Project 91:Mealy Machine for Bit Parity Checker

A Comprehensive Study of Advanced Digital Circuits

By: Abhishek Sharma , Ayush Jain , Gati Goyal, Nikunj Agrawal

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Project Overview

The Mealy Machine for Bit Parity Checker checks the parity of a bit stream, generating an output based on whether the count of 1s is even or odd. It has two states: S0 (Even Parity) and S1 (Odd Parity), with transitions based on input bits (0 or 1). The output, unlike in a Moore machine, depends on both the current state and input bit, providing an immediate response. Implemented using Verilog/VHDL, the design includes a state diagram, state table, and testbench. This project has applications in error detection, communication systems, and embedded systems.

2 Mealy Machine for Bit Parity Checker

2.1 Key Components of Mealy Machine for Bit Parity Checker

A Mealy Machine is a finite-state machine where the outputs depend on both the current state and the input. In the case of a Bit Parity Checker, the machine monitors a serial bit stream and generates a parity bit based on whether the total number of 1s encountered in the stream is even or odd. Below are the key components involved in the design of a Mealy machine for bit parity checking.

States

- **S0 (Even Parity State):** This state indicates that the cumulative number of 1s encountered so far is even.
- **S1 (Odd Parity State):** This state indicates that the cumulative number of 1s encountered so far is odd.
- These two states are sufficient to track whether the parity is even or odd. The state transitions are controlled by the incoming bits, which cause the machine to either stay in the current state or transition to the other state.

Inputs

- The machine accepts a serial bit stream as input, which consists of binary digits (0s and 1s).
- The input at each clock cycle will trigger the state transition logic.

Outputs

- The output, which is the parity bit, is determined by both the current state and the input bit, which is a defining feature of the Mealy machine.
- If the machine is in S0, the output will be 0 (indicating even parity).
- If the machine is in S1, the output will be 1 (indicating odd parity).
- This immediate output generation based on both the state and input distinguishes the Mealy machine from the Moore machine, where outputs are solely based on the state.

State Transition Logic

- The state transitions are governed by the following logic:
- ***From S0:***
  1. Input = 0 → Stay in S0 (parity remains even).
  2. Input = 1 → Transition to S1 (parity becomes odd).
- **From S1:**
  1. Input = 0 → Stay in S1 (parity remains odd).
  2. Input = 1 → Transition to S0 (parity becomes even).

- The state transitions depend on both the current state and the incoming input bit, ensuring the output is generated immediately based on these two factors.

Clock and Control Signals

- The machine operates synchronously, with transitions occurring on each clock cycle. The clock signal triggers the state transitions and ensures that the output is updated correctly with every input bit.

Flip-Flops for State Storage

- Flip-flops (usually D-type flip-flops) are used to store the current state of the machine. The flip-flops hold the state until the next clock cycle, and their outputs feed into the transition logic to determine the next state.

Combinational Logic for Transitions and Outputs

- Combinational logic calculates the next state and the output based on the current state and input. The next state is derived from the current state and input, and the output is determined based on the current state and input bit, according to the transition logic table.

2.2 Working of Mealy Machine for Bit Parity Checker

The Mealy Machine for a Bit Parity Checker operates by monitoring a stream of input bits and determining whether the number of 1s in the stream is even or odd. The machine uses two states: S0 (Even Parity) and S1 (Odd Parity). It transitions between these states based on the input bits, updating the parity each time a new bit is processed. The key feature of a Mealy machine is that the output is determined by both the current state and the input bit, providing a responsive output immediately after each input is received.

The state transitions work as follows: when in S0 (Even Parity), if the input bit is 0, the machine remains in S0 (parity remains even). If the input bit is 1, it transitions to S1 (Odd Parity). Conversely, in S1 (Odd Parity), a 0 keeps the machine in S1, while a 1 will transition it back to S0. The output at each state is determined by both the state and the input: S0 produces an output of 0 (even parity), and S1 produces an output of 1 (odd parity).

This design allows for continuous tracking of the bit stream, updating the parity bit with each new input. For example, given an input stream like 101010, the machine alternates between the states, outputting 1 1 0 0 1 1, reflecting the changing parity after each bit. The Mealy machine's ability to immediately generate output based on both the current state and input makes it an efficient and responsive solution for error detection in communication systems.

2.3 RTL Code

Listing 1: Mealy Machine for Bit Parity Checker

```

module mealy_bit_parity_checker (
    input  logic clk, reset,
    input  logic bit_in,          // Serial input bit stream
    output logic parity_bit       // Output: 1 for even parity, 0 otherwise
);

    // Define states
    typedef enum logic {EVEN, ODD} state_t;
    state_t current_state, next_state;

    // State transition logic
    always_ff @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= EVEN; // Start with even parity
        else
            current_state <= next_state;
    end

    // Next state and output logic (Mealy output depends on state and input)
    always_comb begin
        next_state = current_state;
        case (current_state)
            EVEN: begin
                next_state = (bit_in) ? ODD : EVEN; // Flip to ODD on bit_in = 1
                parity_bit = 1;                      // Output 1 for even parity
            end
            ODD: begin
                next_state = (bit_in) ? EVEN : ODD; // Flip to EVEN on bit_in = 1
                parity_bit = 0;                      // Output 0 for odd parity
            end
        endcase
    end
endmodule
```

2.4 Testbench

Listing 2: Mealy Machine for Bit Parity Checker

```

module tb_mealy_bit_parity_checker();
    logic clk, reset;
    logic bit_in;
    logic parity_bit;

    mealy_bit_parity_checker uut (
        .clk(clk),
        .reset(reset),
        .bit_in(bit_in),
        .parity_bit(parity_bit)
    );

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10ns clock period
    end

    // Test scenario
    initial begin
        reset = 1; bit_in = 0; // Start with reset asserted
        #10 reset = 0;         // Deassert reset

        // Apply serial input bits
        #10 bit_in = 1;  // Transition to ODD
        #10 bit_in = 0;  // Stay in ODD
        #10 bit_in = 1;  // Transition to EVEN
        #10 bit_in = 1;  // Transition to ODD
        #10 bit_in = 0;  // Stay in ODD
        #10 bit_in = 1;  // Transition to EVEN

        #50 $stop; // Stop simulation
    end

    // Monitor outputs
    initial begin
        $monitor("Time: %0t | Bit In: %b | Parity Bit: %b | Current
            State: %s",
                 $time, bit_in, parity_bit, uut.current_state.name());
    end
endmodule
```

3 Results

3.1 Simulation

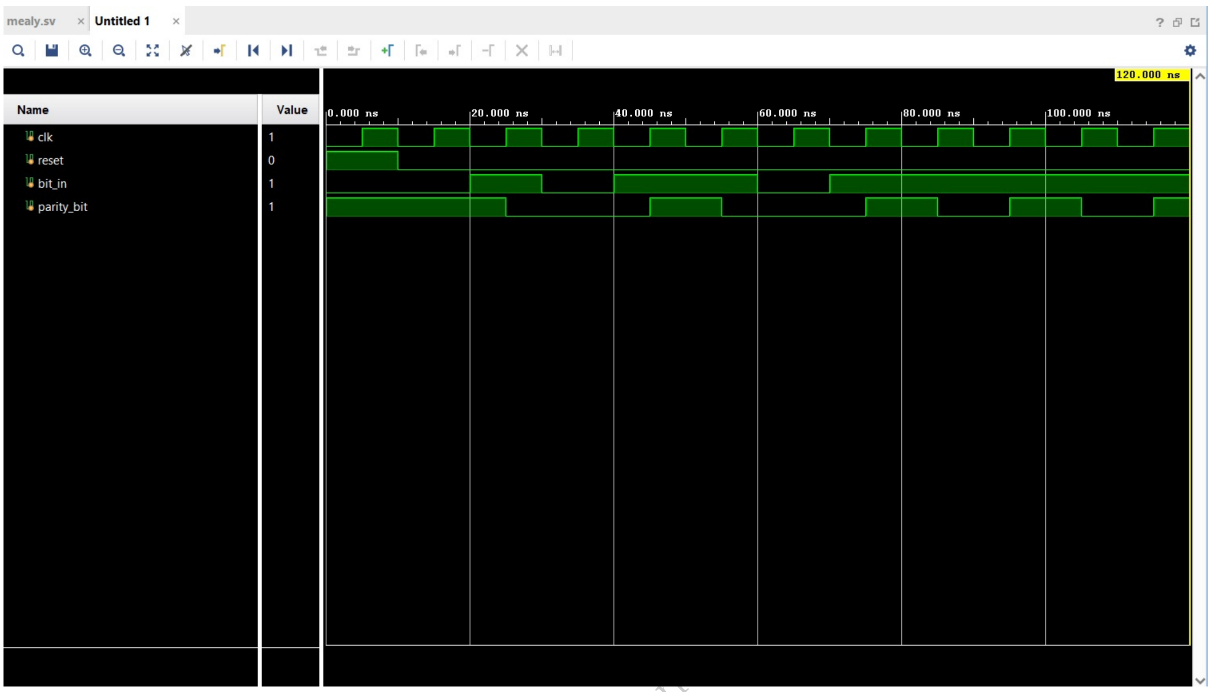

Figure 1: Simulation of Mealy Machine for Bit Parity Checker

3.2 Schematic

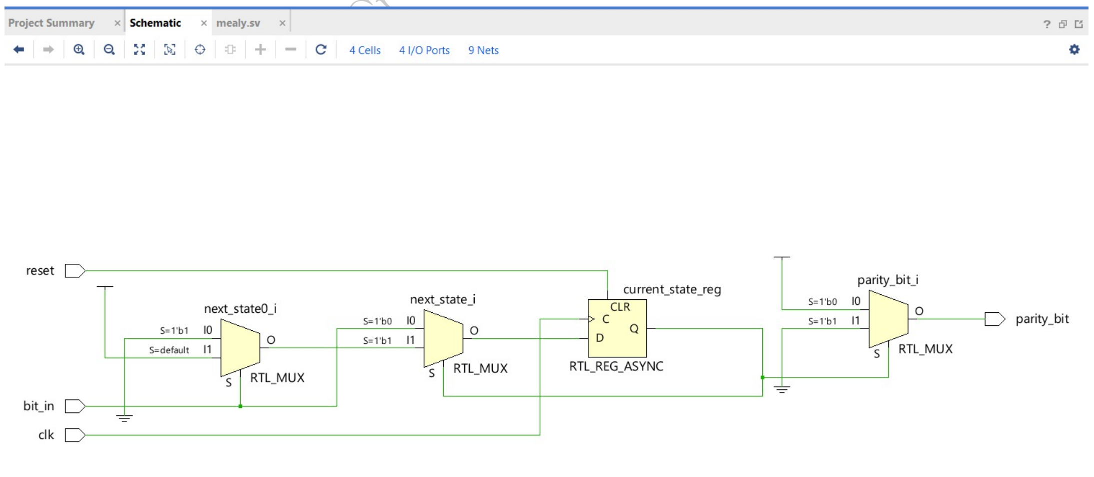

Figure 2: Schematic of Mealy Machine for Bit Parity Checker

3.3 Synthesis Design

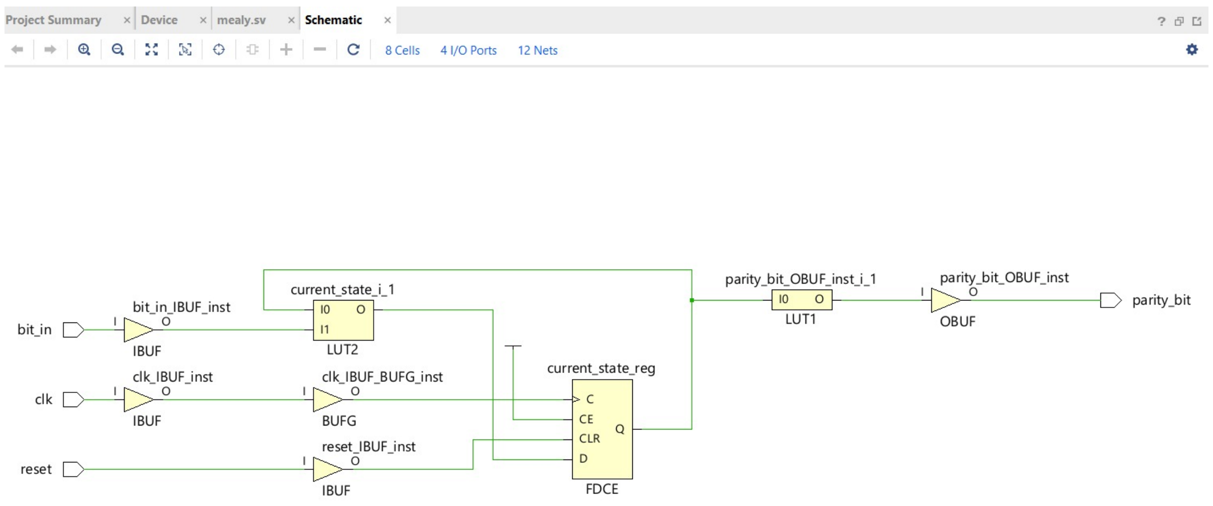

Figure 3: Synthesis Design of Mealy Machine for Bit Parity Checker

4 Advantages of Mealy Machine for Bit Parity Checker

- **Immediate Output:** The output is generated immediately based on both the current state and input, providing real-time feedback.
- **Efficiency:** Requires fewer states and transitions compared to other models like Moore machines, making it more efficient.
- **Responsive:** The design reacts quickly to input changes, ensuring fast parity checking.
- **Simple Design:** The state transition logic is straightforward, making it easy to implement in hardware using tools like Verilog or VHDL.
- **Compact Hardware:** Less complex than alternative designs, requiring fewer resources in hardware implementations, such as FPGAs.

5 Disadvantages of Mealy Machine for Bit Parity Checker

- **Complexity in Output Timing:** Since output depends on both the state and input, it can lead to timing issues or glitches in asynchronous designs.
- **Less Predictable Output:** The output changes immediately with each input, which may be less intuitive than the state-based output of a Moore machine.
- **Harder to Debug:** The combined dependence on state and input for outputs can make debugging more challenging, especially in large systems.
- **Potential for Noise Sensitivity:** Immediate output changes may cause the system to be more susceptible to noise or glitches in input signals.

6 Applications of Mealy Machine for Bit Parity Checker

Error Detection in Communication Systems:

- Used in data transmission protocols (e.g., UART, Ethernet) to detect transmission errors by generating parity bits to ensure data integrity.

Memory Systems:

- Helps in memory modules (e.g., DRAM) to detect and correct errors in data stored or retrieved, ensuring reliable operations.

Digital Circuit Design:

- Used in circuits where real-time parity checking is required, such as in communication interfaces and data bus systems.

Embedded Systems:

- Applied in resource-constrained embedded systems for lightweight error checking, ensuring reliable communication with minimal overhead.

Computer Architecture:

- Utilized in CPU design to detect errors in data paths, ensuring safe data transfers and processing.

Cryptography and Security Systems:

- Parity checkers can be used in secure data exchanges to verify the correctness of transmitted information and prevent data tampering.

7 Conclusion

The Mealy machine for bit parity checking efficiently generates a parity bit based on the input stream and its current state. Each bit causes an immediate transition and output, providing real-time parity checking. The output depends on both the current state and the input bit, making the system highly responsive. This mechanism is particularly useful for real-time error detection in communication systems.

8 FAQs

What is a Mealy machine?

- A Mealy machine is a type of finite-state machine where the output depends on both the current state and the input, unlike a Moore machine, where the output depends only on the current state.

What does a bit parity checker do?

- A bit parity checker determines if the number of 1s in a binary stream is even or odd. It generates a parity bit that reflects this count, which helps in error detection.

How does the Mealy machine work for bit parity checking?

- The Mealy machine keeps track of the parity (even or odd) of the input stream using two states: S0 for even parity and S1 for odd parity. The output is determined by both the current state and the incoming bit.

What are the advantages of a Mealy machine for this task?

- It offers immediate output, is efficient in design, and provides real-time response to input changes, making it suitable for real-time error detection in systems like communication protocols.

What are the disadvantages of using a Mealy machine for bit parity checking?

- The immediate output can lead to timing issues, making the system potentially more susceptible to glitches. It may also be harder to debug and less predictable than Moore machines.

Where is the Mealy machine for bit parity checker applied?

- It is widely used in communication systems, memory systems, digital circuit design, embedded systems, computer architecture, and cryptography to ensure data integrity and error detection.

What tools are used to implement the Mealy machine?

- The Mealy machine can be implemented using hardware description languages like Verilog or VHDL, and tools like ModelSim or Xilinx Vivado for simulation and synthesis.

How does the output of a Mealy machine differ from that of a Moore machine?

- In a Mealy machine, the output is generated immediately based on both the current state and input, while in a Moore machine, the output depends solely on the state.