Draw An ASM For A Sequential Circuit Has One Input And One Output. When Input Sequence "110" Occurs,

Draw An ASM For A Sequential Circuit Has One Input And One Output. When Input Sequence "110" Occurs,

Designing sequential circuits is fundamental in digital electronics, especially when implementing pattern detection, sequence recognition, and control systems. An Algorithmic State Machine (ASM) is a powerful methodology used to design and model such sequential circuits systematically. In this comprehensive guide, we will explore how to draw an ASM for a sequential circuit with one input and one output that detects the specific sequence "110". This article will delve into the fundamental concepts, step-by-step design procedures, and best practices to create an effective ASM for the specified pattern detection.

Understanding Sequential Circuits and ASM Methodology

What Is a Sequential Circuit?

A sequential circuit is a digital logic circuit whose output depends not only on the current input but also on the history of inputs, which is stored in internal states. Unlike combinational circuits, sequential circuits have memory elements such as flip-flops or latches, enabling them to recognize patterns over time.

Key Characteristics:

- Memory-dependent behavior
- State-based operation
- Capable of pattern detection, counting, and control functions

What Is an Algorithmic State Machine (ASM)?

An ASM is a high-level, graphical modeling technique for designing sequential logic. It combines the concepts of state machines and algorithmic flowcharts, providing a clear and structured way to specify the behavior of sequential circuits.

Features of ASM:

- Uses state diagrams to represent states
- Includes actions associated with states and transitions
- Facilitates systematic translation into hardware

Advantages:

- Clear visualization of circuit behavior
- Easier debugging and validation
- Supports modular design approach

Problem Definition: Pattern Detection of "110"

The goal is to design a sequential circuit that outputs a signal (say, 'Z') whenever the input sequence "110" occurs. The circuit should:

- Monitor a serial input stream (single input line)
- Recognize the occurrence of "110" in the input sequence
- Generate a specific output (e.g., Z=1) immediately upon detection
- Reset or continue detecting subsequent occurrences

Important notes:

- The input is a serial data stream (binary)
- The circuit must handle overlapping sequences, e.g., in "11110", the sequence "110" occurs starting at the second input

---

Designing the ASM for the Sequence Detector

Designing an ASM involves several systematic steps: defining states, establishing transition conditions, specifying output actions, and then translating the ASM into hardware.

Step 1: Identify the States

States represent the progress in detecting the sequence "110". Each state indicates how much of the pattern has been matched so far.

Possible states:

- S0: No match yet; initial state
- S1: '1' matched (first bit of "110")
- S2: '11' matched (first two bits of "110")

- S3: "110" matched (full sequence detected)

Step 2: Define State Transitions

The transitions depend on the current input bit and the current state.

Current State	Input Bit	Next State	Output Action
S0	0	S0	Z=0
S0	1	S1	Z=0
S1	0	S2	Z=0
S1	1	S1	Z=0
S2	0	S0	Z=1 (pattern detected)
S2	1	S1	Z=0

Notes:

- When sequence "110" is detected, the output Z becomes '1' immediately.
- Overlapping sequences are handled since after detection, the circuit transitions to the appropriate state based on the input.

Step 3: Draw the State Diagram

Visualize the states and transitions:

- Draw circles representing S0, S1, S2
- Connect them with arrows indicating transitions based on input bits
- Label transitions with input and output actions

Example:

- From S0:
- Input '0': stay in S0
- Input '1': move to S1

- From S1:
- Input '0': move to S2
- Input '1': stay in S1

- From S2:
- Input '0': move back to S0, output '1'
- Input '1': move to S1

Implementing the ASM: State Table and Logic Equations

State Table

Construct a comprehensive table that shows current states, inputs, next states, and outputs.

Current State	Input	Next State	Output Z
S0	0	S0	0
S0	1	S1	0
S1	0	S2	0
S1	1	S1	0
S2	0	S0	1
S2	1	S1	0

Note: The output Z is '1' only when the sequence "110" is detected.

State Encoding

Assign binary codes to states for hardware implementation:

- S0: 00
- S1: 01
- S2: 10

Now, derive logical expressions for next states and output.

Logic Equations for Next State and Output

Using flip-flops (e.g., D flip-flops):

- Let Q1 Q0 represent current state bits
- D1 D0 represent next state bits

Next State Equations:

- D1 = (Q1 AND NOT Q0 AND NOT Input) OR (Q1 AND Input) OR (NOT Q1 AND Q0 AND NOT Input)

- D0 = (Q0 AND NOT Input) OR (NOT Q0 AND Input AND Q1)

Output Equation:

- Z = Q1 AND Q0 AND NOT Input

Hardware Implementation of the ASM

Key Components:

- Two D flip-flops to store state bits
- Logic gates to generate D1 and D0 inputs
- Output logic to generate Z

Implementation Steps:

1. Connect flip-flops Q1 and Q0 to store current state
2. Use combinational logic (AND, OR, NOT gates) to compute next state inputs D1 and D0 based on current state and input
3. Connect flip-flops to update on clock
4. Generate output Z using the logic expression

Validating and Testing the ASM Design

Simulation:

- Apply various input sequences such as "110", "1110", "0110", "11011"
- Observe the output Z:
- Z should be '1' exactly when "110" occurs
- Z should reset to '0' after detection
- Confirm overlapping detection: e.g., in "110110", detection occurs twice

Verification:

- Use state table and transition diagram to verify correctness
- Cross-check logic equations with simulation results

Advantages of Using ASM for Sequence Detection

- Clear visualization of circuit behavior
- Simplifies complex pattern recognition design
- Facilitates systematic hardware translation
- Eases debugging and modification

Conclusion

Designing an ASM for a sequential circuit that detects the sequence "110" involves understanding the pattern, defining states, mapping transitions, and translating these into hardware logic. The process includes creating a state diagram, developing a state table, deriving logic equations, and implementing the logic with flip-flops and gates. This methodology not only ensures correctness but also provides a structured approach to designing complex pattern detection circuits. Utilizing ASM significantly enhances the clarity and efficiency of digital circuit design, making it an essential tool for engineers working in digital electronics and communication systems.

Keywords: ASM, Sequence Detector, Sequential Circuit, Pattern Recognition, State Diagram, State Table, Logic Design, Digital Electronics, Pattern Detection Circuit

Frequently Asked Questions

What is the primary purpose of drawing an ASM for a sequential circuit with one input and one output?

The primary purpose is to model the behavior of the circuit by illustrating state transitions and output responses based on input sequences, such as detecting specific patterns like '110'.

How does the ASM represent the occurrence of the input sequence '110'?

The ASM uses states to track progress toward the sequence '110', transitioning between states with each input, and generating an output when the sequence is detected.

What are the key components involved in designing an ASM for this sequential circuit?

The key components include states representing different partial matches, transition functions dictating state changes based on input, and output functions that produce output upon sequence detection.

How do you determine the number of states needed in the ASM for detecting '110'?

The number of states depends on the partial match process; for '110', typically 3-4 states are used to represent the progress of matching the sequence, including initial, partial, and complete match states.

Can you explain the transition logic when the input '110' occurs in the ASM?

When the input sequence '110' occurs, the ASM transitions through states that represent each matched bit, finally reaching a state that indicates the full sequence has been detected, and triggers the output accordingly.

What is the significance of output generation in the ASM when '110' is detected?

The output signifies that the specific sequence '110' has been recognized by the circuit, which can trigger subsequent actions or signals in the overall system, such as setting an output high or initiating a process.

Additional Resources

Draw An ASM For A Sequential Circuit Has One Input And One Output. When Input Sequence "110" Occurs

Designing sequential circuits is a fundamental aspect of digital systems, enabling devices to respond not just to current inputs but also to their history. In particular, creating an ASM (Algorithmic State Machine) chart for a sequential circuit with one input and one output—especially when the circuit is designed to detect a specific sequence like "110"—is a classic problem that illustrates the principles of finite state machine (FSM) design. This guide provides a comprehensive walkthrough of how to develop such an ASM, covering the underlying concepts, step-by-step construction, and practical implementation tips.

Understanding the Problem

Before diving into the design process, it's crucial to understand what the problem entails:

- The sequential circuit has one input (say, `X`) and one output (say, `Z`).
- The circuit must detect the sequence "110" in the input stream.
- When the sequence "110" occurs, the output `Z` should become '1', indicating detection, and then revert to '0' otherwise.
- The design should incorporate state memory to recognize the sequence amid a continuous input stream.

This problem is a classic example used in digital logic design courses to illustrate finite state machine

(FSM) design, sequence detection, and the use of ASM charts.

Fundamental Concepts

Finite State Machines (FSM)

An FSM is a mathematical model of computation composed of:

- States: Representing the history of inputs.
- Transitions: Moving from one state to another based on input.
- Outputs: Generated based on the current state (Moore machine) or current state and input (Mealy machine).

ASM Charts

The Algorithmic State Machine (ASM) is a graphical representation that combines state machine diagrams with flowchart elements to specify the behavior of digital systems clearly.

An ASM chart typically includes:

- States: Represented as rectangles, with labels.
- Transitions: Arrows indicating movement from one state to another, often labeled with input conditions.
- Conditional statements: Inside states, indicating output and next state logic.
- Output actions: Indicated in the state or transition blocks.

Step-by-Step Guide to Drawing the ASM

Step 1: Identify the Requirements and Inputs

- Input: `X` (binary, 0 or 1)
- Output: `Z` (binary, 0 or 1)
- Operation: Detect the sequence "110" in the input stream.

Step 2: Determine Necessary States

The key is to track how much of the sequence "110" has been detected so far. The states should reflect the current progress:

- S0: No relevant input detected (initial state).
- S1: Detected a '1', waiting for next input.
- S2: Detected "11", waiting for the third input.
- S3: Detected "110" — sequence recognized.

However, to handle overlapping sequences correctly, some states might need to be reentrant or have transitions that consider partial matches.

Step 3: Define State Behavior and Transitions

- From S0:
- If `X=1`, move to S1.
- If `X=0`, stay in S0.
- From S1:
- If `X=1`, stay in S1 (since "11" could overlap).
- If `X=0`, move to S2.
- From S2:
- If `X=1`, move to S3 (sequence "110" detected).
- If `X=0`, revert to S0 (sequence broken).
- From S3:
- Output `Z=1` indicating sequence detection.
- Transition based on input to handle overlapping sequences:
- If `X=1`, go to S1 (since the last input is '1', possibly starting a new sequence).
- If `X=0`, go to S0.

Step 4: Define Output Logic

- The output `Z=1` only when the sequence "110" is detected, i.e., in state S3.
- For all other states, `Z=0`.

Step 5: Construct the ASM Chart

Based on the above, the ASM chart will have:

- States: S0, S1, S2, S3
- Transitions: Based on input `X`.
- Actions: Set output `Z` accordingly.

Detailed ASM Chart Representation

Here's a textual description of the ASM chart:

1. State S0:
- Action: `Z=0`
- Transition:
- If `X=1`, move to S1
- Else (`X=0`), stay in S0

2. State S1:
- Action: `Z=0`
- Transition:
- If `X=1`, stay in S1 (overlapping '1's)
- If `X=0`, move to S2

3. State S2:
- Action: `Z=0`
- Transition:

- If `X=1`, move to S3 (detection)
- If `X=0`, revert to S0

4. State S3:
- Action: `Z=1` (sequence detected)
- Transition:
- If `X=1`, move to S1 (possible overlapping sequence)
- If `X=0`, revert to S0

Visual Representation (Textual)

```
[ S0 ] --X=1--> [ S1 ] --X=0--> [ S2 ] --X=1--> [ S3 ] --X=1--> [ S1 ]
| | | |
| | | --X=0--> [ S0 ]
| | --X=0--> [ S0 ] (or continue)
| |
--X=0--> [ S0 ]
```

In the ASM, each state would be represented as a box with the state's name, and transitions labeled with input conditions.

Practical Implementation

Step 1: State Encoding

Assign binary codes to states for hardware implementation:

State	Code (2 bits)
S0	00
S1	01
S2	10
S3	11

Step 2: State Transition Table

Present State	Input `X`	Next State	Output `Z`
00 (S0)	0	00 (S0)	0
00 (S0)	1	01 (S1)	0
01 (S1)	0	10 (S2)	0
01 (S1)	1	01 (S1)	0
10 (S2)	0	00 (S0)	0
10 (S2)	1	11 (S3)	0
11 (S3)	0	00 (S0)	1

| 11 (S3) | 1 | 01 (S1) | 1 |

Step 3: Implementing the Circuit

The FSM can be implemented using flip-flops (e.g., D flip-flops), combinational logic for next state and output logic, and the input `X`. The logic equations derived from the transition table guide the circuit design.

Summary and Final Remarks

Designing an ASM for a sequential circuit that detects the sequence "110" involves understanding the sequence's partial matches, defining states to encode these, and establishing transition rules that handle overlapping sequences. The key points include:

- Recognizing the importance of states that reflect the current progress in detecting the sequence.
- Using transition logic that accounts for overlapping sequences.
- Assigning output logic that signals detection only when the full sequence is found.
- Implementing the FSM with clear state encoding and logic equations.

This methodology not only facilitates the design of sequence detectors but also provides foundational skills applicable to a wide range of sequential circuit applications, including control systems, communication protocols, and data processing hardware.

Additional Tips

- Always verify the ASM chart by simulating input sequences to ensure correct detection.
- For hardware implementation, optimize state encoding to minimize logic complexity.
- Extend this design approach to detect longer or more complex sequences by adding states accordingly.
- Use simulation tools (e.g., ModelSim, Logisim) to validate your FSM before physical implementation.

By mastering the process of drawing ASM charts for sequence detection, you gain powerful insights into the design of reliable, efficient digital systems capable of complex pattern recognition.

End of Guide

Draw An Asm For A Sequential Circuit Has One Input And One Output When Input Sequence 110 Occurs

Draw An Asm For A Sequential Circuit Has One Input And One Output When Input Sequence 110 Occurs

Related Articles

- Did The USA PATRIOT Act And Other Steps Taken After 911 Upset The Balance Under The U.S Constitution
- During The Summer Months, Glaciers At The Poles Melt, Delivering Fresh Water To The Ocean. What Does
- How Did Caesar Offend The Tribunes? He Refused To Allow Them To Celebrate Sacred Feasts. Marc Antony

Back to Home