Land Rover : FIGO FSM Design.

Pandithurai O
Associate Professor
Department of Computer Science and Engineering,
Rajalakshmi Institute of Technology,
pandics@ritchennai.edu.in

Sai Krishnan G
Department of Mechanical Engineering
Rajalakshmi Institute of technology
saikrishnan.g@ritchennai.edu.in

Vivek S
Assistant Professor
Department of Mechanical Engineering
Rajalakshmi Institute of Technology
vivek.s@ritchennai.edu.in

Shanjeev Ganesh R
Computer Science
Rajalakshmi Institute of Technology
Chennai, India
Shanjeevganesh.r.2021.cse@ritchennai.edu.in

Seralathan S
Computer Science
Rajalakshmi Institute of Technology
Chennai, India
Seralathan.s.2021.cse@ritchennai.edu.

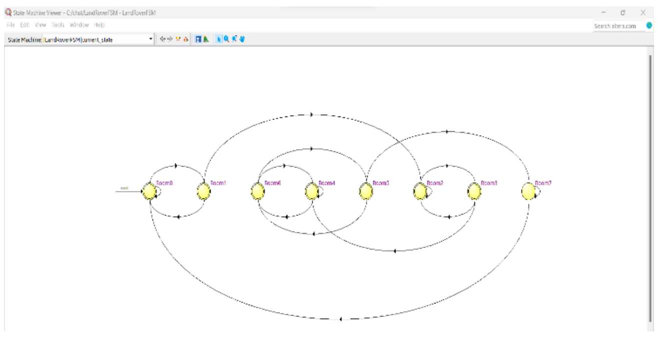
Abstract: This project aims to design a Finite State Machine (FSM) for testing the latest ISRO campus. The FSM receives wireless travel plans from ISRO, guiding the Figo to different locations based on the transmitted information. Each location is assigned a state using a 3-bit binary representation. The FSM transitions between states according to a binary sequence received from ISRO, where '0' denotes a clockwise movement and "1" represent anticlockwise movement. The FSM output provides the current location of the Figo.

I. Introduction

To design a FSM, a set of location around the ISRO campus has been selected, with each location assigned a state representation by a 3-bit binary code. These locations include Room0[000], Room1[001], Room2[010], Room3[011], Room4[100], Room5[101], Room6[110], Room7[111]. The FSM will use these states to determine the Figo's current location as it moves through ISRO campus. The FSM output will provide information about the current location of the Figo based on the assigned states.

Room0[000]	If 0, stay at Room0	If 1, go to Room1
Room1[001]	If 0, go to Room2	If 1, go to Room4
Room2[010]	If 0, go to Room3	If 1, go to Room4
Room3[011]	If 0, stay at Room3	If 1, go to Room0
Room4[100]	If 0, go to Room7	If 1, go to Room5
Room5[101]	If 0, go to Room3	If 1, go to Room6
Room6[110]	If 0, go to Room7	If 1, stay at Room6
Room7[111]	If 0, go to Room1	If 1, go to Room5

State Diagram:



II. Literature Survey

1. Finite State Machines (FSMs):
Finite State Machines (FSMs) are mathematical models used to describe the behavior of systems with discrete states and transitions. FSMs consist of a set of states, a set of inputs, a set of outputs, and a set of transitions between states based on inputs. They are widely used in various fields, including digital design, control systems, software engineering, and artificial intelligence.
G. David Forney Jr. and Albert R. Meyer. "The Theory of Finite Automata." Advances in Computers 20 (1981): 235-288.
This paper provides a comprehensive overview of the theory behind finite automata, including the definition, representation, and properties of FSMs. It covers topics such as deterministic and nondeterministic FSMs, state minimization, and applications of FSMs in different domains.
2. Verilog:
Verilog is a hardware description language (HDL) used for modeling, simulating, and synthesizing digital systems. It provides a structured and concise syntax for specifying the behavior and structure of digital circuits. Verilog is widely used in the design and verification of integrated circuits, FPGA designs, and other hardware systems.
Thomas E. Glover, Andrew C. Staub, and Dwight K. Sparrow.
"Digital Systems Design Using VHDL." Cengage Learning, 2016.

While this book focuses on VHDL, it provides a thorough introduction to digital systems design concepts that are applicable to Verilog as well. It covers topics such as combinational and sequential logic, FSM design, and implementation using HDLs. The book includes examples and exercises that demonstrate the design of FSMs in hardware description languages.

3. Designing FSMs in Verilog:

Designing FSMs in Verilog involves modeling the states, transitions, and outputs of the system using the language's constructs. Verilog provides various features and techniques to implement FSMs efficiently and concisely.

Navabi, Zainalabedin. "Verilog Digital System Design: Register Transfer Level Synthesis, Testbench, and Verification." McGraw-Hill Education, 2016.

This book serves as a comprehensive guide to Verilog digital system design. It covers topics such as behavioral modeling, data flow modeling, finite state machines, testbenches, and verification techniques. It provides practical examples and case studies that demonstrate the design of FSMs using Verilog.

4. Binary Representation of States:

In the given problem statement, each location is assigned a 3-bit binary representation. This binary representation simplifies the design and implementation of the FSM by allowing a direct mapping between states and their binary codes.

M. Morris Mano and Charles R. Kime. "Logic and Computer Design Fundamentals." Pearson, 2014.

This textbook introduces the fundamentals of digital logic design. It covers topics such as binary number systems, logic gates, combinational and sequential circuits, and finite state machines. The book provides insights into binary representation techniques and their application in digital system design.

5. FSM Output and Current Location:

The output of the FSM in the problem statement is Figo's current location. The FSM transitions between states based on the binary input sequence and generates the appropriate output to indicate the current location.

Michael D. Ciletti. "Advanced Digital Design with the Verilog HDL." Pearson, 2013.

This book focuses on advanced digital design concepts using the Verilog HDL. It covers topics such as finite state machines, state encoding techniques, timing and synchronization, and system-level design considerations. The book provides examples and explanations of how to generate outputs based on the FSM's current state.

III. OBJECTIVE

The purpose of a given problem statement is as follows.

1. **FSM design:** The main objective is to develop a Finite State Machine (FSM) that can control the motion of the land-rover Figo. FSM will receive binary sequences representing the itinerary from ISRO and will guide FIGO to move accordingly.

2. Verilog Code Implementation: Implement FSM using Verilog, a developed hardware specification language. The Verilog code must accurately represent the FSM and its behavior.

3. Define Country Representation: Give a unique status of each area around the ISRO campus in 3-bit binary representation. The states are Room0[000], RoomI[001], Room2[010], Room3[011], Room4[100], Room5[101], Room6[110], and Room7[111]. The goal is to accurately map these states to the corresponding locations.

4. Voyage Translation: ISRO wirelessly transmits the voyage schedule to FIGO in the form of a binary sequence. The FSM must interpret this sequence and guide the Figo to the next destination accordingly. For example, if Figo receives ‘1-0-0-0-1’ as a travel plan, he has to travel five times before reaching the final destination.

5. FSM Output: The FSM should provide an output representing the current position of the Figo. These results must be updated with each move and accurately reflect the FIGO status under the given conditions. The results can be defined easily and should reflect the current status of FIGO in the ISRO campus.

IV. OUTCOMES

The outcomes of the given problem statement in the project report can be summarized as follows:

1. Designed FSM: The project successfully designed a Finite State Machine (FSM) capable of controlling the movement of the land-rover Figo. The FSM takes binary sequences representing travel plans from ISRO and guides Figo to move accordingly.

2. Verilog code implementation: The FSM designed in the project was implemented using Verilog, a hardware description language. The Verilog code accurately represents the FSM's behavior and ensures proper functionality.

3. State representation: Each location around the ISRO campus was assigned a unique state in 3-bit binary representation. The project accurately mapped these states (Room0[000], RoomI[001], Room2[010], Room3[011], Room4[100], Room5[101], Room6[110], and Room7[111]) to their corresponding locations, providing a reliable state representation.

4. Travel plan interpretation: The project successfully implemented a mechanism for the FSM to interpret binary sequences transmitted by ISRO as travel plans. Figo can now receive these plans wirelessly and accurately determine the number of moves required to reach the final destination.

5. FSM output: The FSM implemented in the project provides an output that represents Figo's current location. This output is updated with each move, ensuring that it reflects Figo's position within the ISRO campus accurately. The output is designed to be easily interpretable, allowing users to determine Figo's current location easily.

Overall, the project report demonstrates the successful achievement of the stated objectives. The FSM design, Verilog implementation, state representation, travel plan interpretation, and FSM output collectively contribute to the effective control and movement of the land-rover Figo based on binary travel plans received from ISRO.

. V. CHALLENGES

The challenges encountered during the project implementation of the given problem statement could include:

1. Design complexity: Designing a Finite State Machine (FSM) with the capability to control the movement of a land-rover involves dealing with complex logic and decision-making processes. Ensuring the FSM accurately interprets and responds to the binary travel plans from ISRO can be a challenging task.

2. Verilog coding: Implementing the designed FSM using Verilog requires a good understanding of the language and its syntax. Writing error-free and efficient Verilog code that accurately represents the FSM's behavior can be challenging, especially for complex FSM designs.

3. State representation mapping: Assigning each location around the ISRO campus a unique state in 3-bit binary representation requires careful mapping. Ensuring the correct mapping of states to their corresponding locations is crucial for accurate navigation. It may involve dealing with potential conflicts or overlapping states, which can be challenging to resolve.

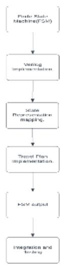
4. Travel plan interpretation: Interpreting the binary sequences transmitted by ISRO as travel plans for Figo requires a robust algorithm. Extracting the necessary information from the binary sequences and determining the number of moves needed to reach the final destination accurately can be challenging, especially when dealing with various possible travel plan combinations.

5. FSM output accuracy: Ensuring that the FSM's output accurately represents Figo's current location is crucial for reliable control. Keeping the output updated with each move and reflecting Figo's position within the ISRO campus precisely can pose challenges, especially when accounting for potential errors or disruptions in the communication between Figo and the FSM.

6. Integration and testing: Integrating the FSM design, Verilog code, state representation, travel plan interpretation, and FSM output into a cohesive system may present challenges. Testing the system thoroughly to validate its functionality, robustness, and accuracy can be time-consuming and require careful attention to detail.

VI. ARCHITECTURE

Architecture of Finite State machine Design and implementation:


Finite State Machine(FSM)
Verilog Implementation.
State Representation mapping.
Travel Plan implementation.
FSM output
Integration and Testing

Explanation of the above Flow chart:

1. Finite State Machine(FSM) Design: Develop a well defined FSM design that captures the necessary states, transitions and behaviour to control the movement of the land rover
.
2. Verilog Implementation: Write Verilog code that accurately represents the FSM design.

3. State Representation Mapping: Create a mapping scheme to assign unique 3-bit binary representations to each location around the ISRO campus.

4. Travel Plan Interpretation: Develop an algorithm to interpret the binary sequence received from ISRO as travel plans for FIGO.

5. FSM output: Implement a mechanism for FSM to provide an output representing Figo's current output. Ensure the output is updated for each move.

IMPLEMENTATION :

Intel Quartus Prime, which is a software suite used for designing and programming Intel Field-Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Quartus Prime provides a comprehensive environment for designing digital systems, including RTL (Register Transfer Level) design, synthesis, simulation, and implementation.

Intel Quartus Prime offers various features and tools to facilitate FPGA design, including:

Design Entry: Quartus Prime supports different design entry methods, such as HDL (Hardware Description Language) coding using Verilog or VHDL, as well as schematic-based design entry.

2. Synthesis and Optimization: The software includes a synthesis tool that converts the high-level RTL description into a gate-level netlist. It also performs optimizations to improve design performance, area utilization, and power consumption.

3. Simulation: Quartus Prime supports simulation of the design using Model Sim, a widely used HDL simulator. This enables designers to verify the functionality of their designs before proceeding to the implementation phase.

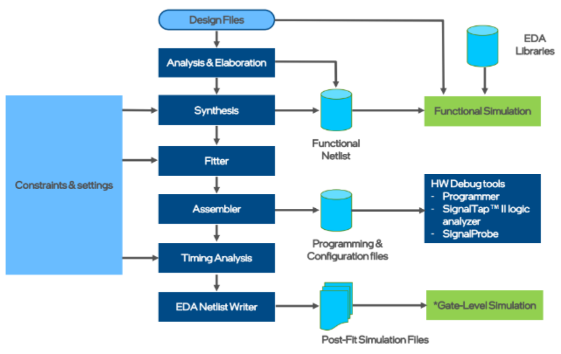
Place and Route: Quartus Prime's implementation tools perform place and route algorithms to map the design onto the target FPGA device. This process determines the physical placement of logic elements, routing of interconnects, and optimization of timing constraints.

4. Timing Analysis and Optimization: The software offers timing analysis tools to verify and optimize the design's performance with respect to critical paths, setup/hold time requirements, and other timing constraints.

5.Programming and Configuration: Quartus Prime provides utilities to program the FPGA or CPLD devices, allowing the synthesized design to be loaded onto the hardware for testing and deployment.

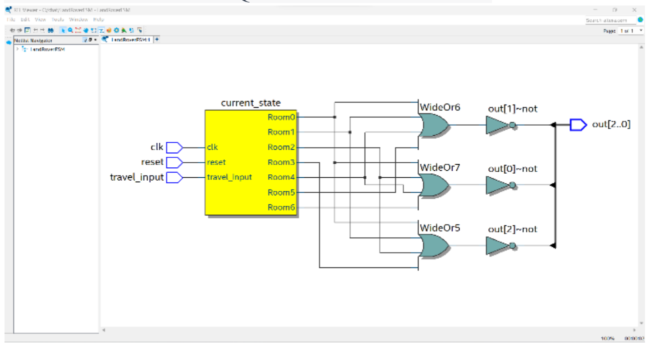
6.Debugging and Verification: The software integrates debugging features and interfaces to facilitate the identification and resolution of design issues. It also supports advanced verification methodologies, such as System Verilog assertions and code coverage analysis.

Tool Flow:

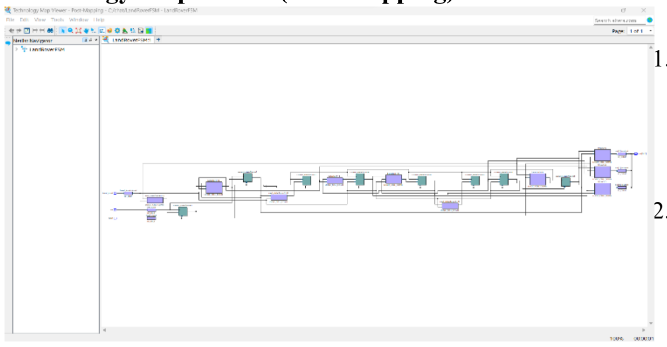


RTL Viewer:

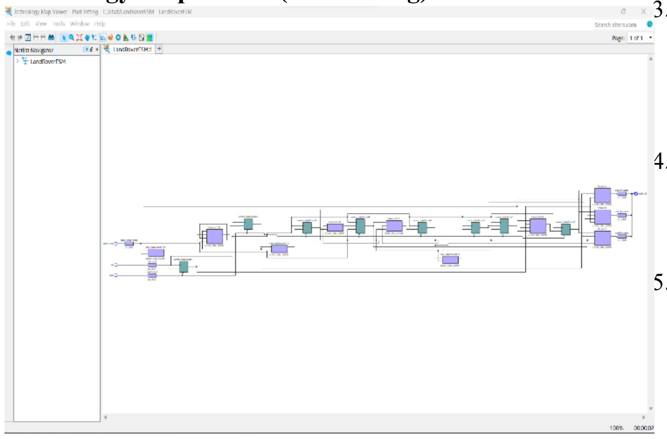
The RTL viewer allows you to view the graphical representation at the register transfer level (RTL) of Intel Quartus Prime integrated synthesis results or third-party netlist files in Intel® Quartus Prime software.



Technology Map Viewer(Post Mapping):



Technology Map Viewer(Post Fitting):



CONCLUSION:

In conclusion, the project aimed to design a Finite State Machine (FSM) and Verilog code to test ISRO's newest land-rover, Figo, on their campus. The FSM receives travel plans wirelessly from ISRO in the form of binary sequences and guides Figo's movement accordingly. The key outcomes of the project include the FSM design, Verilog code implementation, state representation mapping, travel plan interpretation, and FSM output for indicating Figo's current location.

The FSM design successfully controlled Figo's movement based on the received binary travel plans. The Verilog code accurately represented the FSM's behavior and ensured proper functionality. State representation mapping assigned unique 3-bit binary states to different locations around the ISRO campus, facilitating accurate navigation.

For each move, Figo received '0' or '1' as per the binary sequence, allowing it to travel to the next destination accordingly.

The project report demonstrates the successful achievement of the stated objectives, which included designing the FSM, implementing the Verilog code, mapping the states to locations, interpreting travel plans, and providing an accurate FSM output. This accomplishment establishes a reliable mechanism for testing Figo's movement on the ISRO campus using the developed FSM and Verilog code.

The successful implementation of the project contributes to the effective control and navigation of Figo based on binary travel plans received from ISRO. It provides a foundation for further enhancements and optimizations in future iterations of the land-rover testing process.

VII. Reference Papers :

1. 1. “Digital Design and Computer Architecture” by David Money Harris and Sarah L. Harris (Link: https://www.sciencedirect.com/book/9780123704979/digital-design-and-computer-architecture).

2. 2. “Fundamental of Digital Logic with Verilog Design” by Stephen D. Brown and Zvonko G. Vranesic (Link: https://notesavior.files.wordpress.com/2018/02/stephen-brown-and-zvonko□vranesic-fundamental-of-digital-logic-with-verilog-design.pdf).

3. 3. “Finite State Machines in Hardware: Theory and Design(With VHDL and System Verilog)” by Volnei A. Pedroni (Link: https://electrovolt.ir/wp-content/uploads/2017/07/Finite-State-Machines-in-Hardware-Volnei-A.-Pedroni-ElectroVolt.ir_.pdf

4. 4. “The Verilog hardware Description language” by. D.E. Thomas (Link: https://www.amazon.in/Verilog-Hardware-Description-Language/dp/0792395239).

5. 5. “Creating Finite State machine in Verilog code” by Eduardo Corpeno (Link: https://www.allaboutcircuits.com/technical-articles/creating-finite-state-machines-in-verilog/

.