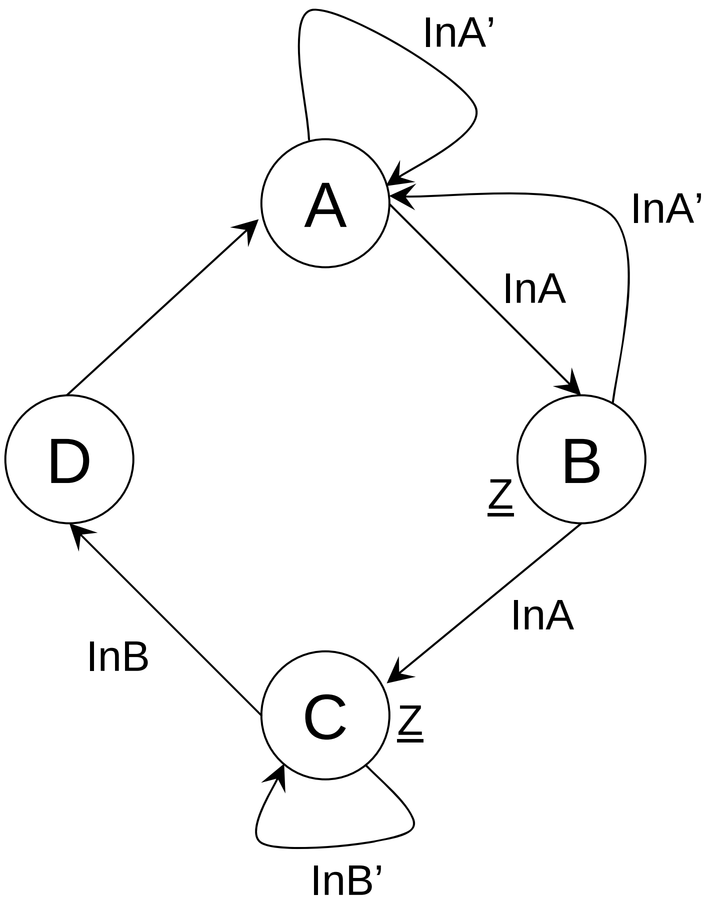
# One-Hot Encoded Finite State Machines

# Example State Machine

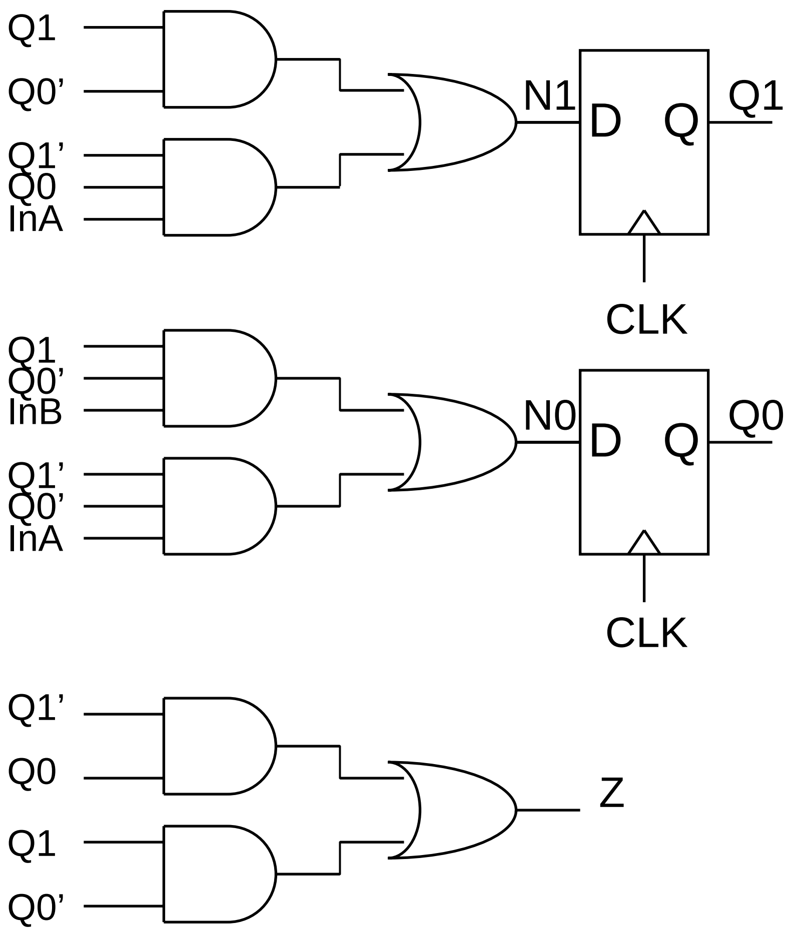



| InA | InB | CS | NS | Z |
|---|---|---|---|---|
| 0 | - | A | A | 0 |
| 1 | - | A | B | 0 |
| 0 | - | B | A | 1 |
| 1 | - | B | C | 1 |
| - | 0 | C | C | 1 |
| - | 1 | C | D | 1 |
| - | - | D | A | 0 |

State A = 00
State B = 01
State C = 10
State D = 11

| InA | InB | CS | NS | Z |
|---|---|---|---|---|
| 0 | - | 00 | 00 | 0 |
| 1 | - | 00 | 01 | 0 |
| 0 | - | 01 | 00 | 1 |
| 1 | - | 01 | 10 | 1 |
| - | 0 | 10 | 10 | 1 |
| - | 1 | 10 | 11 | 1 |
| - | - | 11 | 00 | 0 |

# Example Machine Implementation




**N1 = Q1•Q0' + Q1'•Q0•InA**
**N0 = Q1•Q0'•InB + Q1'•Q0'•InA**
**Z = Q1'•Q0 + Q1•Q0'**

9 gates
21 gate inputs

# Choose a Different Encoding

- A=1000, B=0100, C=0010, D=0001

| InA | InB | CS | NS | Z |
|---|---|---|---|---|
| 0 | - | 1000 | 1000 | 0 |
| 1 | - | 1000 | 0100 | 0 |
| 0 | - | 0100 | 1000 | 1 |
| 1 | - | 0100 | 0010 | 1 |
| - | 0 | 0010 | 0010 | 1 |
| - | 1 | 0010 | 0001 | 1 |
| - | - | 0001 | 1000 | 0 |

| InA | InB | CS | NS | Z |
|---|---|---|---|---|
| 0 | - | 1--- | 1000 | 0 |
| 1 | - | 1--- | 0100 | 0 |
| 0 | - | -1-- | 1000 | 1 |
| 1 | - | -1-- | 0010 | 1 |
| - | 0 | --1- | 0010 | 1 |
| - | 1 | --1- | 0001 | 1 |
| - | - | ---1 | 1000 | 0 |

This is called a *one-hot* encoding.

Only one state bit is on at a time

Because of the state encodings, there are many illegal states.

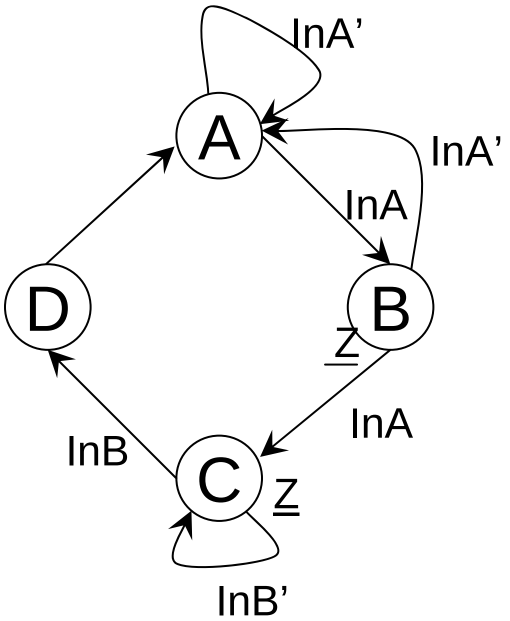
This TT with all these input don't cares is the result

# One-Hot Encoding Results

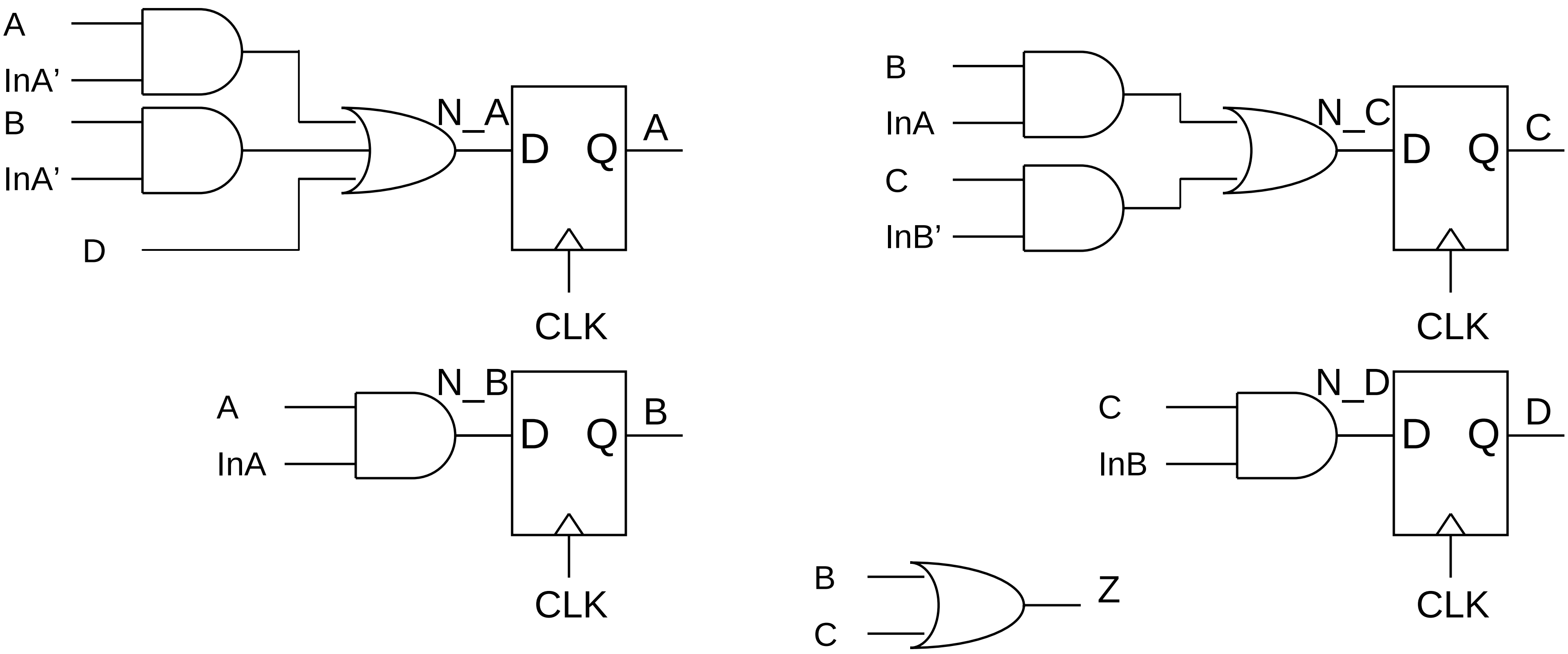
- Will require 4 flip flops
  - One per state
  - Call the current state bits A, B, C, and D
  - Call the next state bits NA, NB, NC, and ND

***By inspection we see:***

**NA = A•InA' + B•InA' + D**
**NB = A•InA**
**NC = B•InA + C•InB'**
**ND = C•InB**
**Z = B + C**



# One-Hot Implementation

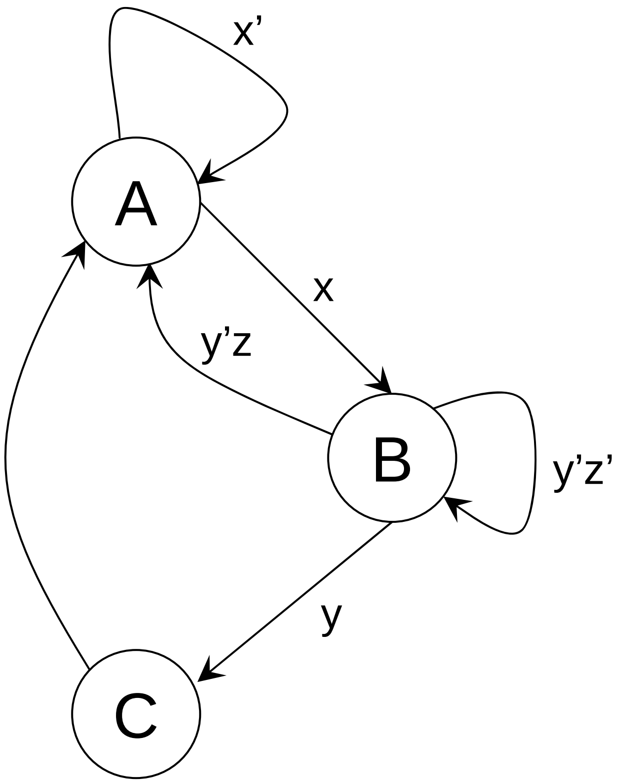



9 gates
19 gate inputs

# One-Hot - Observations

- Choosing a one-hot encoding results in many, many don't cares in transition table
- Minimization results in simpler IFL and OFL
- Can do one-hot design by inspection
  - ***without using transition tables…***

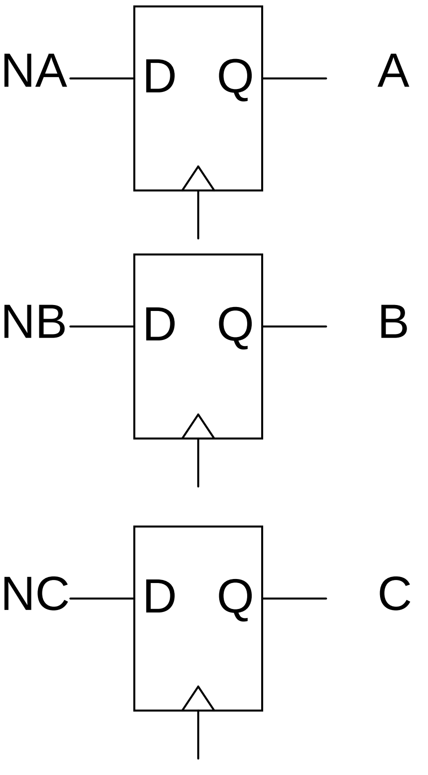
# Another One-Hot Example




| State | Encoding |
|---|---|
| A | 100 |
| B | 010 |
| C | 001 |

# State Encoding and Structure

| State | Encoding |
| --- | --- |
| A | 100 |
| B | 010 |
| C | 001 |

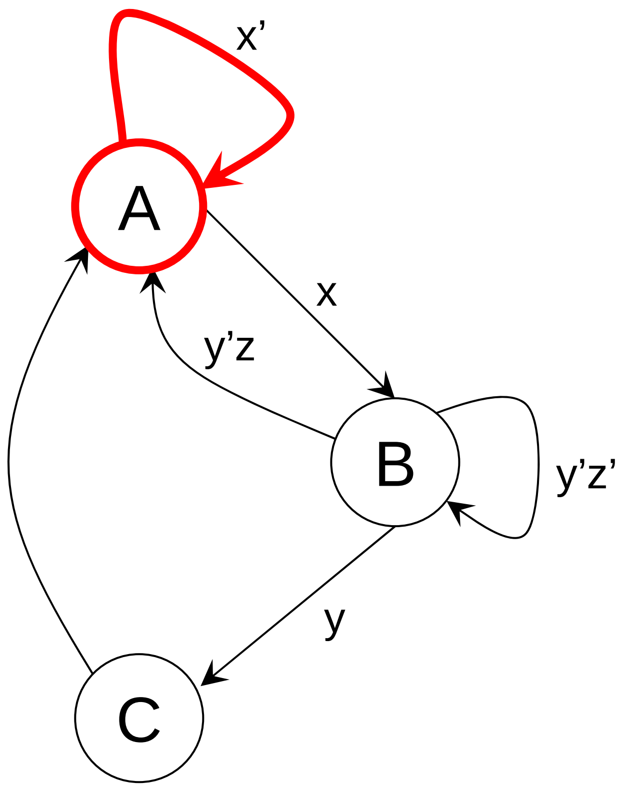



With one-hot encoding, each state has its own flip flop.

Note: 'A' is the name of a state. It is also the name of the wire coming out from the flip flop for state 'A'.

The same holds true for states 'B' and 'C'

# One-Hot Encodings By Inspection

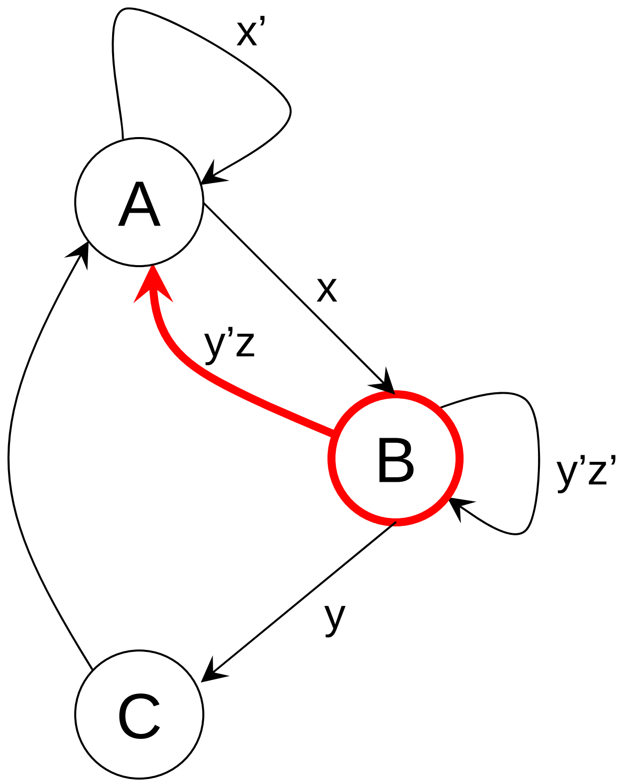



When is A the next state?

Look at the arcs entering state A

**NA = A•x' + B•y'z + C**

# One-Hot Encodings By Inspection

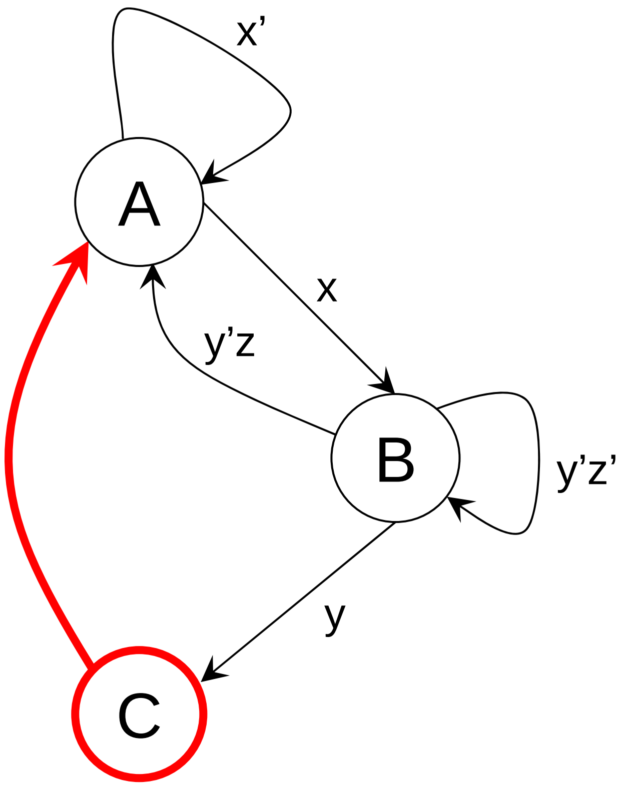



When is A the next state?

Look at the arcs entering state A

**NA = A•x’ + B•y’z + C**

# One-Hot Encodings By Inspection

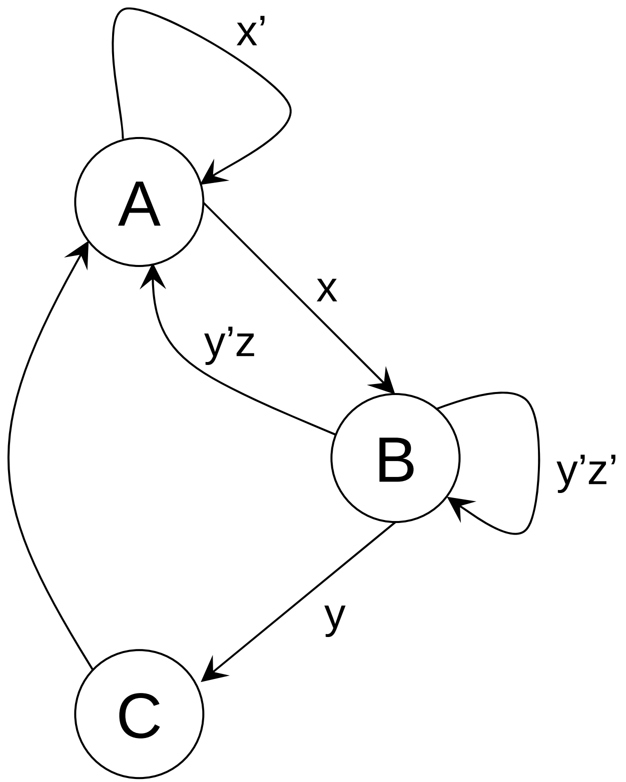



When is A the next state?

Look at the arcs entering state A

**NA = A•x' + B•y'z + C**

# One-Hot Encodings By Inspection




When is A the next state?

Look at the arcs entering state A

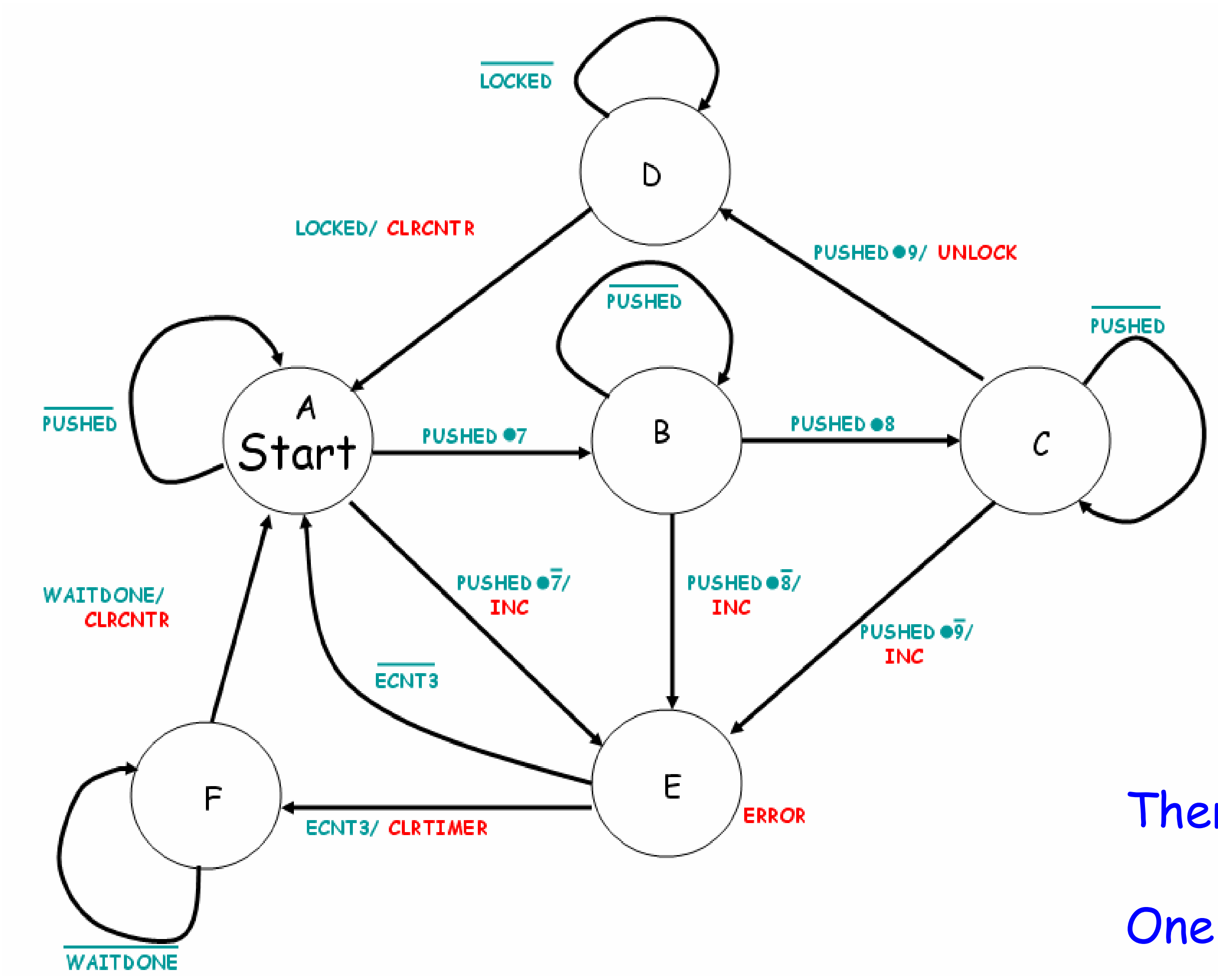
**NA = A•x' + B•y'z + C**

**Similar reasoning leads to:**

**NB = A•x + B•y'z'**

**NC = B•y**

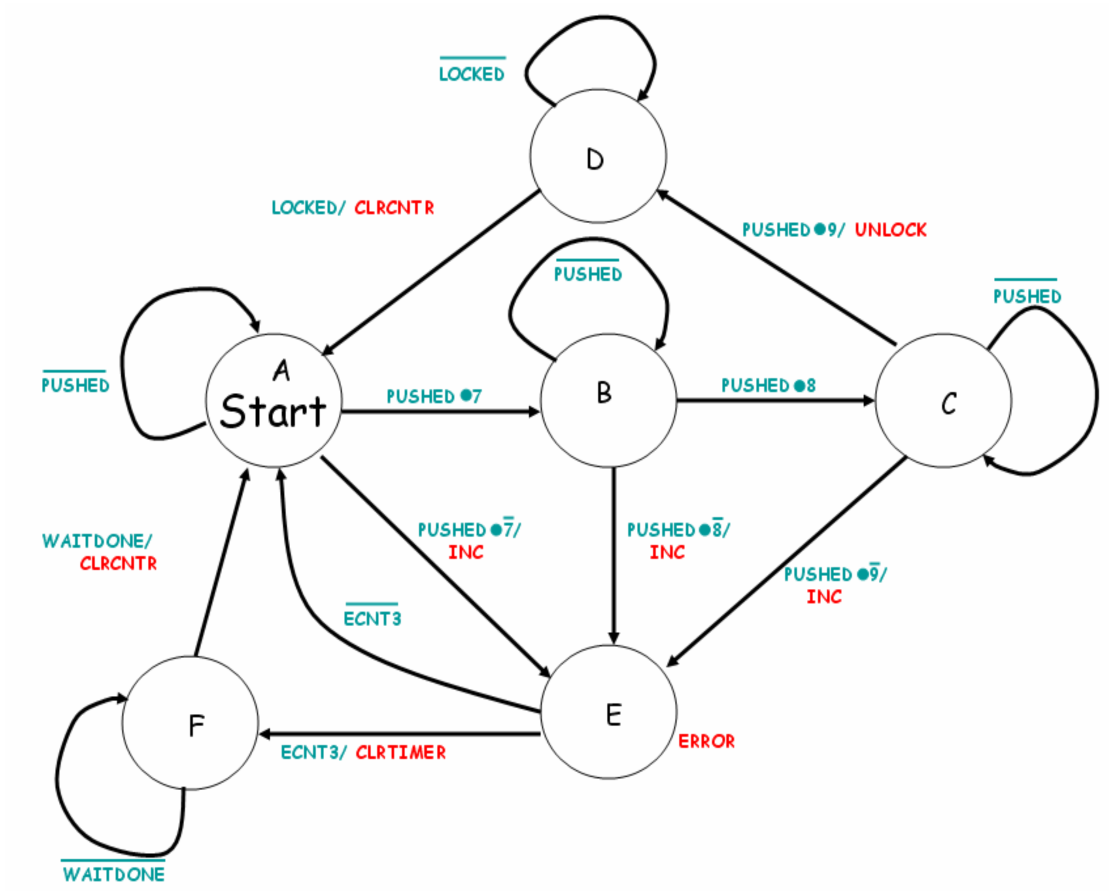
# The Key Lock Problem - One-Hot Version



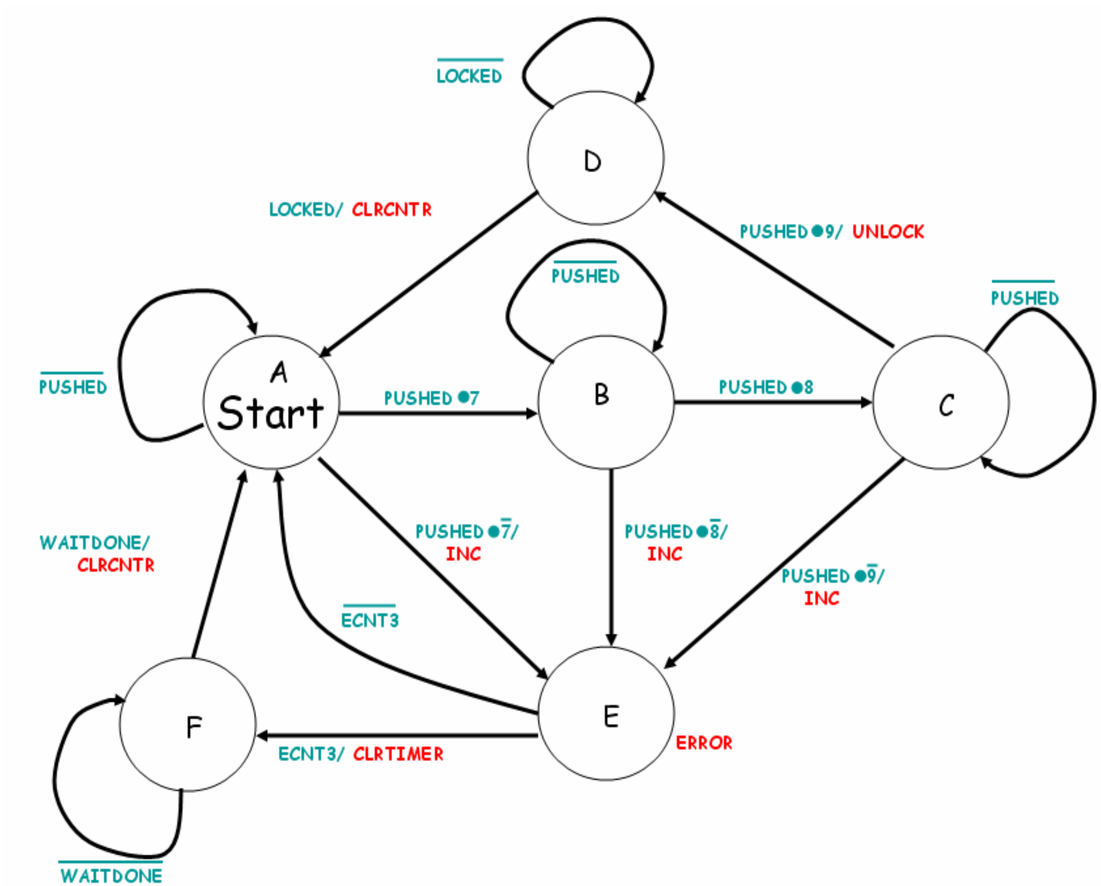

There will be 6 flip flops

One for each state

# Key Lock Input Forming Logic

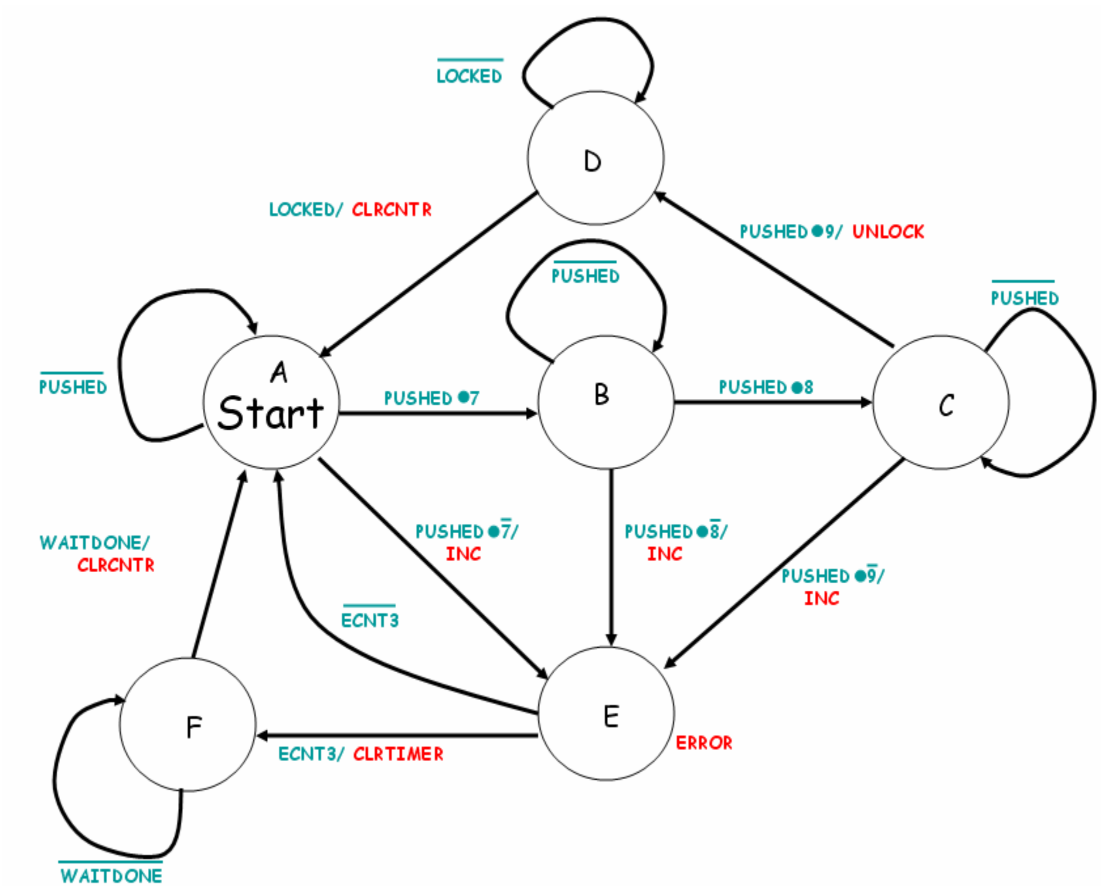


# Key Lock Input Forming Logic




A+ = (PUSHED'●A) + (ECNT3'● E)
+ (WAITDONE●F)
+ (LOCKED●D)

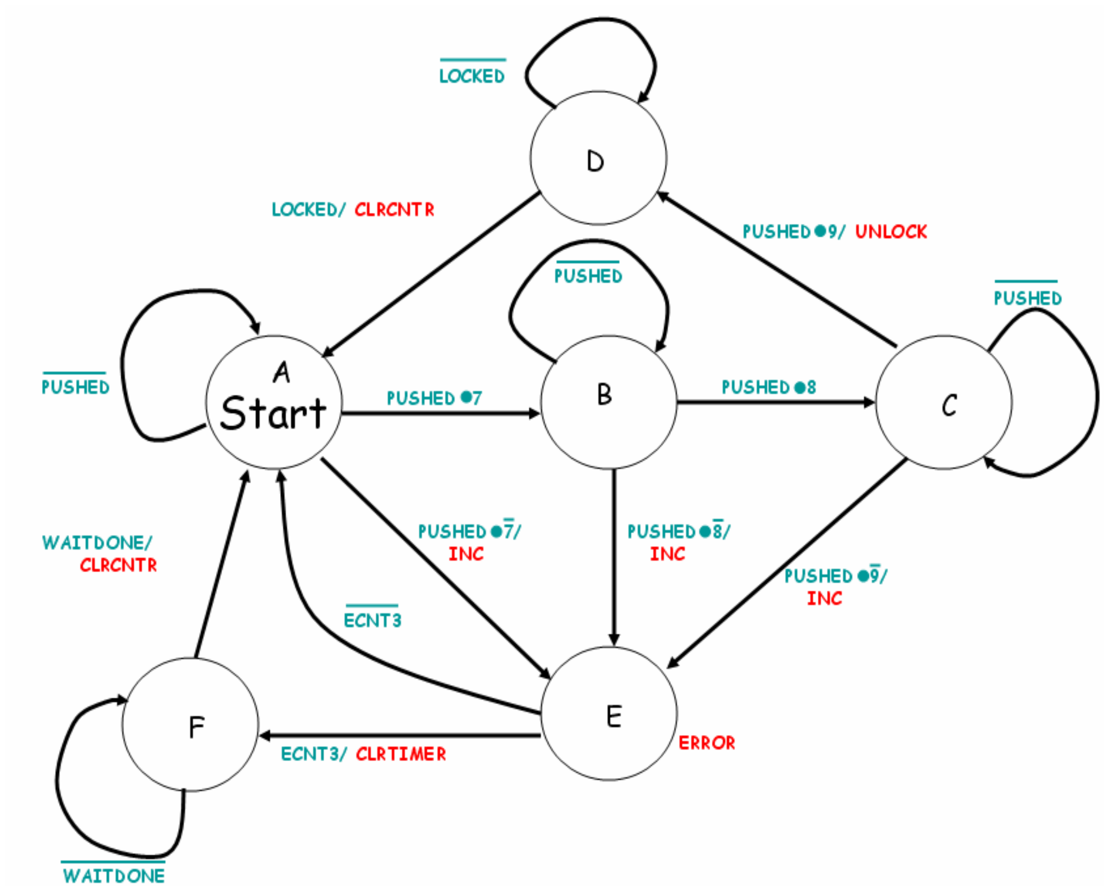
# Key Lock Input Forming Logic




A+ = (PUSHED'•A) + (ECNT3'•E)
+ (WAITDONE•F)
+ (LOCKED•D)

B+ = (PUSHED'•B)+(PUSHED•7•A)

# Key Lock Input Forming Logic

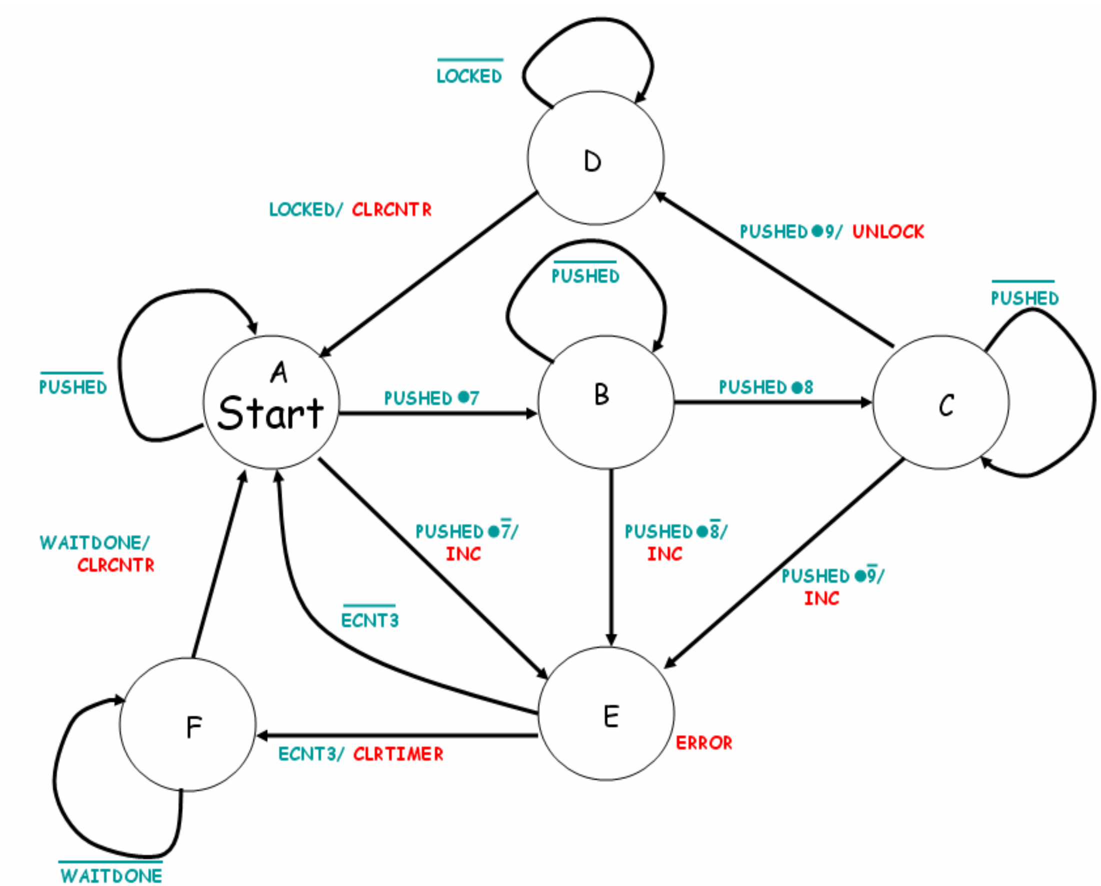



A+ = (PUSHED'•A) + (ECNT3'•E)
+ (WAITDONE•F)
+ (LOCKED•D)

B+ = (PUSHED'•B)+(PUSHED•7•A)

C+ = (PUSHED'•C)+(PUSHED•8•B)

# Key Lock Input Forming Logic



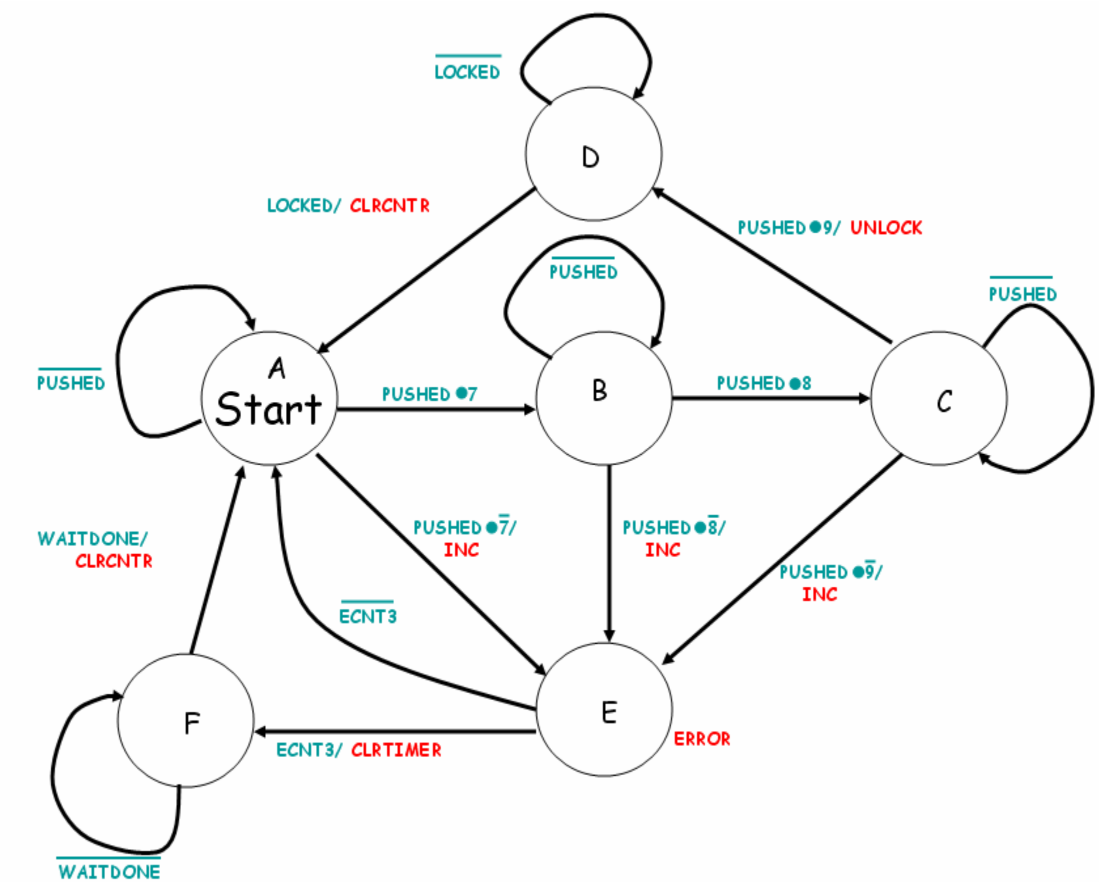

$$A+ = (PUSHED' \bullet A) + (ECNT3' \bullet E) + (WAITDONE \bullet F) + (LOCKED \bullet D)$$

$$B+ = (PUSHED' \bullet B) + (PUSHED \bullet 7 \bullet A)$$

$$C+ = (PUSHED' \bullet C) + (PUSHED \bullet 8 \bullet B)$$

$$D+ = (LOCKED' \bullet D) + (PUSHED \bullet 9 \bullet C)$$

# Key Lock Input Forming Logic




A+ = (PUSHED'•A) + (ECNT3'•E)
+ (WAITDONE•F)
+ (LOCKED•D)

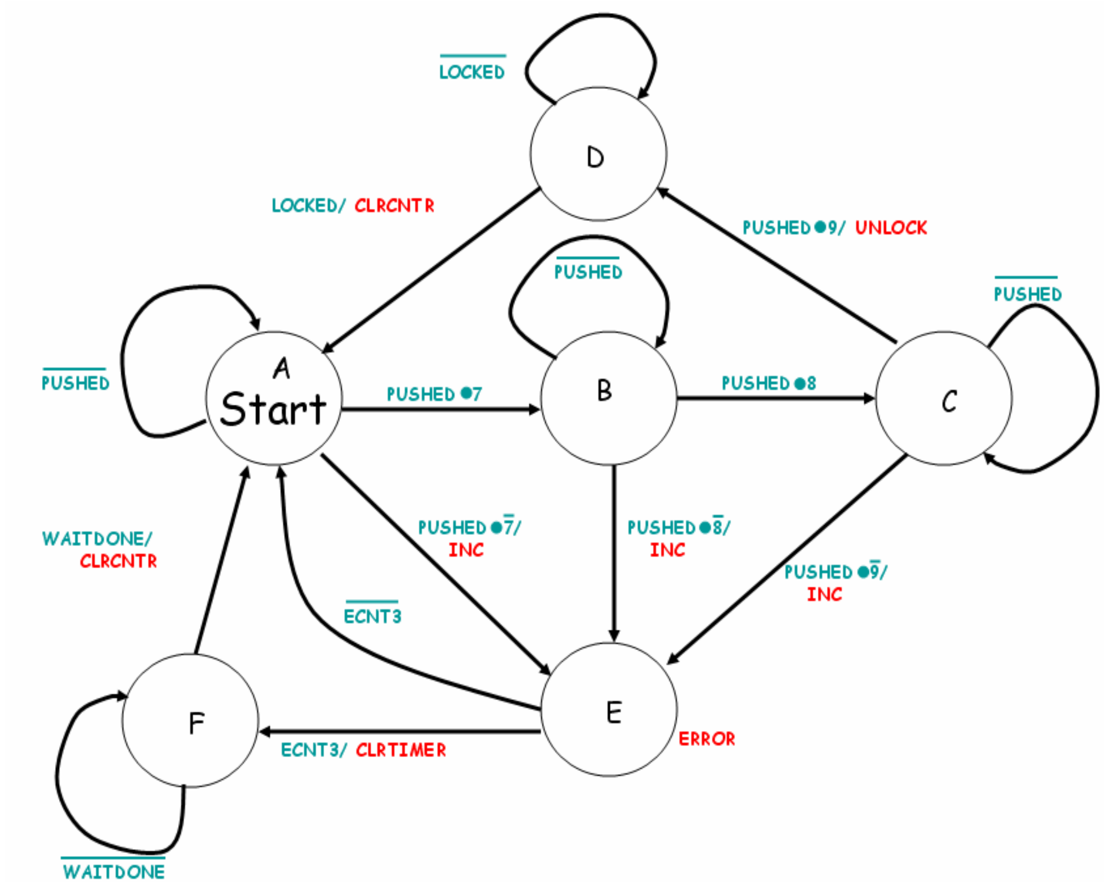
B+ = (PUSHED'•B)+(PUSHED•7•A)

C+ = (PUSHED'•C)+(PUSHED•8•B)

D+ = (LOCKED'•D)+(PUSHED•9•C)

E+ = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)

# Key Lock Input Forming Logic




A+ = (PUSHED'•A) + (ECNT3'•E)
+ (WAITDONE•F)
+ (LOCKED•D)

B+ = (PUSHED'•B)+(PUSHED•7•A)

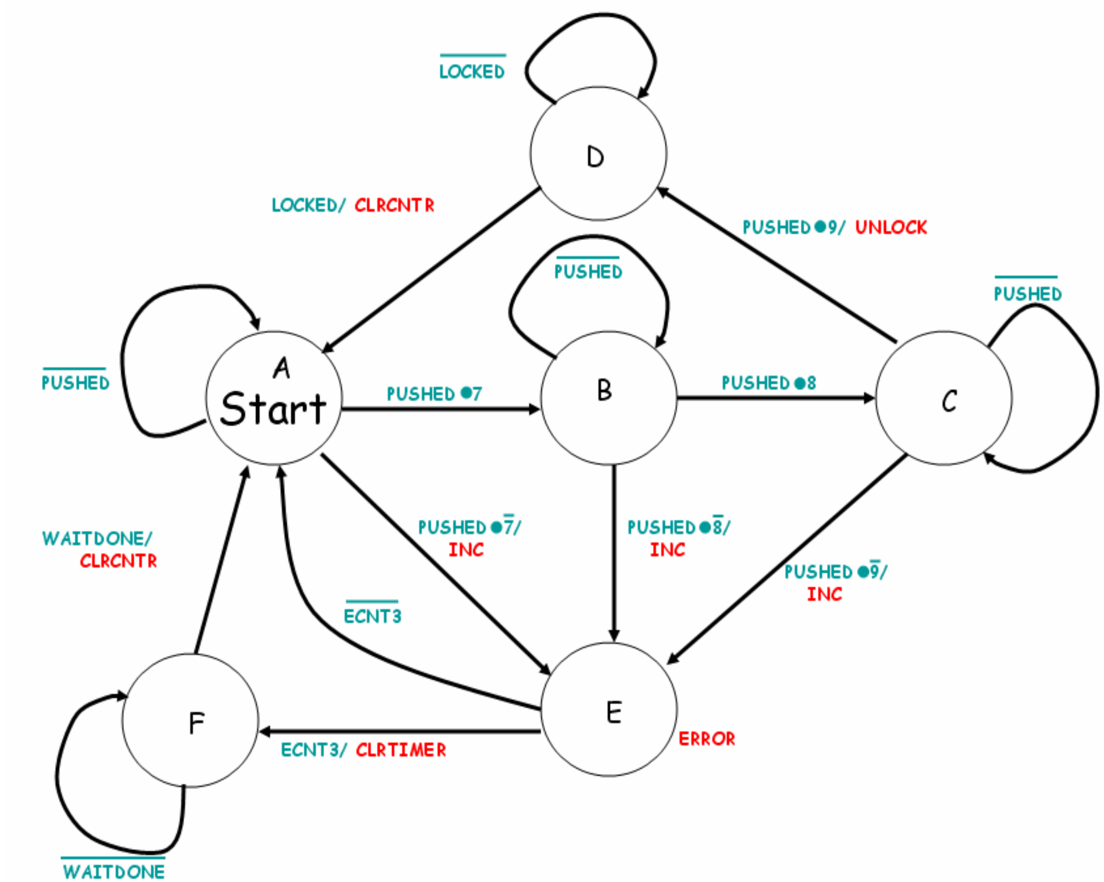
C+ = (PUSHED'•C)+(PUSHED•8•B)

D+ = (LOCKED'•D)+(PUSHED•9•C)

E+ = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)

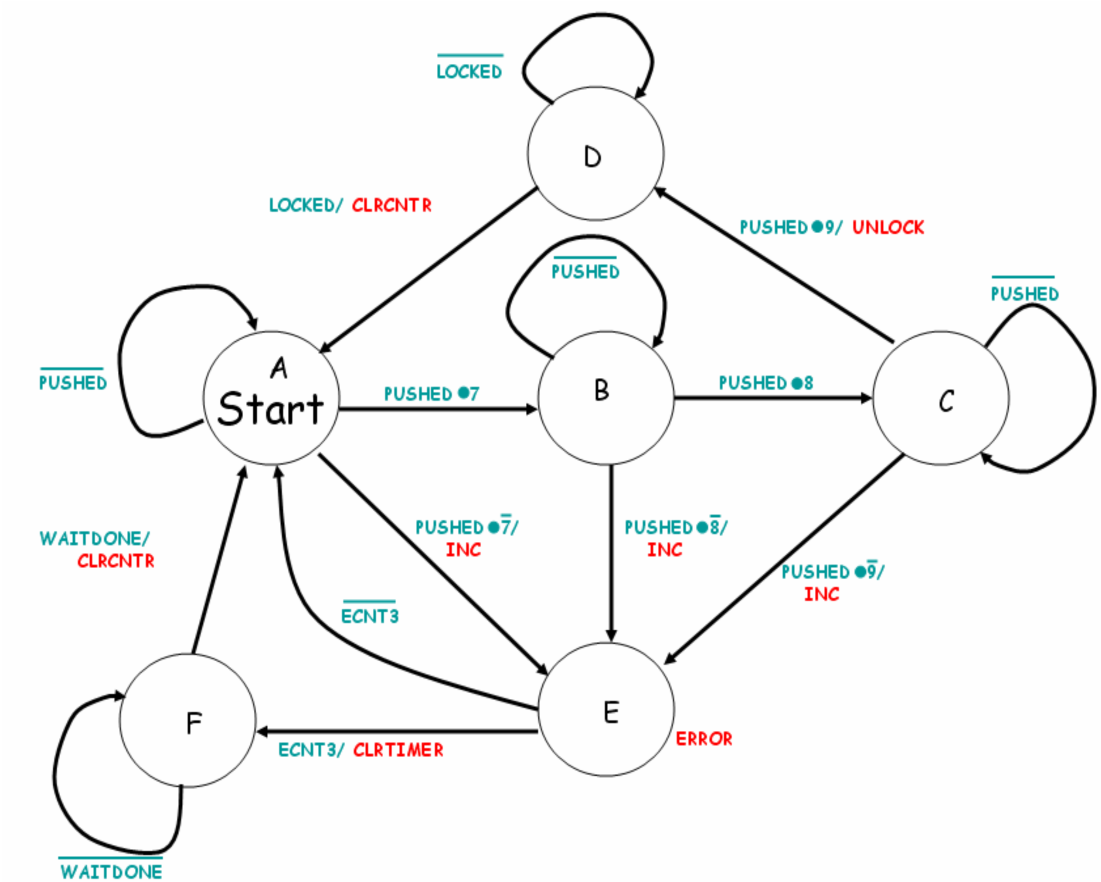
F+ = (ECNT3•E)
+ (WAITDONE'•F)

# Key Lock Output Forming Logic




ERROR = E

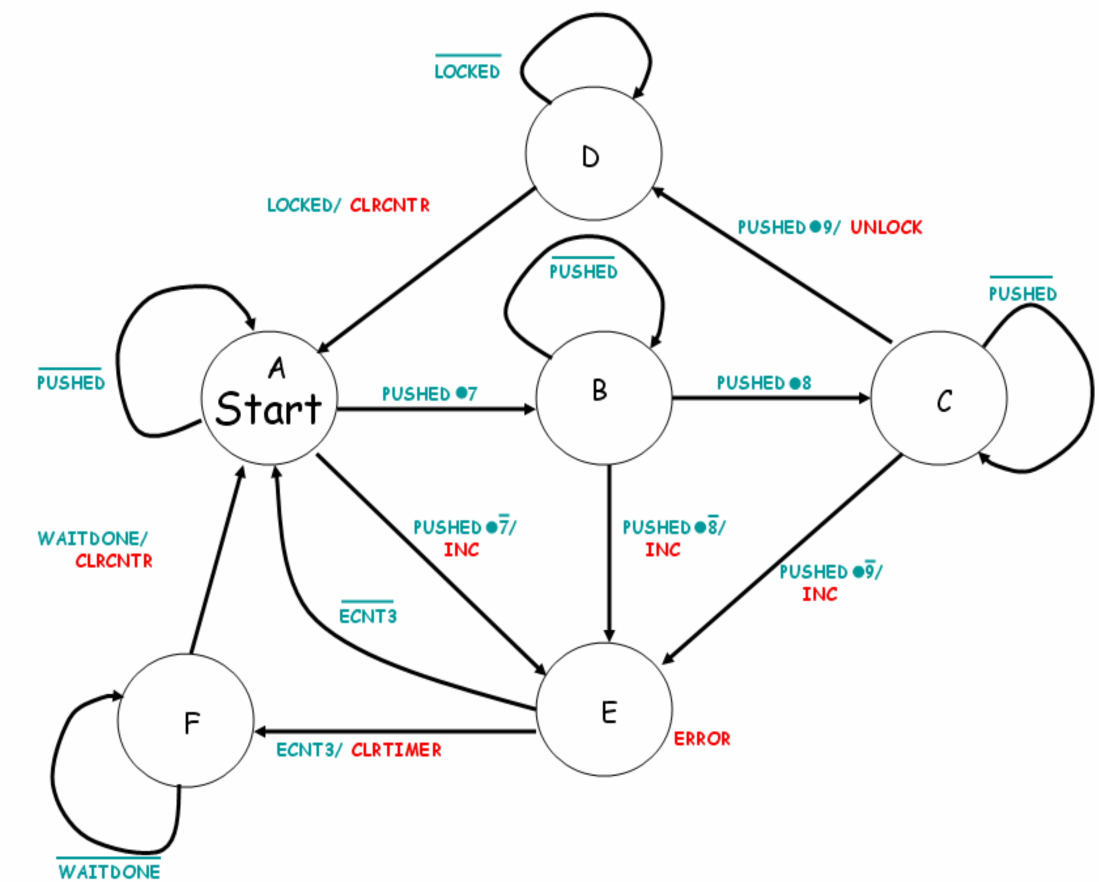
# Key Lock Output Forming Logic




ERROR = E

INC = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)

# Key Lock Output Forming Logic

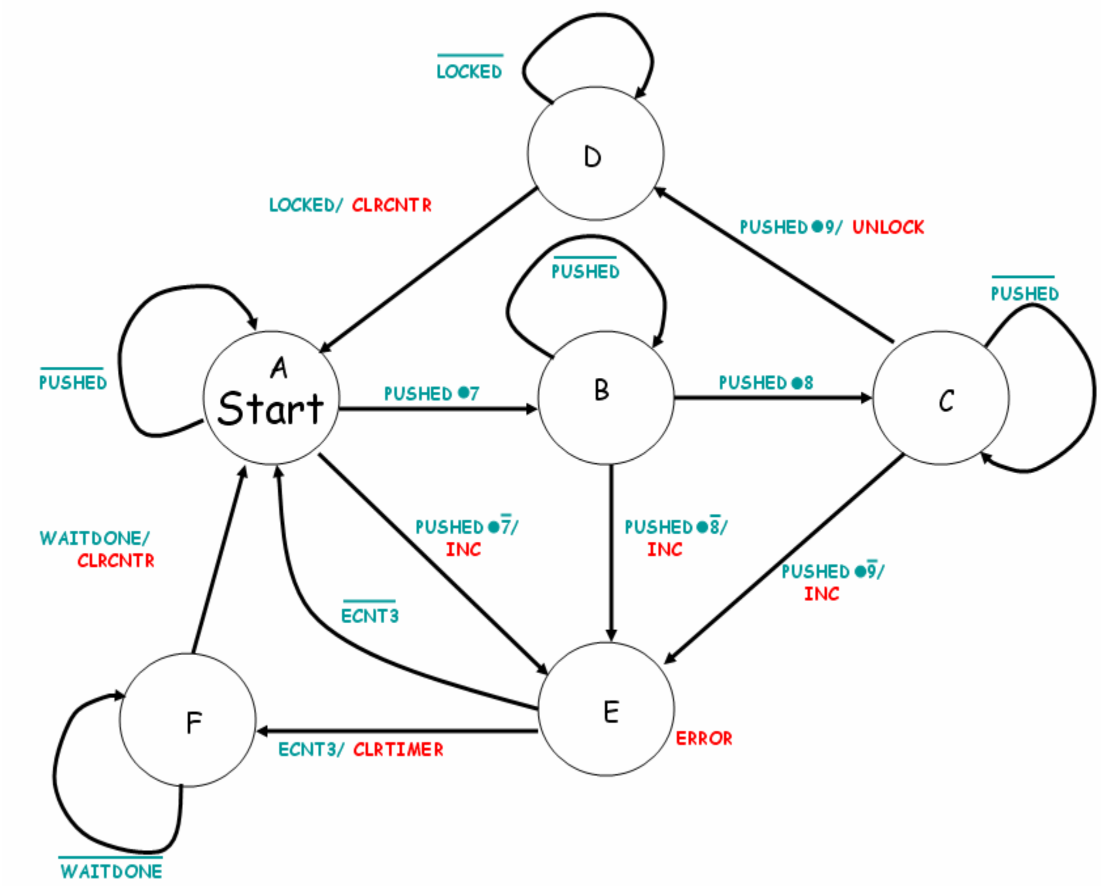



ERROR = E

INC = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)

CLRTIMER = ECNT3•E

# Key Lock Output Forming Logic



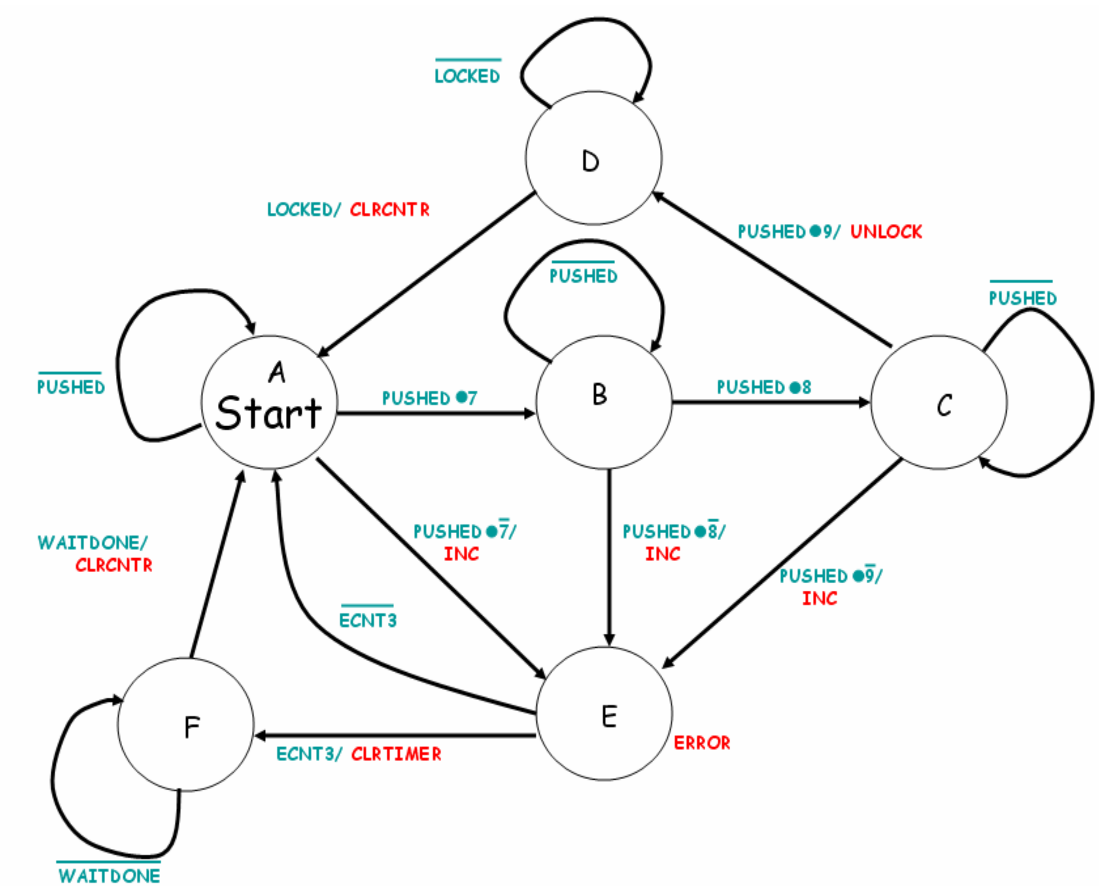

ERROR = E

INC = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)

CLRTIMER = ECNT3•E

CLRCNTR = (WAITDONE•F)
+ ((LOCKED•D)

# Key Lock Output Forming Logic




ERROR = E
INC = (PUSHED•7'•A)
+ (PUSHED•8'•B)
+ (PUSHED•9'•C)
CLRTIMER = ECNT3•E
CLRCNTR = (WAITDONE•F)
+ ((LOCKED•D)
UNLOCK = (PUSHED•9•C)

# Other State Encoding Techniques

- You have learned the 2 extremes
  - Fully encoded (8 states ⇔ 3 state bits)
  - One-hot encoded (8 states ⇔ 8 state bits)
- A range of options exist in between

- A good choice of encoding
  - Can minimize IFL and OFL complexity
  - Algorithms have been developed for this…
  - Beyond the scope of this class