Designing a Finite State Machine for Robot

Abstract:

This report outlines the process of designing a Finite State Machine (FSM) for controlling the behavior of a robot. The FSM methodology is explored, and its application in robotics is discussed, highlighting its significance in enhancing robot autonomy and task performance.

Introduction:

In the field of robotics, controlling the behavior of robots is crucial for accomplishing various tasks efficiently and autonomously. One approach to achieve this is by utilizing Finite State Machines (FSM), which model the behavior of the robot as a set of distinct states and transitions between them. This report discusses the methodology of designing an FSM for robot behavior control, exploring its applications, challenges, and benefits in robotics.

Verilog HDL:

- Verilog: Verilog is a hardware description language (HDL) used in digital circuit and system design. It allows engineers to model, simulate, and synthesize digital systems. Verilog is widely used in the design and verification of digital integrated circuits, field-programmable gate arrays (FPGAs), and other electronic systems, Some key concepts in Verilog:
    - **Modules**: Verilog designs are typically composed of modules, which represent different functional units of the design.
    - **Behavioral Modeling**: Verilog supports behavioral modeling, where the functionality of a module is described in terms of its behavior or operation, which’s used for describing our FSM module, some number of states and every clock cycle the states changes based on the inputs and the current state.
    - **Structural Modeling**: Verilog also supports structural modeling, where the module's functionality is described in terms of interconnected components.
    - **Data Types**: Verilog provides various data types for representing different types of signals and variables, such as wire, reg, integer, and real.
    - **Simulation**: Verilog designs can be simulated using simulators to verify their correctness and functionality before synthesis.

Digital Design Concepts [ECE 230]:

Boolean Algebra: The mathematical foundation for digital logic design, including basic operations such as AND, OR, and NOT, as well as more complex operations like XOR and NAND.

Combinational Logic: The design of digital circuits where the output depends only on the current inputs, with no memory elements involved.

Sequential Logic: The design of digital circuits where the output depends on both the current inputs and the past history of inputs, typically involving flip-flops or latches, the sequential logic is used to describe our FSM.

Finite State Machines (FSMs): A theoretical and practical framework for modeling and designing sequential logic circuits, where the system's behavior is defined by a finite number of states and transitions between them.

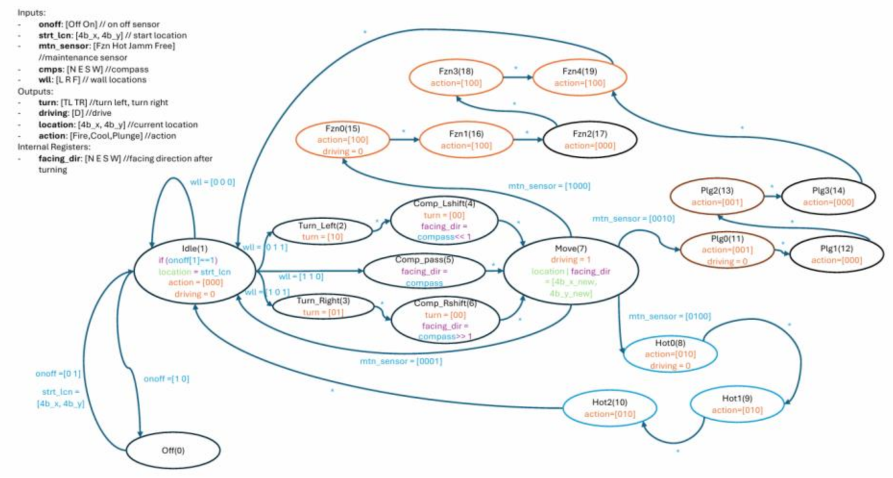
Digital Circuit Design: Techniques for designing and implementing digital circuits using basic logic gates, multiplexers, decoders, encoders, and other components.

Thesis Sentence:

This report aims to demonstrate the systematic process of designing an FSM for a robot, highlighting its effectiveness in enabling autonomous decision-making and task execution.

Methods:

The design process of the FSM for robot behavior control involves several key steps:


Inputs:
- onoff: [Off On] // on off sensor
- strt_lcn: [4b_x, 4b_y] // start location
- mtn_sensor: [Fzn Hot Jamm Free] //maintenance sensor
- cmps: [N E S W] //compass
- wll: [L R F] // wall locations
Outputs:
- turn: [TL TR] //turn left, turn right
- driving: [D] //drive
- location: [4b_x, 4b_y] //current location
- action: [Fire,Cool,Plunge] //action
Internal Registers:
- facing_dir: [N E S W] //facing direction after turning
Fzn3(18) action=[100]
Fzn4(19) action=[100]
Fzn0(15) action=[100] driving = 0
Fzn1(16) action=[100]
Fzn2(17) action=[000]
wll = [0 0 0]
mtn_sensor = [1000]
Plg2(13) action=[001]
Plg3(14) action=[000]
Comp_Lshift(4) turn = [00] facing_dir = compass<< 1
Turn_Left(2) turn = [10]
mtn_sensor = [0010]
Plg0(11) action=[001] driving = 0
Plg1(12) action=[000]
Idle(1) if (onoff[1]==1) location = strt_lcn action = [000] driving = 0
wll = [0 1 1]
wll = [1 1 0]
wll = [1 0 1]
Comp_pass(5) facing_dir = compass
Move(7) driving = 1 location | facing_dir = [4b_x_new, 4b_y_new]
Turn_Right(3) turn = [01]
Comp_Rshift(6) turn = [00] facing_dir = compass>> 1
mtn_sensor = [0100]
Hot0(8) action=[010] driving = 0
mtn_sensor = [0001]
onoff =[0 1]
strt_lcn = [4b_x, 4b_y]
onoff =[1 0]
Hot2(10) action=[010]
Hot1(9) action=[010]
Off(0)

State Definition: Identify the distinct states that encapsulate the robot's behavior, considering factors such as task requirements, environmental conditions, and sensor inputs.

Transition Specification: Define the transitions between states and the conditions or events that trigger them, ensuring logical flow and efficient task execution.

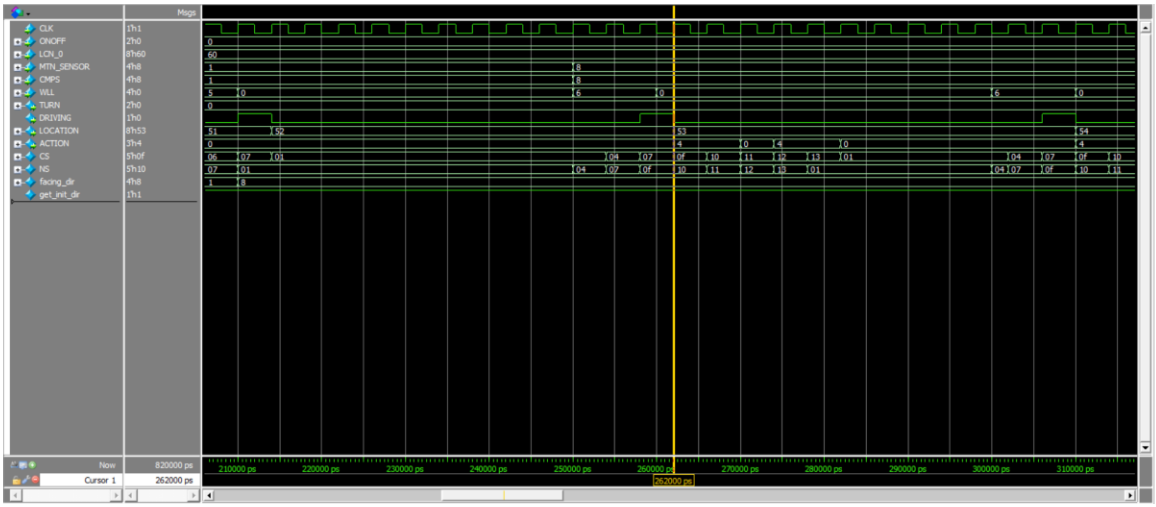
Conflict Resolution: Address any potential conflicts or concurrent states within the FSM, implementing mechanisms to prioritize actions and maintain coherence.

Implementation: Translate the designed FSM into executable code, integrating it into the robot's control system using suitable programming languages or tools.

Results:

The designed FSM was successfully implemented in a simulated environment using ModelSim, and the provided testbench is used the verify the functionality of the FSM.

Also, another testbench of another map for extra verification purpose.



Discussion:

The application of FSM in robot behavior control offers several advantages, including modularity, scalability, and ease of implementation. By organizing the robot's behavior into distinct states and transitions, FSM enables autonomous decision-making and facilitates task execution in complex environments. However, challenges such as state explosion and transition complexity must be addressed through careful design and optimization techniques.

Conclusion:

In conclusion, the systematic design and simulation of an FSM for robot behavior control provides a structured approach to enhance robot autonomy and task performance. By defining states, transitions, and conditions, FSM enables efficient decision-making and execution, paving the way for the development of more capable and adaptive robotic system.

References:

[1] FPGA Prototyping by Verilog Examples.

[2] Digital design 5th ed by moris mano.