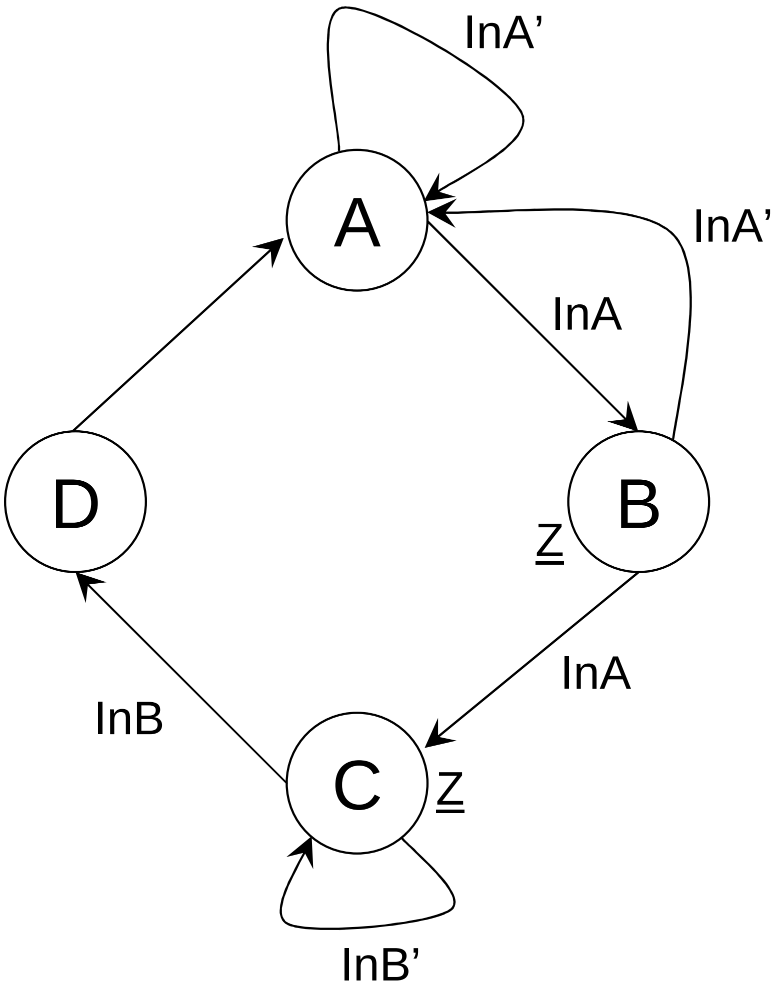
One-Hot Encoded Finite State Machines

Example State Machine

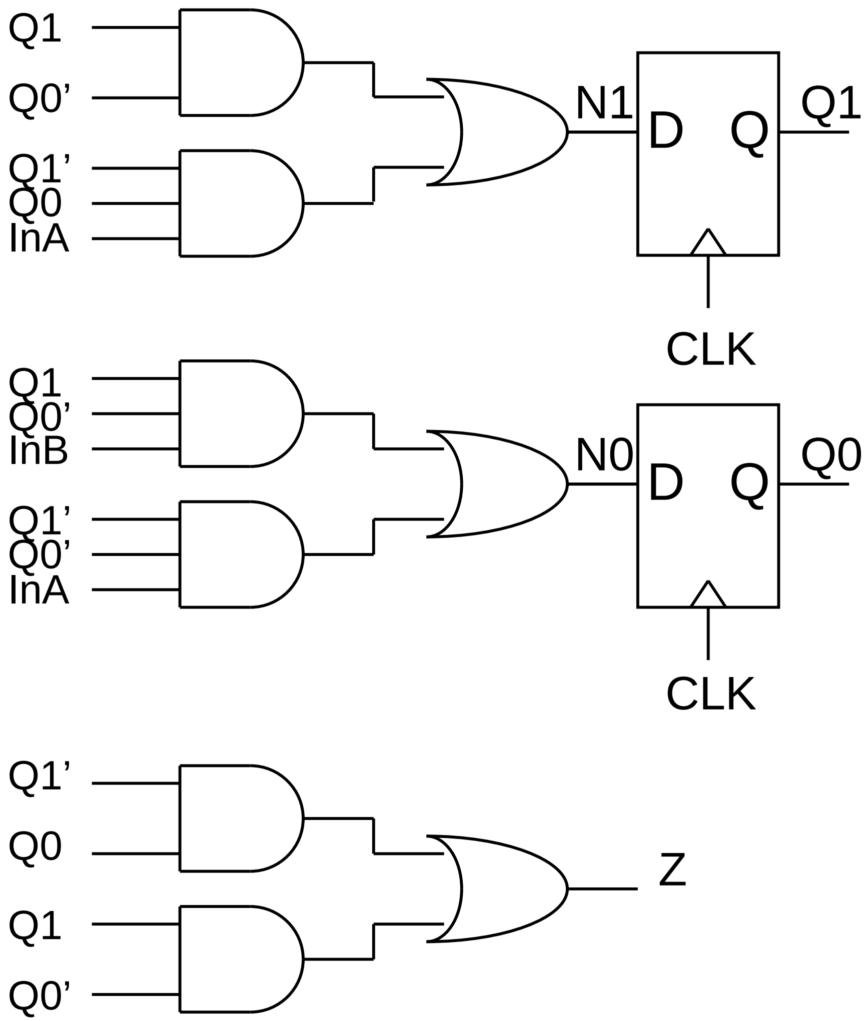


InA	InB	CS	NS	Z
0	-	A	A	0
1	-	A	B	0
0	-	B	A	1
1	-	B	C	1
-	0	C	C	1
-	1	C	D	1
-	-	D	A	0

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'

Logic Used:
- 9 gates
- 21 gate inputs
- 2 flip flops

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.

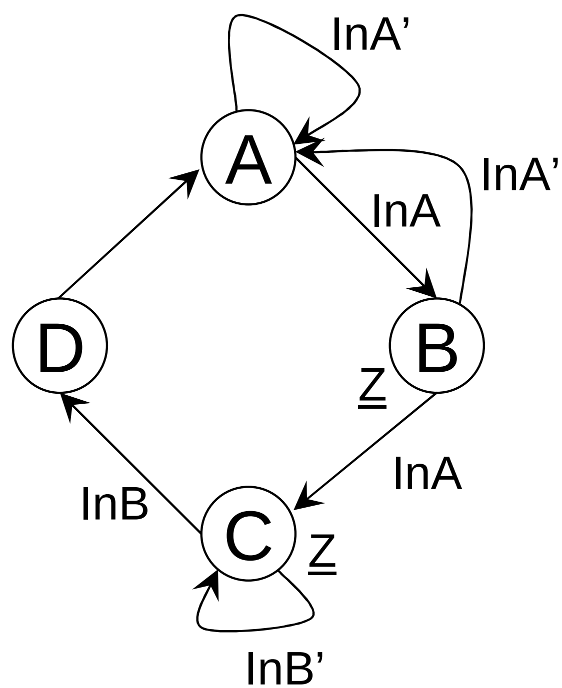
This TT with all these input don't cares is the result

One-Hot Encoding Results

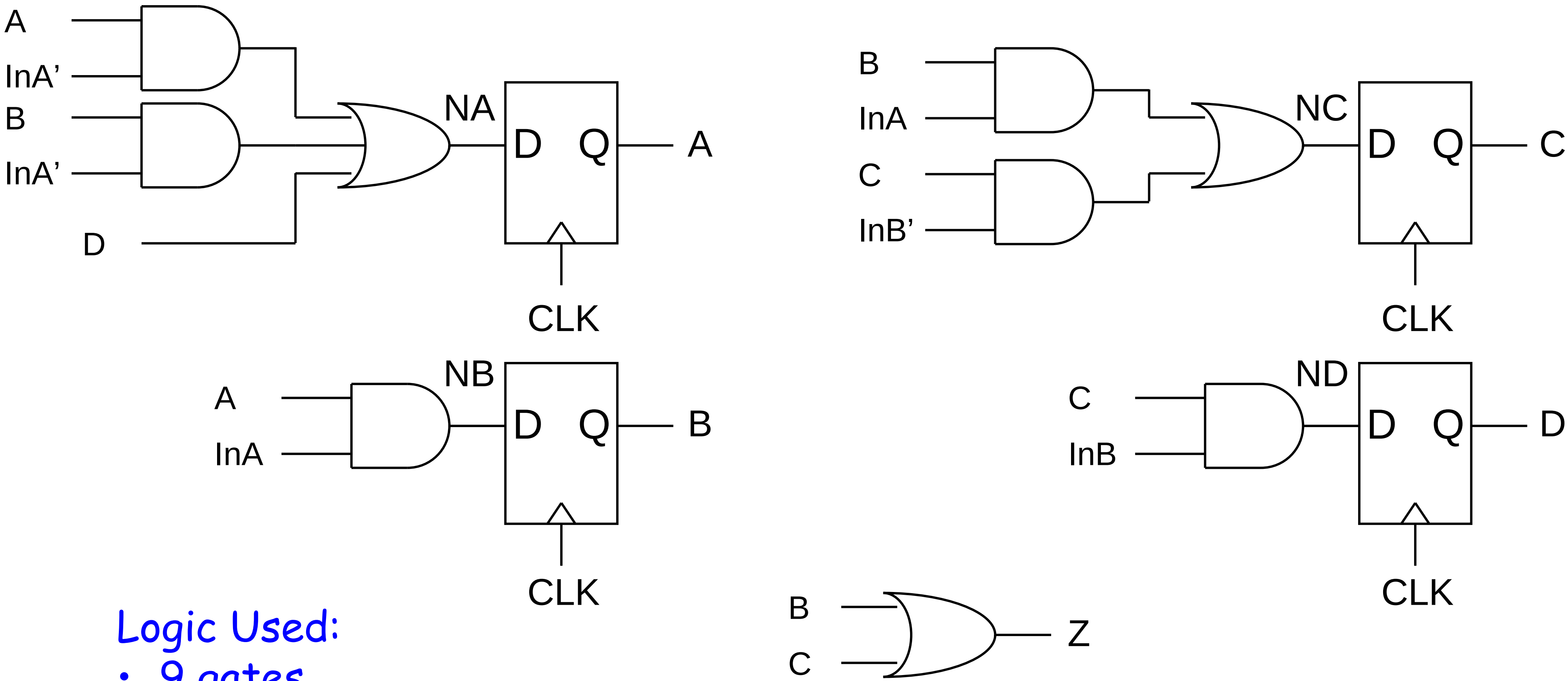
- Will require 4 flip flops
  - One per state
  - Call the current state bits A, B, C, and D
  - Call the next state bits NA, NB, NC, and ND

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



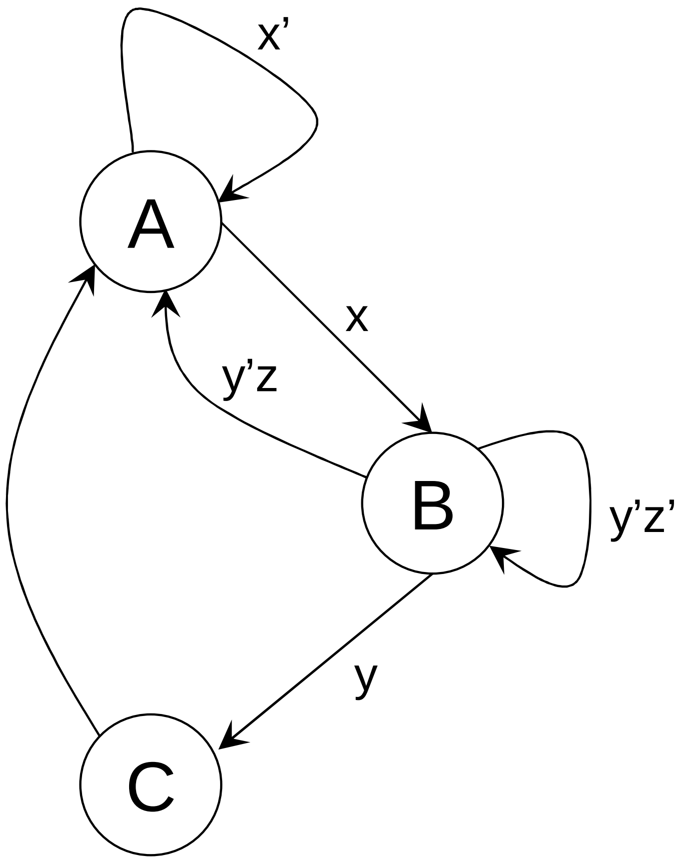
Logic Used:

- 9 gates
- 19 gate inputs
- 4 flip flops

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
- Requires more flip flops
- Can do one-hot design by inspection
  - ***Without using transition tables…***

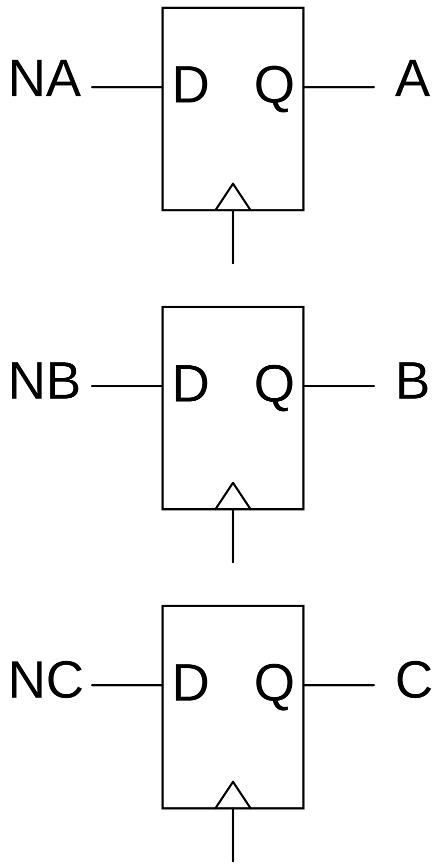
Another One-Hot Example



State	Encoding
A	100
B	010
C	001

State Encoding and Structure

State	Encoding
A	100
B	010
C	001

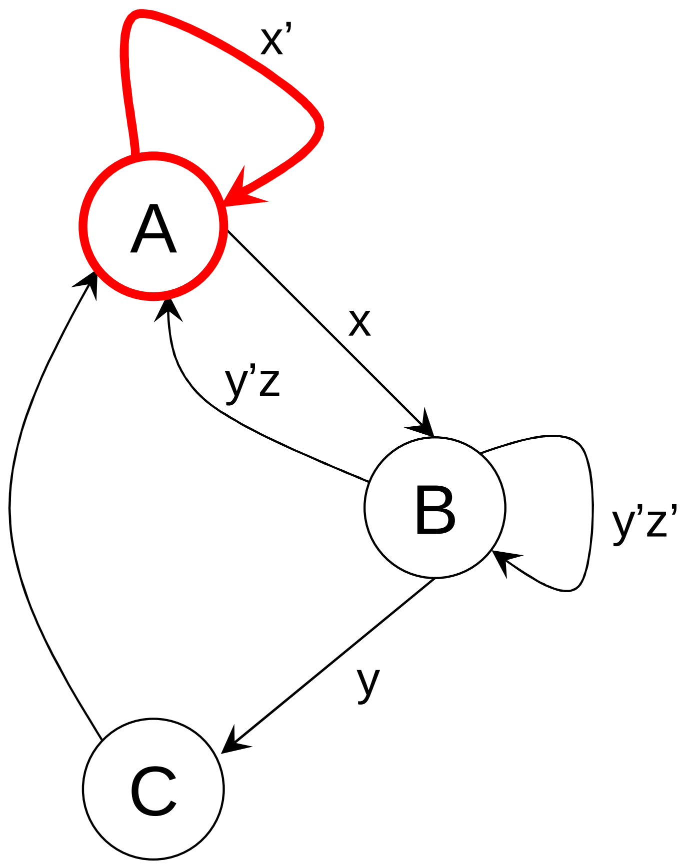


With one-hot encoding, each state has its own flip flop.

'A' is the name of a state *and* it is the name of the wire coming out of the flip flop for state 'A'.

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

One-Hot Encodings By Inspection

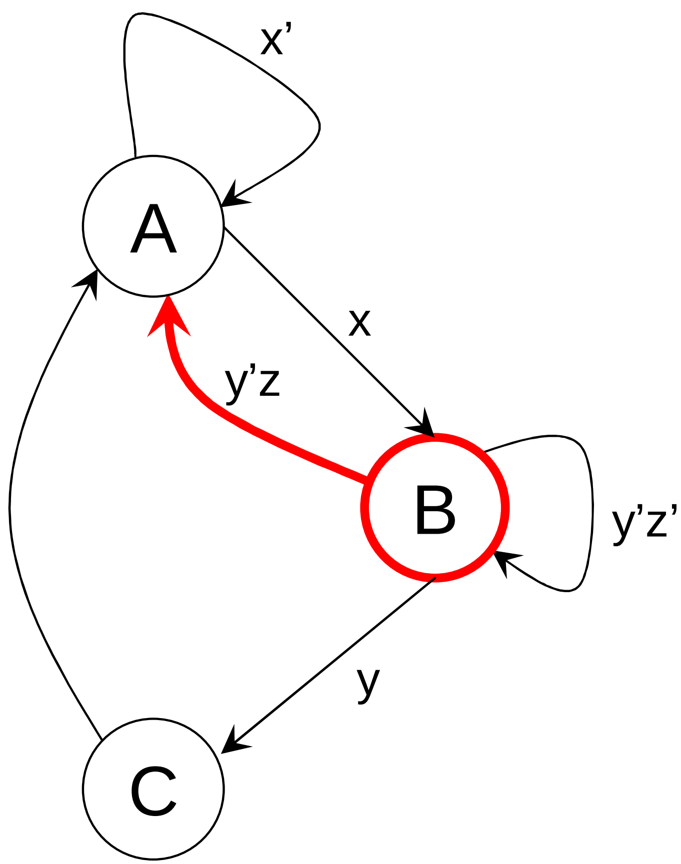


When is A the next state?

Look at the arcs entering state A

NA = A•x' + B•y'z + C

One-Hot Encodings By Inspection

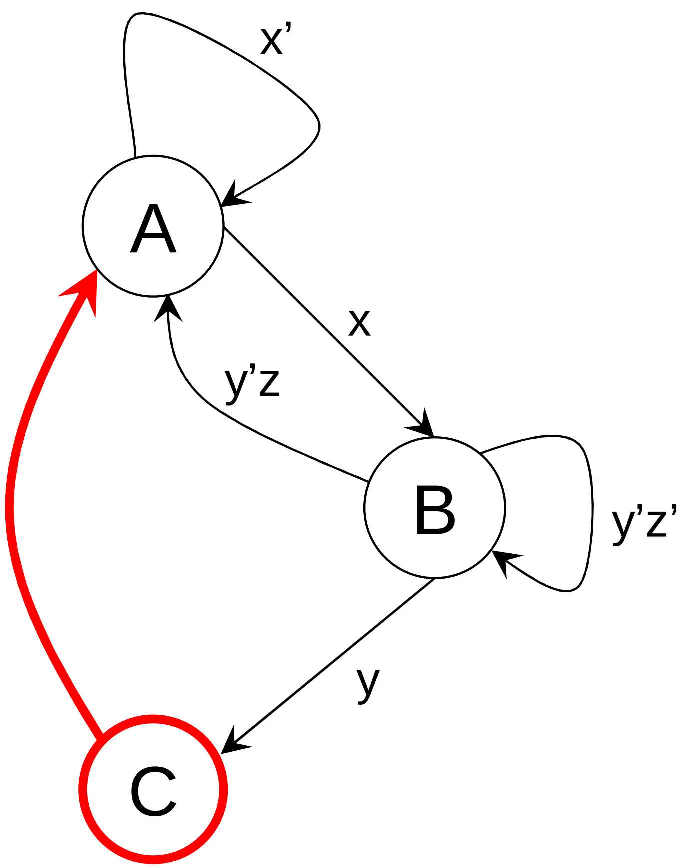


When is A the next state?

Look at the arcs entering state A

NA = A•x’ + B•y’z + C

One-Hot Encodings By Inspection

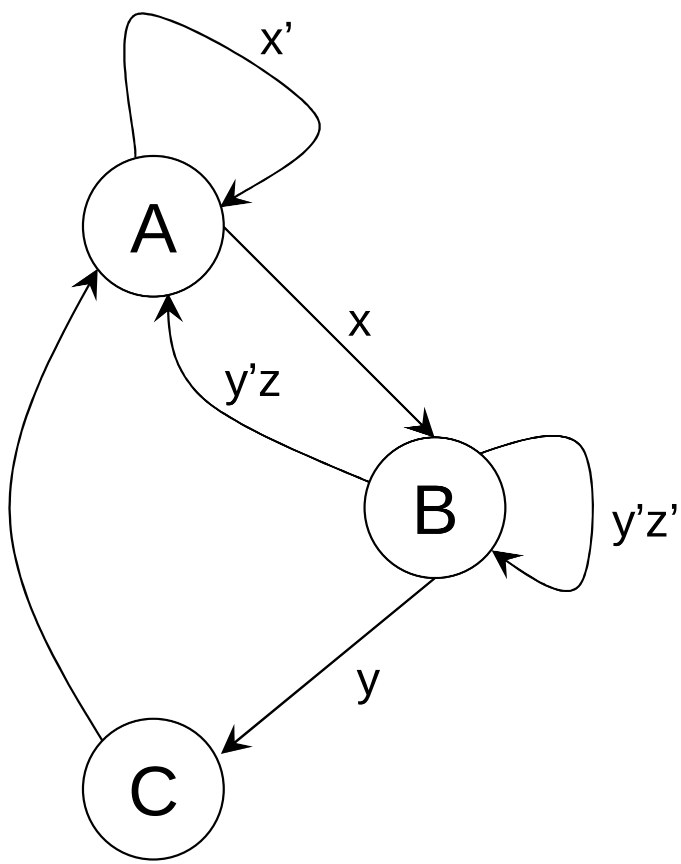


When is A the next state?

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

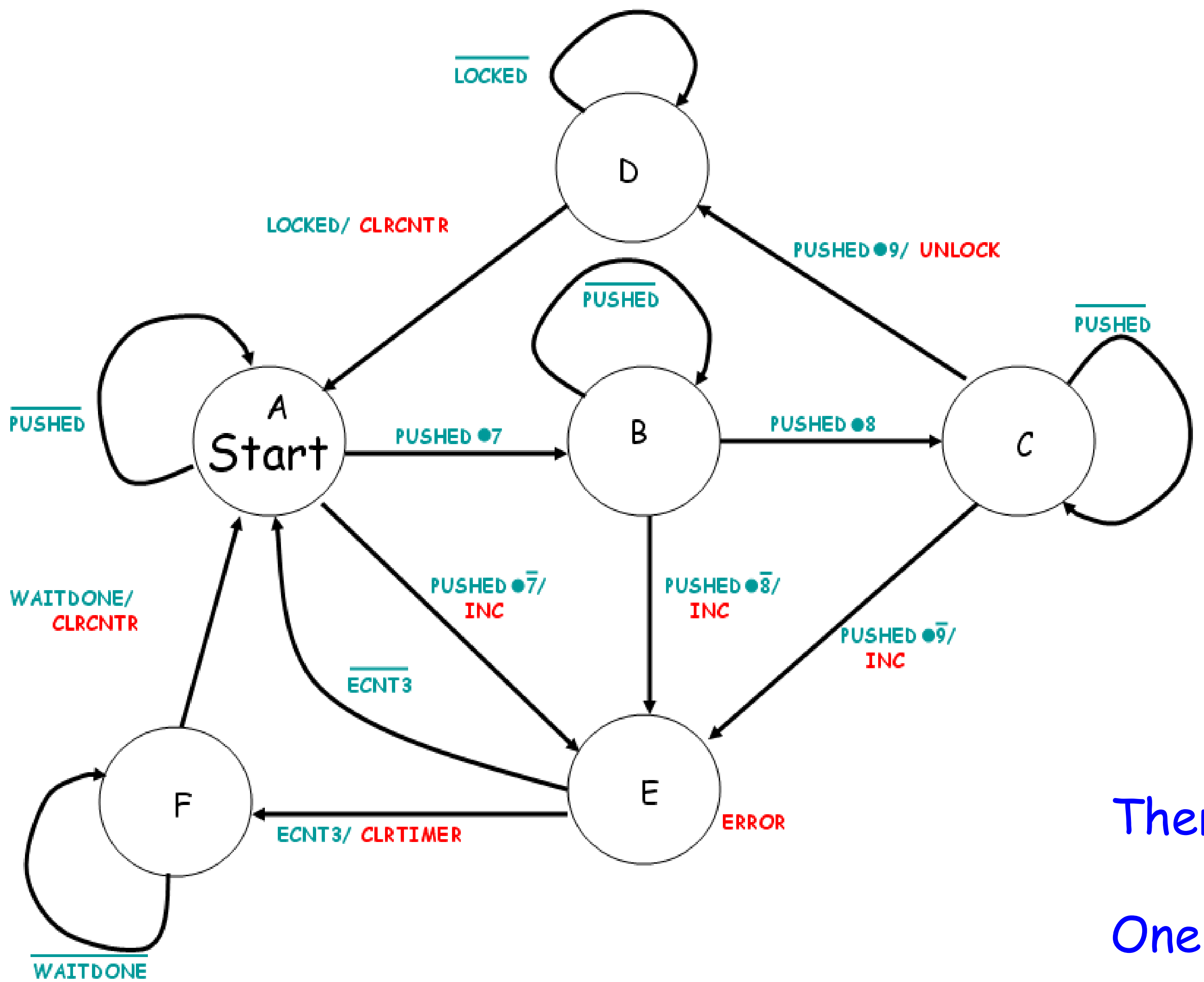
NA = A•x' + B•y'z + C

Similar reasoning leads to:

NB = A•x + B•y'z'

NC = B•y

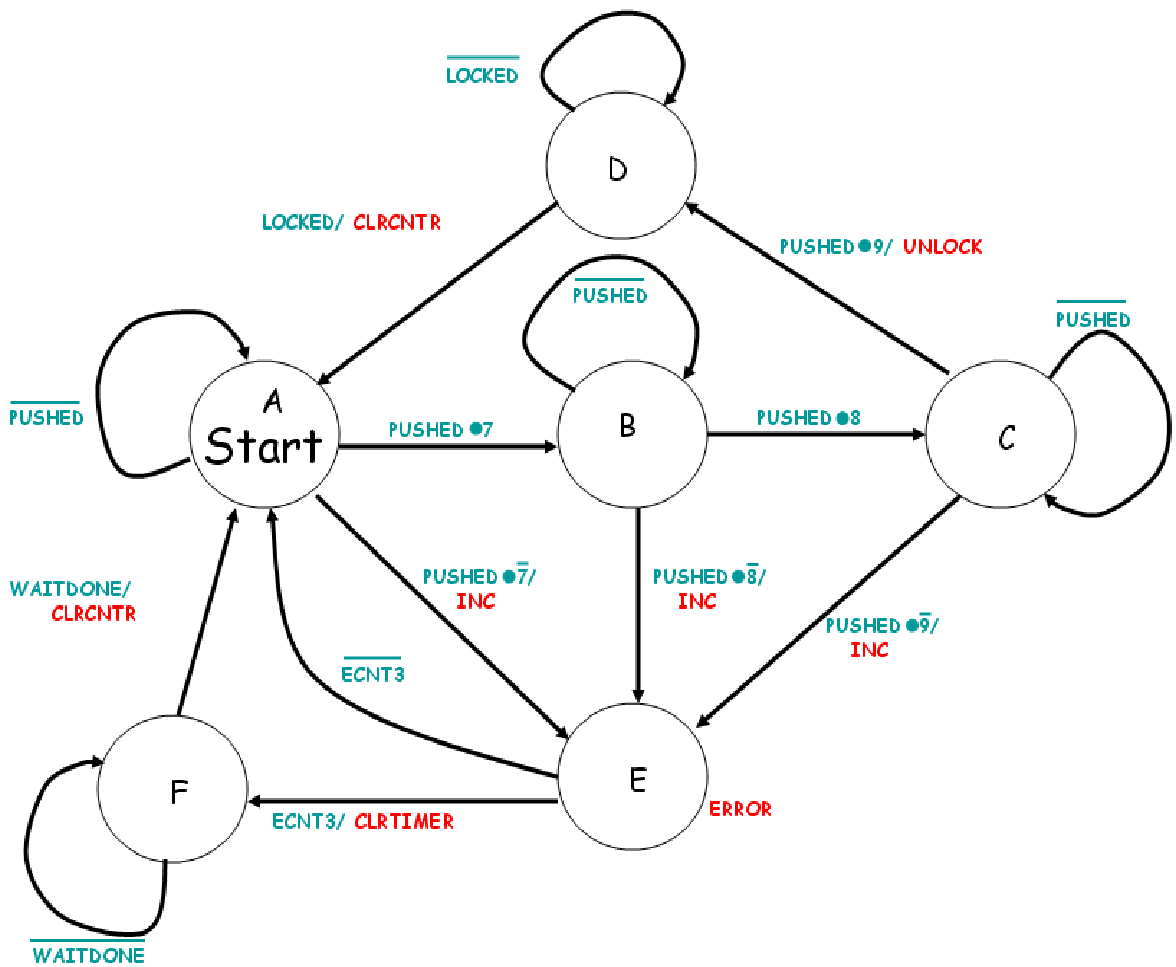
The Key Lock Problem - One-Hot Version



There will be 6 flip flops

One for each state

Key Lock Input Forming Logic



$NA = (PUSHED' \cdot A) + (ECNT3' \cdot E)$
$\quad + (WAITDONE \cdot F)$
$\quad + (LOCKED \cdot D)$

$NB = (PUSHED' \cdot B) + (PUSHED \cdot 7 \cdot A)$

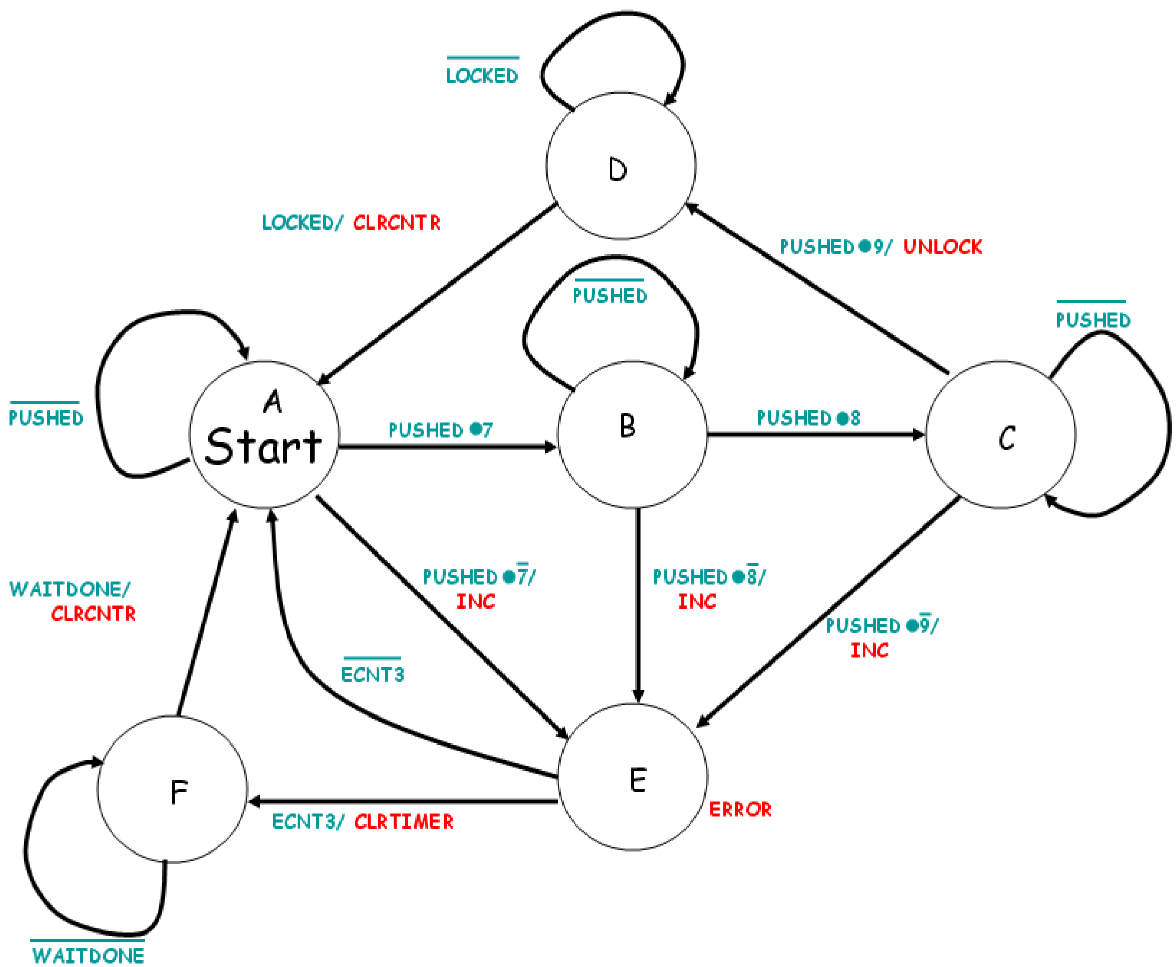
$NC = (PUSHED' \cdot C) + (PUSHED \cdot 8 \cdot B)$

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

$NE = (PUSHED \cdot 7' \cdot A)$
$\quad + (PUSHED \cdot 8' \cdot B)$
$\quad + (PUSHED \cdot 9' \cdot C)$

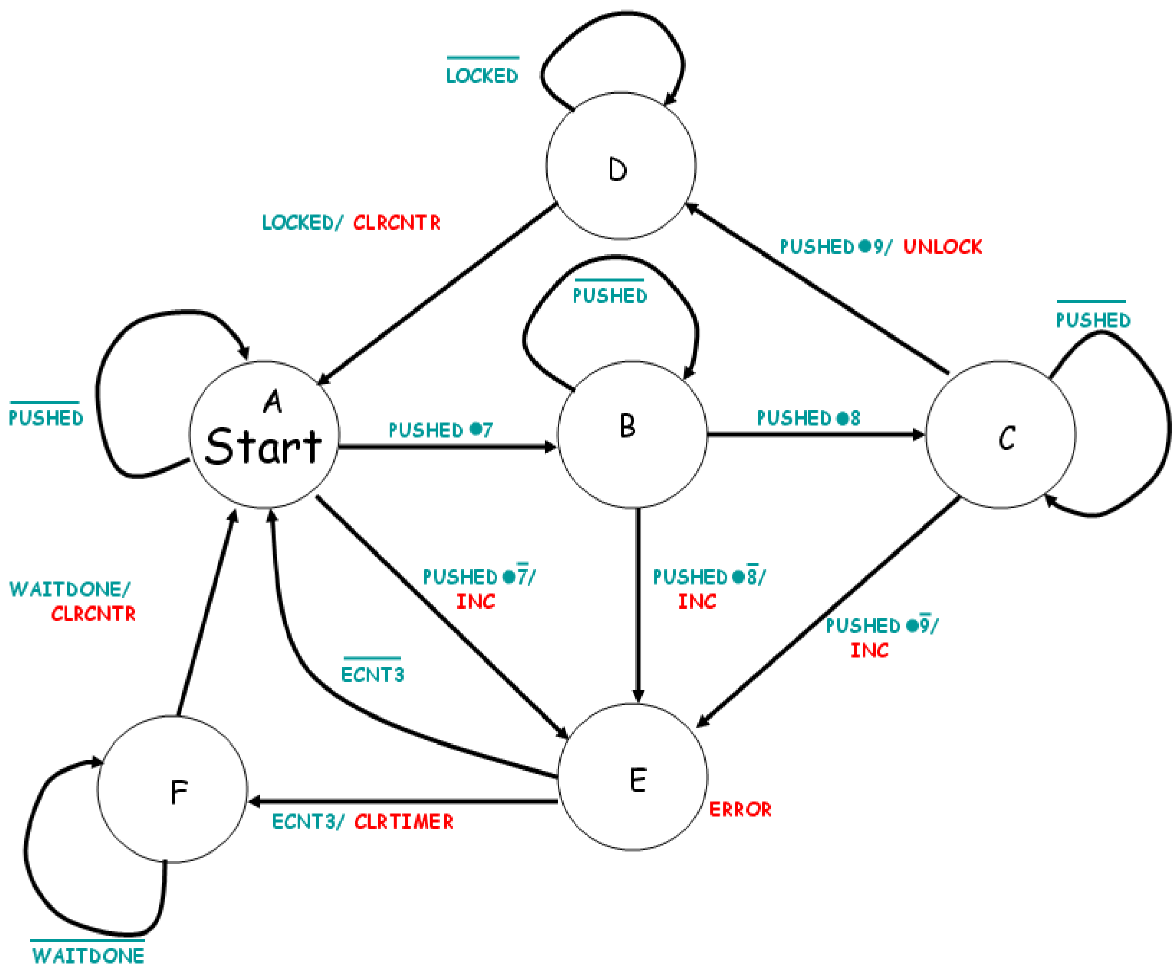
$NF = (ECNT3 \cdot E)$
$\quad + (WAITDONE' \cdot F)$

Key Lock Output Forming Logic



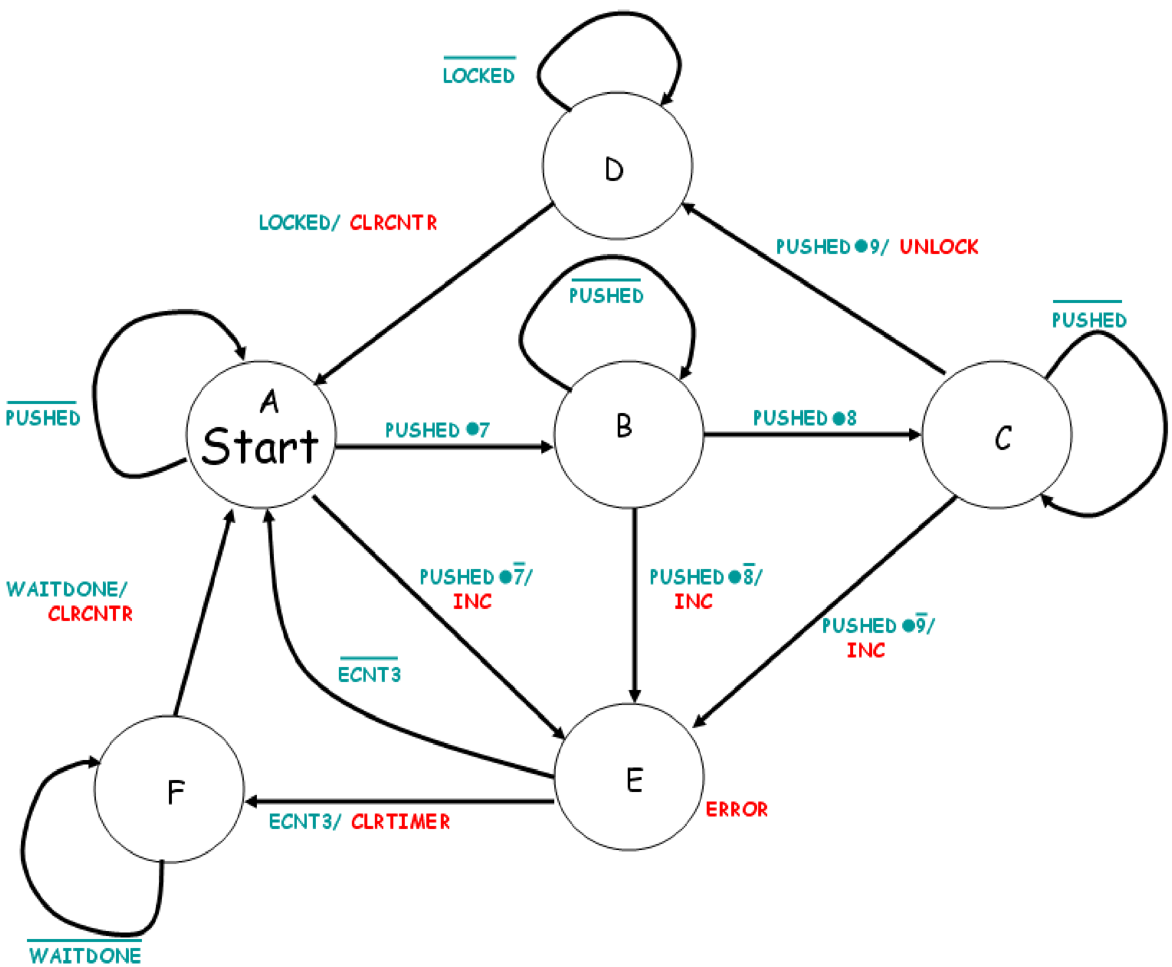
ERROR =

Key Lock Output Forming Logic



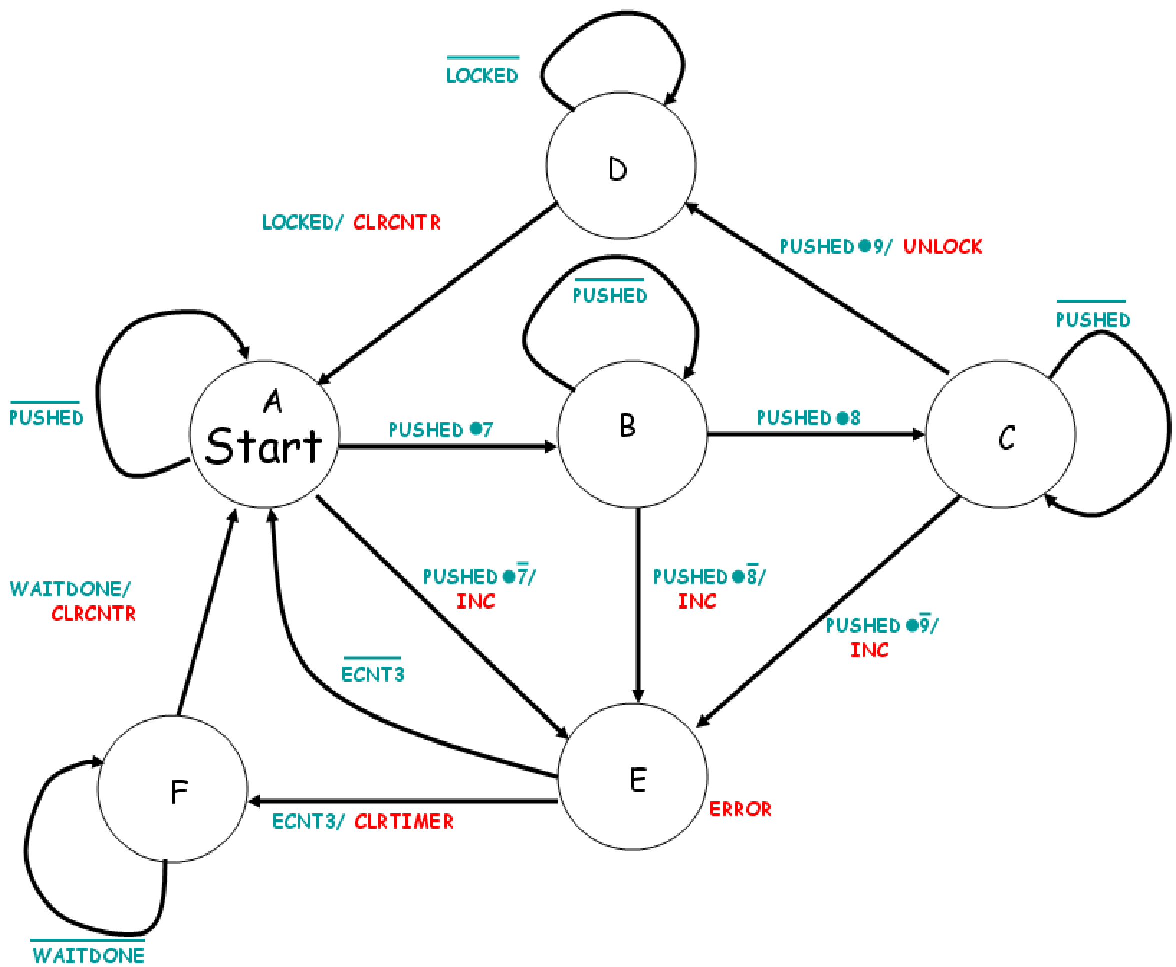
ERROR = E
INC =

Key Lock Output Forming Logic



ERROR = E
INC = (PUSHED·7'·A)
+ (PUSHED·8'·B)
+ (PUSHED·9'·C)
CLRTIMER =

Key Lock Output Forming Logic



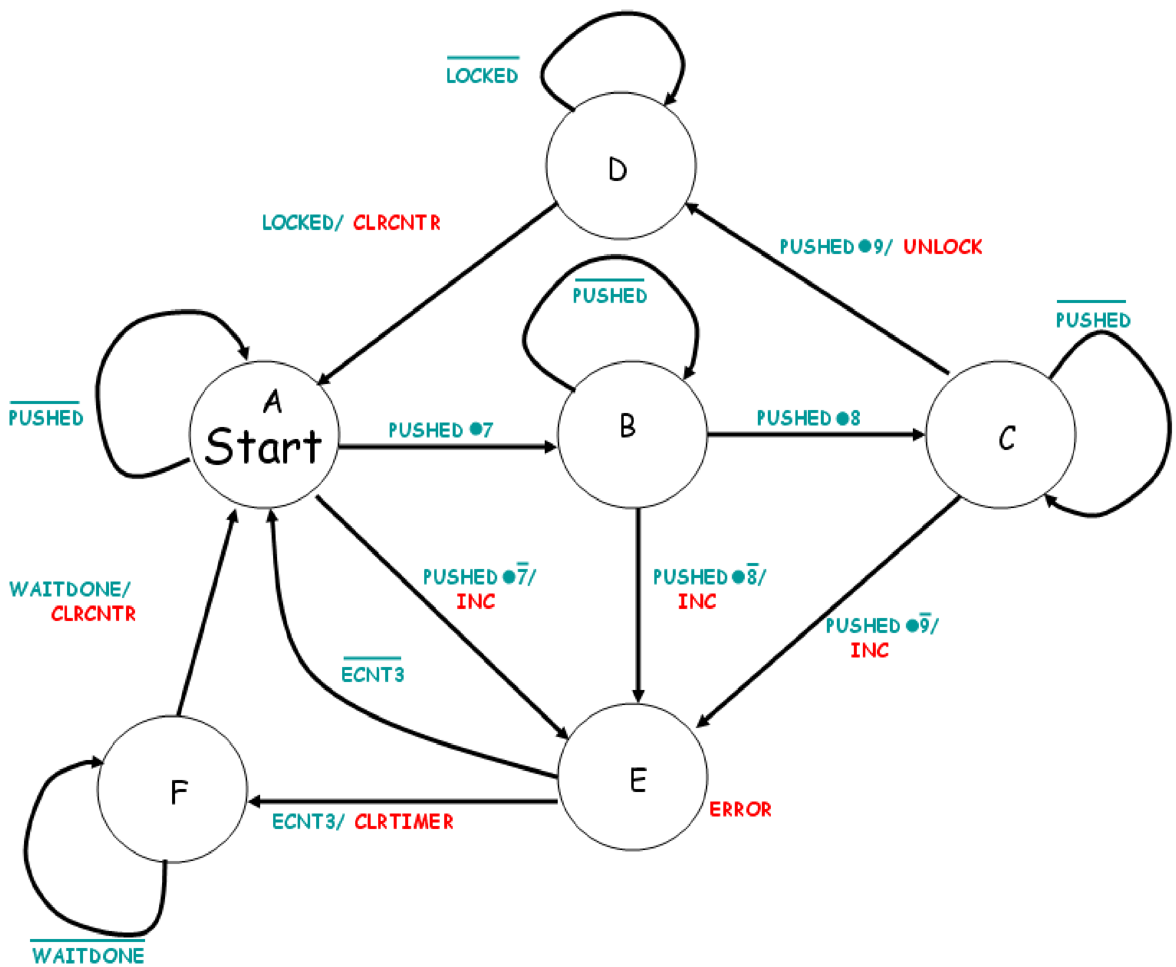
ERROR = E

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

CLRTIMER = ECNT3·E

CLRCNTR =

Key Lock Output Forming Logic



ERROR = E

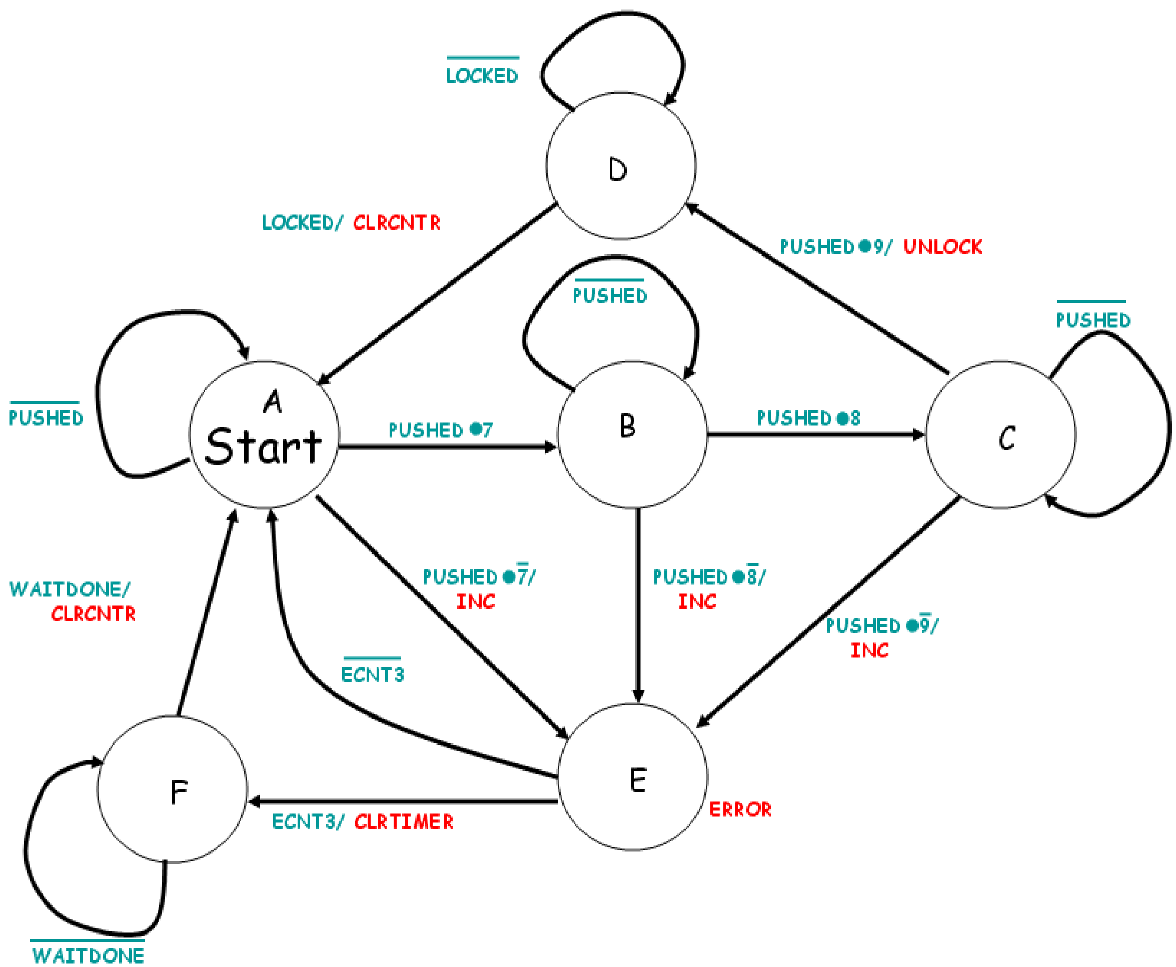
INC = (PUSHED·7'·A)
+ (PUSHED·8'·B)
+ (PUSHED·9'·C)

CLRTIMER = ECNT3·E

CLRCNTR = (WAITDONE·F)
+ (LOCKED·D)

UNLOCK =

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