

Chapter #8: Finite State Machine Design

Contemporary Logic Design
Assignment Project Exam Help

<https://powcoder.com>
Randy H. Katz

University of California, Berkeley
Add WeChat powcoder

June 1993

- **Counters:** Sequential Circuits where State = Output
- **Generalizes to Finite State Machines:**
Outputs are Function of State (and Inputs)
Next States are Functions of State and Inputs
Used to implement circuits that control other circuits
"Decision Making" logic
- **Application of Sequential Logic Design Techniques**
Word Problems
Mapping into formal representations of FSM behavior
Case Studies

Assignment Project Exam Help

<https://powcoder.com>
Add WeChat powcoder

Chapter Overview

Concept of the State Machine

- Partitioning into Datapath and Control
- When Inputs are Sampled and Outputs Asserted

Basic Design Approach

- Six Step Design Process

<https://powcoder.com>
Alternative State Machine Representations

- State Diagram, ASM Notation, VHDL, ABEL Description Language

Moore and Mealy Machines

- Definitions, Implementation Examples

Word Problems

- Case Studies

Computer Hardware = Datapath + Control

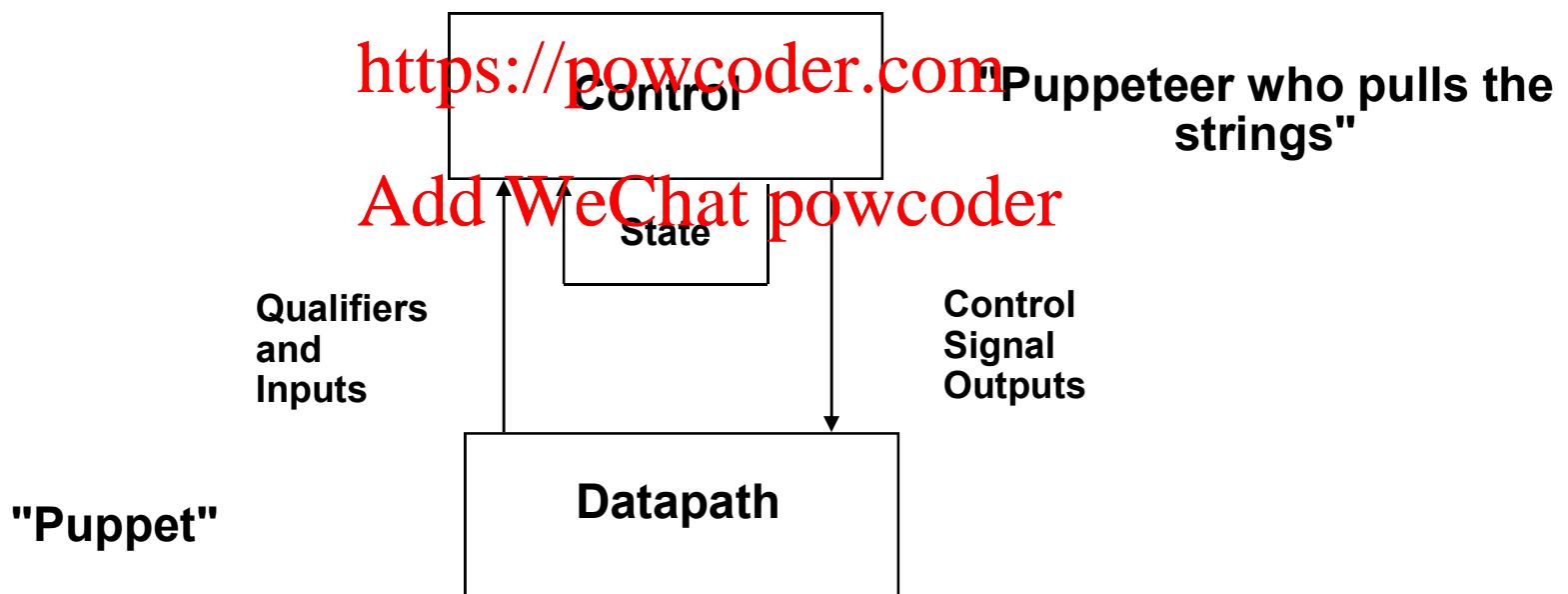
Registers
Combinational Functional
Units (e.g., ALU)
Busses

Qualifiers

FSM generating sequences
of control signals
Instructs datapath what to
do next

Control

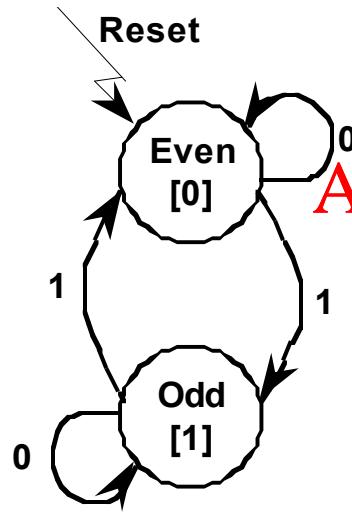
Assignment Project Exam Help



Concept of the State Machine

Example: Odd Parity Checker

Assert output whenever input bit stream has odd # of 1's



State
Diagram

Present State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Assignment Project Exam Help

<https://powcoder.com>

Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

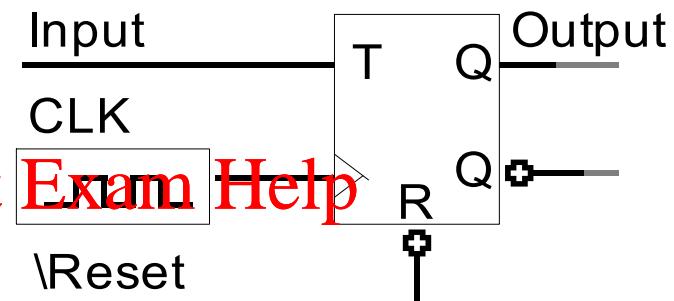
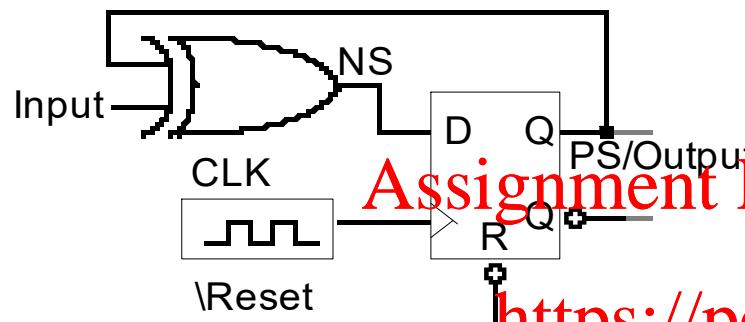
Encoded State Transition Table

Concept of the State Machine

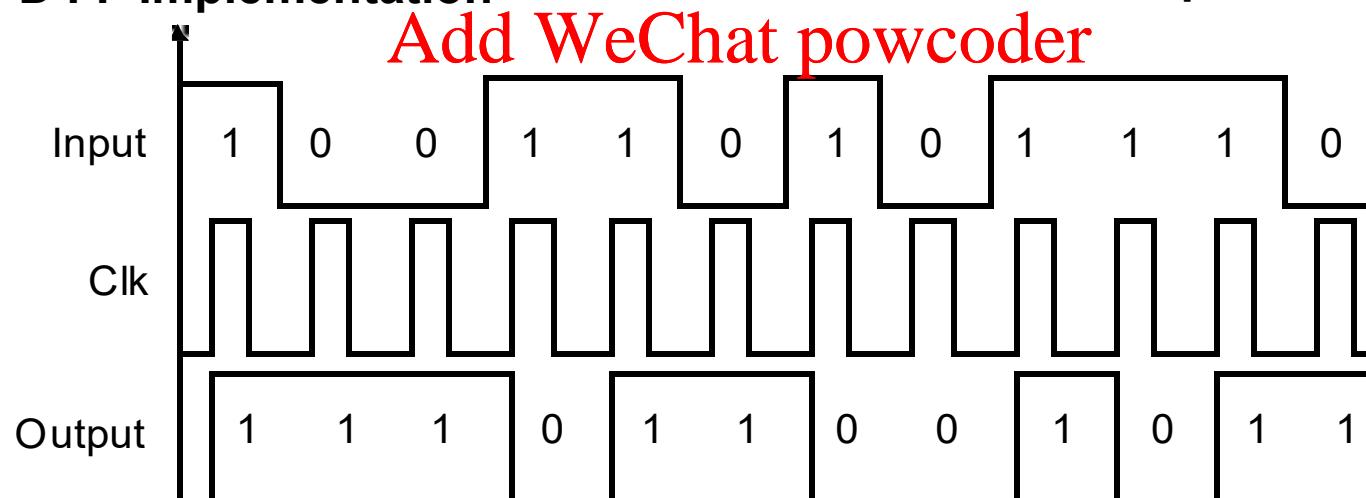
Example: Odd Parity Checker

Next State/Output Functions

$$NS = PS \text{ xor } PI; \quad OUT = PS$$



D FF Implementation



T FF Implementation

Add WeChat powcoder

Timing Behavior: Input 1 0 0 1 1 0 1 0 1 1 1 0

Timing:

When are inputs sampled, next state computed, outputs asserted?

State Time: Time between clocking events

- Clocking event causes state/outputs to transition, based on inputs

Assignment Project Exam Help

- For set-up/hold time considerations:

Inputs should be stable before clocking event

- After propagation delay, Next State entered, Outputs are stable

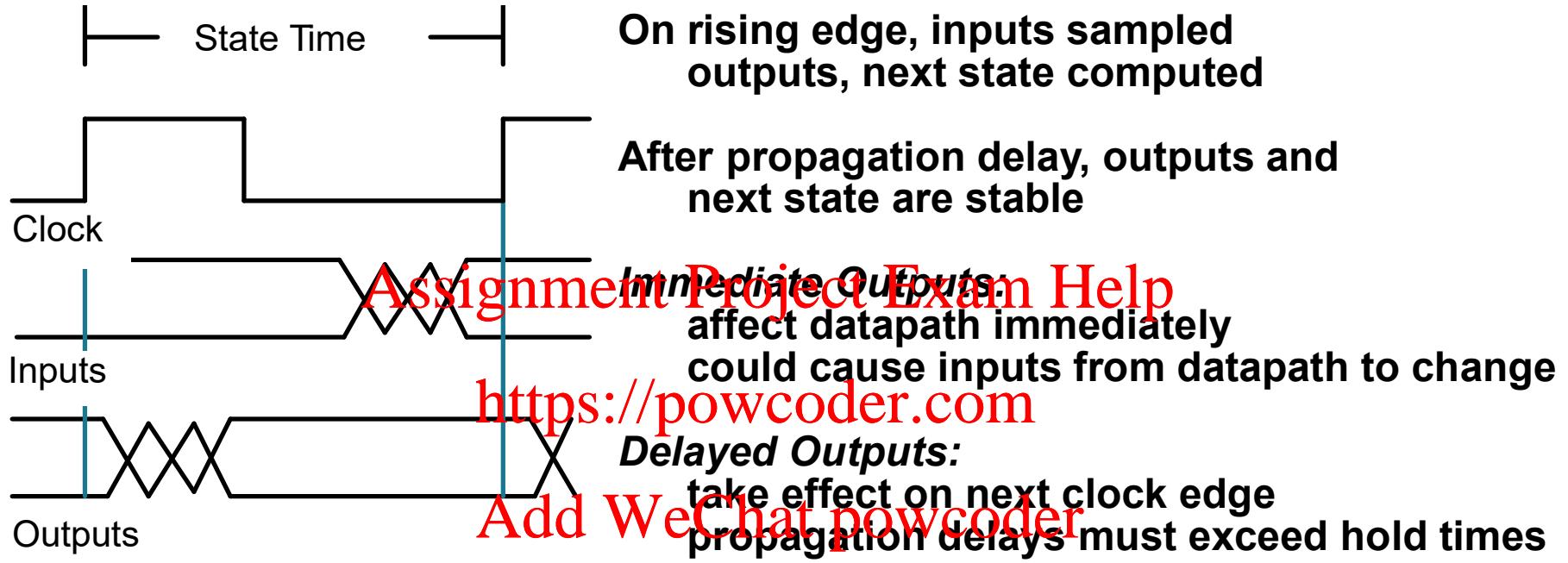
NOTE: Asynchronous signals take effect immediately

Synchronous signals take effect at the next clocking event

E.g., tri-state enable: effective immediately

sync. counter clear: effective at next clock event

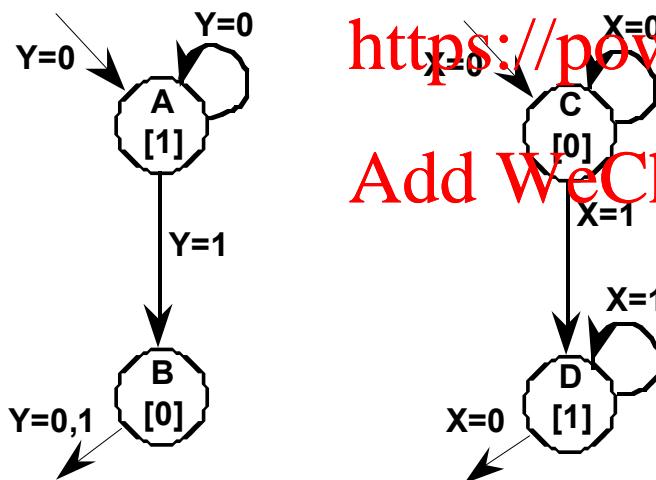
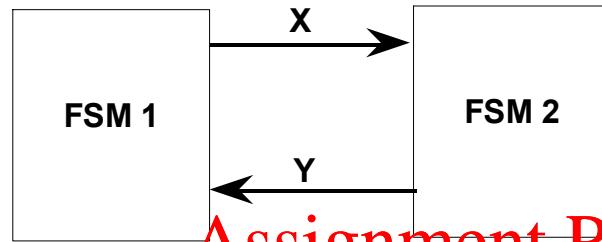
Example: Positive Edge Triggered Synchronous System



Concept of the State Machine

Communicating State Machines

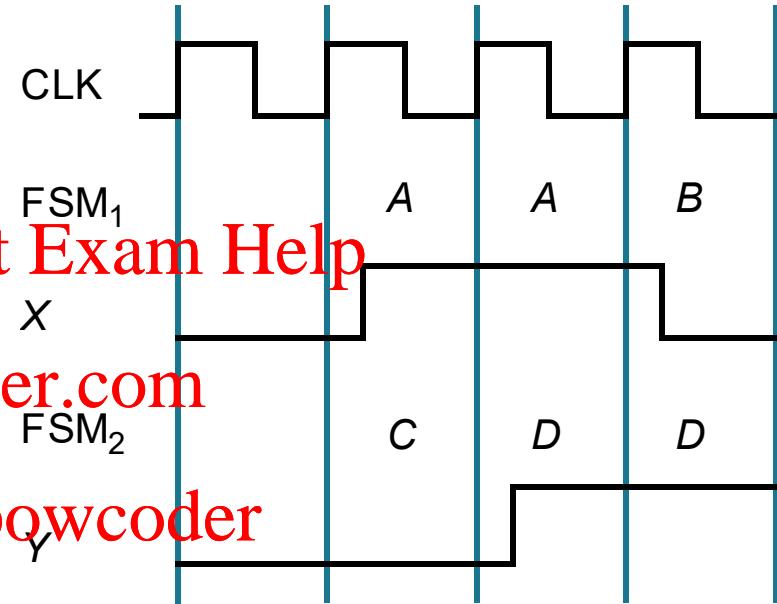
One machine's output is another machine's input



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Machines advance in lock step

Initial inputs/outputs: X = 0, Y = 0

Basic Design Approach

Six Step Process

- 1. Understand the statement of the Specification**

- 2. Obtain an abstract specification of the FSM**

- 3. Perform a state minimization**
Assignment Project Exam Help

- 4. Perform state assignment**
<https://powcoder.com>

- 5. Choose FF types to implement FSM state register**
Add WeChat powcoder

- 6. Implement the FSM**

***1, 2 covered now; 3, 4, 5 covered later;
4, 5 generalized from the counter design procedure***

Basic Design Approach

Example: Vending Machine FSM

General Machine Concept:

deliver package of gum after 15 cents deposited

single coin slot for dimes, nickels

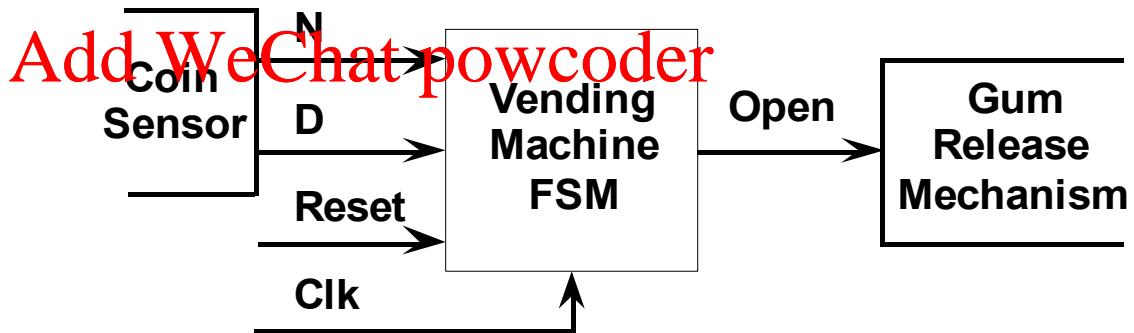
no change

Assignment Project Exam Help

Step 1. Understand the problem:

Draw a picture! <https://powcoder.com>

Block Diagram



Vending Machine Example

Step 2. Map into more suitable abstract representation

Tabulate typical input sequences:

three nickels
nickel, dime
dime, nickel
two dimes
two nickels, dime

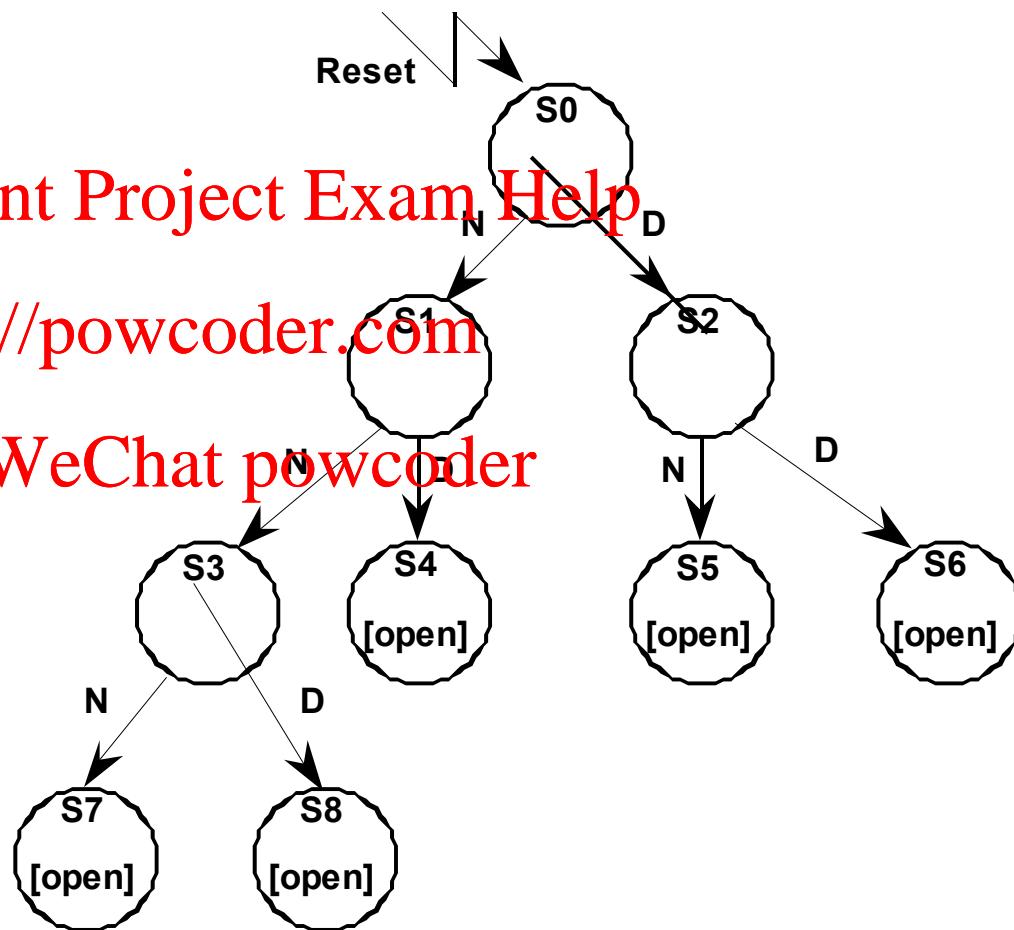
Assignment Project Exam Help

Draw state diagram:

Inputs: N, D, reset <https://powcoder.com>

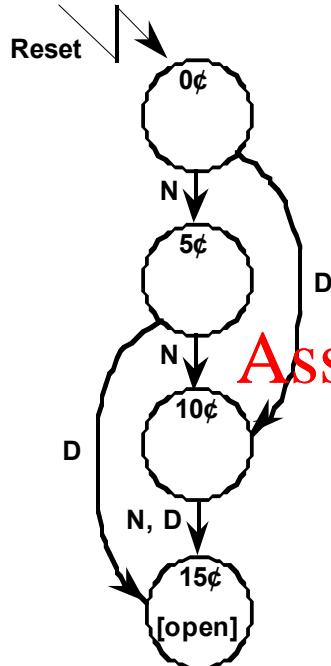
Output: open

Add WeChat powcoder



Vending Machine Example

Step 3: State Minimization



reuse states
whenever
possible

Present State	Inputs		Next State	Output
	D	N		Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Symbolic State Table

Step 4: State Encoding

Present State Q_1 Q_0	Inputs D N		Next State D_1 D_0	Output Open
0 0	0	0	0 0	0
	0	1	0 1	0
	1	0	1 0	0
	1	1	X X	X
0 1	0	0	0 1	0
	0	1	1 0	0
	1	0	1 1	0
	1	1	X X	X
1 0	0	0	1 0	0
	0	1	1 1	0
	1	0	1 1	1
	1	1	X X	X
1 1	0	0	1 1	1
	0	1	1 1	1
	1	0	1 1	1
	1	1	X X	X

Assignment Project Exam Help

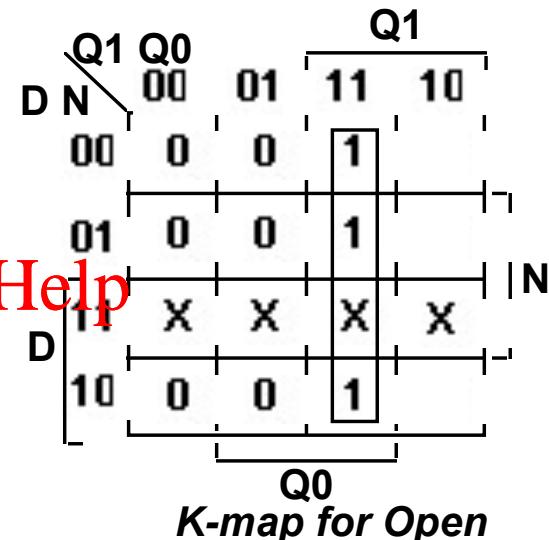
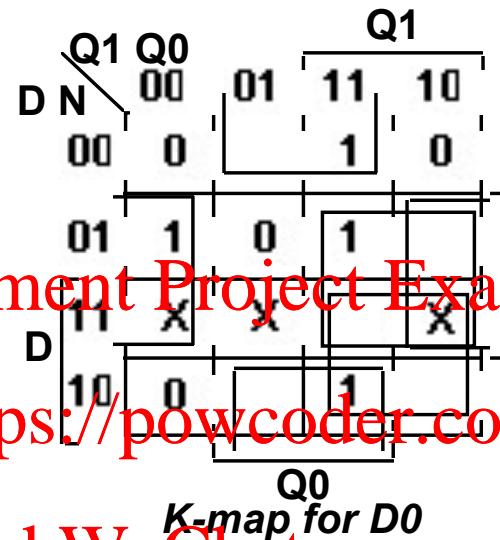
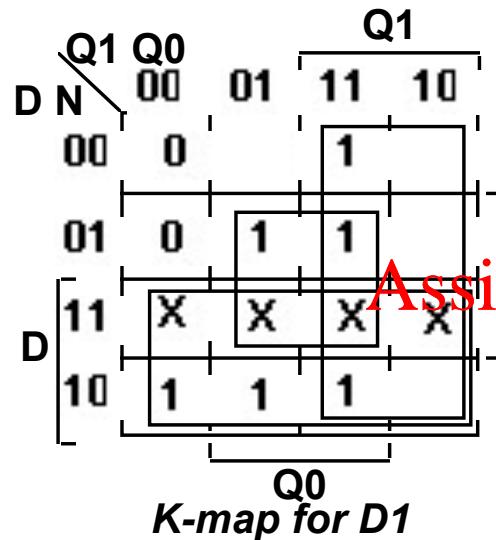
<https://powcoder.com>

Add WeChat powcoder

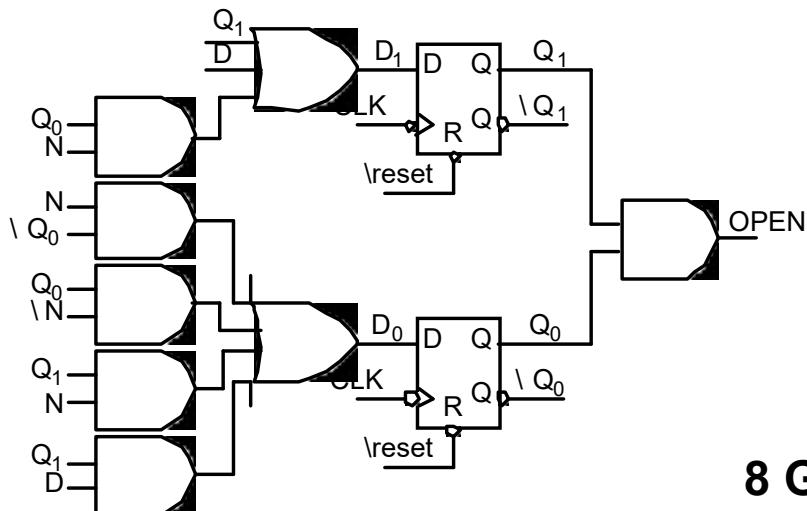
Vending Machine Example

Step 5. Choose FFs for implementation

D FF easiest to use



Assignment Project Exam Help
<https://powcoder.com>
 Add WeChat powcoder



$$D_1 = Q_1 + D + Q_0 N$$

$$D_0 = N \bar{Q}_0 + Q_0 \bar{N} + Q_1 N + Q_1 D$$

$$OPEN = Q_1 Q_0$$

8 Gates

Step 5. Choosing FF for Implementation

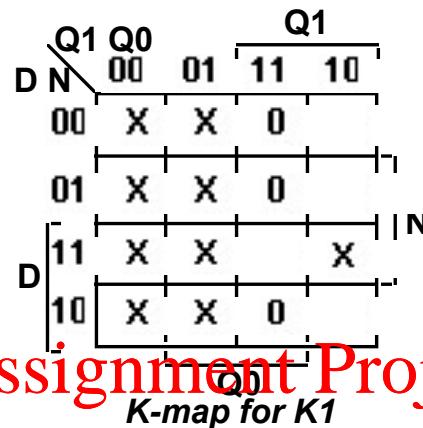
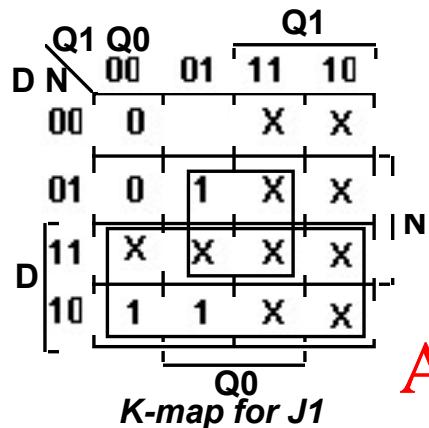
J-K FF

Present State		Inputs		Next State		J ₁	K ₁	J ₀	K ₀
Q ₁	Q ₀	D	N	D ₁	D ₀				
0	0	0	0	0	0	0	X	0	X
	0	1	0	0	1	0	X	1	X
	1	0	0	1	0	1	X	0	X
	1	1	0	X	X	X	X	X	X
0	1	0	0	0	1	0	X	X	0
	0	1	0	1	0	1	X	X	1
	1	0	0	1	1	1	X	X	0
	1	1	0	X	X	X	X	X	X
1	0	0	0	1	0	X	0	0	X
	0	1	0	1	1	X	0	1	X
	1	0	0	1	1	X	0	1	X
	1	1	0	X	X	X	X	X	X
1	1	0	0	1	1	X	0	X	0
	0	1	0	1	1	X	0	X	0
	1	0	0	1	1	X	0	X	0
	1	1	0	X	X	X	X	X	X

Remapped encoded state transition table

Vending Machine Example

Implementation:

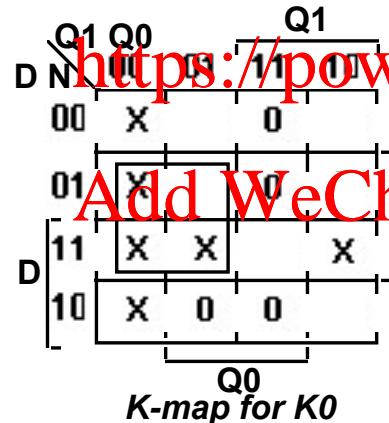
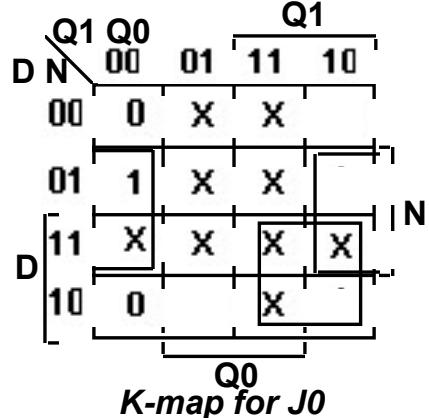


$$J_1 = D + Q_0 N$$

$$K_1 = 0$$

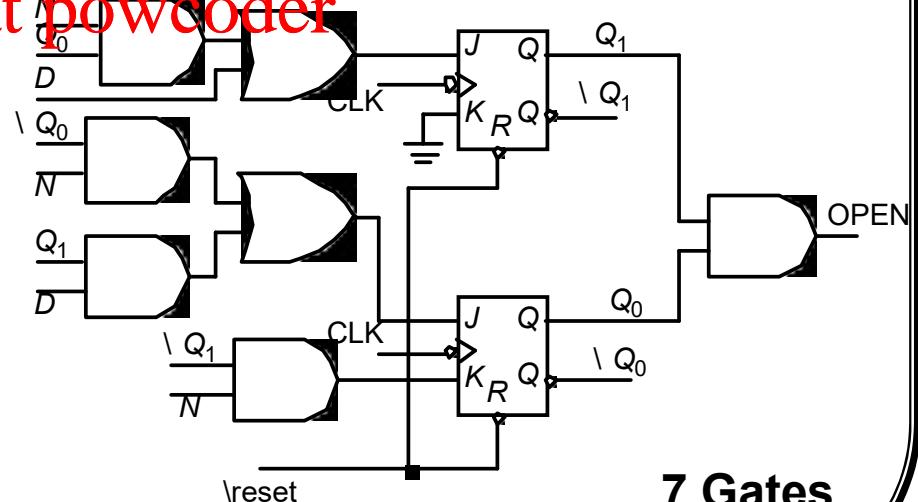
$$J_0 = \overline{Q_0} N + Q_1 D$$

$$K_0 = Q_1 N$$



<https://powcoder.com>

Add WeChat powcoder

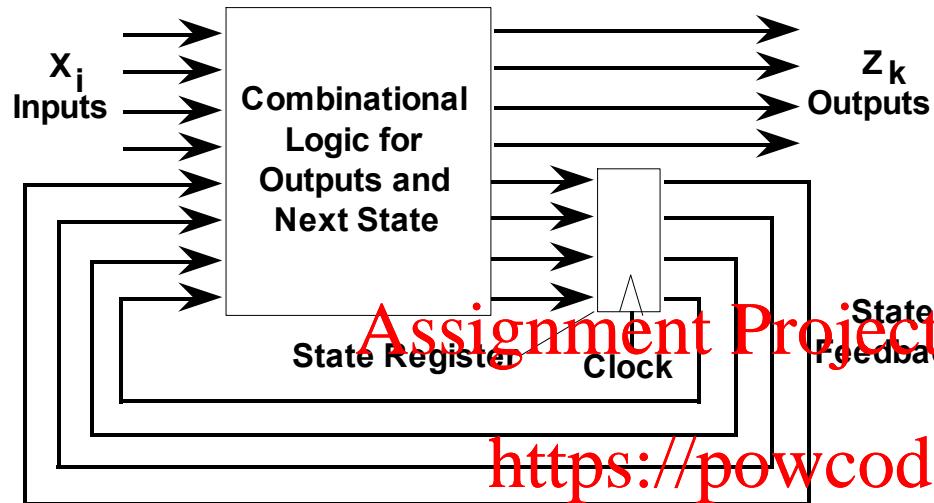


7 Gates

Moore and Mealy Machine Design Procedure

Contemporary Logic Design
Finite State Machine Design

Definitions



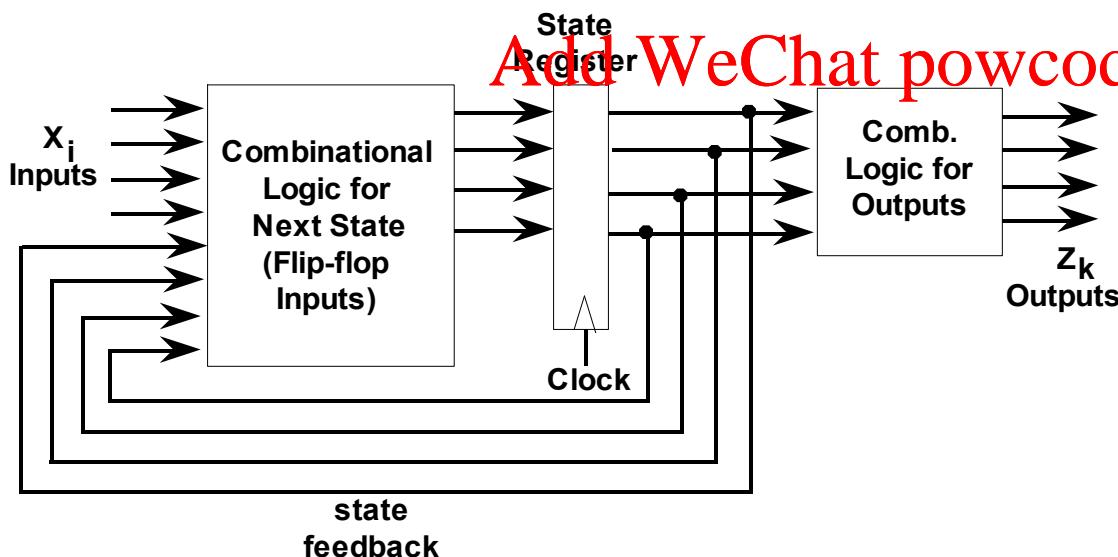
Mealy Machine

Outputs depend on state AND inputs

Input change causes an immediate output change

Asynchronous signals

Assignment Project Exam Help
<https://powcoder.com>



Moore Machine

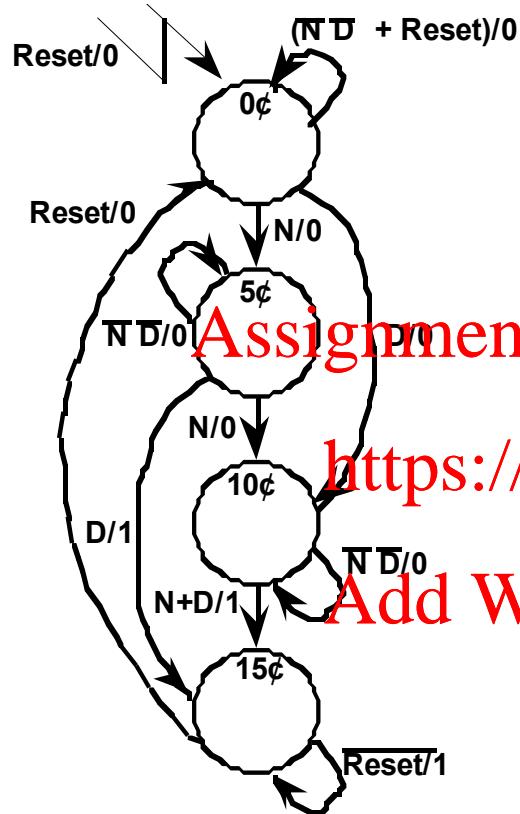
Outputs are function solely of the current state

Outputs change synchronously with state changes

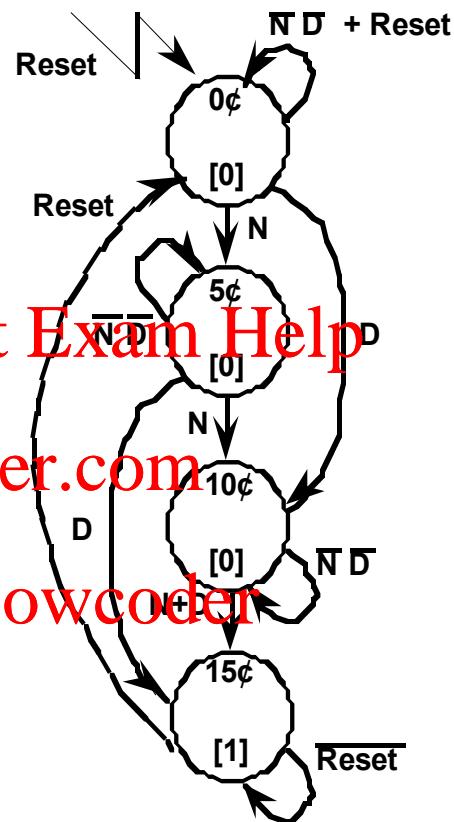
Moore and Mealy Machines

State Diagram Equivalents

**Mealy
Machine**



**Moore
Machine**



Outputs are associated
with Transitions

Outputs are associated
with State

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Moore and Mealy Machines

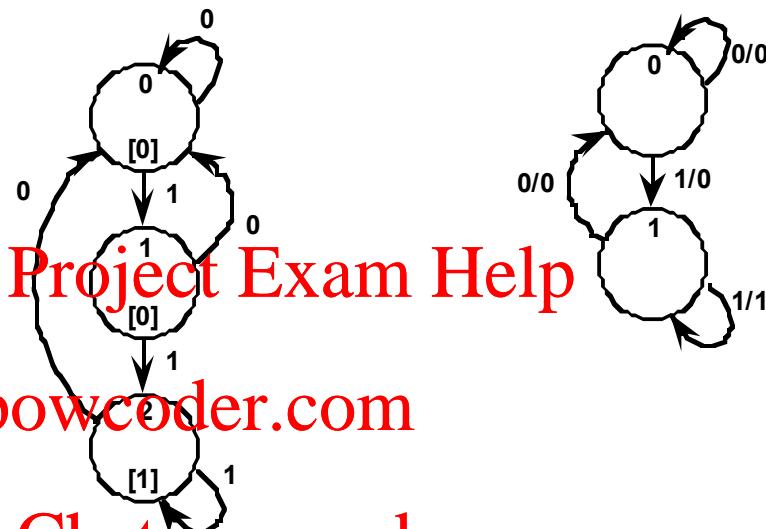
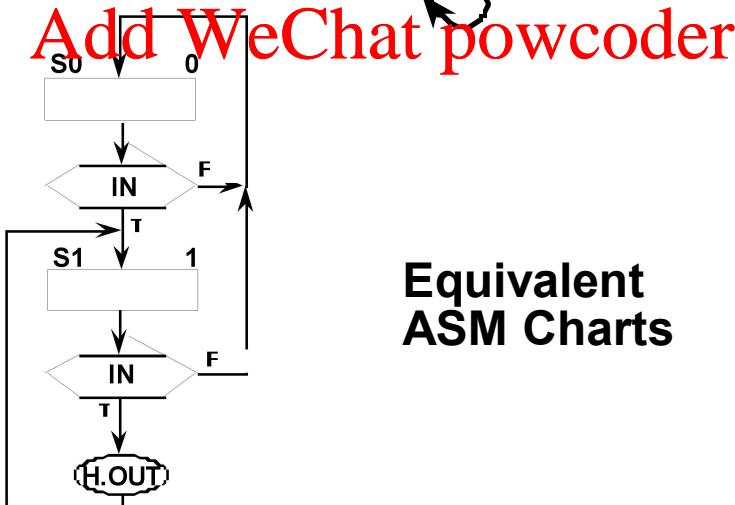
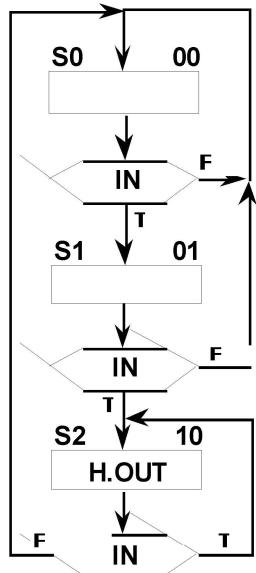
States vs. Transitions

Mealy Machine typically has fewer states than Moore Machine for same output sequence

Same I/O behavior:
Assert a single output whenever at least two 1's have been received in a sequence.

Assignment Project Exam Help

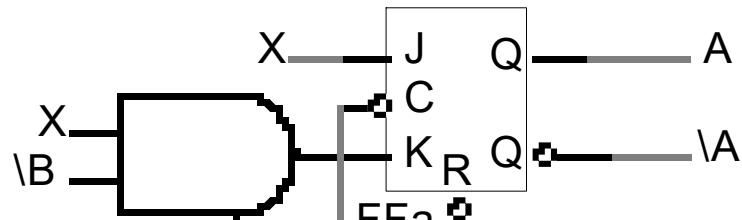
Different # of states <https://powcoder.com>



Equivalent
ASM Charts

Timing Behavior of Moore Machines

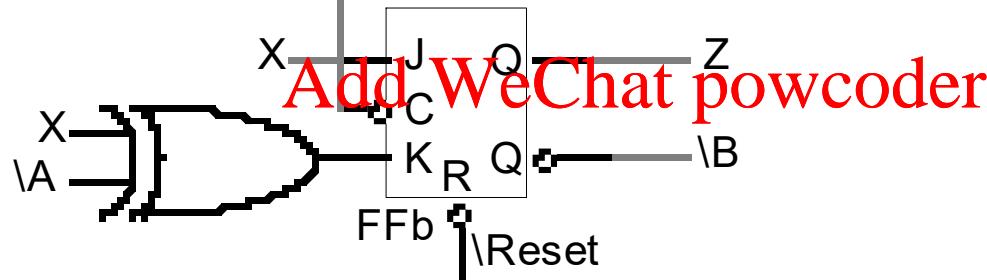
Reverse engineer the following:



Input X
Output Z
State A, B = Z

Assignment Project Exam Help

Clk
<https://powcoder.com>



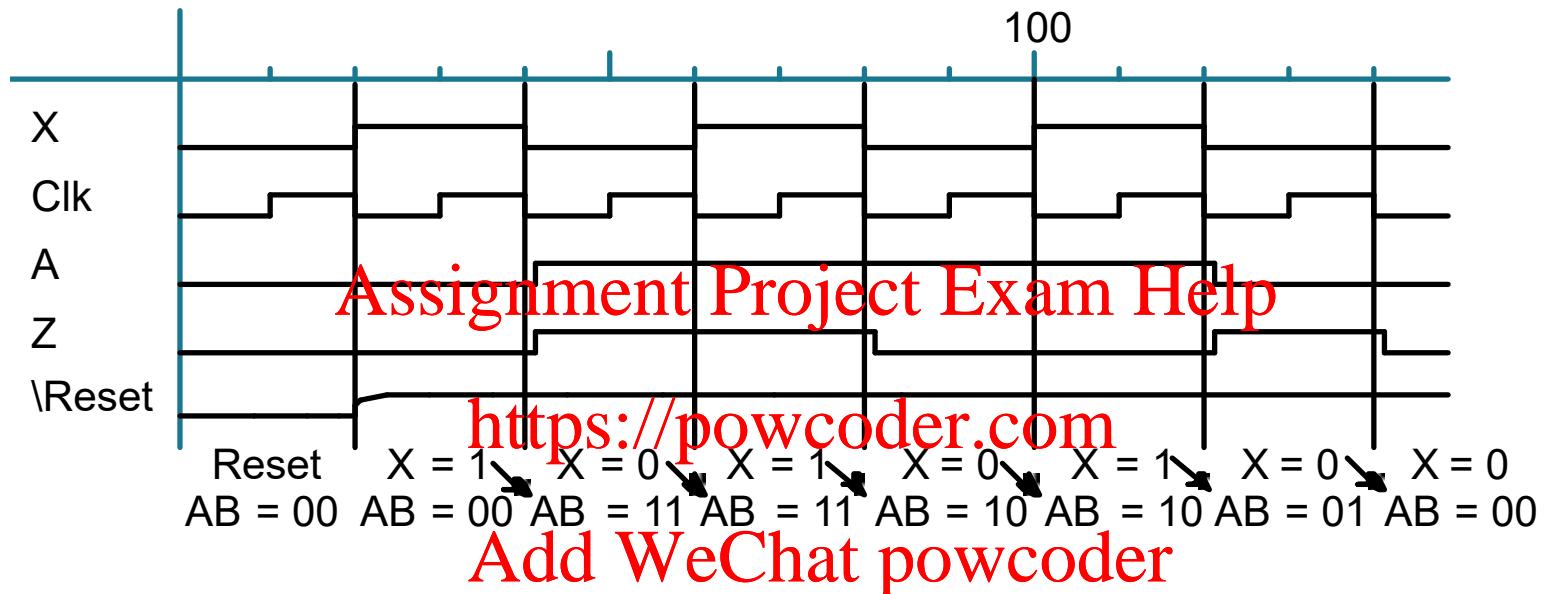
Add WeChat powcoder

Two Techniques for Reverse Engineering:

- Ad Hoc: Try input combinations to derive transition table
- Formal: Derive transition by analyzing the circuit

Ad Hoc Reverse Engineering

Behavior in response to input sequence 1 0 1 0 1 0:



**Partially Derived
State Transition
Table**

A	B	X	A+	B+	Z
0	0	0	?	?	0
		1		1	0
0	1	0	0	0	1
		1			1
1	0	0	1	0	0
		1			0
1	1	0	1	1	1
		1			1

Moore and Mealy Machines

Formal Reverse Engineering

Derive transition table from next state and output combinational functions presented to the flipflops!

$$\begin{aligned} J_a &= X \\ J_b &= X \end{aligned}$$

$$\begin{aligned} K_a &= X \cdot \bar{B} \\ K_b &= X \text{ xor } A \end{aligned}$$

$$Z = B$$

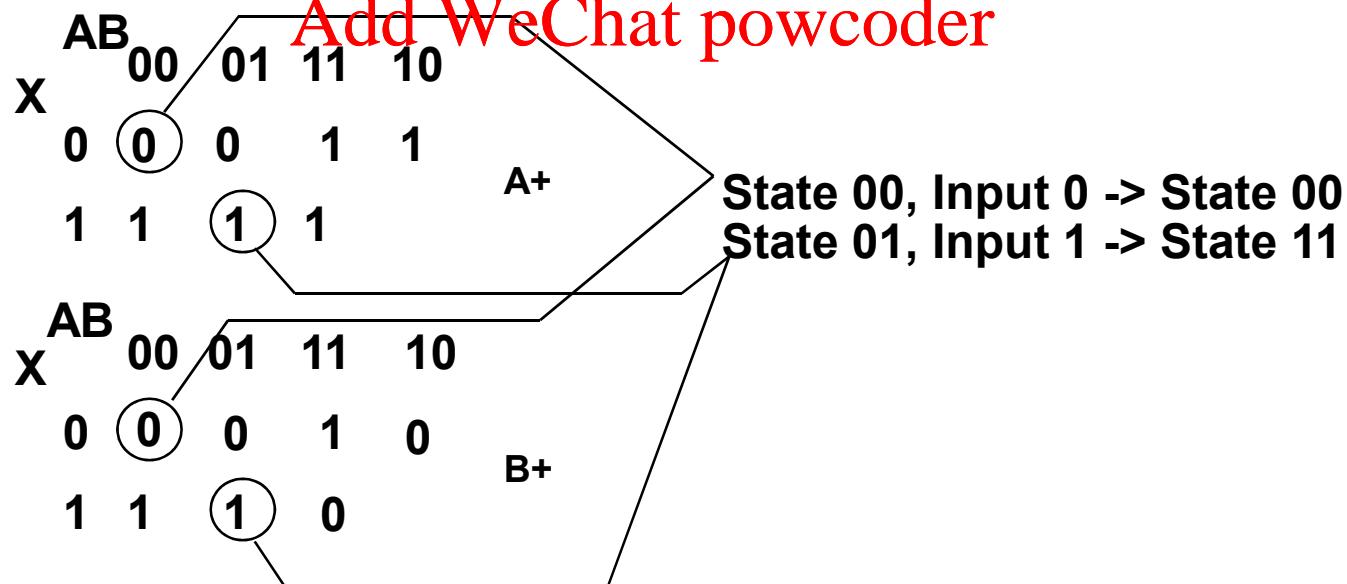
FF excitation equations for J-K flipflop:

$$A+ = J_a \cdot \bar{A} + K_a \cdot A = X \cdot \bar{A} + (\bar{X} + \bar{B}) \cdot A$$

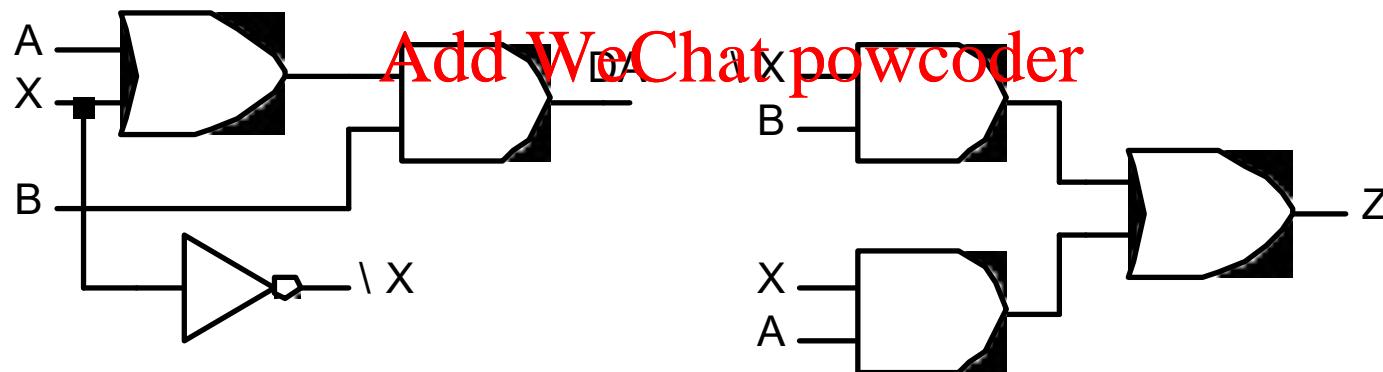
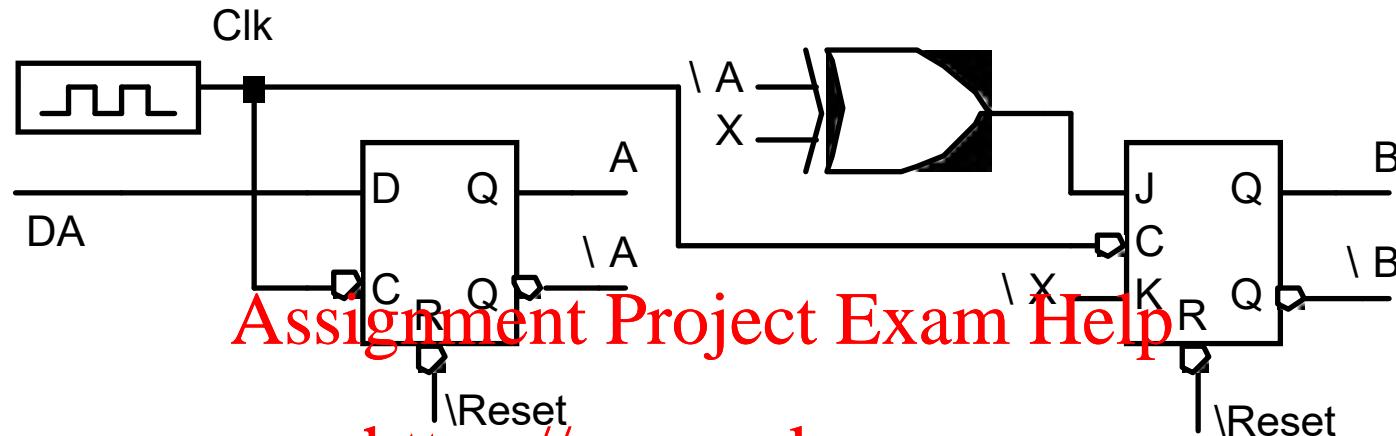
$$B+ = J_b \cdot B + K_b \cdot \bar{B} = X \cdot B + (X \cdot A + \bar{X} \cdot A) \cdot B$$

<https://powcoder.com>

Next State K-Maps:



Reverse Engineering a Mealy Machine

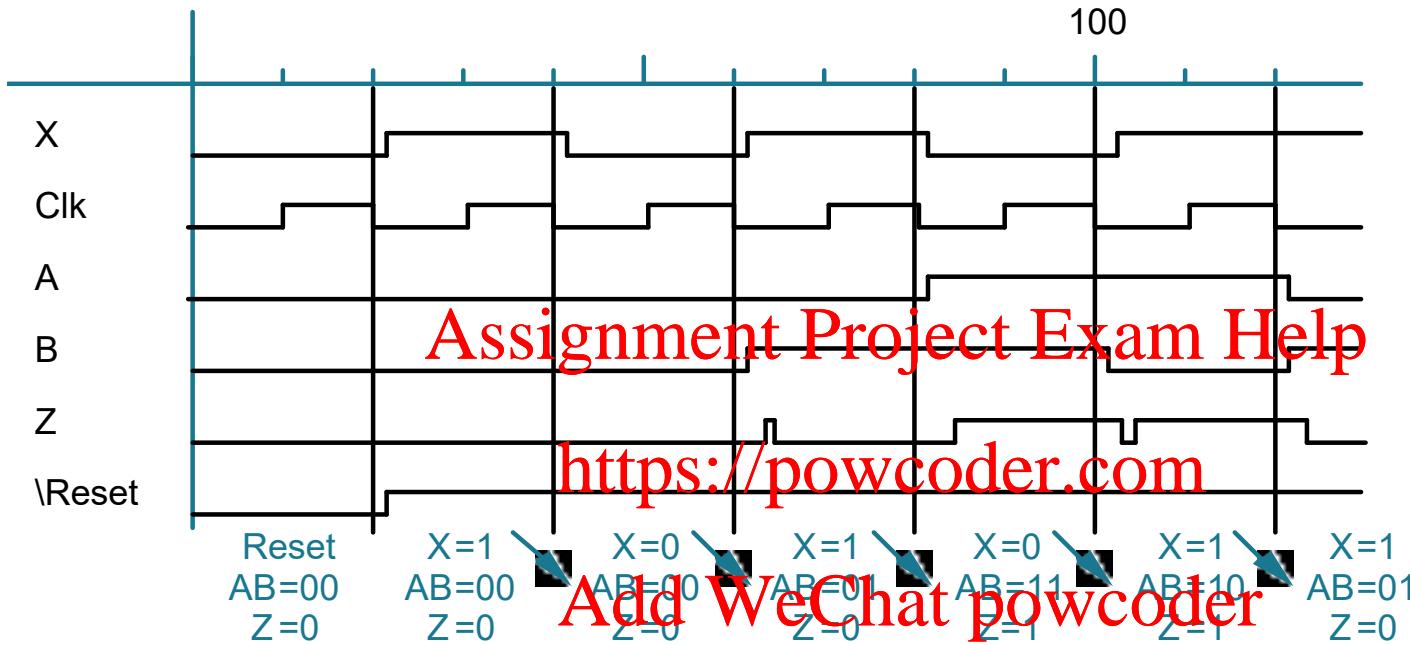


Input X, Output Z, State A, B

State register consists of D FF and J-K FF

Ad Hoc Method

Signal Trace of Input Sequence 101011:



Note glitches
in Z!

Outputs valid at
following falling
clock edge

Partially completed
state transition table
based on the signal
trace

A	B	X	A+	B+	Z
0	0	0	0	1	0
0	1	1	0	0	0
1	0	0	?	?	?
1	0	1	1	1	0
1	1	0	?	?	?
1	1	1	0	1	1
			1	0	1
			1	?	?

Formal Method

$$A^+ = B \cdot (A + X) = A \cdot B + B \cdot X$$

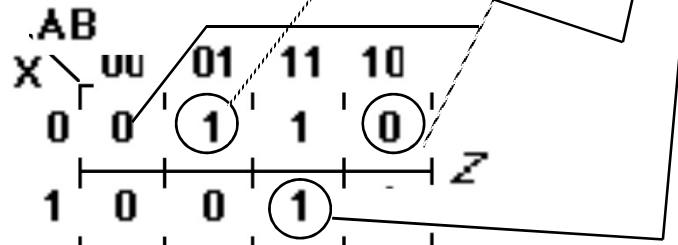
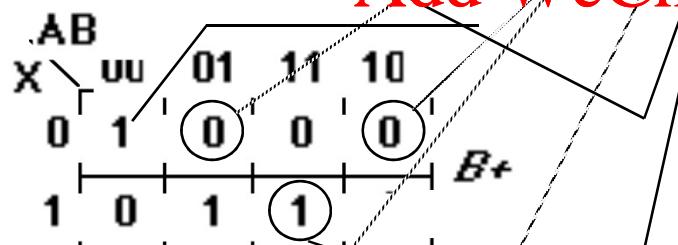
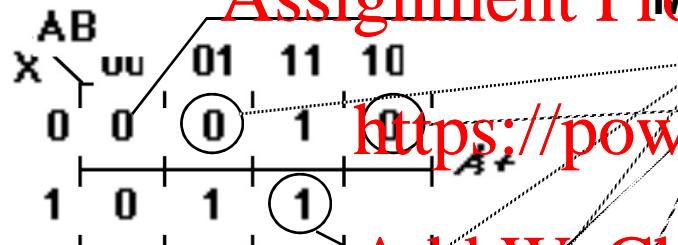
$$\begin{aligned} B^+ &= Jb \cdot \bar{B} + K_b \cdot B = (\bar{A} \text{ xor } X) \cdot \bar{B} + X \cdot B \\ &= A \cdot \bar{B} \cdot X + \bar{A} \cdot B \cdot \bar{X} + B \cdot X \end{aligned}$$

$$Z = A \cdot X + B \cdot \bar{X}$$

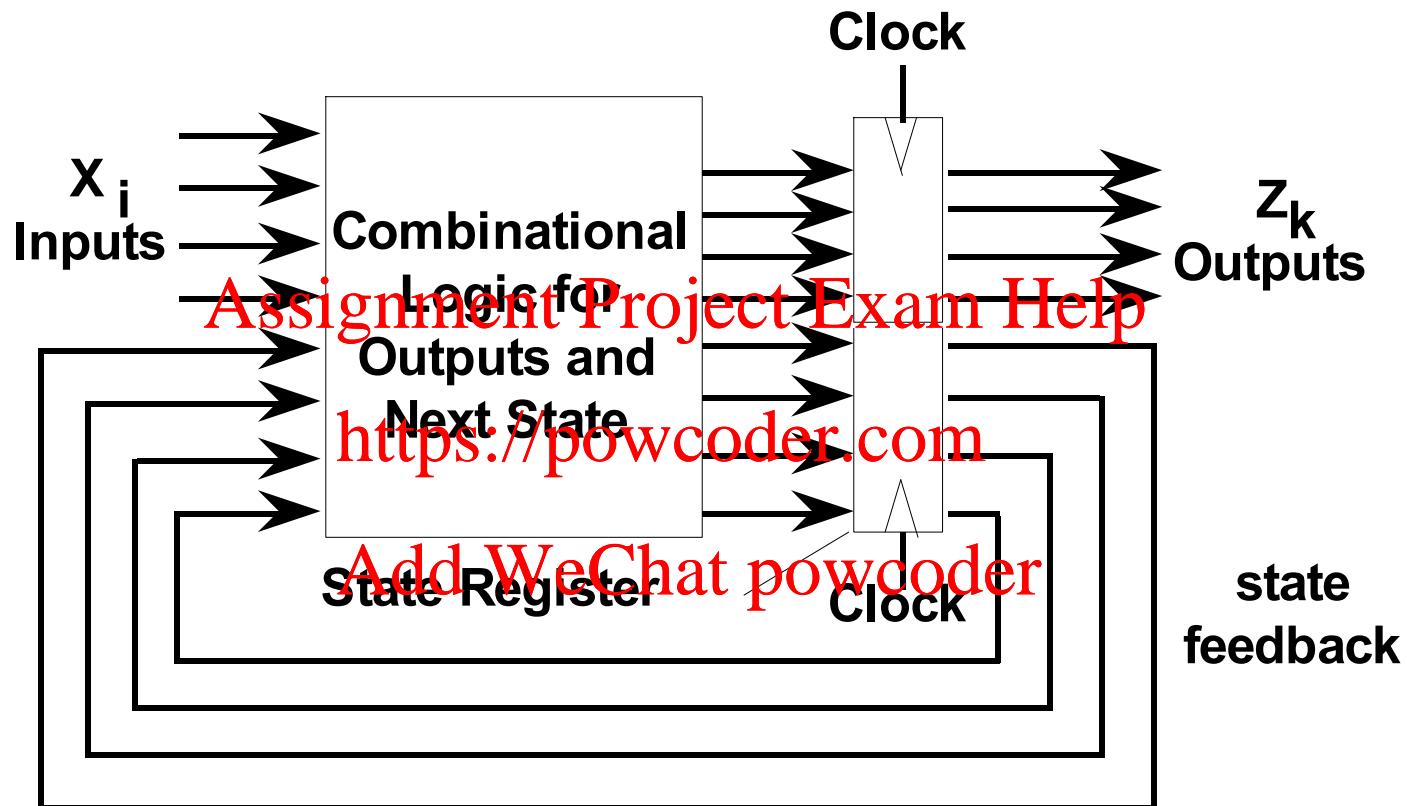
Assignment Project Exam Help
<https://powcoder.com>

Missing Transitions and Outputs:

State 01, Input 0 -> State 00, Output 1
State 10, Input 0 -> State 00, Output 0
State 11, Input 1 -> State 11, Output 1



Synchronous Mealy Machine



latched state AND outputs

avoids glitchy outputs!

Mapping English Language Description to Formal Specifications

Four Case Studies:

- Finite String Pattern Recognizer
- Complex Counter with Decision Making
- Traffic Light Controller
- Digital Combination Lock

Assignment Project Exam Help

<https://powcoder.com>

We will use state diagrams and ASM Charts

Add WeChat powcoder

Finite String Pattern Recognizer

A finite string recognizer has one input (X) and one output (Z).
The output is asserted whenever the input sequence ...010...
has been observed, as long as the sequence 100 has never been
seen.

Step 1. Understanding the problem statement

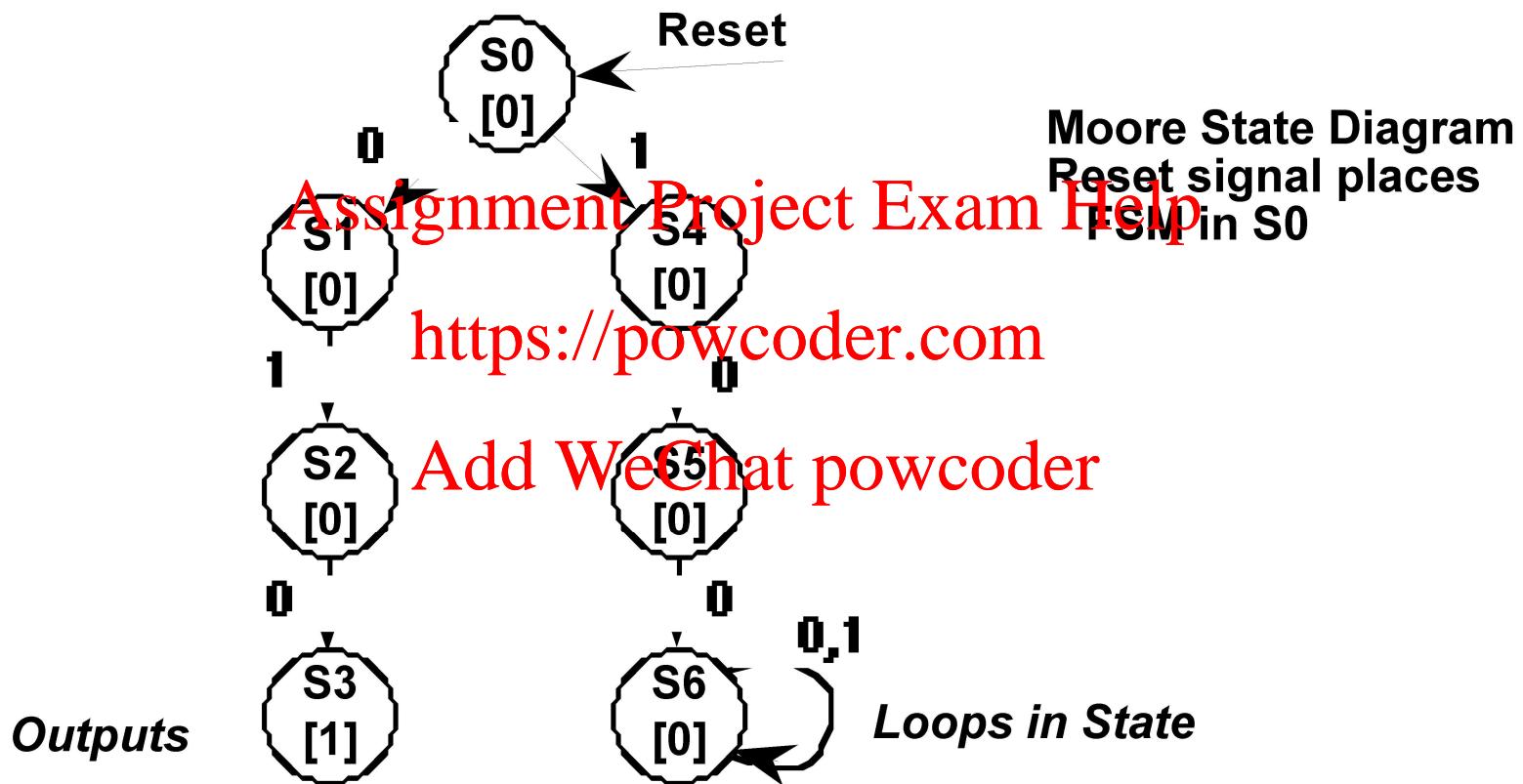
Assignment Project Exam Help
Sample input/output behavior:

X: 00101010010
Z: 00010101000...

X: 1A011010010...
Z: 00000001000...

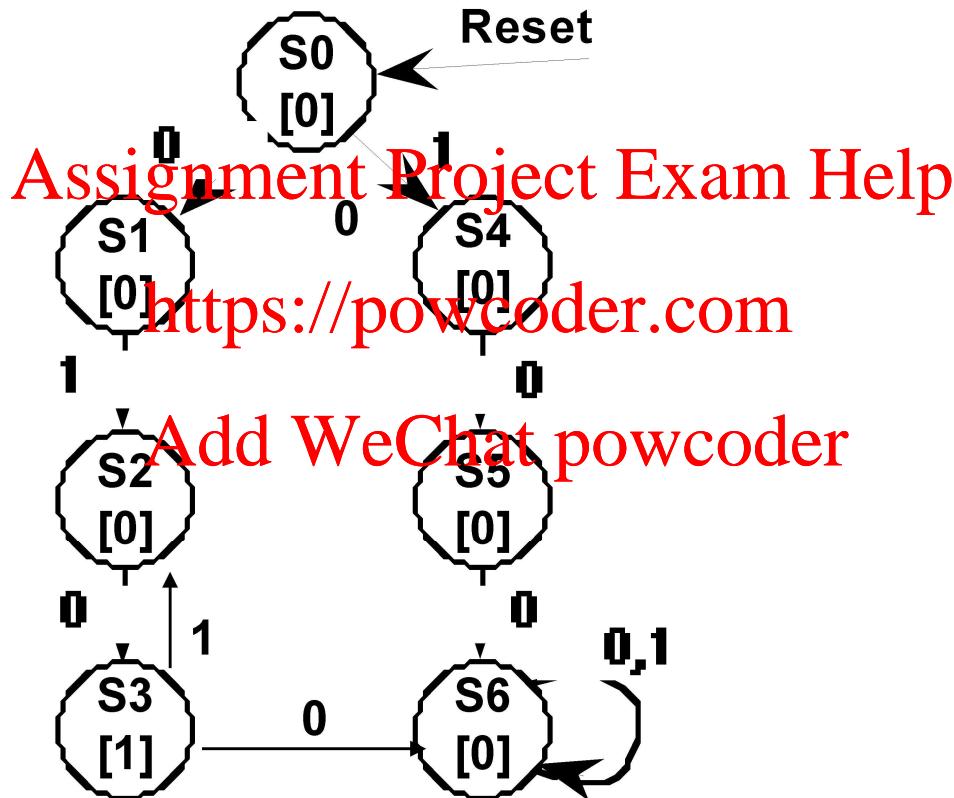
Finite String Recognizer

Step 2. Draw State Diagrams/ASM Charts for the strings that must be recognized. I.e., 010 and 100.



Finite String Recognizer

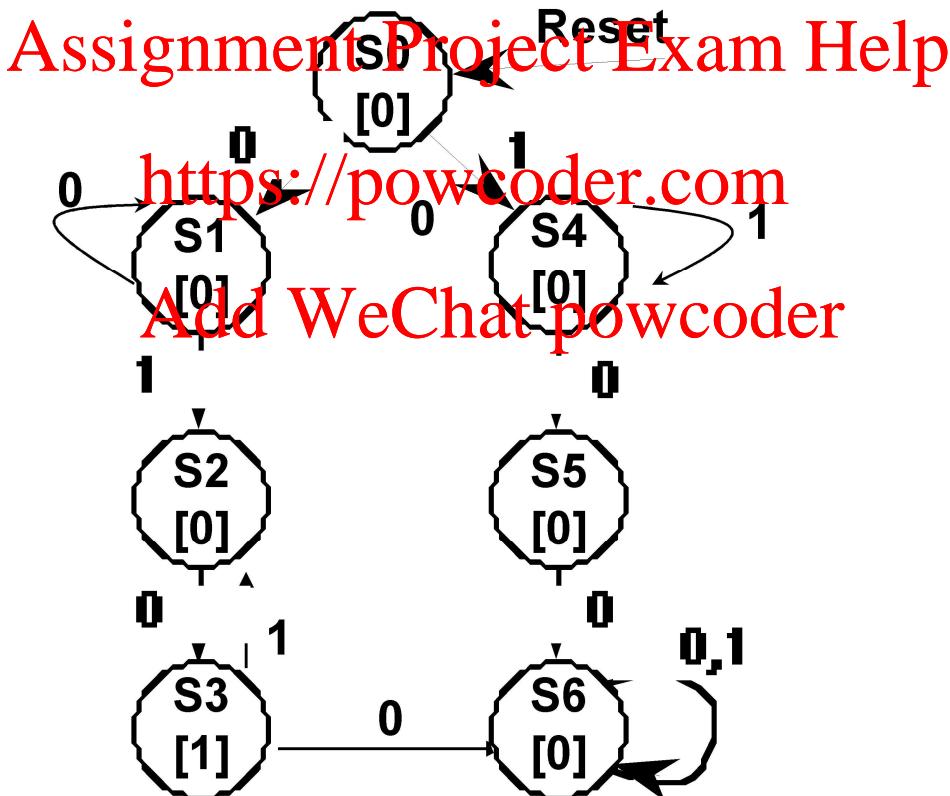
Exit conditions from state S3: have recognized ...010
if next input is 0 then have ...0100!
if next input is 1 then have ...0101 = ...01 (state S2)



Finite String Recognizer

**Exit conditions from S1: recognizes strings of form ...0 (no 1 seen)
loop back to S1 if input is 0**

**Exit conditions from S4: recognizes strings of form ...1 (no 0 seen)
loop back to S4 if input is 1**



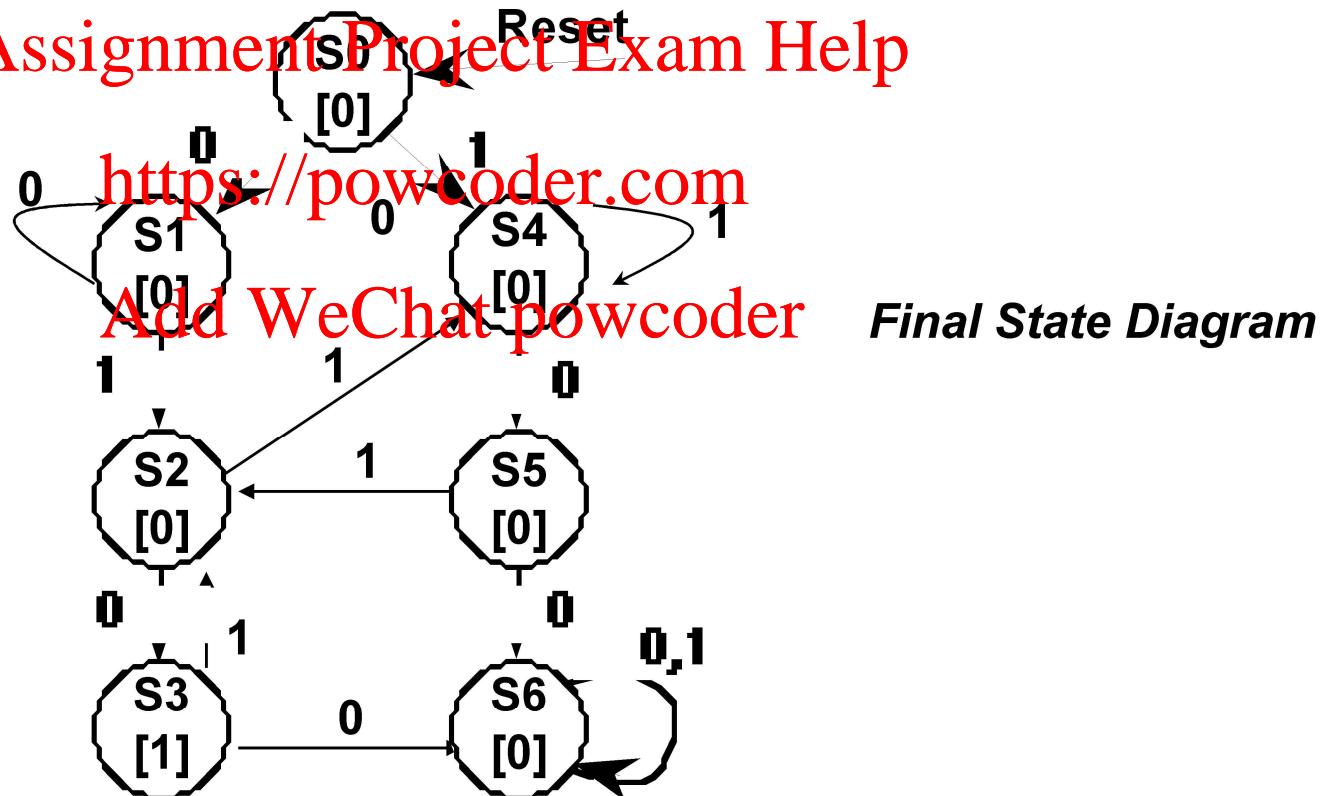
Finite String Recognizer

S2, S5 with incomplete transitions

S2 = ...01; If next input is 1, then string could be prefix of (01)1(00)
S4 handles just this case!

S5 = ...10; If next input is 1, then string could be prefix of (10)1(0)
S2 handles just this case!

Assignments Project Exam Help



Finite String Recognizer

Review of Process:

- Write down sample inputs and outputs to understand specification
- Write down sequences of states and transitions for the sequences to be recognized

Assignment Project Exam Help

- Add missing transitions; reuse states as much as possible
<https://powcoder.com>
- Verify I/O behavior of your state diagram to insure it functions like the specification

Add WeChat powcoder

Complex Counter

A sync. 3 bit counter has a mode control M. When M = 0, the counter counts up in the binary sequence. When M = 1, the counter advances through the Gray code sequence.

Binary: 000, 001, 010, 011, 100, 101, 110, 111

Gray: 000, 001, 011, 010, 110, 111, 101, 100

Assignment Project Exam Help

Valid I/O behavior:

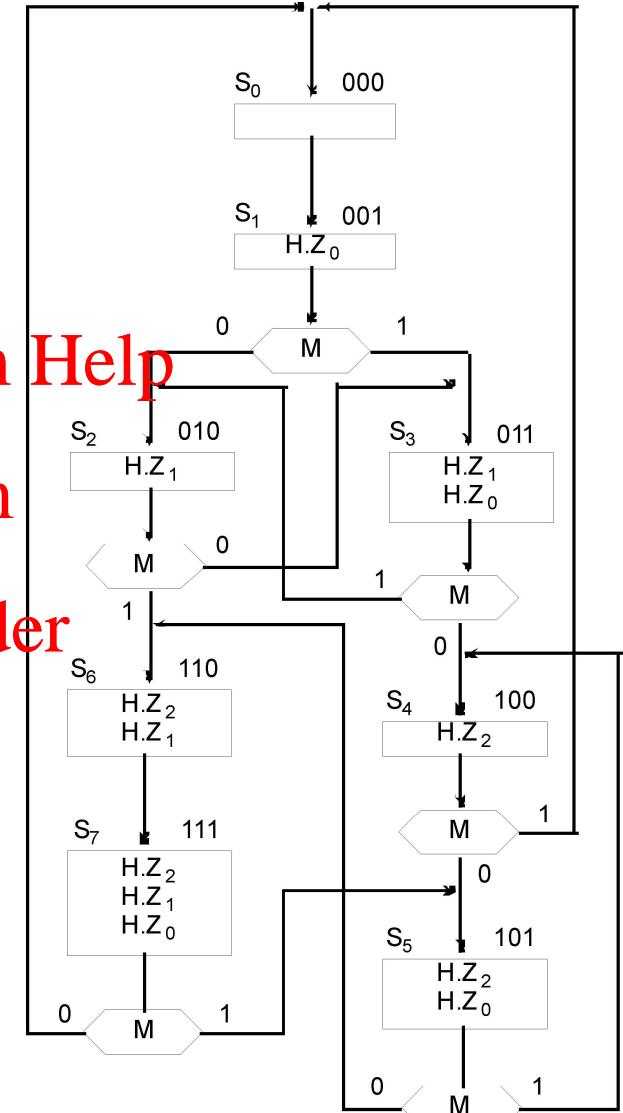
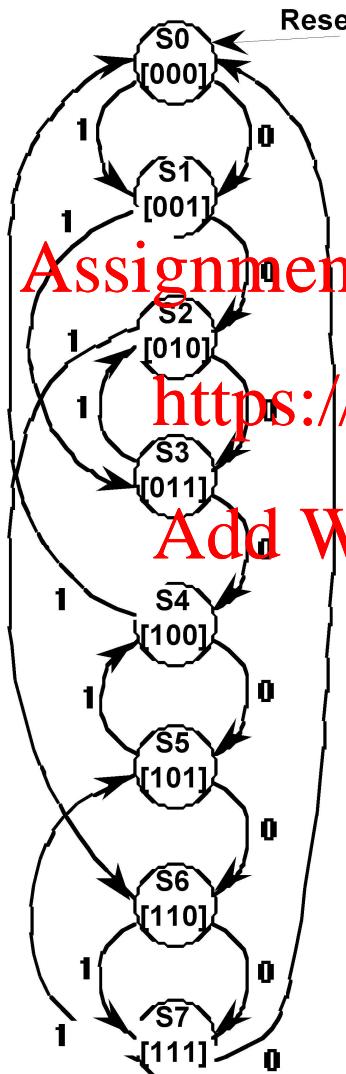
Mode Input M	Current State	Next State (Z2 Z1 Z0)
0	000	001
0	001	010
1	010	110
1	110	111
1	111	101
0	101	110
0	110	111

Add WeChat powcoder

Complex Counter

One state for each output combination

Add appropriate arcs for the mode control

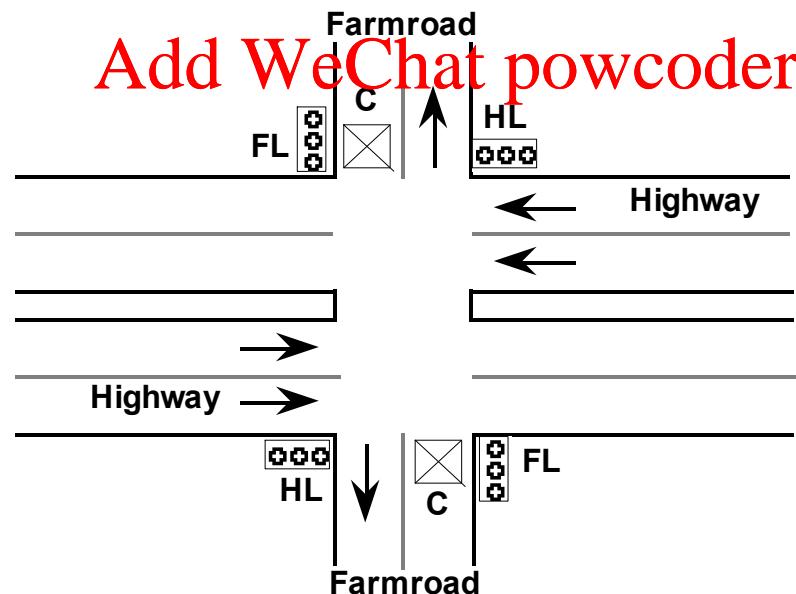


Traffic Light Controller

A busy highway is intersected by a little used farmroad. Detectors C sense the presence of cars waiting on the farmroad. With no car on farmroad, lights remain green in highway direction. If vehicle on farmroad, highway lights go from Green to Yellow to Red, allowing the farmroad lights to become green. These stay green only as long as a farmroad car is detected but never longer than a set interval. When these are met, farm lights transition from Green to Yellow to Red, allowing highway to return to green. Even if farmroad vehicles are waiting, highway gets at least a set interval as green.

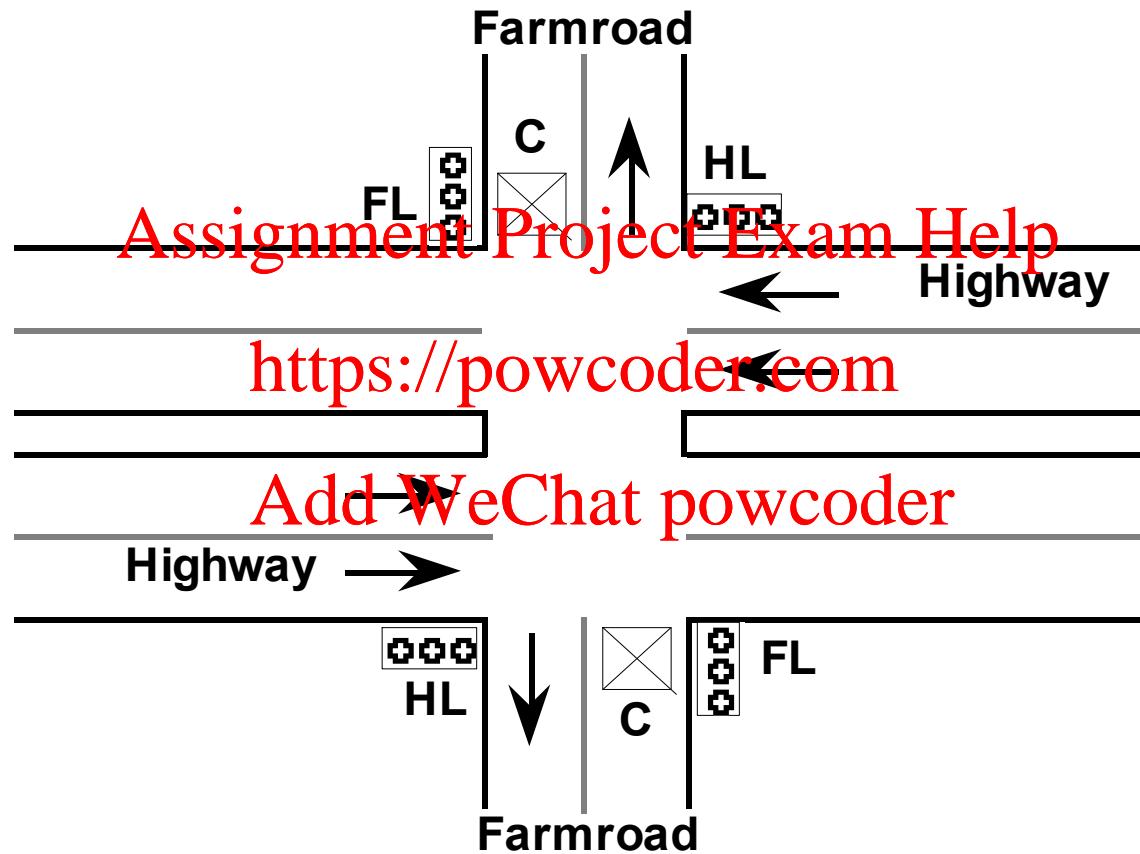
Assignment Project Exam Help

Assume you have an interval timer that generates a short time pulse (TS) and a long time pulse (TL) in response to a set (ST) signal. TS is to be used for timing yellow lights and TL for green lights.



Traffic Light Controller

Picture of Highway/Farmroad Intersection:



Traffic Light Controller

- Tabulation of Inputs and Outputs:

Input Signal

reset
C
TS
TL

Description

place FSM in initial state
detect vehicle on farmroad
short time interval expired
long time interval expired

Output Signal

HG, HY, HR
FG, FY, FR
ST

Description

assert green/yellow/red highway lights
assert green/yellow/red farmroad lights
start timing a short or long interval

Assignment Project Exam Help

<https://powcoder.com>

- Tabulation of Unique States: Some light configuration imply others

State

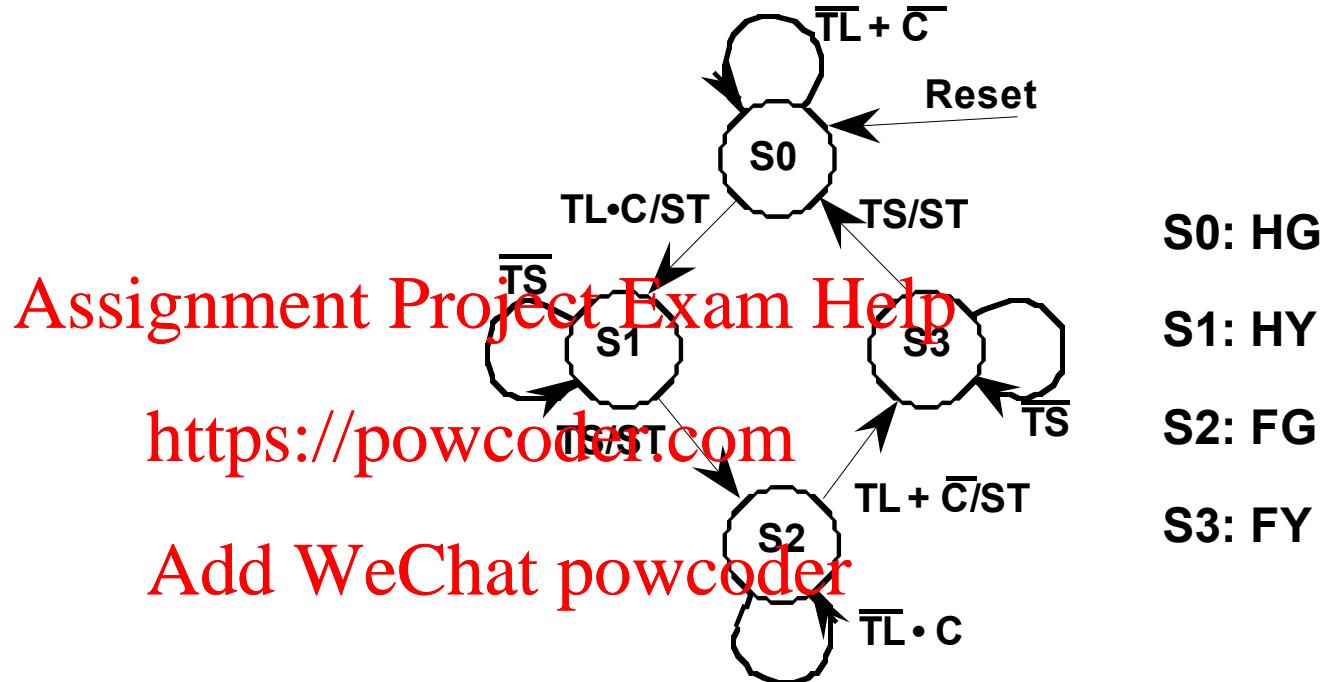
S0
S1
S2
S3

Description

Highway green (farmroad red)
Highway yellow (farmroad red)
Farmroad green (highway red)
Farmroad yellow (highway red)

Traffic Light Controller

Compare with state diagram:



Advantages of State Charts:

- Concentrates on paths and conditions for exiting a state
- Exit conditions built up incrementally, later combined into single Boolean condition for exit
- Easier to understand the design as an algorithm

Digital Combination Lock

"3 bit serial lock controls entry to locked room. Inputs are RESET, ENTER, 2 position switch for bit of key data. Locks generates an UNLOCK signal when key matches internal combination. ERROR light illuminated if key does not match combination. Sequence is: (1) Press RESET, (2) enter key bit, (3) Press ENTER, (4) repeat (2) & (3) two more times."

Assignment Project Exam Help

Problem specification is incomplete:

- how do you set the internal combination?
- exactly when is the ERROR light asserted?

Add WeChat powcoder

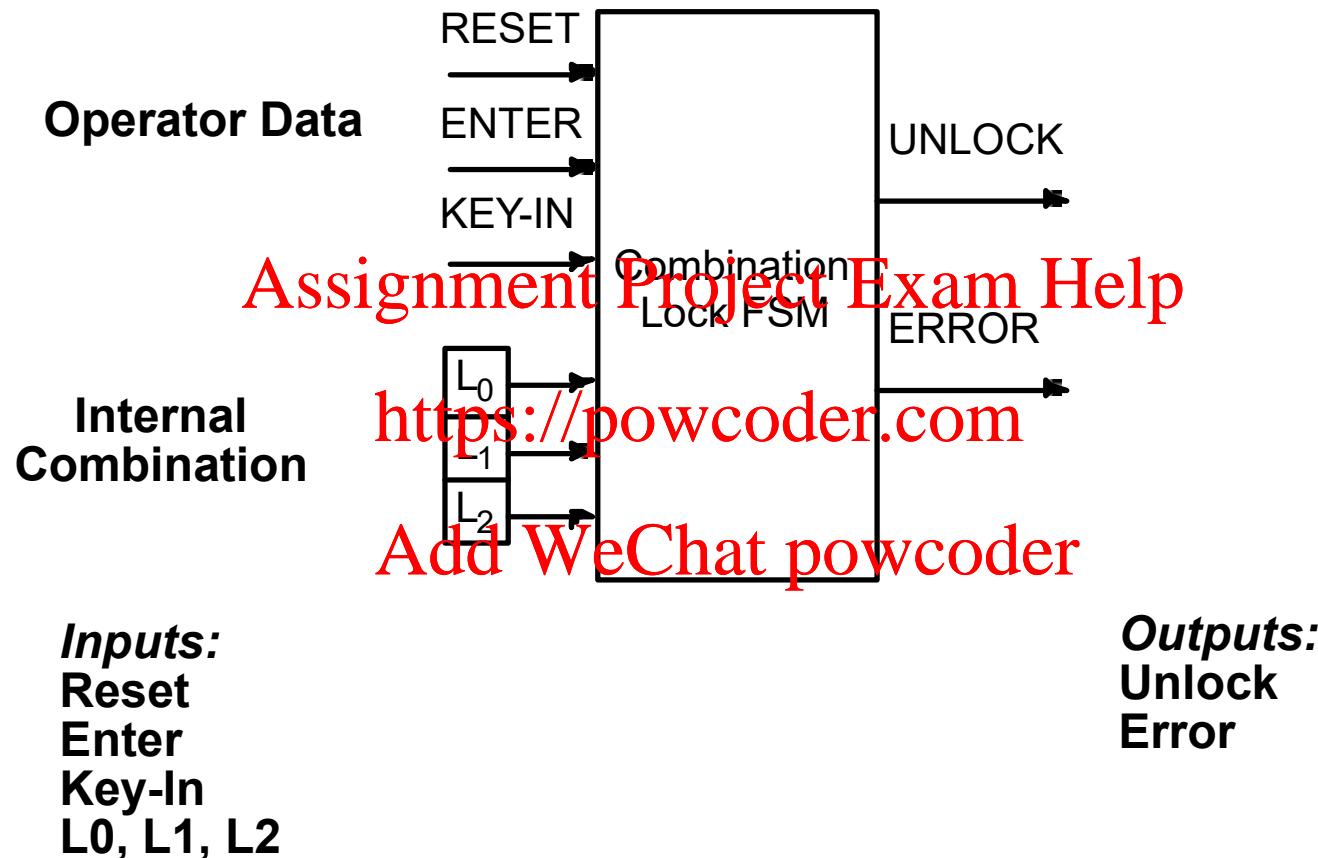
Make reasonable assumptions:

- hardwired into next state logic vs. stored in internal register
- assert as soon as error is detected vs. wait until full combination has been entered

Our design: registered combination plus error after full combination

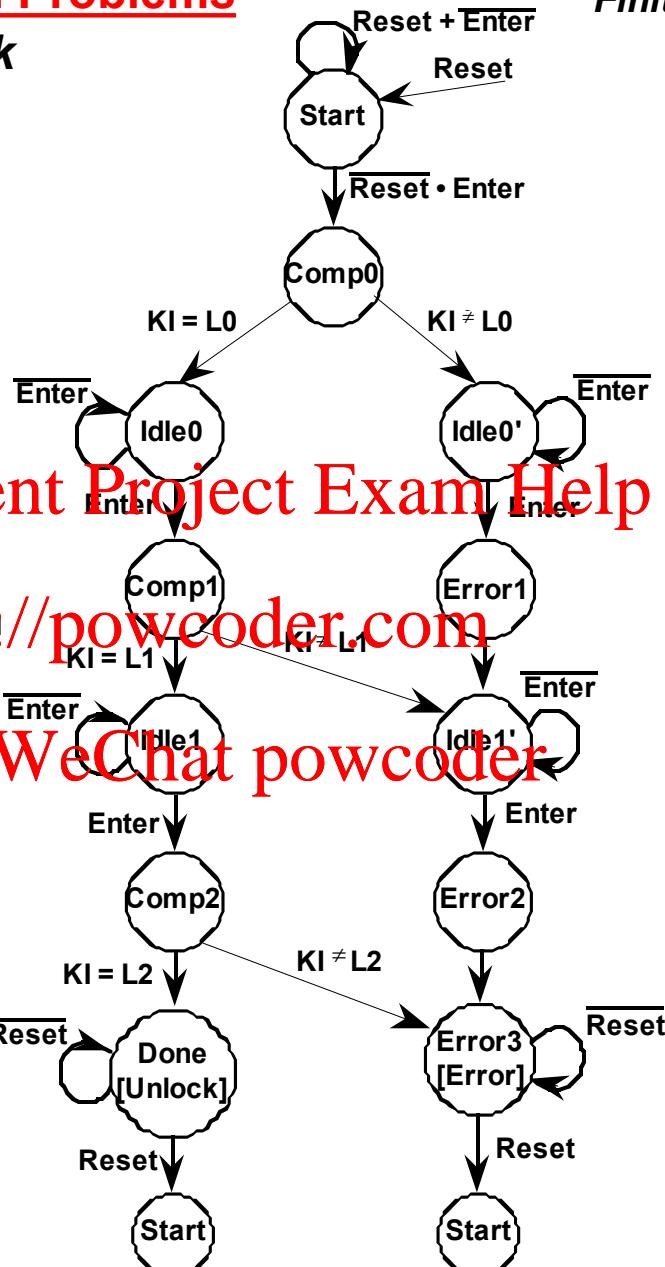
Digital Combination Lock

Understanding the problem: draw a block diagram ...



Finite State Machine Word Problems

Digital Combination Lock



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Equivalent State Diagram

Basic Timing Behavior an FSM

- when are inputs sampled, next state/outputs transition and stabilize
- Moore and Mealy (Async and Sync) machine organizations
outputs = F(state) vs. outputs = F(state, inputs)

First Two Steps of the Six Step Procedure for FSM Design

Assignment Project Exam Help

- understanding the problem
- abstract representation of the FSM
<https://powcoder.com>

Abstract Representations of an FSM

Add WeChat powcoder

- ASM Charts, Hardware Description Languages

Word Problems

- understand I/O behavior; draw diagrams
- enumerate states for the "goal"; expand with error conditions
- reuse states whenever possible