Design of a Moore Machine Sequence Detector for ”11011” with Overlapping

Digital Electronics Laboratory

December 18, 2025

1 Objective

To design and implement a Moore finite state machine (FSM) sequence detector that detects the binary sequence **11011** with overlapping allowed. The detector should output ’1’ whenever the sequence is detected, even if the detected bits overlap with the next sequence.

2 Theory

2.1 Sequence Detection

A sequence detector is a digital circuit that scans an input stream of bits and produces an output when a specific pattern (sequence) is found. Sequence detectors are widely used in digital communication, data parsing, and protocol analysis.

2.2 Moore Machine vs. Mealy Machine

Moore Machine: The output depends only on the current state, not on the input. Outputs change synchronously with state transitions, making the design more predictable and less sensitive to glitches.

Mealy Machine: The output depends on both the current state and the current input. Mealy machines often require fewer states but can produce outputs that change asynchronously with the clock.

2.3 Overlapping Detection

With overlapping allowed, the detector can use bits from the end of one detected sequence as the start of the next. For example, in the input “11011011011”, the sequence “11011” appears multiple times, and overlapping allows detection at every valid occurrence.

2.4 Real-World Applications

- Data communication protocol analyzers
- Serial data stream pattern matching
- Error detection and correction
- Digital locks and security systems

3 State Assignment and Meaning

For the sequence **11011** (5 bits), we need 6 states (including the initial state):

State	Meaning
A	Initial state, no bits matched
B	Matched '1'
C	Matched '11'
D	Matched '110'
E	Matched '1101'
F	Matched '11011' (sequence found)

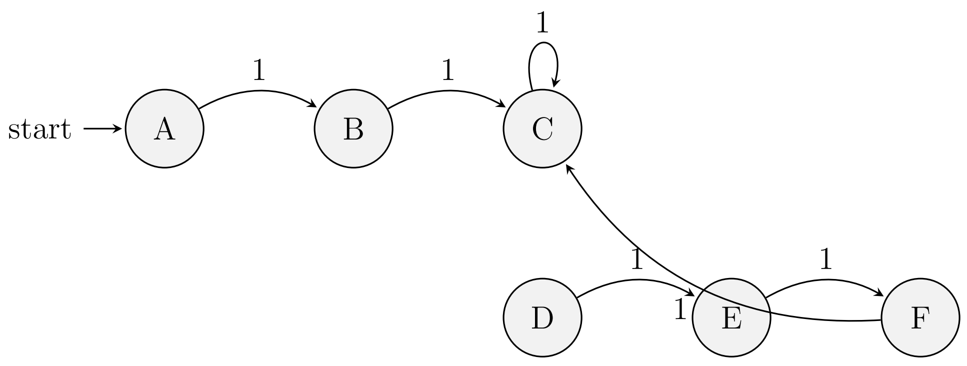
After detecting the full sequence (F), transition to the appropriate state to allow overlapping (in this case, C, since the last two bits "11" can start a new sequence).

4 Transition Table

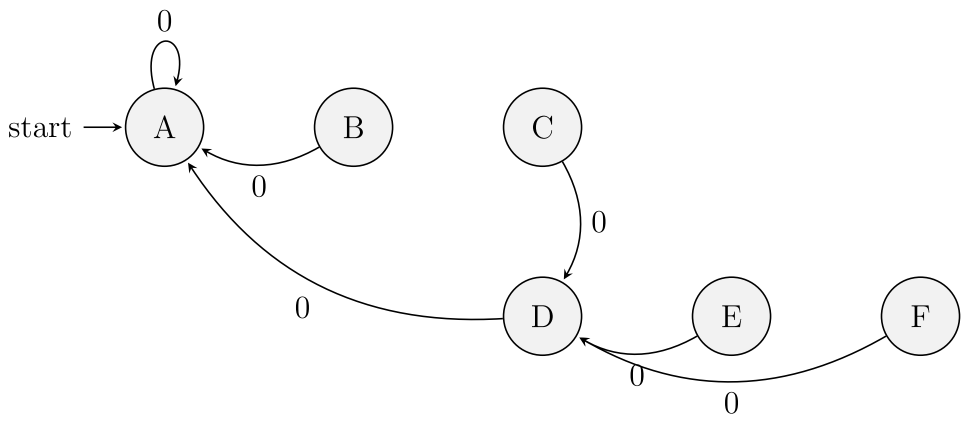
Current State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	A	0
B	1	C	0
C	0	D	0
C	1	C	0
D	0	A	0
D	1	E	0
E	0	D	0
E	1	F	0
F	0	D	1
F	1	C	1

5 State Diagrams

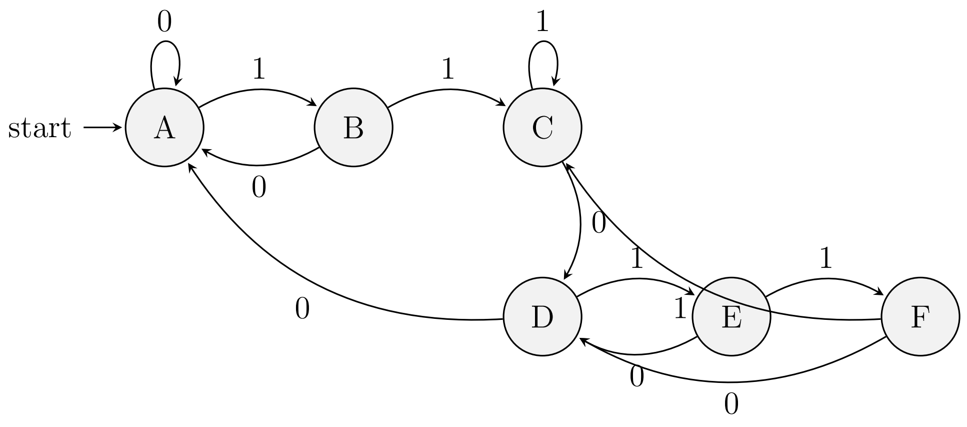
Transitions on Input = 1



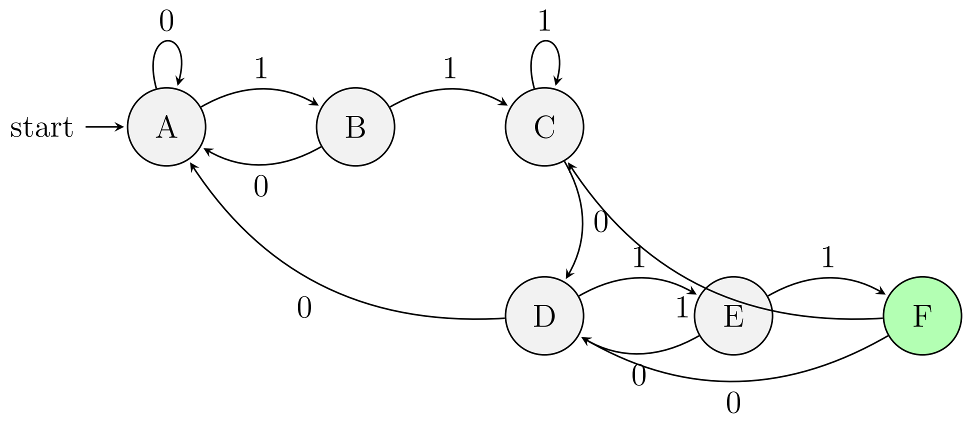
Transitions on Input = 0



Combined State Diagram

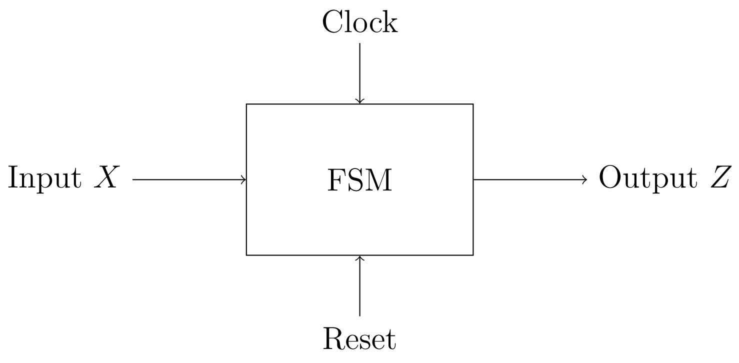


Output States Diagram (Output = 1)



6 FSM Block Diagram

A high-level block diagram of the sequence detector FSM is shown below. The FSM receives the serial input, clock, and reset signals, and produces the output signal.



7 Reset Functionality

The FSM includes a reset input that, when activated, returns the machine to the initial state (A), regardless of the current state. This ensures reliable operation after power-up or in case of errors. The reset can be implemented as either synchronous or asynchronous, depending on the design requirements.

8 State Encoding Table

State	Encoding ($Y_2Y_1Y_0$)
A	000
B	001
C	010
D	011
E	100
F	101

States 110 and 111 are unused.

9 Truth Table (Sample Sequence)

Suppose input: `1 1 0 1 1 0 1 1`

Step	Input	State	Output
0	-	A	0
1	1	B	0
2	1	C	0
3	0	D	0
4	1	E	0
5	1	F	1
6	0	D	0
7	1	E	0
8	1	F	1

Output is '1' at steps 5 and 8 (sequence detected twice, with overlap).

10 Testing and Verification

To verify the correct operation of the sequence detector, a series of input sequences—including those with multiple overlapping occurrences of "11011"—should be applied. The state transitions and output waveform can be observed using simulation tools. The output should go high exactly when the sequence is detected, including for overlapping instances.

A sample simulation waveform is shown below:

```
Time:   0 1 2 3 4 5 6 7 8 9 10
Input:  1 1 0 1 1 0 1 1 0 1 1
Output: 0 0 0 0 0 1 0 0 1 0 0
```

11 Design Procedure

1. Define the sequence and determine required states.
2. Assign lettered states (A-F) and describe each state's meaning.
3. Draw the state diagram for all transitions (inputs 0 and 1).
4. Create the transition table with current state, input, next state, and output.
5. Encode states in binary for hardware implementation.
6. Derive logic equations for next state and output using Karnaugh maps or Boolean algebra (optional).
7. Implement the FSM using flip-flops (JK, D, or T) and combinational logic.
8. Simulate with test inputs to verify correct sequence detection and overlapping behavior.

Example Boolean Equations (for D Flip-Flops)

$$\begin{aligned} D_2 &= X'Y_1 + XY_2Y_0' \\ D_1 &= XY_0 \\ D_0 &= X \\ Z &= Y_2Y_0 \end{aligned}$$

Where X is the input, $Y_2Y_1Y_0$ is the current state encoding, and Z is the output.

12 Hardware Implementation Notes

The FSM can be implemented using D flip-flops for state memory and combinational logic for next-state and output logic. Alternatively, the FSM can be described in a hardware description language (HDL) such as Verilog or VHDL and synthesized onto an FPGA or CPLD. Proper handling of unused state encodings should be ensured to prevent undefined behavior.

13 Simulation or Timing Diagram

A typical simulation waveform would show the input stream, the current state, and the output. For input "11011011011", the output would pulse high at each detection, showing overlapping detection:

```
Input:   1 1 0 1 1 0 1 1 0 1 1
State:   A B C D E F D E F D E
Output:  0 0 0 0 0 1 0 0 1 0 0
```

Output '1' at positions where the sequence "11011" is detected, including overlapping positions.

14 Limitations and Observations

- **State Explosion:** For longer sequences, the number of required states increases linearly.
- **Unused States:** Some binary encodings may be unused, requiring careful handling in hardware.
- **Glitches:** Moore machines are less susceptible to output glitches since outputs change only on state transitions.
- **Overlapping:** Proper state transitions after detection are crucial to allow overlapping.

15 Error Handling and Unused States

In hardware, unused state encodings (such as 110 and 111 in this design) should be managed to prevent the FSM from entering illegal states. This can be achieved by resetting the FSM to the initial state if an unused state is detected, or by designing the next-state logic to redirect any unused state back to the initial state.

16 Comparison with Mealy Machine

Feature	Moore Machine	Mealy Machine
Output Depends	Only on state	On state and input
Number of States	More (for same function)	Fewer
Output Timing	Synchronous with clock	May change asynchronously
Design Simplicity	Simpler output logic	More compact state diagram
Overlap Handling	Both can handle, but transitions differ	Both can handle, often with fewer states

For the same pattern, a Mealy machine would require fewer states and may react faster, but at the cost of potentially less predictable output timing.

17 Conclusion

A Moore FSM sequence detector for the binary pattern **11011** with overlapping allowed was designed and analyzed. The detector uses six clearly defined states to track progress through the sequence, with outputs assigned only to the final state. Overlapping detection is implemented by transitioning to the appropriate state after a match, enabling detection of consecutive, overlapping patterns. The design is robust, with outputs synchronized to state changes, making it well-suited for hardware implementation and resistant to glitches.

The approach demonstrates the clarity and reliability of Moore machine logic for sequence detection, especially where output stability is critical.

18 References

1. M. Morris Mano, *Digital Design*, 5th Edition, Pearson.
2. John F. Wakerly, *Digital Design: Principles and Practices*, 4th Edition, Pearson.
3. Any additional datasheets, lecture notes, or online resources used.