

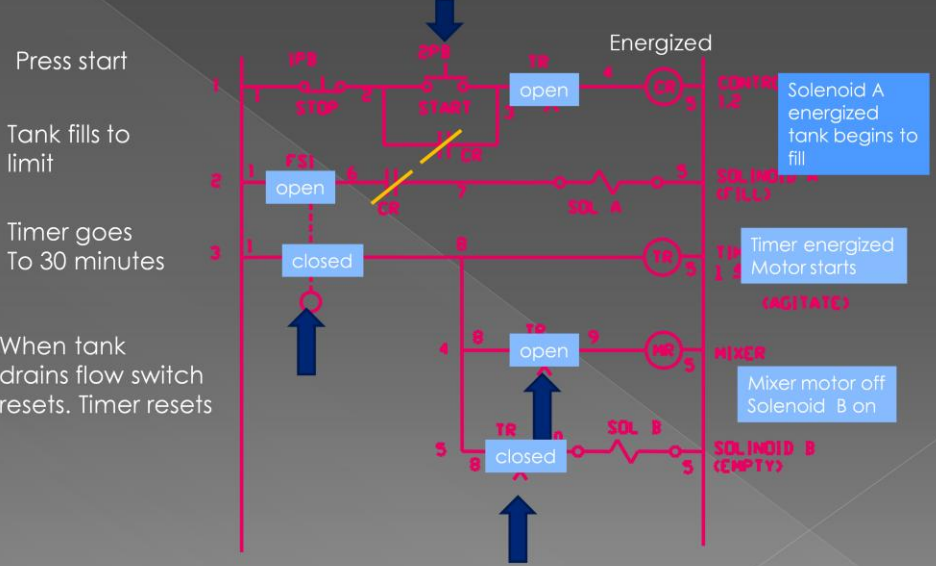
Ladder Diagram Example

A manual mixing operation is to be automated using sequential process control methods. The process composed of three steps:

- a.) filling a tank to a predetermined level
- b.) agitating the liquid for 30 minutes
- c.) draining the tank for use in another part of process

Does the ladder logic schematic that follows perform this function correctly?

Ladder Diagram Example



Press start


Tank fills to limit


Timer goes To 30 minutes

When tank drains flow switch resets. Timer resets

Combinational and Sequential Logic with Relays and Contacts

Let contact state represent a logical value

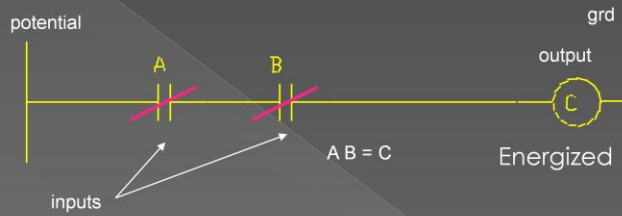
 = Logic 0 Called Form A Contact

 = Logic 1 Called Form B Contact

Implement AND gate



Combinational and Sequential Logic with Relays and Contacts



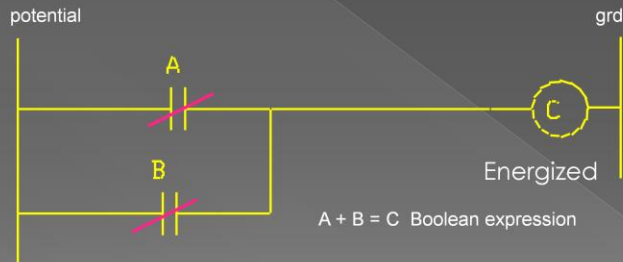
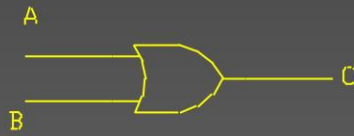
Conditions A **AND** B must be present to energize output C

Note: all contacts are considered instantaneous and not held unless modified

With electromechanical relays fan-in and fan-out limited by number of contacts in relays

More Logic Functions

OR
Function



Either A **OR** B will cause coil C to be energized
Contacts A, B represent conditions or states in
the sequential process

More Logic Functions

NOT Function



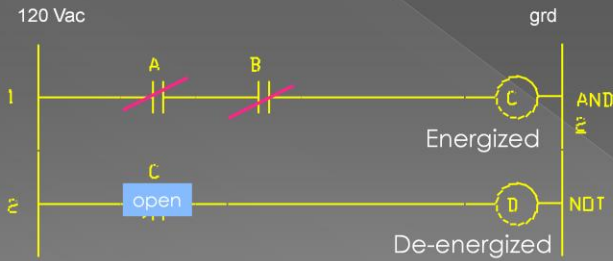
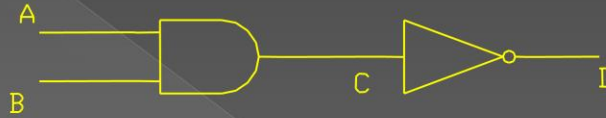
Boolean Expression
 $B = \overline{A}$



Contact of opposite state creates inversion

Constructing Other Logic Functions

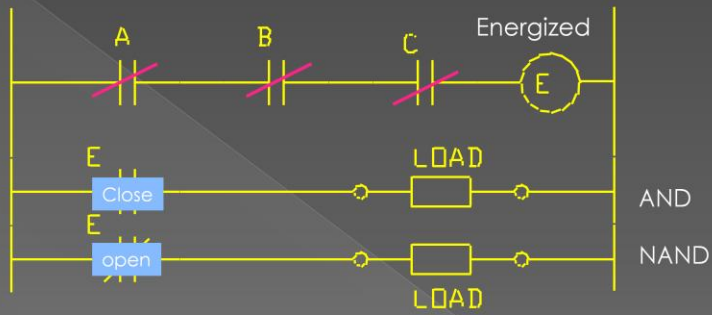
Combine the **AND** function with the **NOT** function to get a **NAND** operation.



Rung 1 implements the AND function
Rung 2 implements the NOT function

Any contact associated with coil D will change state like a NAND TTL gate.

Multiple Input AND/NAND

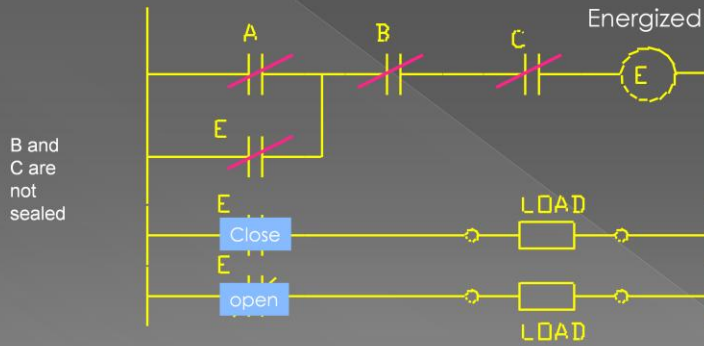


$$ABC = E \text{ and } \overline{ABC} = E$$

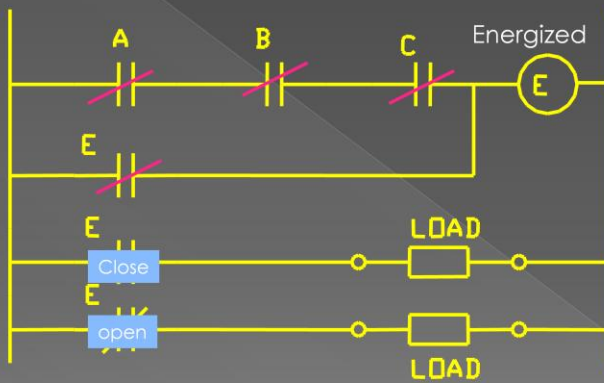
Can add a memory action to the above by including a feedback from the output coil to the inputs

Memory Action AND/NAND

Can add a memory action to the above by including a feedback from the output coil to the inputs



All Inputs Latched AND/NAND

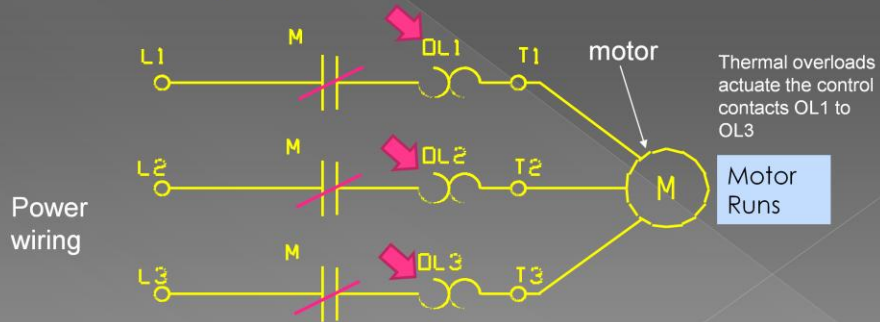
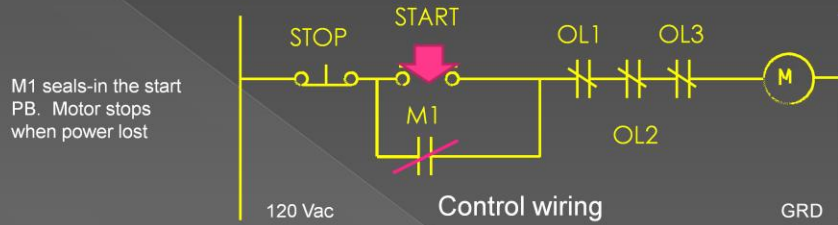


The output can not change unless the circuit is de-energized.

Contact E in rung 2 is a feedback from the output that makes circuit ignore state changes of A, B and C after the condition A B C is detected.

Motor Control Example

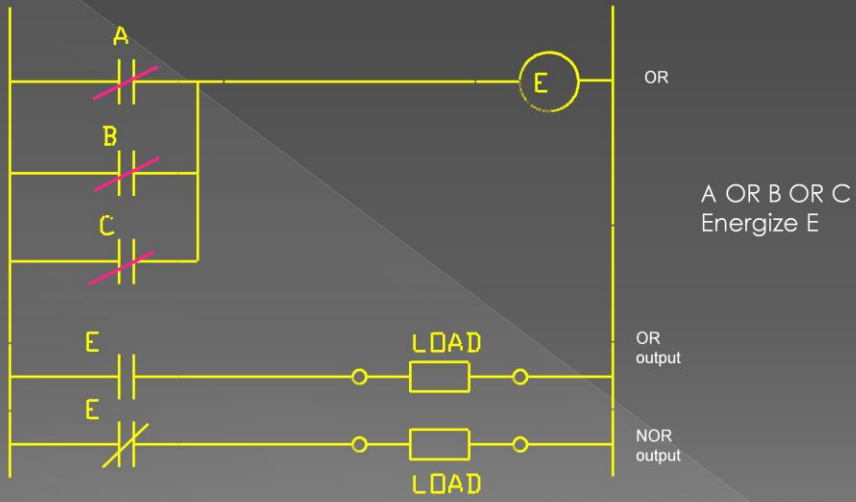
Three-wire control- used for manual and automatic motor starting.



e1438b-7.pptx

11

Multiple Input OR/NOR Function



OR

A OR B OR C
Energize E

OR
output

NOR
output

Outputs $A+B+C = E$

 $A+B+C = E$

Notice that Relay logic is similar to TTL. Can use Truth tables and Boolean expressions to do designs

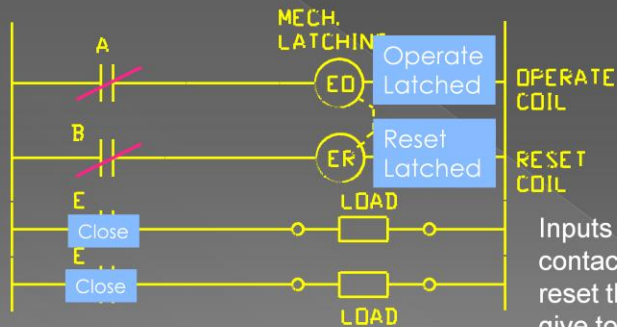
ef438b-7.pptx

12

Ladder Logic Memory Elements

Mechanically latched relay - maintains state even when power removed.
Has two coils (operate, reset)

Typical wiring



Inputs A and B set the output contacts E and reset then respectively. This give toggle action that "remembers" the last input state even when power is removed

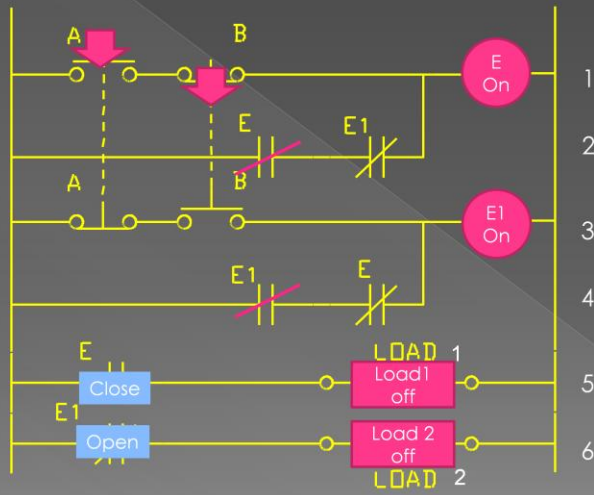
Typical Applications

Reversing Motor starters. Reclose Relay Cut-out

et438b-7.pptx

13

Off-Return Memory



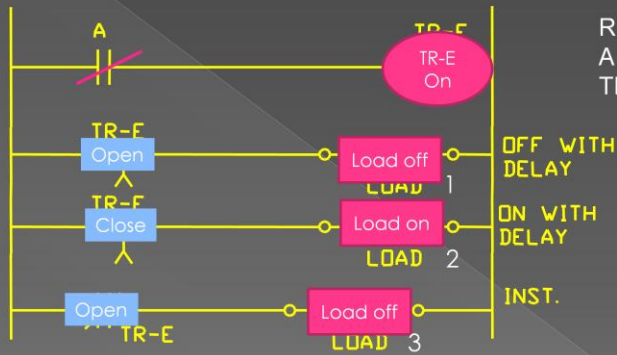
Energize and re-energize circuit - Load 2 on
 No continuity in rungs 1-4
 Continuity in rung 6

Press A: continuity rung 2 Both loads on

Press B: continuity rung 4 Both loads off

Remember all contacts are drawn with the coils de-energized

Timer Sub-Circuits



Rung 1: when input A is energized timer TR-E starts

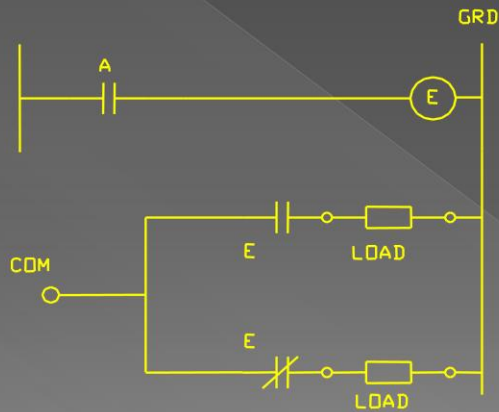
Schematic indicates that this is a on-delay timer. After defined interval TR-E in rung 2 opens and TR-E in rung 3 closes

- Load 1 is deactivated after time delay
- Load 2 is activated after time delay

Load 3 is instantaneously deactivated by TR-E

Form "C" Contact

Loads are toggled between a common point



Typical "Form C" contacts include both a NO and NC contact arrangement.

Used in some sensors for more flexibility

Contact A creates a remote control toggle switch

Designing Sequential Control Systems

Combinational Systems

- Use true tables, Boolean Algebra
- Multiple inputs and/or outputs
- Sum of Products or product of sums Boolean Implementations
- Reduce to minimum implementation




Sequential Systems




- Follow steps, transition from one step to another.
- Use state transition diagrams or tables with Boolean Algebra
- State Machine implemented in software or hardware
- Decisions made base on current condition of system and input information

Review of Logic Gates and Boolean Algebra

Boolean Variables False = 0 True = 1

Boolean Operators

AND			OR			NOT	
							
$X = A \cdot B$			$X = A + B$			$X = \bar{A}$	
A	B	X	A	B	X	A	X
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

NAND			NOR			EOR		
								
$X = \overline{A \cdot B}$			$X = \overline{A + B}$			$X = A \oplus B$		
A	B	X	A	B	X	A	B	X
0	0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1
1	1	0	1	1	0	1	1	0

EOR=XOR

Alternate Implementation

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

Review of Logic Gates and Boolean Algebra

Axioms of Boolean Algebra

Idempotent

$$A + A = A$$

$$A \cdot A = A$$

Associative

$$(A + B) + C = A + (B + C)$$

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Distributive

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

Identity

$$A + 0 = A$$

$$A + 1 = 1$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

Complement

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

$$\overline{(\bar{A})} = A$$

$$\bar{\bar{1}} = 0$$

DeMorgan's Theorem

$$\overline{(A + B)} = \bar{A} \cdot \bar{B}$$

$$\overline{(A \cdot B)} = \bar{A} + \bar{B}$$

Absorption

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

$$A + A \cdot B = A$$

Order of Operations

1. NOT
2. AND
3. OR

Review of Logic Gates and Boolean Algebra

Example: Simplify the following expression using the axioms of Boolean Algebra.

$$X = \overline{(A + B \cdot C)} + A \cdot (B + \overline{C})$$

$$X = \overline{(A)} + \overline{(B \cdot C)} + A \cdot (B + \overline{C})$$

Add
Parentheses

Apply DeMorgan's Theorem to first term

$$\overline{A} \cdot \overline{(B \cdot C)} = \overline{(A)} + \overline{(B \cdot C)}$$

Apply
DeMorgan's
Here

$$X = \overline{A} \cdot \overline{(B \cdot C)} + A \cdot (B + \overline{C})$$

$$X = \overline{A} \cdot (\overline{B} + \overline{C}) + A \cdot (B + \overline{C})$$

$$(\overline{B \cdot C}) = (\overline{B} + \overline{C})$$

$$X = \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C} + A \cdot B + A \cdot \overline{C}$$

Collect common terms and factor

Expand
Expressions

$$\overline{C} \cdot (A + \overline{A}) = A \cdot \overline{C} + \overline{A} \cdot \overline{C}$$

Review of Logic Gates and Boolean Algebra

Example Continued

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

Use
Complement
Axiom

$$A + \bar{A} = 1$$

$$X = \bar{A} \cdot \bar{B} + A \cdot B + \bar{C} \cdot 1$$

Use Identity
Axiom

$$\bar{C} \cdot 1 = \bar{C}$$

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

Simplified Expression

Logic Design

- 1.) Obtain description of process
- 2.) Define control action
- 3.) Define Inputs and Outputs
- 4.) Develop Truth Table or Boolean Equation of Process

Process control description

A heating oven with two bays can heat one ingot in each bay. When the heater is on it provides enough heat for two ingots. If only one ingot is present, the oven may overheat so a fan is used to cool the oven when it exceeds a set temperature.

Control Action

When only one ingot is in the oven and the temperature exceeds the setpoint, turn on the fan

Logic Design

Define I/O variables Inputs: B1 = bay1 ingot present
B2 = bay2 ingot present
T = temperature sensor

Create Truth Table

Output: F= fan start

T	B2	B1	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

If there is no over temperature don't start the fan

Over temperature in empty oven: safety fan start

Start fan in lightly load ovens with over temp.

Over temperature in full oven: safety fan start

Logic Design

Select elements from truth table in SOP (sum-of-products) form then simplify.

T	B2	B1	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$F = T \cdot \overline{B1} \cdot \overline{B2} + T \cdot B1 \cdot \overline{B2} + T \cdot \overline{B1} \cdot B2 + T \cdot B1 \cdot B2$$

$$F = T \cdot (\overline{B1} \cdot \overline{B2} + B1 \cdot \overline{B2} + \overline{B1} \cdot B2 + B1 \cdot B2)$$

$$F = T \cdot (\overline{B2} \cdot (\overline{B1} + B1) + B2 \cdot (\overline{B1} + B1))$$

$$F = T \cdot (\overline{B2} + B2)$$

$$F = T$$

Requires only Temp control

Ignore unloaded and full load cases and try again

Logic Design

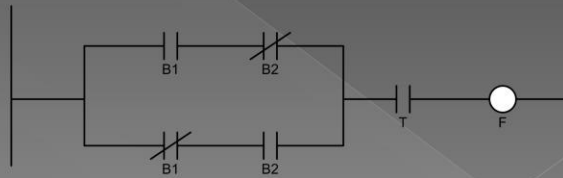
$$F = T \cdot B1 \cdot \overline{B2} + T \cdot \overline{B1} \cdot B2$$

$$F = T \cdot (B1 \cdot \overline{B2} + \overline{B1} \cdot B2)$$

Revised Truth Table

T	B2	B1	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	<u>0</u>
1	0	1	1
1	1	0	1
1	1	1	0

Ladder Logic Representation

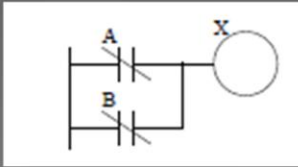


Simplified Forms of Functions

Avoid multiple complemented variables in ladder logic (No NAND, NOR)

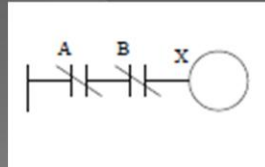
NAND

$$X = \overline{A \cdot B} = \overline{A} + \overline{B}$$



NOR

$$X = \overline{A + B} = \overline{A} \cdot \overline{B}$$



NAND/NOR can not be implemented effectively using software.
(Programmable Logic Controllers)

State-Based Designs

Definitions

State - current operational mode of system

Examples: On/Off, Idle, Tank filling, dispensing product.

Conditions (inputs) - inputs required for leaving the current state and moving to another state

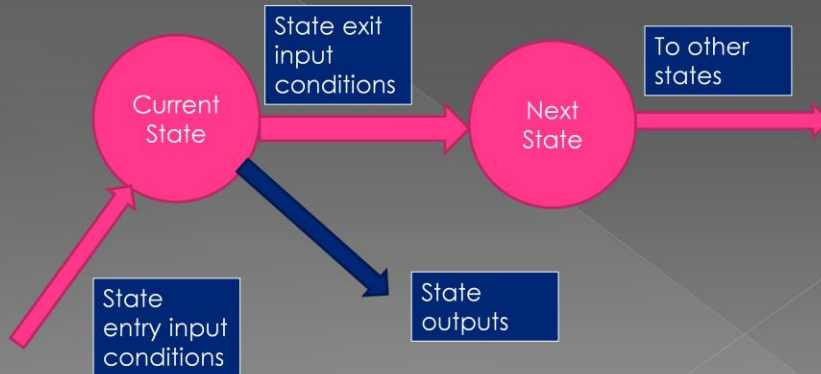
Examples: Coins inserted, button pressed, OL activated

Actions (outputs) - actions performed by system when the transition from one state to another take place

Examples: Start motor, turn on light, sound alarm.

State-Based Designs

When a set of inputs (conditions) become valid for leaving a state, the system is directed to the destination state

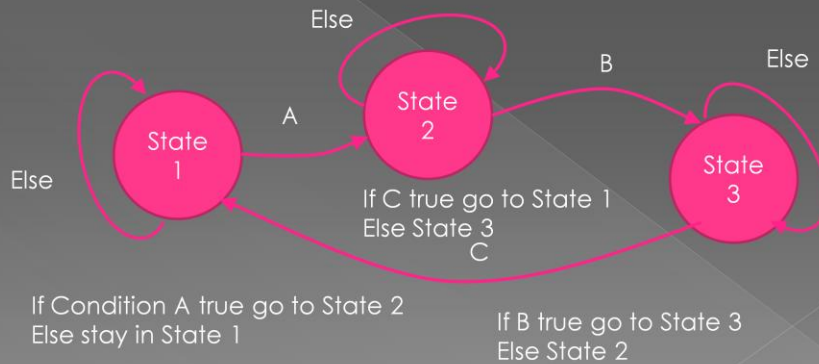


e1438b-7.pptx

28

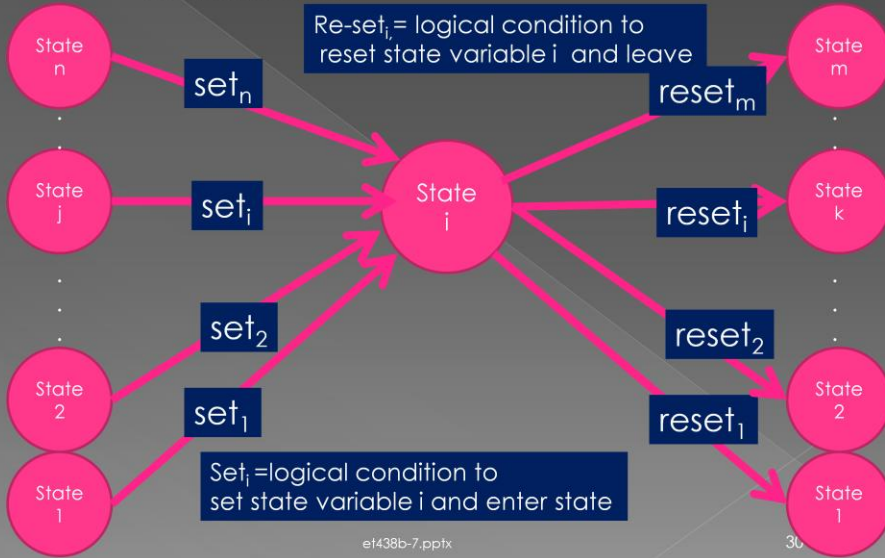
State Transition Diagrams

State transition diagrams allow designers to examine the interaction between desired conditions and find their logical relationships and sequence. Use in digital computer design



State Equations

Informal: State X = (State X + Arrival from another state) and has not left for another state



State Equations

Formal Definition:

$$\text{state}_i^{+1} = \left(\text{state}_i + \sum_{j=1}^n (\text{set}(\text{state}_j, I_i)) \right) \bullet \sum_{k=1}^m (\overline{\text{reset}(\text{state}_i, I_k)})$$

$$\text{out}_i = h_i(\text{state}_1, \text{state}_2, \dots, \text{state}_N)$$

Where:

state_i = a variable that reflects state i is on

state_i^{+1} = next value of state variable

out_i = desired outputs of state i

$h_i(\)$ = output function of state variables

n = number of transitions **into** state i

m = number of transitions **out of** state i

N = total number of system states

set_i = logical condition to set state variable i

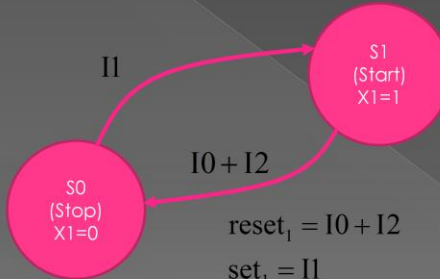
reset_i = logical condition to reset state variable i

Set
Conditions
Functions of
state and inputs

Reset
Conditions
Functions of
state and
inputs

Example

Write the state equation for a motor starting control described in the state diagram below with the following input and outputs



I0=pressed stop button (PB1)
I1= pressed start button (PB2)
I2 = motor overload condition (OL)

O1 = start motor (M)

Only 1 state variable required for two conditions $X1=0$ or $X1=1$

$$\text{reset}_1 = I0 + I2$$

$$\text{set}_1 = I1$$

$$X1^{+1} = (X1 + \text{set}_1) \cdot (\overline{\text{reset}_1})$$

$$X1^{+1} = (X1 + I1) \cdot (\overline{I0 + I2})$$

$$X1^{+1} = (X1 + I1) \cdot \overline{I0} \cdot \overline{I2}$$

Output equation

$$M = O1 = X1$$

Example

Boolean Equation to ladder logic diagram

$$X1^{+1} = (X1 + I1) \cdot \overline{I0} \cdot \overline{I2}$$

Substitute variable names

$$M = (M + PB2) \cdot \overline{PB1} \cdot \overline{OL}$$

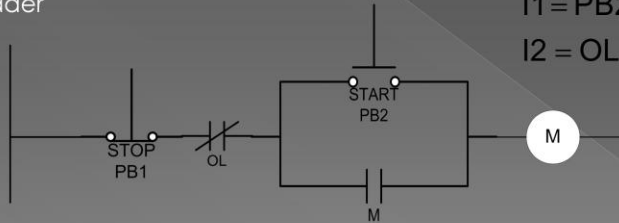
$$X1 = M$$

$$I0 = PB1 \text{ Stop}$$

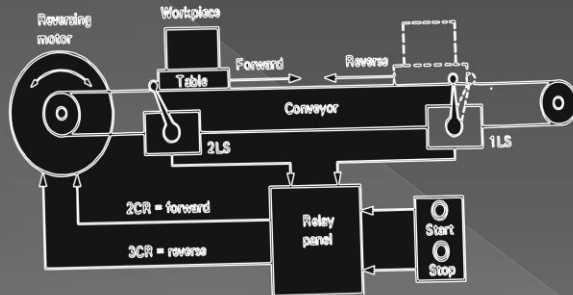
$$I1 = PB2 \text{ Start}$$

$$I2 = OL \text{ Overload}$$

Construct Ladder



Design Example: Reciprocating Motion Process



A work piece must travel back and forth on a conveyor. The location of the work piece is determined by two limit switches. When the location is detected control signals are sent to a reversing motor contactor. The machine is started and stopped from a local set of push button switches. Develop a ladder logic diagram to implement this control.

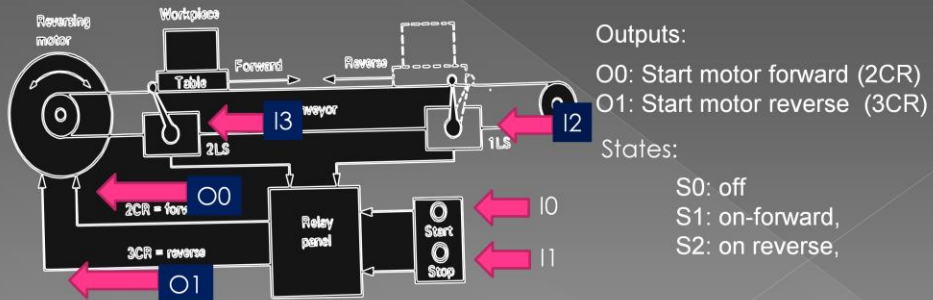
e1438b-7.pptx

34

Design Example: Reciprocating Motion Process

Determine the inputs, outputs and states of system

Inputs: I0: press start
I1: press stop
I2: Table at reverse limit (1LS)
I3: Table at forward limit (2LS)

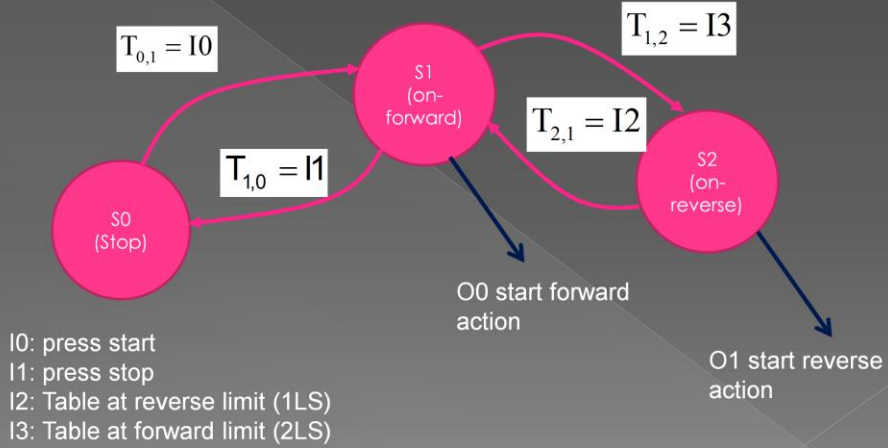


et438b-7.pptx

35

Design Example: Reciprocating Motion Process

Assume machine starts at reverse limit. (1LS changes state)



Design Example: Reciprocating Motion Process

Define set and reset conditions

Define 2 state variables X1 and X2

X2	X1	Condition
0	0	Off (S0)
0	1	On-Forward (S1)
1	0	On-Reverse (S2)
1	1	Not allowed

$$\text{set}_{x1} = I0 + I2 \cdot X2$$

$$\text{reset}_{x1} = I1 + I3$$

$$\text{set}_{x2} = I3 \cdot X1$$

$$\text{reset}_{x2} = I2$$

$$X1^{+1} = (X1 + \text{set}_{x1})(\overline{\text{reset}_{x1}})$$

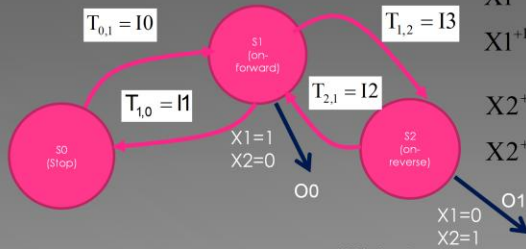
$$X1^{+1} = (X1 + (I0 + I2 \cdot X2))(\overline{I1 + I3})$$

$$X1^{+1} = (X1 + (I0 + I2 \cdot X2))(\overline{I1} \cdot \overline{I3})$$

$$X2^{+1} = (X2 + \text{set}_{x2})(\overline{\text{reset}_{x2}})$$

$$X2^{+1} = (X2 + I3 \cdot X1)(\overline{I2})$$

Outputs X1 = 00, X2 = 01



et438b-7.pptx

37

Design Example: Reciprocating Motion Process

Convert state equations into ladder diagram

$$2CR^{+1} = (2CR + \text{start} + 1LS \cdot 3CR) (\overline{\text{Stop}} \cdot \overline{2LS})$$

$$3CR^{+1} = (3CR + 2LS \cdot 2CR) \cdot 1LS$$

2CR = O0

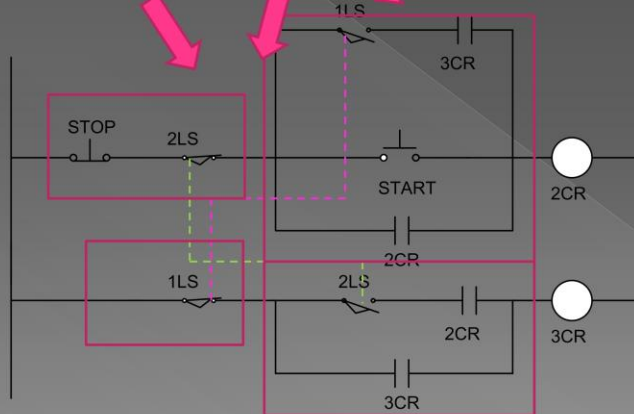
3CR = O1

I0=start

I1=stop

I2=1LS

I3=2LS

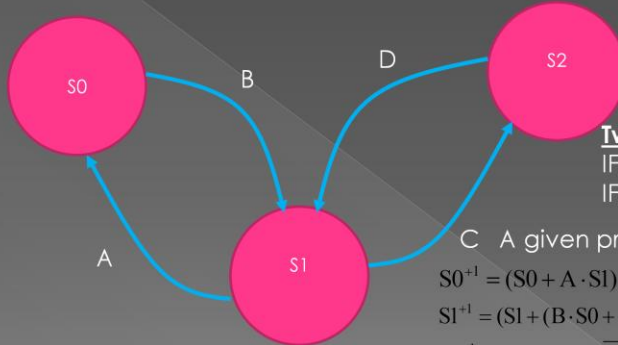


et438b-7.pptx

38

States With Prioritization

Systems with multiple entries and exits from a state require blocking of Alternatives.



Two Choices

IF A THEN block C
IF C THEN block A

C A given priority to C

$$S0^{+1} = (S0 + A \cdot S1) \cdot \overline{(B \cdot S0)}$$

$$S1^{+1} = (S1 + (B \cdot S0 + D \cdot S2)) \cdot \overline{(A \cdot S1 + C \cdot S1)}$$

$$S2^{+1} = (S2 + C \cdot S1 \cdot \overline{A}) \cdot \overline{(D \cdot S2)}$$

A or C can occur independently to exit S1.
Must give one transition priority over other.
Block setting of conflicting state

C over A

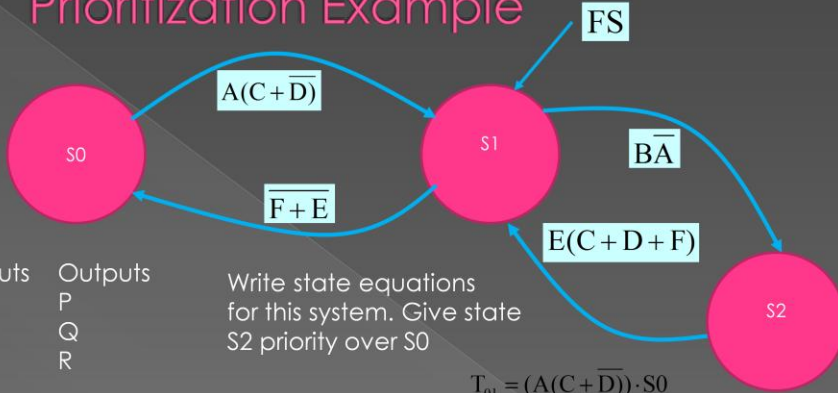
$$S0^{+1} = (S0 + A \cdot S1 \cdot \overline{C}) \cdot \overline{(B \cdot S0)}$$

$$S2^{+1} = (S2 + C \cdot S1) \cdot \overline{(D \cdot S2)}$$

e1438b-7.pptx

39

Prioritization Example



Inputs
A
B
C
D
E
F
FS

Outputs
P
Q
R

Write state equations for this system. Give state S2 priority over S0

Output Map

State	P	Q	R
S0	0	1	1
S1	1	0	1
S2	1	1	0

$$T_{01} = (A(C + \overline{D})) \cdot S0$$

$$T_{10} = (\overline{F + E}) \cdot S1$$

$$T_1 = FS$$

$$T_{12} = B \cdot \overline{A} \cdot S1$$

$$T_{21} = (E(C + D + F)) \cdot S2$$

Prioritization Example

Write state equations using transitions

$$S0^{+1} = (S0 + T_{10} \cdot \overline{T_{12}}) \cdot \overline{T_{21}}$$

S0 blocked if
S2 is active

$$S1^{+1} = (S1 + T_{01} + T_{21} + T_1) \cdot (\overline{T_{10}} + \overline{T_{12}})$$

$$S1^{+1} = (S1 + T_{01} + T_{21} + T_1) \cdot (\overline{T_{10}} \cdot \overline{T_{12}})$$

Simplify using
DeMorgan's Theorem

$$S2^{+1} = (S2 + T_{12}) \cdot \overline{T_{21}}$$

Output Map

State	P	Q	R
S0	0	1	1
S1	1	0	1
S2	1	1	0

Output Equations

$$P = S1 + S2$$

$$Q = S0 + S2$$

$$R = S0 + S1$$