**ELX304 – Electronic Systems**

**Individual Coursework Assignment**

**Digital Design**

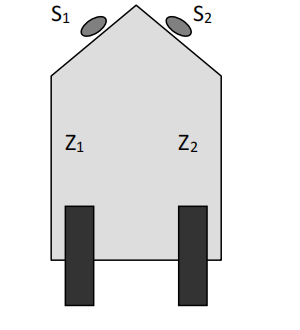


**WERE VINCENT**

1. **Task 1: Synchronous Design Problem**

* **Operational Limitations**

***Analysis of the given control system***



The control system depends on proximity sensors (S1, S2) and outputs (Z1, Z2) to control the robot's motors.

The potential limitations includes;

* *Lack of memory****-*** The absence of memory mechanism to store previous states, making the control system react only to the current inputs, which may cause instability if the sensor data fluctuates quickly.
* *Ambiguity in Inputs-* If the sensors provide inconsistent signals (S1=0, S2=1), how does the system react?
* *Power Consumption-* Continuous sensing and motor operations might lead to higher power usage, limiting battery life.
* *Scalability-* The system might not easily scale if more sensors or more complex movements are added.
* *Sensor Sensitivity-* The system might not handle small obstacles well due to limitations in sensor range or accuracy.
* **Design a minimal synchronous solution using D-type flip-flops.**

***Steps to approach***

* *State Diagram-* Created a state diagram for the robot's movement based on sensor inputs S1 and S2, and motor outputs Z1 and Z2.
* *State Table-* Translated the state diagram into a state transition table. The table should map (S1, S2) to the corresponding outputs (Z1, Z2) and the next states.

The state table lists the present state, the sensor inputs (S1,S2), and the corresponding motor outputs (Z1,Z2).It also shows the next state based on the current inputs as shown below.



* *Flip-flop Selection-* Used D-type flip-flops to store the states of the system. The inputs to the flip-flops was determined from the state transition table.
* *Minimized Logic-* Simplify the logic using Karnaugh Maps (K-maps) to minimize the number of logic gates required for the system as shown below. To minimize the logic for K1and K2, K-maps was based on the state table.

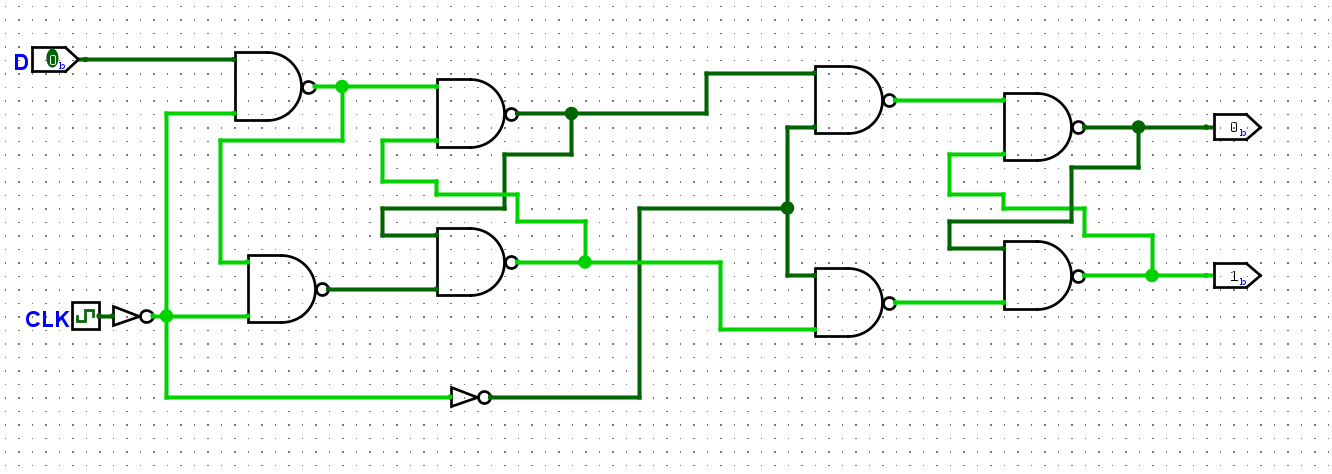
1. *K-map for K1*



1. *K-map for K2*



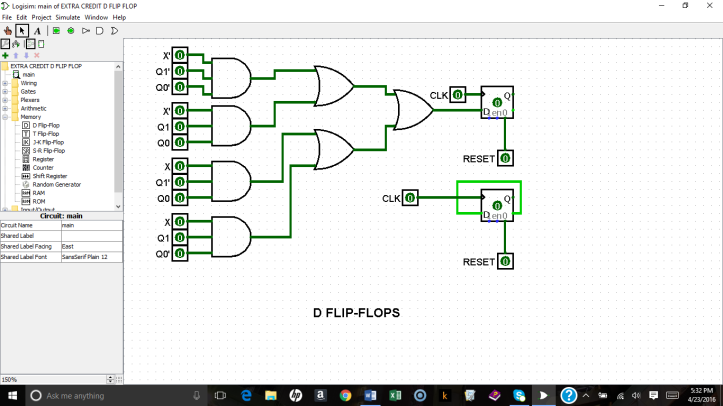
* *Circuit Design-* Drew the final circuit diagram including D flip-flops, logic gates, and inputs/outputs.



***Here's a brief overview of how the circuit work***

* Use the proximity sensors (S1, S2), U1A,U1B,U2A,U2B,U3A,as inputs to drive the state machine, with outputs Z1 and Z2 ,clock controlling the robot's motors when d is Hi and Low.
* D flip-flops will hold the state information, and the logic gates will ensure the correct transitions based on the sensor inputs.
* **Produce a simulation of the system using suitable software.**

1. ***Simulation-*** Logisim simulation software was used to simulate the design.
2. Input the designed circuit into the simulation tool, apply inputs (S1, S2), and observe the output (Z1, Z2).
3. Lastly verified that the outputs behave as expected in response to different sensor inputs.



1. **Task 2 : Asynchronous Design Problem**

***Understanding the problem***

The system should rotate left or right based on the sensor (Z) detecting an obstacle. If Z=0 (no obstacle), the robot continues rotating in the current direction. After losing the signal (Z=0), the system alternates between left and right rotations.

In this case, we have a single proximity sensor Z, and the system alternates between left and right rotations based on whether the sensor detects an obstacle (Z=1) or no obstacle (Z=0). The robot’s behavior is to rotate in a sequence: left-right-left, alternating directions based on the sensor signal. When an obstacle is detected (Z=1), the robot continues rotating in the same direction until the sensor signal is lost (Z=0), at which point it switches direction.

***Approach***

* *State Diagram*-Created a state diagram with two main states: Left Rotation and Right Rotation. The sensor (Z) determines the transitions between states i.e.

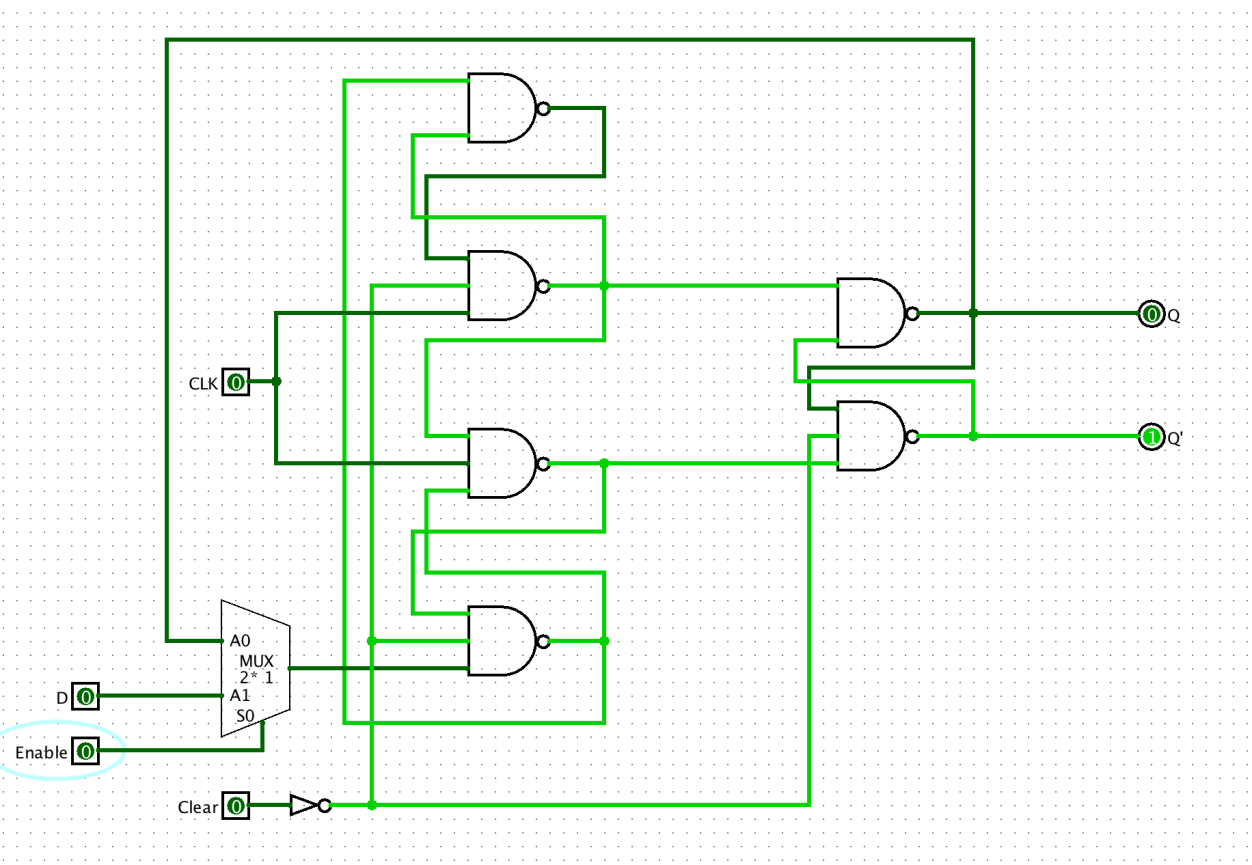
1. If Z = 1, the robot continues rotating in its current state (no change in direction).
2. If Z = 0, the robot switches to the opposite state (left to right or right to left).

* *State Table-*Develop a state table that maps Z=0 (no obstacle) to the next state, alternating between left and right.



In this case, we have a single proximity sensor Z, and the system alternates between left and right rotations based on whether the sensor detects an obstacle (Z=1) or no obstacle (Z=0)

* *Circuit Design-* Used flip-flops and logic gates to implement the state transitions. Since this is an asynchronous system, care was taken to avoid race conditions.



**How the designed Circuit Works**

* ***When Z = 1***

The inputs to the SR latch do not change, so the robot remains in the current state (either left or right).

* ***When Z = 0***
* If the robot is currently rotating right (R state), the Set input of the SR latch will be triggered, switching the robot to the left (L state).
* If the robot is currently rotating left (L state), the Reset input of the SR latch will be triggered, switching the robot to the right (R state).
* *Hazard Elimination-* Used techniques like covering hazards in the logic design to avoid race conditions, ensuring a stable circuit behavior when switching between states.
* *Race-free Design-* Ensured that the changes in the proximity sensor input (Z) do not cause unstable behavior. For asynchronous designs, use techniques like inserting delays (using gates or flip-flops) to prevent state races.
* In order to avoid race conditions in the design, stability of the transition logic was ensured. This was done using de-bouncing circuits or ensuring that there was a clear delay between changing the input (Z) and the state switching, preventing any unwanted glitches.
* The use of an SR latch helped to eliminate hazards and races, as the latch only changed state when there is a clear change in the input (Z = 0).