
- `TC` = 0 $\Rightarrow T_1T_0 = 01$ (center).
- `TR` = 0 $\Rightarrow T_1T_0 = 11$ (right).
- No sensor active $\Rightarrow T_1T_0 = 01$ (center as default).

2.4.2 State Diagram

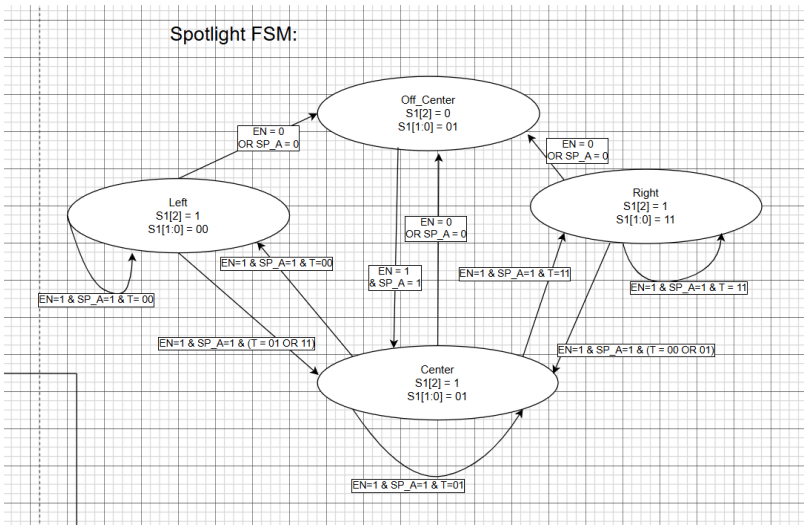

Figure 2.6: Spotlight FSM state diagram using R_1, R_0.

The Spotlight FSM only updates its position when $E = EN \cdot SP_A = 1$. Otherwise, it holds its state.

2.4.3 State Table, State Assignment, and Transition Table

State Table (Next State)

PS	E=0	E=1, T=00	E=1, T=01	E=1, T=11
Off_Center	Off_Center	Center	Center	Center
Center	Off_Center	Left	Center	Right
Left	Off_Center	Left	Center	Center
Right	Off_Center	Center	Center	Right

Table 2.5: Spotlight FSM State Table (Next State)

State Table (Output)

Let $E = EN \cdot SP_A$.

State	S1[2]	S1[1:0]
Off_Center	0	01
Center	1	01
Left	1	00
Right	1	11

Table 2.6: Spotlight FSM State Table (Output)

State Assignment

State	Q_1 Q_0
Off_Center	00
Center	01
Left	10
Right	11

Table 2.7: Binary Encoding for Spotlight FSM States

Transition Table

Let $E = EN \cdot SP_A$.

PS (R1R0)	E = 0	E = 1, T = 00	E = 1, T = 01	E = 1, T = 11	S1[2]	S1[1:0]
00	00	01	01	01	0	01
01	00	10	01	11	1	01
10	00	10	01	01	1	00
11	00	01	01	11	1	11

Table 2.8: Spotlight FSM Transition Table (Encoded)

2.4.4 Transition Equations and K-maps

For the Spotlight FSM, the state bits are R_1, R_0, the sensor encoder outputs are T_1, T_0, and we define a combined enable

$$E = EN \cdot SP_A.$$

Using the transition table and K-maps over R_1, R_0, T_1, T_0, the simplified next-state equations are:

$$R_1^* = E\big(R_0T_1 + T_0'(R_0 \oplus R_1)\big),$$
$$R_0^* = E\big(T_0 + R_1R_0 + R_1'R_0'\big).$$

K-map Derivations

The equations above were obtained from K-maps for R_1^* and R_0^*. The completed K-maps are shown below.
Note: In the spotlight K-map derivations, the notation Q_1, Q_1^*, Q_0, Q_0^* appears in place of R_1, R_1^*, R_0, and R_0^*. These symbols refer to the same spotlight state bits; only the variable labels differ during the derivation process.

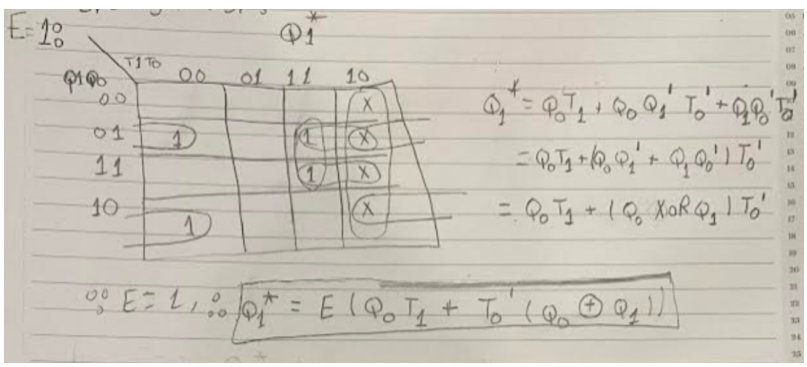

Figure 2.7: K-map used to derive R_1^*.

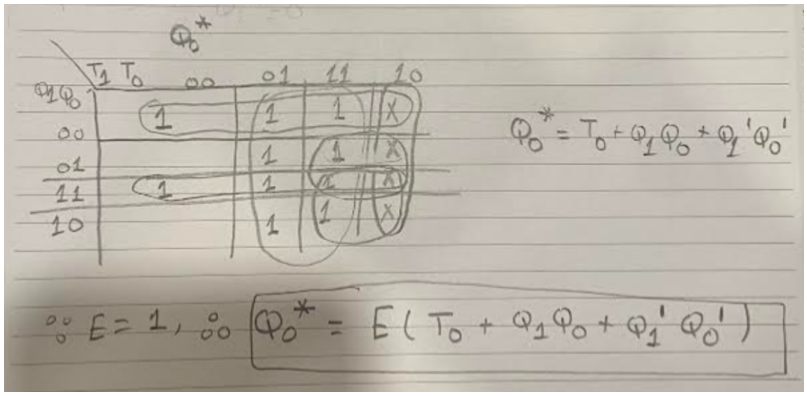

Figure 2.8: K-map used to derive R_0^*.

2.5 Output Equations and Logic Diagram

The output logic combines the mode FSM state bits (Q_2, Q_1, Q_0) and the spotlight FSM state bits (R_1, R_0). The simplified output equations are:

$$
\begin{aligned}
HL &= \overline{Q_2}\,\overline{Q_1}, \\
VD &= \overline{Q_0}\,(Q_1 \oplus Q_2), \\
SP_A &= \overline{Q_2}Q_1Q_0 \;+\; Q_2\overline{Q_1}\,\overline{Q_0}, \\
S1[2] &= R_1 + R_0, \\
S1[1] &= R_1R_0, \\
S1[0] &= \overline{R_1} + R_0.
\end{aligned}
$$

2.5.1 K-Maps Used to Derive Output Equations

The following K maps were used to obtain the minimized output expressions. Two K-maps are shown: one for the Mode FSM outputs (HL, VD, SP_A) and one for the Spotlight FSM outputs $(S1[2:0])$.

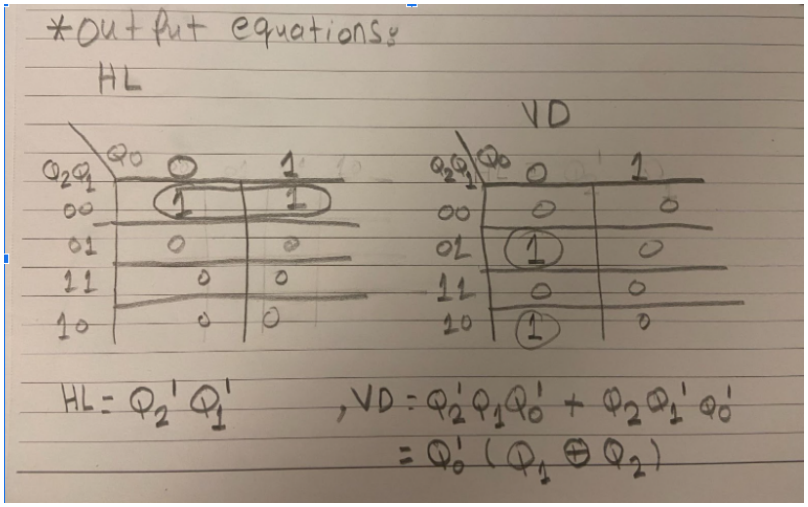

Figure 2.9: K-map derivation for Mode FSM outputs HL, VD, SP_A.

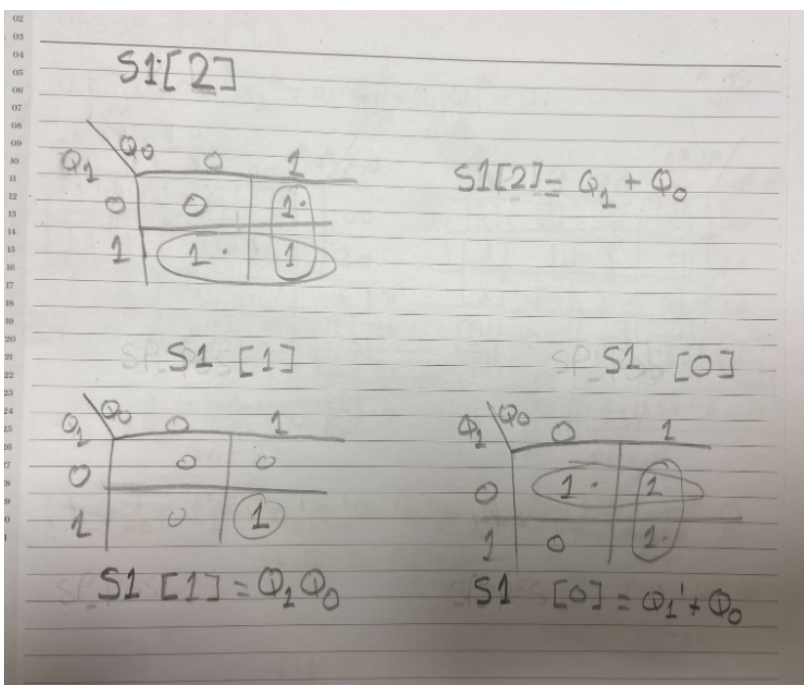

Figure 2.10: K-map derivation for spotlight output bits $S1[2], S1[1], S1[0]$.

2.5.2 Output Logic Diagrams

Two logic diagrams are included below: (1) Mode FSM output logic, (2) Spotlight FSM output logic.

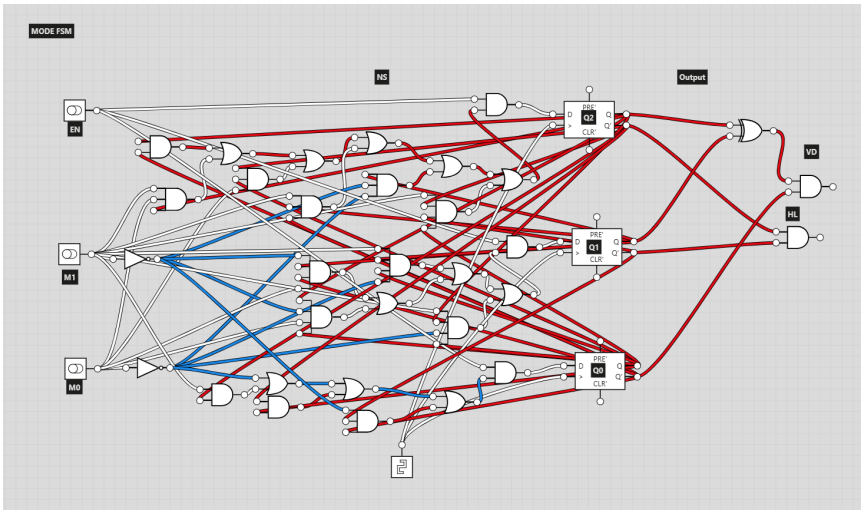

Figure 2.11: Logic diagram for Mode FSM outputs HL, VD, SP_A.

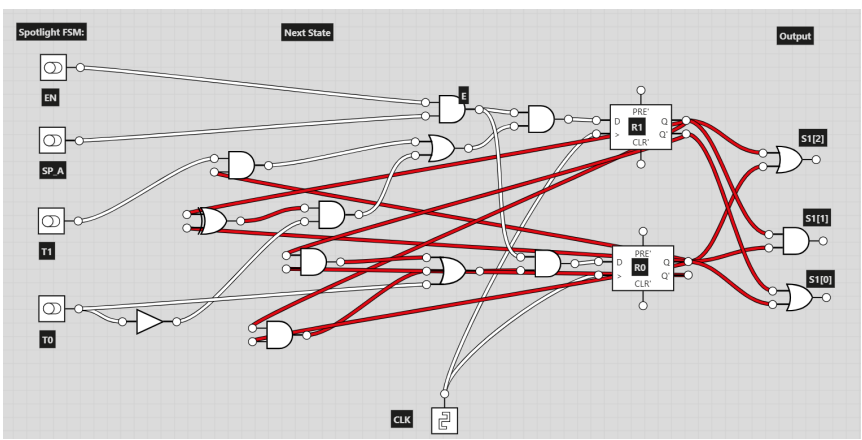

Figure 2.12: Logic diagram for Spotlight FSM outputs $S1[2], S1[1], S1[0]$.

3 Implementation, Testing, and Results

3.1 SystemVerilog Implementation Overview

The implementation follows a structural SystemVerilog style using separate modules for:

- A D flip-flop with asynchronous reset.
- 4-to-2 mode encoder (`encoder4to2`).
- 3-to-2 sensor encoder (`encoder3to2`).
- Mode FSM next-state logic (`ns_Mode_FSM`).
- Spotlight FSM next-state logic (`ns_Spotlight_FSM`).
- Output equations (`theatre_outputs`).
- Overall module (`theatre`) that combines everything.

The full System Verilog design and testbench files were submitted separately as required.

3.2 Test Case Descriptions

1. **Case 1: Enable on, no mode**
   - Test that when system turns on after reset, system is active at disabled state.
   - Initially: set EN = 1
   - Expected output: HL = 1, VD = 0, S1 = 001

2. **Case 2: House mode**
   - Test house mode
   - EN = 1, House = 1, TL = TC = TR = 1
   - Expected output: HL = 1, VD = 0, S1 = 001

3. **Case 3: Music Mode**
   - Test Music mode
   - EN = 1, Music = 1
   - Expected output: HL = 0, VD = 1, S1 = 001

4. **Case 4: Speaker Mode**
   - Test Speaker mode
   - EN = 1, Speaker = 1, then test different TL, TC and TR patterns of 00, 01, 11
   - Expected output: HL = 0, VD = 0, S1 = 100, 101, 111. (as the pattern of spotlight sensors above)

5. **Case 5: Play mode**
   - Test Play mode
   - EN = 1, Play = 1, then test different TL, TC and TR patterns of 00, 01, 11
   - Expected output: HL = 0, VD = 1, S1 = 100, 101, 111. (as the pattern of spotlight sensors above)

6. **Case 6: Asynchronous reset test**
   - Test that when reset turns to 1, everything resets regardless of clock.
   - Expected output: HL = 1, VD = 0, S1 = 001 (off, and position at center)

7. **Case 7: Transition test**
   - Between mode transitions, the system should enter a buffer state where everything is off HL = 0, VD = 0, S1 = 001 (off and center).
   - Start in play = 1, then turn play = 0 and Play = 1
   - Expected output: when play turns to 0, Mode FSM goes to buffer state where HL = 0 and VD = 0, S1 = 001, then it turns to play mode outputs

8. **Case 8: Global Disable override**
   - Test EN override behaviour
   - Start in any mode (I choose play mode for the sake of testing) with EN = 1. Then turn EN = 0.
   - Expected Output: Mode FSM goes to disabled state, where HL = 1, VD = 0, S1 = 001 (off and center)

3.3 Waveforms, Logs, and synthesis

Figure 3.1: Simulation waveform showing theatre system Behaviour.

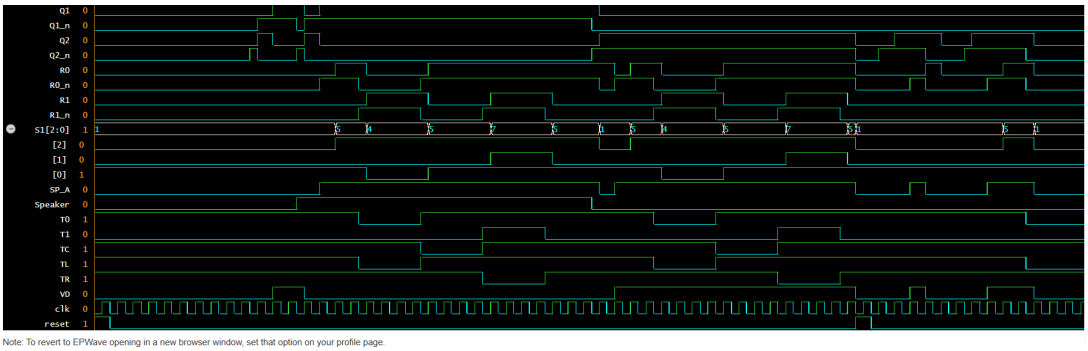

Figure 3.2: Simulation waveform showing theatre system Behaviour.

clk	rst	EN	H	M	Sp	P	TL	TC	TR	HL	VD	S1
0	1	0	0	0	0	0	1	1	1	1	0	001
1	1	0	0	0	0	0	1	1	1	1	0	001
0	0	0	0	0	0	0	1	1	1	1	0	001
1	0	0	0	0	0	0	1	1	1	1	0	001
0	0	0	0	0	0	0	1	1	1	1	0	001
1	0	0	0	0	0	0	1	1	1	1	0	001
0	0	0	0	0	0	0	1	1	1	1	0	001
1	0	0	0	0	0	0	1	1	1	1	0	001
0	0	1	0	0	0	0	1	1	1	1	0	001
1	0	1	0	0	0	0	1	1	1	1	0	001
0	0	1	0	0	0	0	1	1	1	1	0	001
1	0	1	0	0	0	0	1	1	1	1	0	001
0	0	1	0	0	0	0	1	1	1	1	0	001
1	0	1	0	0	0	0	1	1	1	1	0	001
0	0	1	1	0	0	0	1	1	1	1	0	001
1	0	1	1	0	0	0	1	1	1	1	0	001
0	0	1	1	0	0	0	1	1	1	1	0	001
1	0	1	1	0	0	0	1	1	1	1	0	001
0	0	1	1	0	0	0	1	1	1	1	0	001
1	0	1	1	0	0	0	1	1	1	1	0	001
0	0	1	0	1	0	0	1	1	1	1	0	001
1	0	1	0	1	0	0	1	1	1	0	0	001
0	0	1	0	1	0	0	1	1	1	0	0	001
1	0	1	0	1	0	0	1	1	1	0	1	001
0	0	1	0	1	0	0	1	1	1	0	1	001
1	0	1	0	1	0	0	1	1	1	0	1	001

Figure 3.3: log output produced.

0	0	1	0	0	1	0	1	1	1	|	0	1	001
1	0	1	0	0	1	0	1	1	1	|	0	0	001
0	0	1	0	0	1	0	1	1	1	|	0	0	001
1	0	1	0	0	1	0	1	1	1	|	0	0	001
0	0	1	0	0	1	0	1	1	1	|	0	0	001
1	0	1	0	0	1	0	1	1	1	|	0	0	101
0	0	1	0	0	1	0	1	1	1	|	0	0	101
1	0	1	0	0	1	0	1	1	1	|	0	0	101
0	0	1	0	0	1	0	0	1	1	|	0	0	101
1	0	1	0	0	1	0	0	1	1	|	0	0	100
0	0	1	0	0	1	0	0	1	1	|	0	0	100
1	0	1	0	0	1	0	0	1	1	|	0	0	100
0	0	1	0	0	1	0	0	1	1	|	0	0	100
1	0	1	0	0	1	0	0	1	1	|	0	0	100
0	0	1	0	0	1	0	0	1	1	|	0	0	100
1	0	1	0	0	1	0	0	1	1	|	0	0	100
0	0	1	0	0	1	0	1	0	1	|	0	0	100
1	0	1	0	0	1	0	1	0	1	|	0	0	101
0	0	1	0	0	1	0	1	0	1	|	0	0	101
1	0	1	0	0	1	0	1	0	1	|	0	0	101
0	0	1	0	0	1	0	1	0	1	|	0	0	101
1	0	1	0	0	1	0	1	0	1	|	0	0	101
0	0	1	0	0	1	0	1	0	1	|	0	0	101
1	0	1	0	0	1	0	1	0	1	|	0	0	101
0	0	1	0	0	1	0	1	1	0	|	0	0	101
1	0	1	0	0	1	0	1	1	0	|	0	0	111
0	0	1	0	0	1	0	1	1	0	|	0	0	111
1	0	1	0	0	1	0	1	1	0	|	0	0	111

Figure 3.4: log output produced.

0	0	1	0	0	1	0	1	1	0	|	0	0	111
1	0	1	0	0	1	0	1	1	0	|	0	0	111
0	0	1	0	0	1	0	1	1	0	|	0	0	111
1	0	1	0	0	1	0	1	1	0	|	0	0	111
0	0	1	0	0	1	0	1	1	1	|	0	0	111
1	0	1	0	0	1	0	1	1	1	|	0	0	101
0	0	1	0	0	1	0	1	1	1	|	0	0	101
1	0	1	0	0	1	0	1	1	1	|	0	0	101
0	0	1	0	0	1	0	1	1	1	|	0	0	101
1	0	1	0	0	1	0	1	1	1	|	0	0	101
0	0	1	0	0	0	1	1	1	1	|	0	0	101
1	0	1	0	0	0	1	1	1	1	|	0	0	001
0	0	1	0	0	0	1	1	1	1	|	0	0	001
1	0	1	0	0	0	1	1	1	1	|	0	1	001
0	0	1	0	0	0	1	1	1	1	|	0	1	001
1	0	1	0	0	0	1	1	1	1	|	0	1	101
0	0	1	0	0	0	1	1	1	1	|	0	1	101
1	0	1	0	0	0	1	1	1	1	|	0	1	101
0	0	1	0	0	0	1	0	1	1	|	0	1	101
1	0	1	0	0	0	1	0	1	1	|	0	1	100
0	0	1	0	0	0	1	0	1	1	|	0	1	100
1	0	1	0	0	0	1	0	1	1	|	0	1	100
0	0	1	0	0	0	1	0	1	1	|	0	1	100
1	0	1	0	0	0	1	0	1	1	|	0	1	100
0	0	1	0	0	0	1	0	1	1	|	0	1	100
1	0	1	0	0	0	1	0	1	1	|	0	1	100
0	0	1	0	0	0	1	1	0	1	|	0	1	100

Figure 3.5: log output produced.

1	0	1	0	0	0	1	1	0	1	|	0	1	101
0	0	1	0	0	0	1	1	0	1	|	0	1	101
1	0	1	0	0	0	1	1	0	1	|	0	1	101
0	0	1	0	0	0	1	1	0	1	|	0	1	101
1	0	1	0	0	0	1	1	0	1	|	0	1	101
0	0	1	0	0	0	1	1	0	1	|	0	1	101
1	0	1	0	0	0	1	1	0	1	|	0	1	101
0	0	1	0	0	0	1	1	1	0	|	0	1	101
1	0	1	0	0	0	1	1	1	0	|	0	1	111
0	0	1	0	0	0	1	1	1	0	|	0	1	111
1	0	1	0	0	0	1	1	1	0	|	0	1	111
0	0	1	0	0	0	1	1	1	0	|	0	1	111
1	0	1	0	0	0	1	1	1	0	|	0	1	111
0	0	1	0	0	0	1	1	1	0	|	0	1	111
1	0	1	0	0	0	1	1	1	0	|	0	1	111
0	0	1	0	0	0	1	1	1	1	|	0	1	111
1	0	1	0	0	0	1	1	1	1	|	0	1	101
0	1	1	0	0	0	1	1	1	1	|	1	0	001
1	1	1	0	0	0	1	1	1	1	|	1	0	001
0	0	1	0	0	0	1	1	1	1	|	1	0	001
1	0	1	0	0	0	1	1	1	1	|	1	0	001
0	0	1	0	0	0	1	1	1	1	|	1	0	001
1	0	1	0	0	0	1	1	1	1	|	0	0	001
0	0	1	0	0	0	1	1	1	1	|	0	0	001
1	0	1	0	0	0	1	1	1	1	|	0	1	001
0	0	1	0	0	0	0	1	1	1	|	0	1	001
1	0	1	0	0	0	0	1	1	1	|	0	0	001

Figure 3.6: log output produced.

```
0   0   1  0 0 0 0  1  1  1  | 0  0  001
1   0   1  0 0 0 0  1  1  1  | 1  0  001
0   0   1  0 0 0 0  1  1  1  | 1  0  001
1   0   1  0 0 0 0  1  1  1  | 1  0  001
0   0   1  0 0 0 1  1  1  1  | 1  0  001
1   0   1  0 0 0 1  1  1  1  | 0  0  001
0   0   1  0 0 0 1  1  1  1  | 0  0  001
1   0   1  0 0 0 1  1  1  1  | 0  1  001
0   0   1  0 0 0 1  1  1  1  | 0  1  001
1   0   1  0 0 0 1  1  1  1  | 0  1  101
0   0   1  0 0 0 1  1  1  1  | 0  1  101
1   0   1  0 0 0 1  1  1  1  | 0  1  101
0   0   0  0 0 0 1  0  1  1  | 0  1  101
1   0   0  0 0 0 1  0  1  1  | 1  0  001
0   0   0  0 0 0 1  0  1  1  | 1  0  001
1   0   0  0 0 0 1  0  1  1  | 1  0  001
0   0   0  0 0 0 1  0  1  1  | 1  0  001
1   0   0  0 0 0 1  0  1  1  | 1  0  001
0   0   0  0 0 0 1  0  1  1  | 1  0  001
1   0   0  0 0 0 1  0  1  1  | 1  0  001
testbench.sv:73: $finish called at 6400 (100ps)
0   0   0  0 0 0 1  0  1  1  | 1  0  001
Finding VCD file...
./theatre_wave.vcd
[2025-12-05 01:05:07 UTC] Opening EPWave...
Done
```

Figure 3.7: log output produced.

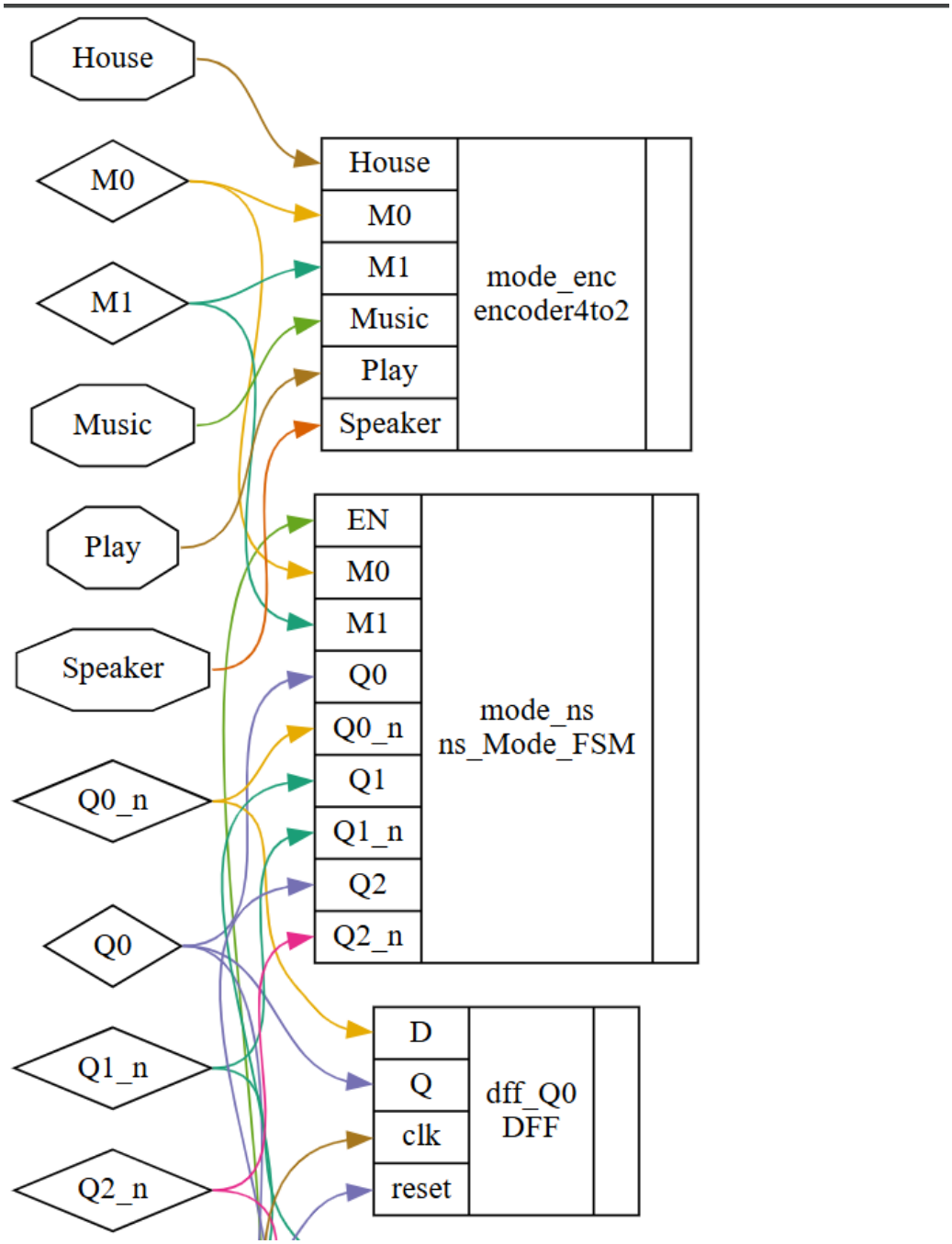

Figure 3.8: Synthesis produced.

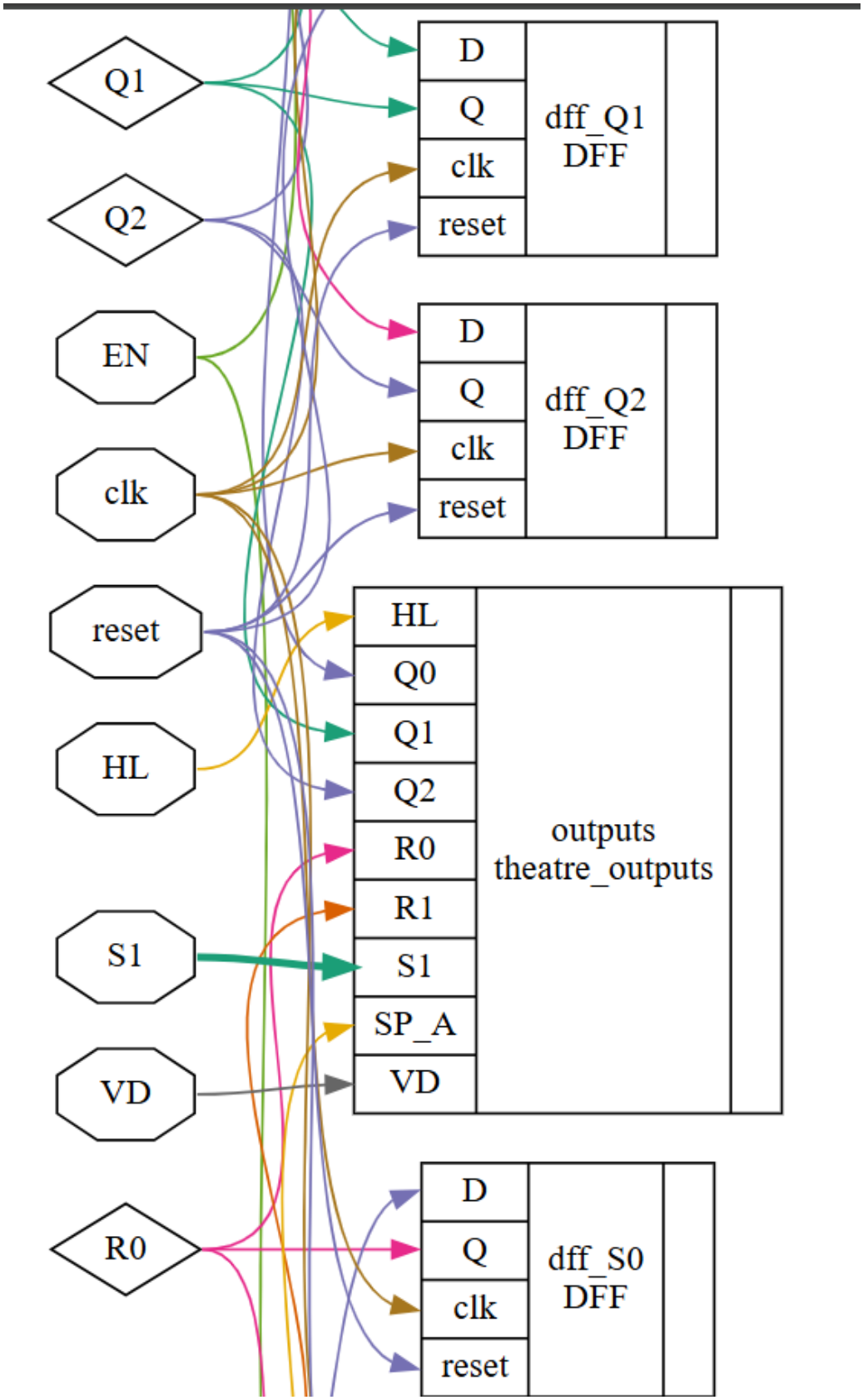

Figure 3.9: Synthesis produced.

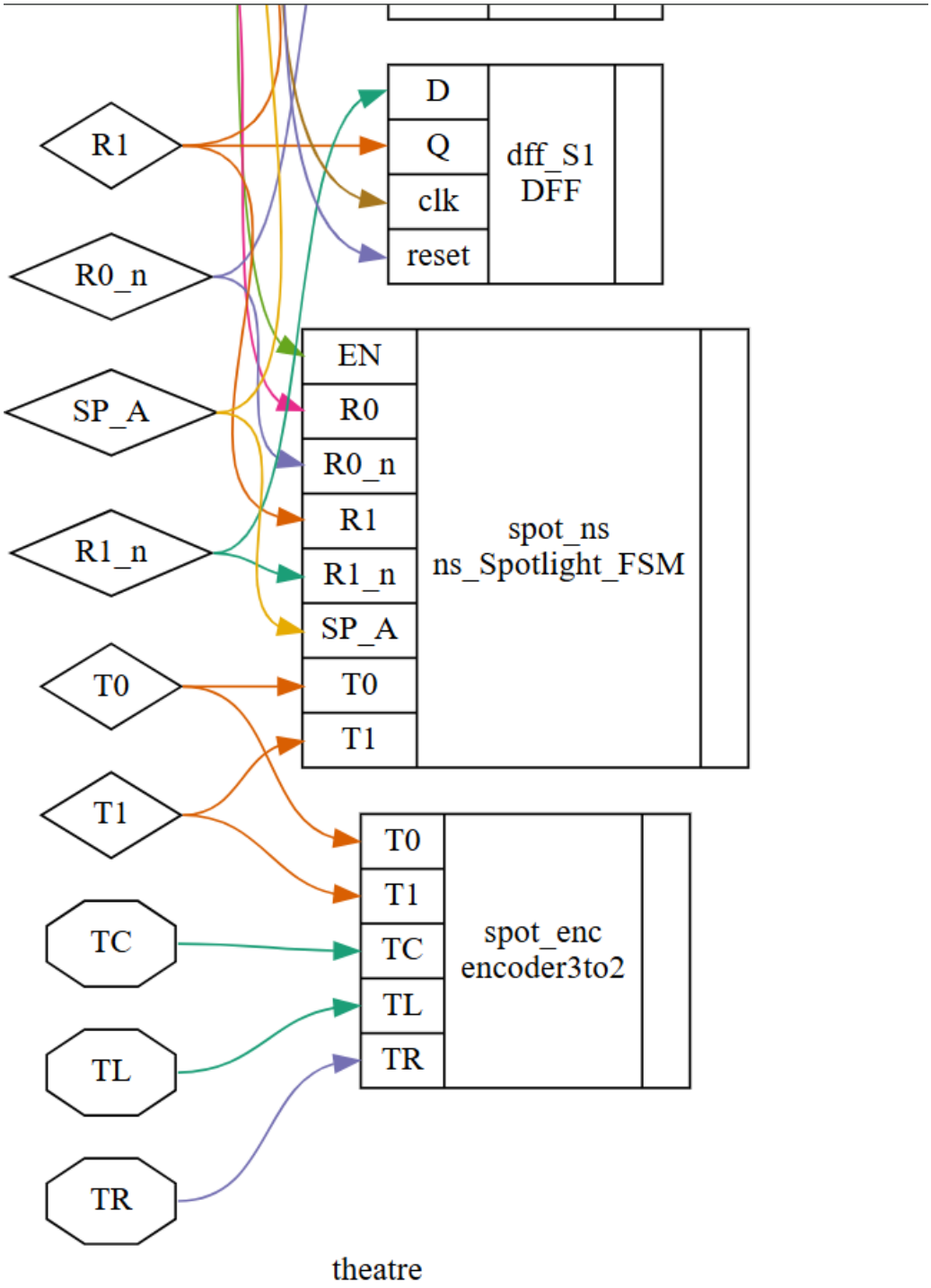

Figure 3.10: Synthesis produced.

Conclusion

From the waveform and log, we conclude:

- After reset, all states are cleared and the system starts at the Initial state
- When `EN` is with no mode active, the system moves to a state where `HL = 1` and visualization/spotlight are off, matching the requirement that the default enabled state is house lights on.
- In Speaker and Play modes, the spotlight turns on (`S1[2]` = 1) and its encoded position `S1[1:0]` follows the active-low sensors, with center as default when no sensor is active.
- When switching modes, the Mode FSM passes through the all-lights-off (buffer) state before entering the new mode.
- When `EN = 0`, outputs go to the off state, overriding any active mode.

This confirms that the SystemVerilog implementation matches the behavior specified in the PRD.