Digital Logic Circuits
'Sequential Logic Design’
ELEC2200
Summer 2009

David J. Broderick
brodedj@auburn.edu
http://www.auburn.edu/~brodedj
Office: Broun 360

Example

- Looking up circuits in a book only useful to a point
- Example: Design a 3 bit gray code counter with an active low, synchronous reset. Transitions should occur on the rising edge.
- Not a common circuit, we'll have to design ourselves
- What steps do we need to take?

Basic Steps

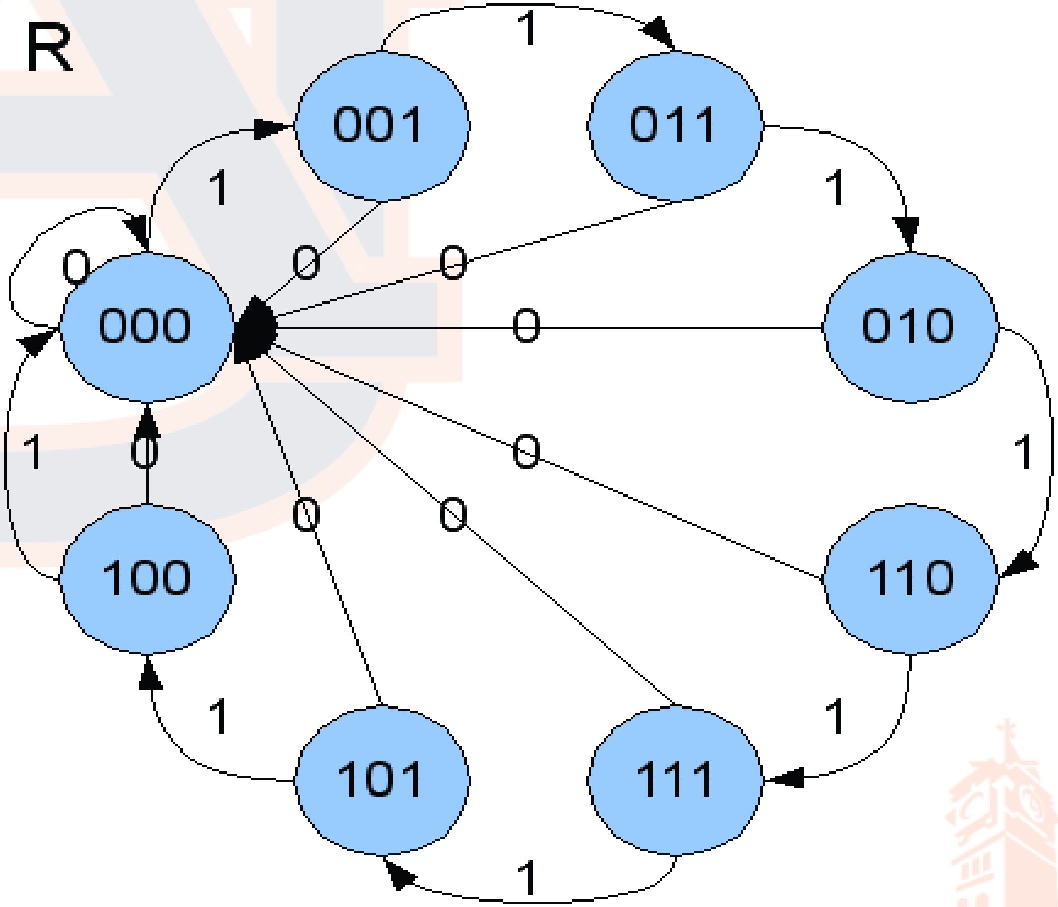
1) **Describe the state machine based on the design spec**
   - 3 different ways
2) Choose flip-flops
3) Create circuit excitation table
4) Construct K-maps for:
   - flip-flop inputs
   - primary outputs
5) Obtain minimized SOP equations
6) Draw logic diagram
7) Simulate to verify design & debug as needed
8) Perform circuit analysis & logic optimization

State Machine

- A state diagram describes the behavior of a finite state machine (FSM). Can also use:
  - A state table
  - Truth table format
- A FSM is a model of a device that can be described by:
  - A finite number of states
  - Stimuli and related transitions
  - Outputs

State Diagram

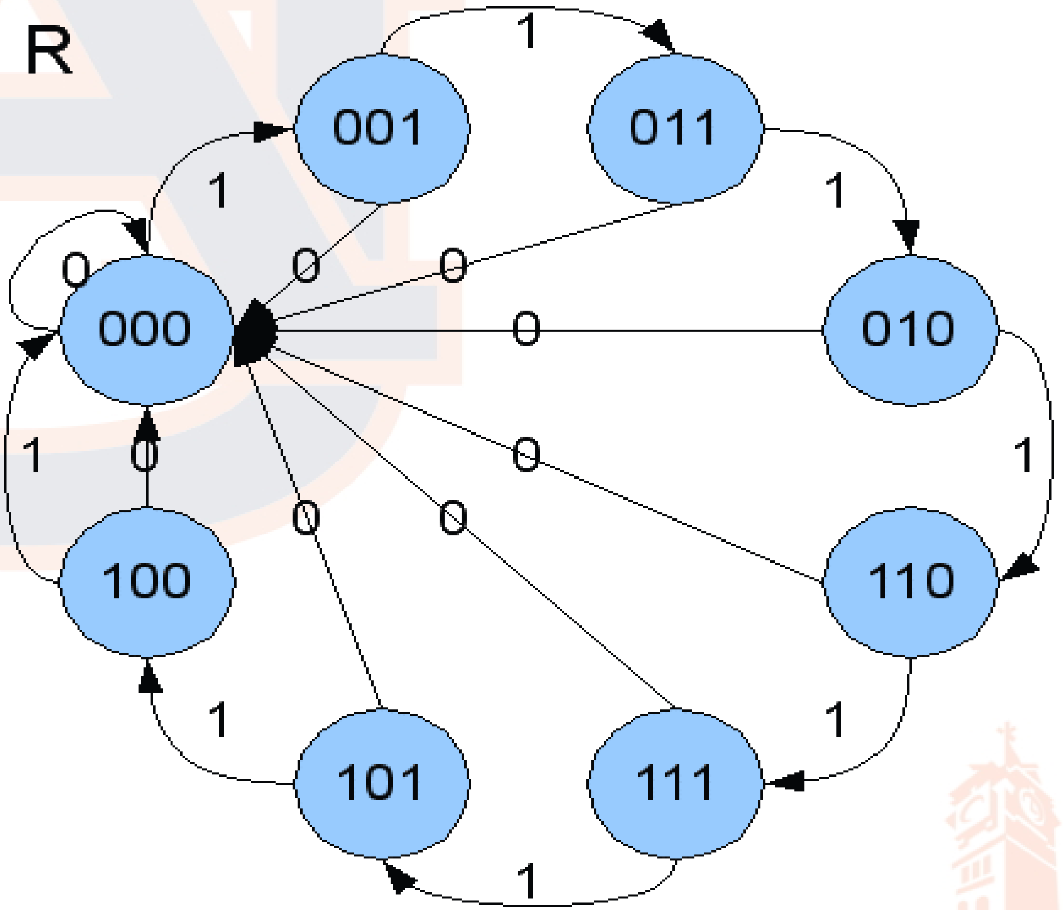
- In this case output is equivalent to state
- FSM is initialized to 000_2 state
- If R=1, move to next count
- If R=0, reset to zero
- Active low reset
- Design calls for rising edge triggered flip-flops



State Table

- Contains the same information in a different format
- Current state and input as column, row headings
- Next state as cell in table

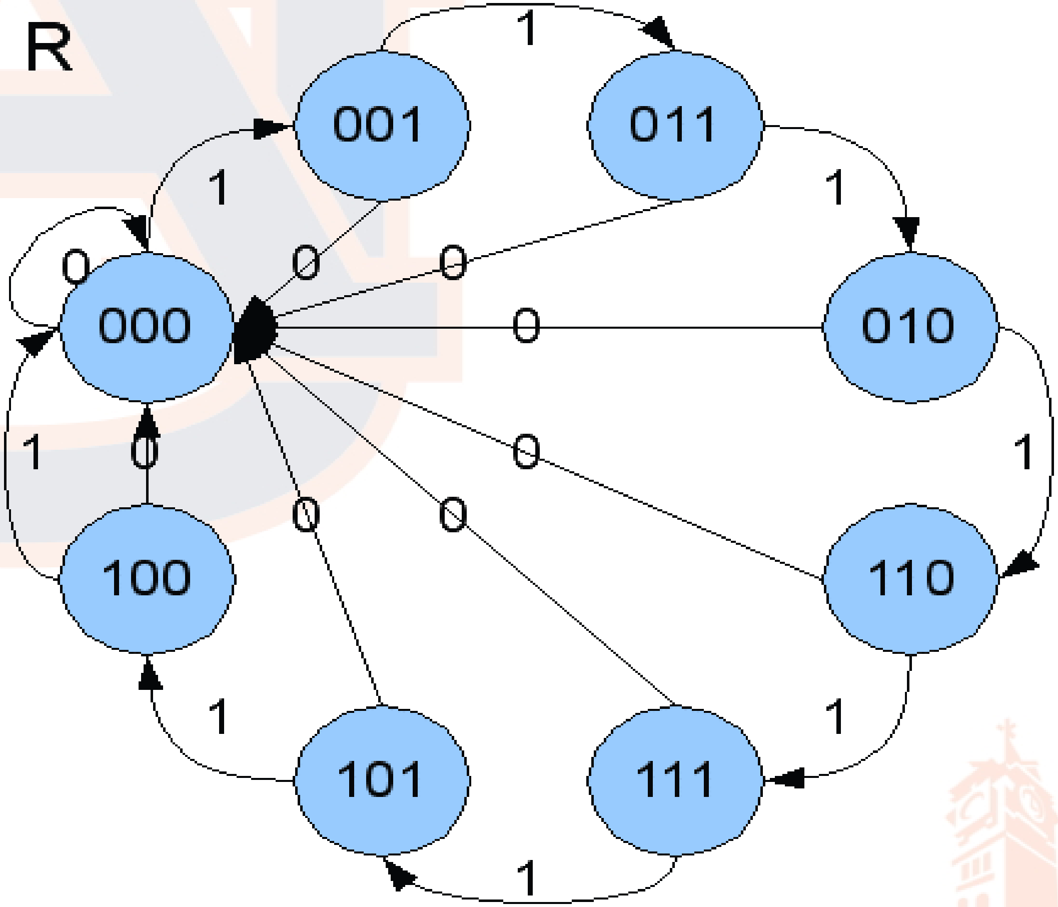
State\R	0	1
000	000	001
001	000	011
011	000	010
010	000	110
110	000	111
111	000	101
101	000	100
100	000	000



Truth Table

- Contains the same information in a different format
- If it's not in the table, we "don't care"

Input	Curr. State			Next State		
R	**2**	**1**	**0**	**2**	**1**	**0**
0	x	x	x	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	0	1	0
1	0	1	0	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1
1	1	0	1	1	0	0
1	1	0	0	0	0	0



Basic Steps

1) Describe the state machine based on the design spec
   - 3 different ways ☑

2) **Choose flip-flops**

3) Create circuit excitation table

4) Construct K-maps for:
   - flip-flop inputs
   - primary outputs

5) Obtain minimized SOP equations

6) Draw logic diagram

7) Simulate to verify design & debug as needed

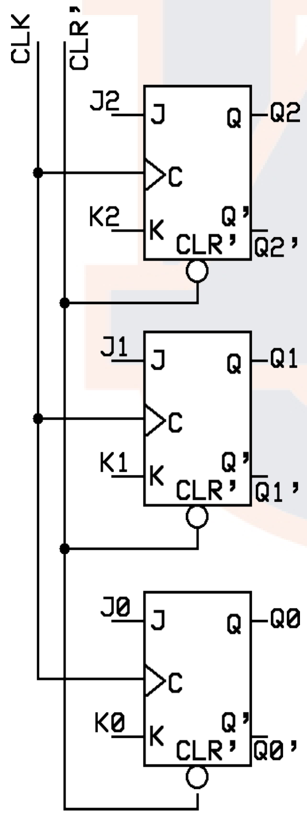
8) Perform circuit analysis & logic optimization

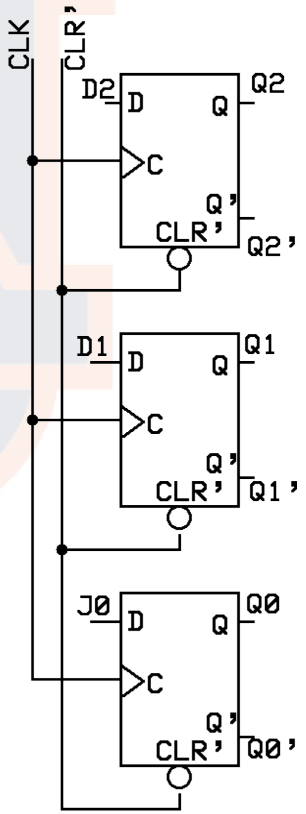
Choosing Flip-Flops

- Any type will work
- Some will result in a simpler, smaller circuit
- For sequential design typically JK and D flip-flops
- T flip-flops typically used for counter designs
- Let's mix it up for our example and use all types

Choosing Flip-Flops

- Each state is represented by a binary value
- Binary value must have enough bits to represent all of the states
- Here, 8 states → 3 bits





Basic Steps

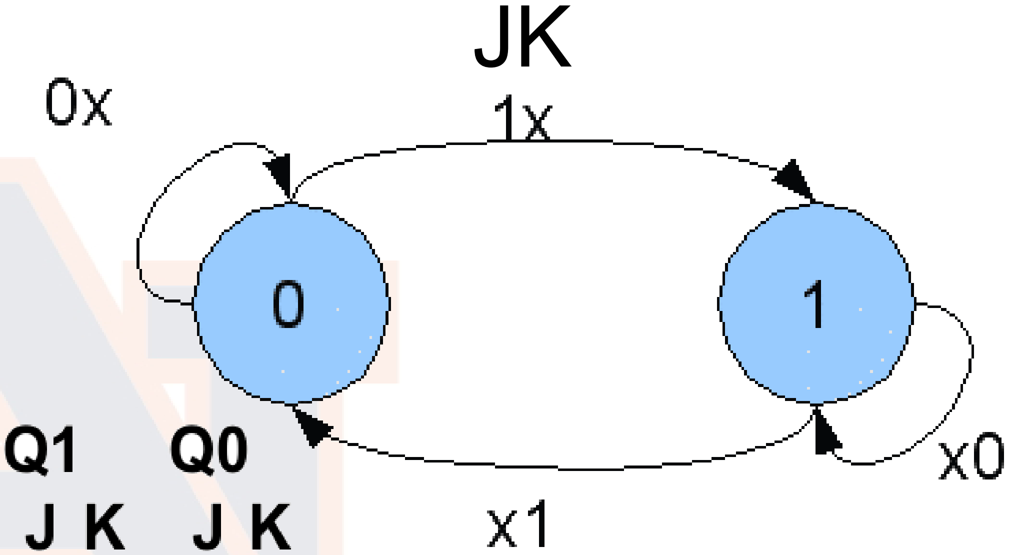
1) Describe the state machine based on the design spec
    - 3 different ways ✓
2) Choose flip-flops ✓
3) **Create circuit excitation table**
4) Construct K-maps for:
    - flip-flop inputs
    - primary outputs
5) Obtain minimized SOP equations
6) Draw logic diagram
7) Simulate to verify design & debug as needed
8) Perform circuit analysis & logic optimization

Circuit Excitation Table

- Excitation table contains information to answer: “What inputs are needed to move a flip-flop from the current state to the next state?”
- Build table by:
  - Copying the state machine truth table
  - Adding columns for each flip-flop input

Circuit Excitation Table

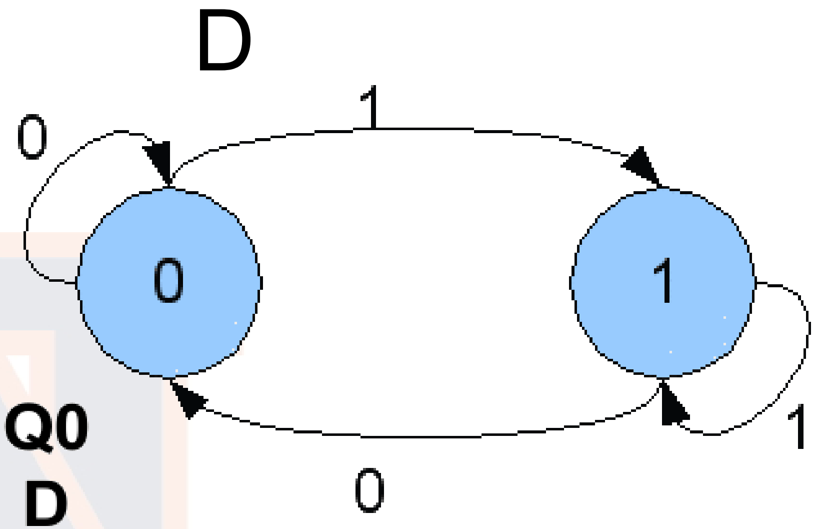
- For JK flip-flops:



Input	Curr. State			Next State			Q2		Q1		Q0	
R	**2**	**1**	**0**	**2**	**1**	**0**	**J**	**K**	**J**	**K**	**J**	**K**
0	x	x	x	0	0	0	0	1	0	1	0	1
1	0	0	0	0	0	1	0	x	0	x	1	x
1	0	0	1	0	1	1	0	x	1	x	x	0
1	0	1	1	0	1	0	0	x	x	0	x	1
1	0	1	0	1	1	0	1	x	x	0	0	x
1	1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	1	1	0	1	x	0	x	1	x	0
1	1	0	1	1	0	0	x	0	0	x	x	1
1	1	0	0	0	0	0	x	1	0	x	0	x

Circuit Excitation Table

- For D flip-flops:



Input	Curr. State			Next State			Q2	Q1	Q0
R	**2**	**1**	**0**	**2**	**1**	**0**	**D**	**D**	**D**
0	x	x	x	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	1
1	0	1	1	0	1	0	0	1	0
1	0	1	0	1	1	0	1	1	0
1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	0	1	1	0	1
1	1	0	1	1	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0

Basic Steps

1) Describe the state machine based on the design spec
   - 3 different ways ✓

2) Choose flip-flops ✓

3) Create circuit excitation table ✓

4) **Construct K-maps for:**
   - **flip-flop inputs**
   - **primary outputs**

5) **Obtain minimized SOP equations**

6) Draw logic diagram

7) Simulate to verify design & debug as needed

8) Perform circuit analysis & logic optimization

Create K-Maps

- Each flip-flop input is driven by a combinational circuit
- Inputs to each circuit are:
  - the current state as indicated by the current flip-flop outputs
  - External inputs (reset in this case) used to control which transtion occurs

JK K-Maps

J2

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	x	x	x	x
10	0	0	0	1

J1

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	x	x
10	0	1	x	x

J0

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	x	x	1
10	1	x	x	0

K2

R,Q2\ Q1,Q0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	0	0	0
10	x	x	x	x

$J2=R\cdot Q1\cdot\overline{Q0}$

$K2=\overline{R}+\overline{Q1}\cdot\overline{Q0}$

K1

R,Q2\ Q1,Q0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	1	0
10	x	x	0	0

$J1=R\cdot\overline{Q2}\cdot Q0$

$K1=\overline{R}+Q2\cdot Q0$

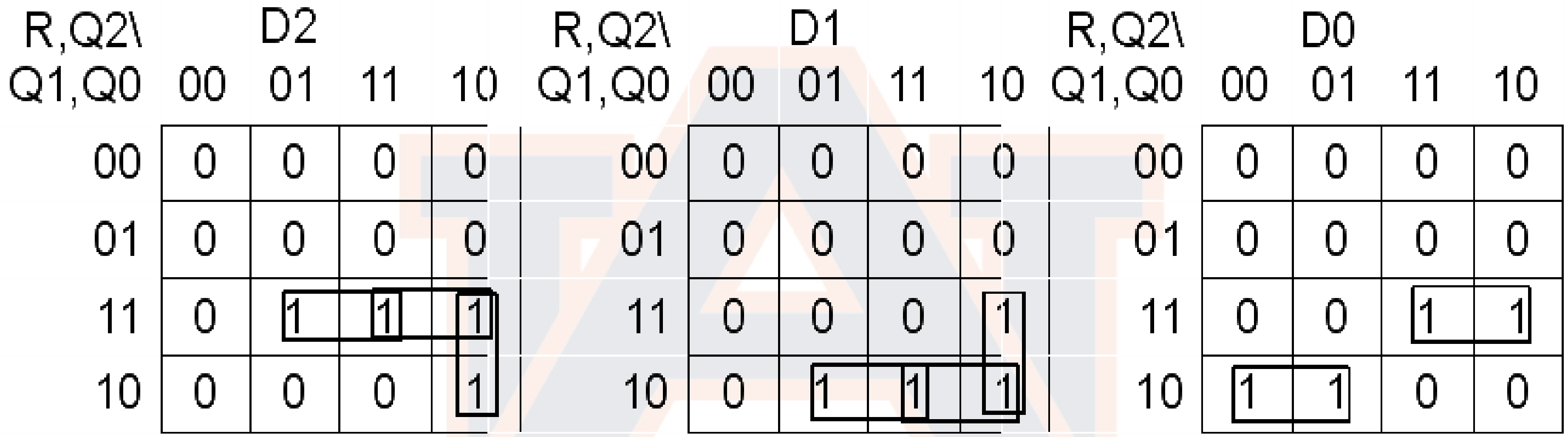
K0

R,Q2\ Q1,Q0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	1	0	x
10	x	0	1	x

$J0=R\cdot\overline{Q2}\cdot\overline{Q1}+R\cdot Q2\cdot Q1$

$K0=\overline{R}+Q2\cdot\overline{Q1}+\overline{Q2}\cdot Q1$

D K-Maps



D2

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	0	1

D1

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	1
10	0	1	1	1

D0

R,Q2\ Q1,Q0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	1	1	0	0

$D2 = R \cdot Q2 \cdot Q0 + R \cdot Q2 \cdot Q1 + R \cdot Q1 \cdot \overline{Q0}$

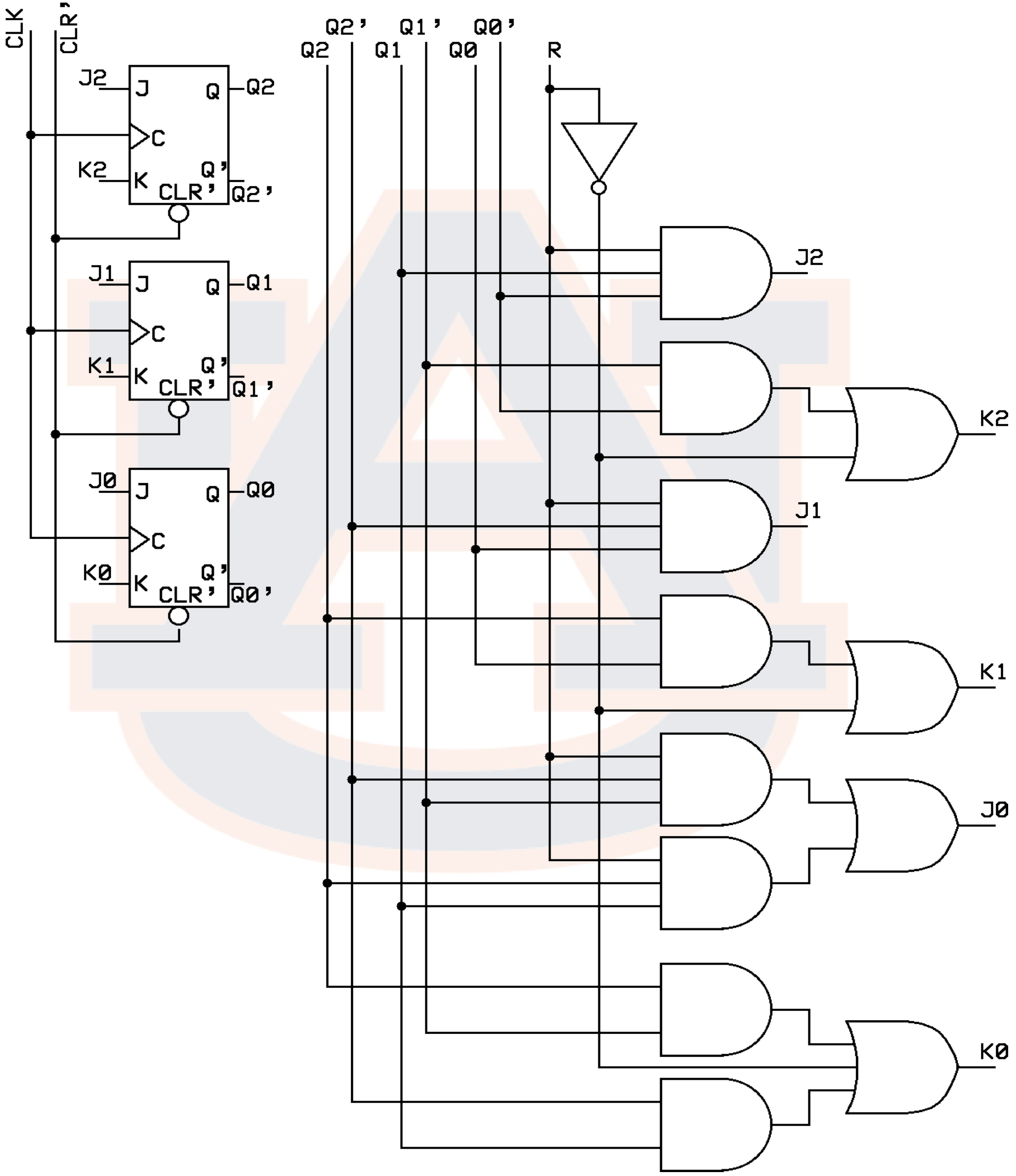
$D0 = R \cdot \overline{Q2} \cdot \overline{Q1} + R \cdot Q2 \cdot Q1$

$D1 = R \cdot \overline{Q2} \cdot Q0 + R \cdot \overline{Q2} \cdot Q1 + R \cdot Q1 \cdot \overline{Q0}$

Basic Steps

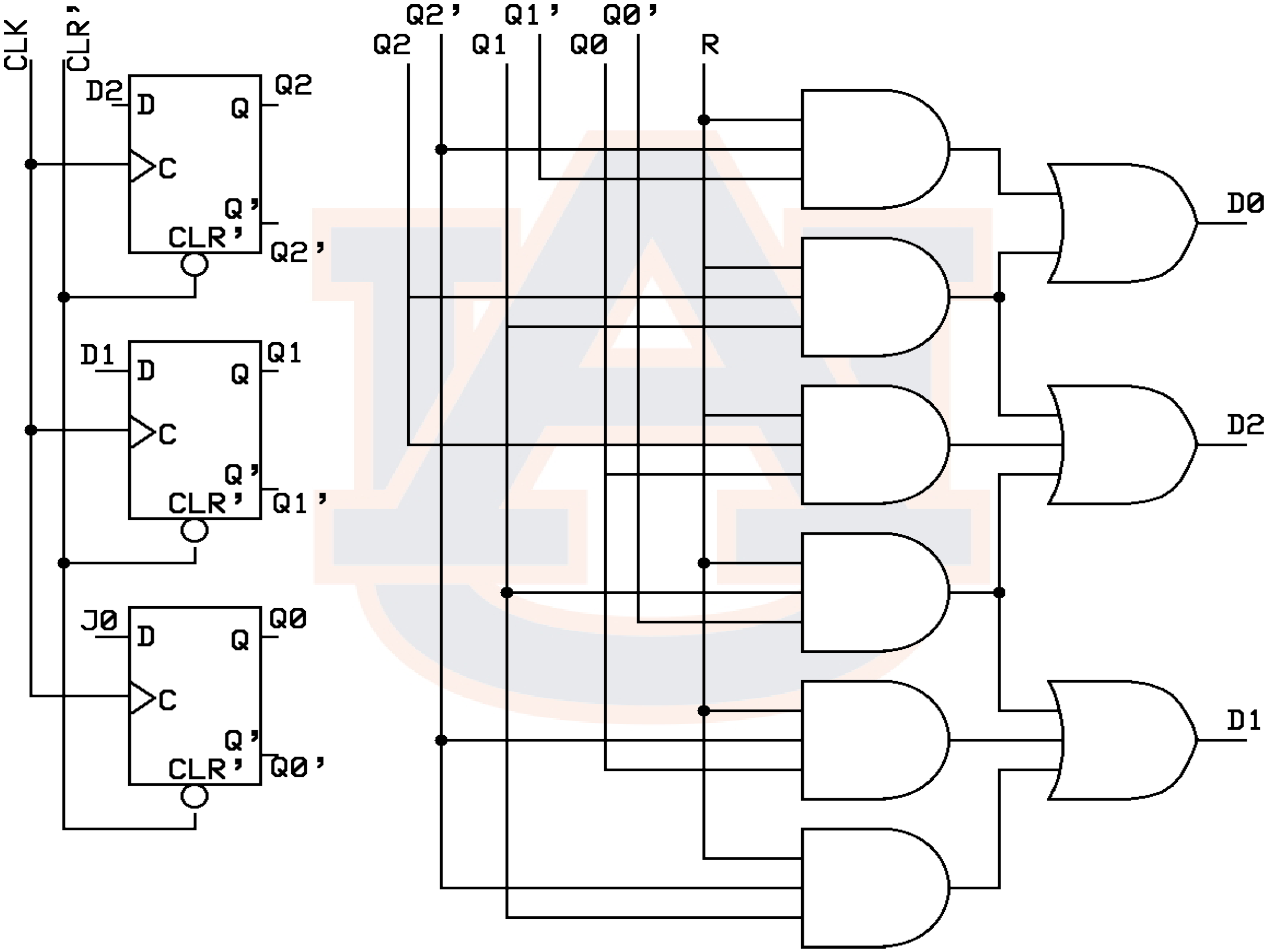
1) Describe the state machine based on the design spec
   - 3 different ways ✓
2) Choose flip-flops ✓
3) Create circuit excitation table ✓
4) Construct K-maps for: ✓
   - flip-flop inputs
   - primary outputs
5) Obtain minimized SOP equations ✓
6) **Draw logic diagram**
7) Simulate to verify design & debug as needed
8) Perform circuit analysis & logic optimization

Circuit Diagram – See Handout



David J. Broderick

Circuit Diagram – See Handout



Basic Steps

1) Describe the state machine based on the design spec
   - 3 different ways ✓
2) Choose flip-flops ✓
3) Create circuit excitation table ✓
4) Construct K-maps for: ✓
   - flip-flop inputs
   - primary outputs
5) Obtain minimized SOP equations ✓
6) Draw logic diagram ✓
7) **Simulate to verify design & debug as needed**
8) Perform circuit analysis & logic optimization

Simulation

- AUSIM can simulate sequential circuits too
- Flip-flops in AUSIM:
  - dff: FF1 in: CLK D out: Q QN ; (D flip-flop)
  - jkff: FF2 in: CLK J K out: Q QN ; (JK flip-flop)
  - srff: FF3 in: CLK S R out: Q QN ; (SR flip-flop)
- Q's and QN's become input to the combinational circuit designed above in addition to R
- Input to this circuit is the Clock(C) and the Reset (R)
- Output of this circuit is Q2,Q1,Q0

Simulation

DFF Circuit

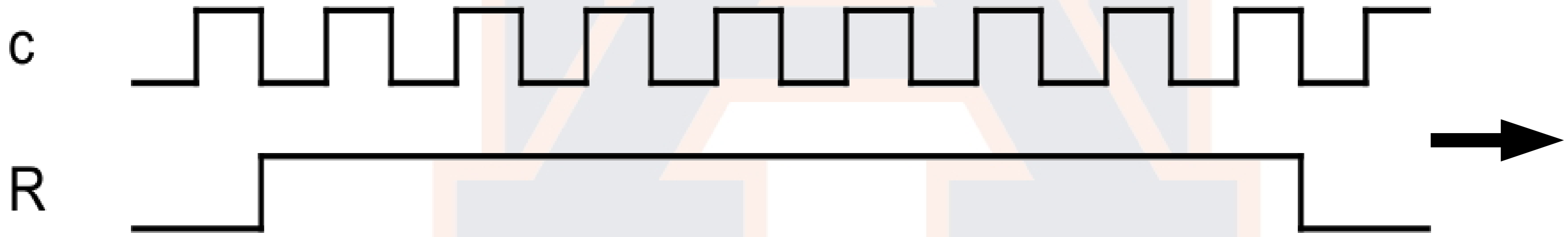
```
ckt: 3bitgray in: c R out: Q2 Q1 Q0 ;
dff: DFF2 in: c d2 out: Q2 Q2N ;
dff: DFF1 in: c d1 out: Q1 Q1N ;
dff: DFF0 in: c d0 out: Q0 Q0N ;
and: G1 in: R Q2 Q0 out: G1 ;
and: G2 in: R Q2 Q1 out: G2 ;
and: G3 in: R Q1 Q0N out: G3 ;
or: G4 in: G1 G2 G3 out: d2 ;
and: G5 in: R Q2N Q0 out: G5 ;
and: G6 in: R Q2N Q1 out: G6 ;
or: G7 in: G3 G5 G6 out: d1 ;
and: G8 in: R Q2N Q1N out: G8 ;
or: G9 in: G2 G8 out: d0 ;
```

JKFF Circuit

```
ckt: 3bitgray in: c R out: Q2 Q1 Q0 ;
jkff: JKFF2 in: c J2 K2 out: Q2 Q2N ;
jkff: JKFF1 in: c J1 K1 out: Q1 Q1N ;
jkff: JKFF0 in: c J0 K0 out: Q0 Q0N ;
not: G0 in: R out: RN ;
and: G1 in: R Q1 Q0N out: J2 ;
and: G2 in: Q1N Q0N out: G2 ;
or: G3 in: RN G2 out: K2 ;
and: G4 in: R Q2N Q0 out: J1 ;
and: G5 in: Q2 Q0 out: G5 ;
or: G6 in: RN G5 out: K1 ;
and: G7 in: R Q2N Q1N out: G7 ;
and: G8 in: R Q2 Q1 out: G8 ;
or: G9 in: G7 G8 out: J0 ;
and: G10 in: Q2 Q1N out: G10 ;
and: G11 in: Q2N Q1 out: G11 ;
or: G12 in: RN G10 G11 out: K0 ;
```

Test Vectors

- Unlike with combinational design, order of vectors now matters
- Think of the list of vectors (top to bottom) as you would a timing diagram (left to right)



- Ideally each transition will be tested
  - 90 vectors!

cR
00
10
01
11
01
11
01
11
01
11
01
11
01
11
01
11
01
11
00
10

Output

- Unlike with combinational design, state of output can be unknown
    - Indicated by a '2' in the .out file
- Using the same vector file output of both circuits should be identical
- Let's take a look...

Basic Steps

1) Describe the state machine based on the design spec
   - 3 different ways ✓

2) Choose flip-flops ✓

3) Create circuit excitation table ✓

4) Construct K-maps for: ✓
   - flip-flop inputs
   - primary outputs

5) Obtain minimized SOP equations ✓

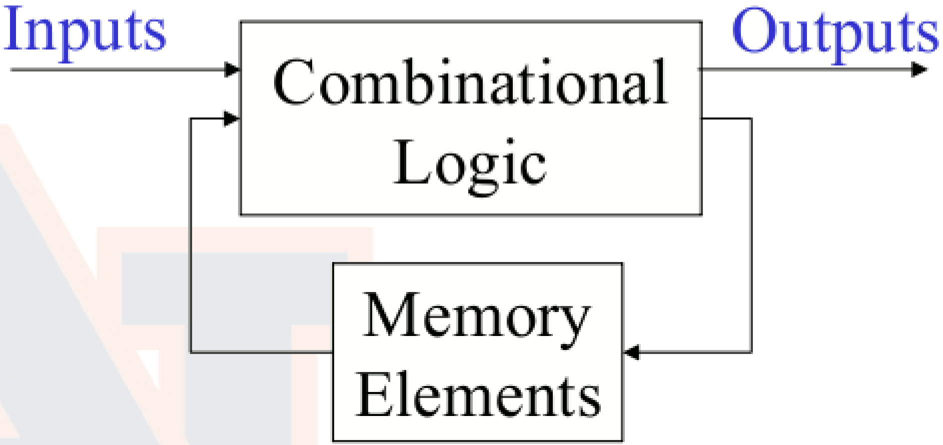
6) Draw logic diagram ✓

7) Simulate to verify design & debug as needed ✓

8) Perform circuit analysis & logic optimization

Analysis & Optimization

- To minimize the area of the circuit:
    - Use the smallest number of memory elements
    - Minimize area of combinational logic
- For timing analysis
    - With less control over the flip-flop propagation delay focus on combinational logic
    - Path analysis from inputs to outputs
        - Inputs: flip-flop outputs (Q's), external inputs ®
        - Outputs: flip-flop inputs (J, K, D's), external outputs (not present in this example)
    - Consider each path...



Analysis & Optimization

- JK circuit
  - $G=13$
  - $G_{IO}=43$
  - $G_{del}=2$
    All paths
  - $P_{del}=13$
    R->K0

- D circuit
  - $G=9$
  - $G_{IO}=35$
  - $G_{del}=2$
    All paths
  - $P_{del}=15$
    R->All outputs

- JK is faster, D is smaller
- What effect do the flip-flops have?

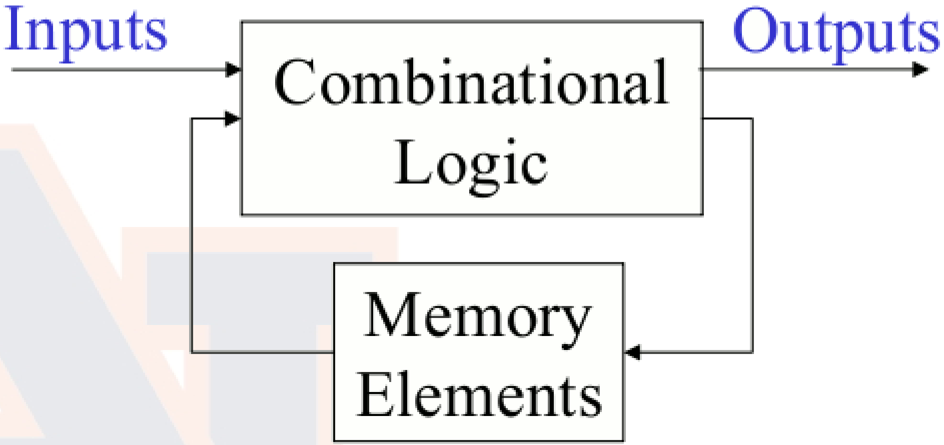
Analysis & Optimization

- These circuits are not necessarily optimal
    - Make changes
    - Perform analysis again to show improvement
    - **Re-**check design (simulation)

Basic Steps

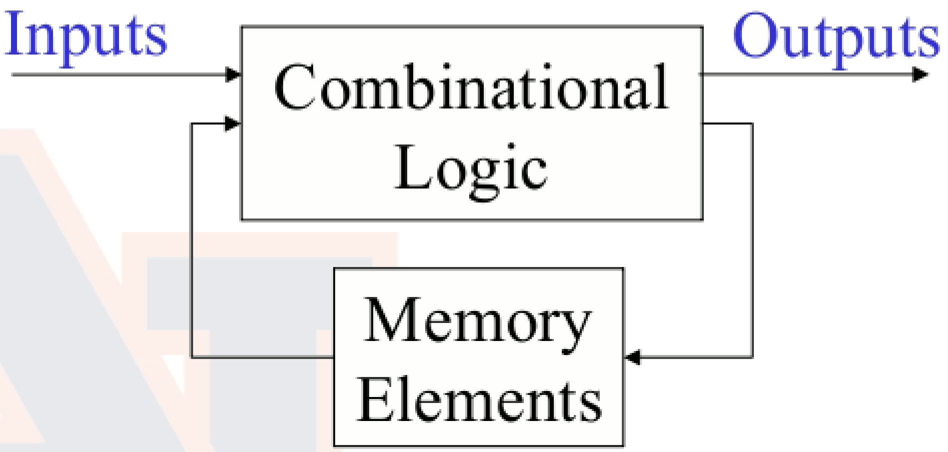
1) Describe the state machine based on the design spec
   - 3 different ways ✓
2) Choose flip-flops ✓
3) Create circuit excitation table ✓
4) Construct K-maps for: ✓
   - flip-flop inputs
   - primary outputs
5) Obtain minimized SOP equations ✓
6) Draw logic diagram ✓
7) Simulate to verify design & debug as needed ✓
8) **Perform circuit analysis & logic optimization** ✓

Huffman: Mealy & Moore



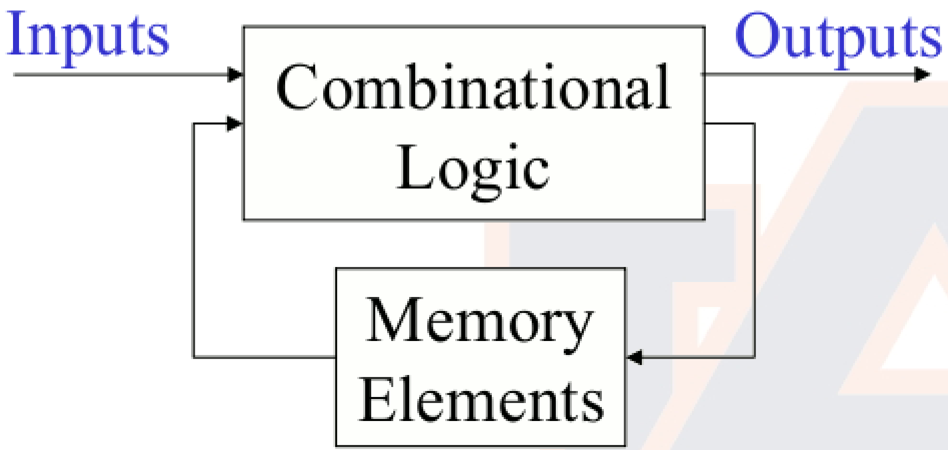
- Outputs are not always the state of the machine directly
- Huffman Model divided into two sub-types
- Mealy Model
  - Outputs dependent on current state **and** inputs
- Moore Model
  - Outputs dependent on current state **only**

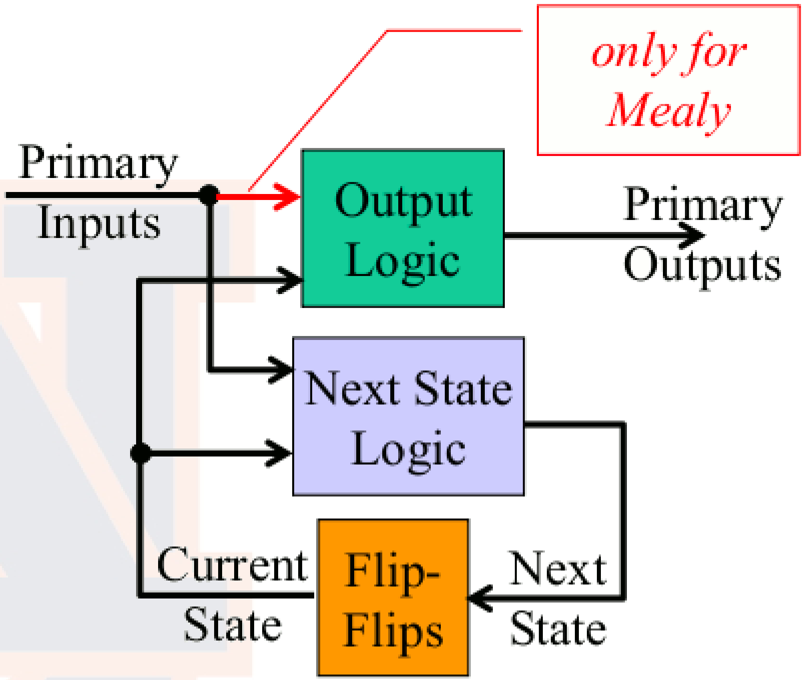
Huffman: Mealy & Moore



- Outputs are not always the state of the machine directly
- Huffman Model divided into two sub-types
- Mealy Model
    - Outputs dependent on current state **and** inputs
- Moore Model
    - Outputs dependent on current state **only**

Huffman: Mealy & Moore





- More detail in Huffman model reveals difference
- Next state logic the same for Mealy and Moore. Only output logic differs

Mealy Example

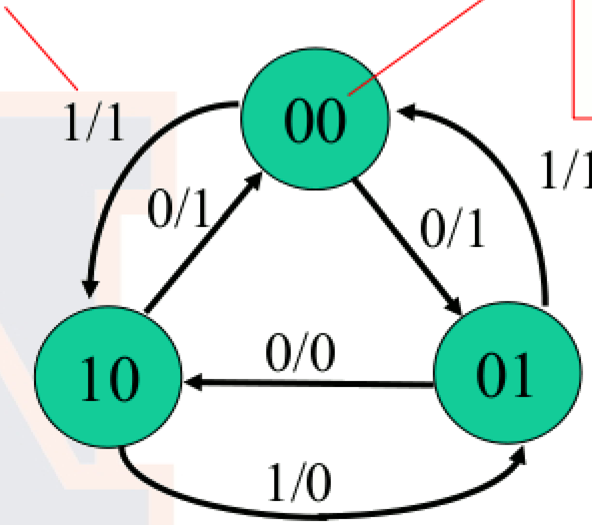
In \ X Y	00	01	11	10
0	0	1	X	0
1	1	0	X	0

In \ X Y	00	01	11	10
0	1	0	X	0
1	0	0	X	1

$D_X = In'Y + InX'Y'$ $D_Y = InX + In'X'Y'$

Input	Curr. State		Next State		X	Y	Output
In	X	Y	X	Y	D	D	
0	0	0	0	1	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	0	0	0	0	1
1	1	0	0	1	0	1	0
0	1	1	X	X	X	X	X

Input / Output



State order XY

In \ X Y	00	01	11	10
0	1	0	X	1
1	1	1	X	0

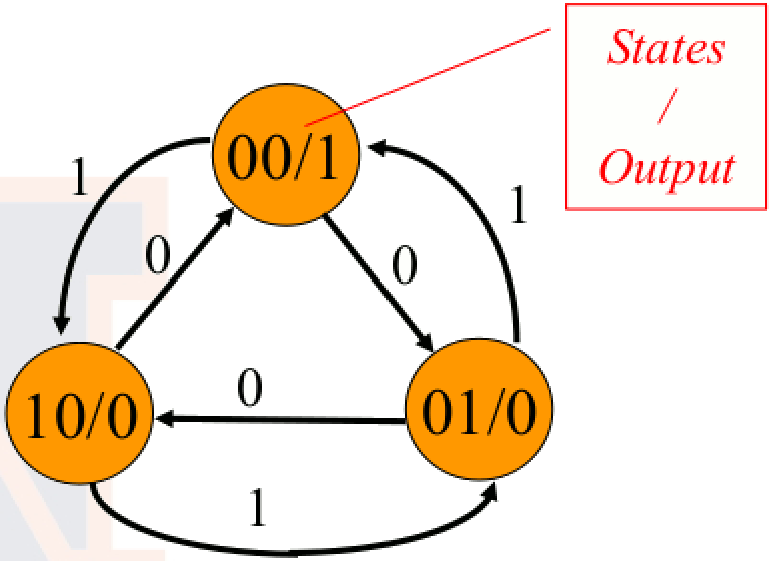
$O_{Mealy} = In'Y' + InX'$

Moore Example

In \ X Y	00	01	11	10
0	0	1	X	0
1	1	0	X	0

In \ X Y	00	01	11	10
0	1	0	X	0
1	0	0	X	1

$D_X = In'Y + InX'Y'$ $D_Y = InX + In'X'Y'$



States / Output

Input	Curr. State		Next State		X	Y	Output
In	X	Y	X	Y	D	D	
0	0	0	0	1	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	0	0	0	0
1	1	0	0	1	0	1	0
0	1	1	X	X	X	X	X

In \ X Y	00	01	11	10
0	1	0	X	0
1	1	0	X	0

$O_{Moore} = X'Y'$

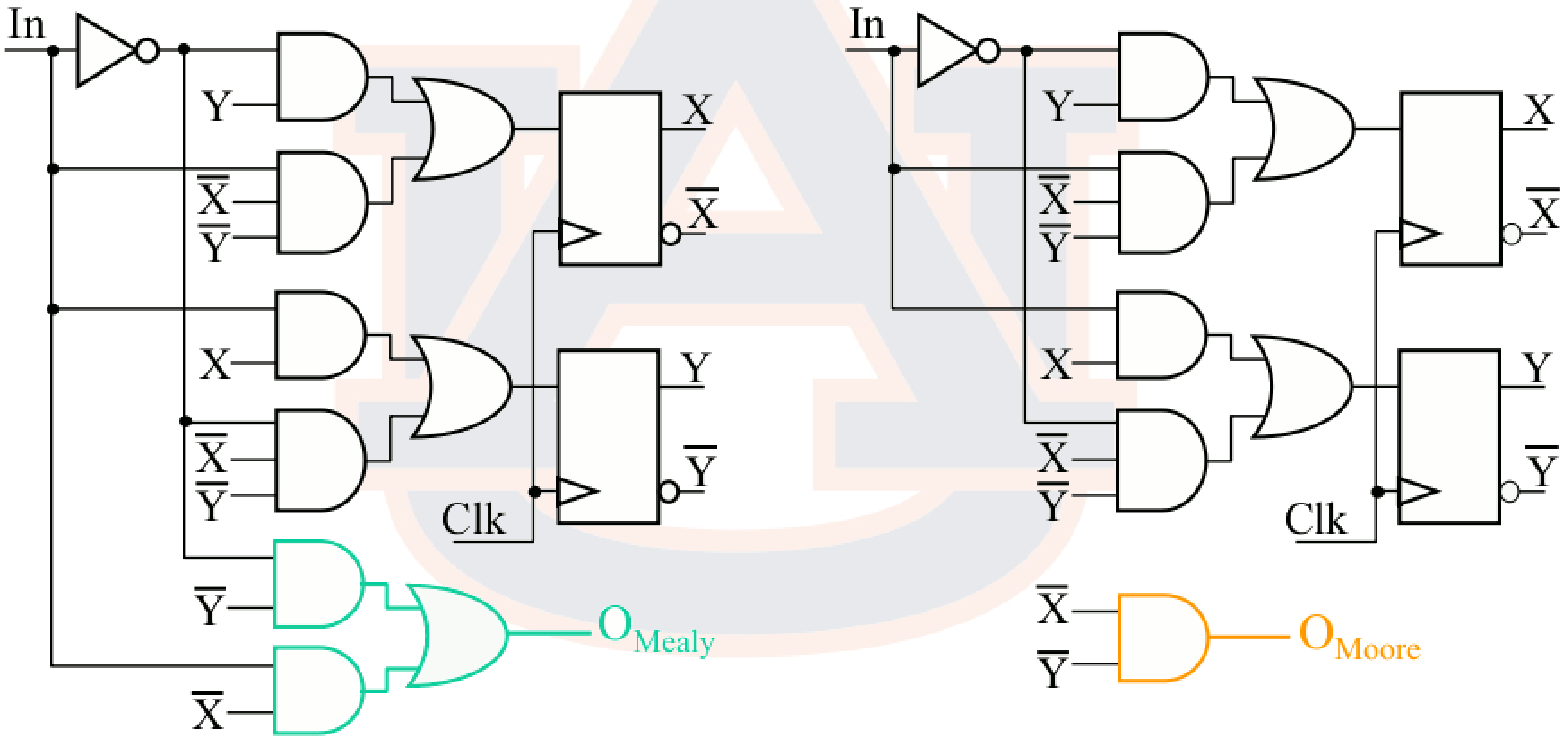
Mealy & Moore Circuits

$D_X = In'Y + InX'Y'$

$O_{Mealy} = In'Y' + InX'$

$O_{Moore} = X'Y'$

$D_Y = InX + In'X'Y'$



- Note: Similarity in next state logic, difference in output

Mealy/Moore Summary

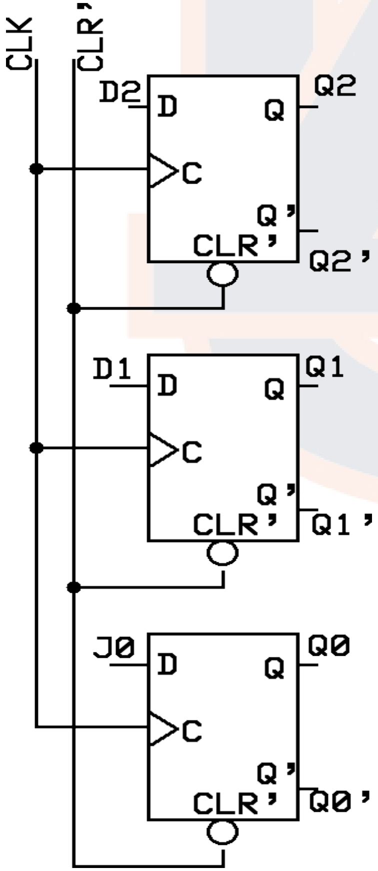
- Mealy
  - Outputs dependent on current state **and** inputs
  - More flexible model, can result in fewer states
  - Care must be taken not to transition inputs near time to read outputs
- Moore
  - Outputs dependent on current state **only**
  - Less flexible model, may require more states
  - Output only changes on clock, easier to time reads of outputs

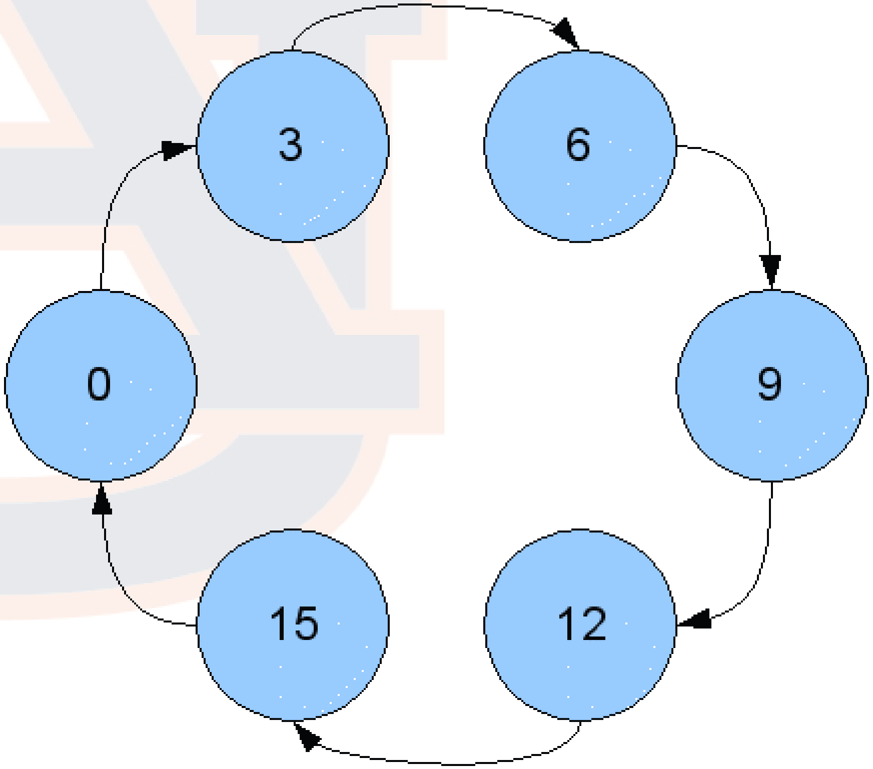
Example

- Design a synchronous circuit that will count from 0 to 15 by 3's and then rollover to 0. Output should change on the positive edge of the clock.

State Diagram/Flip-Flops

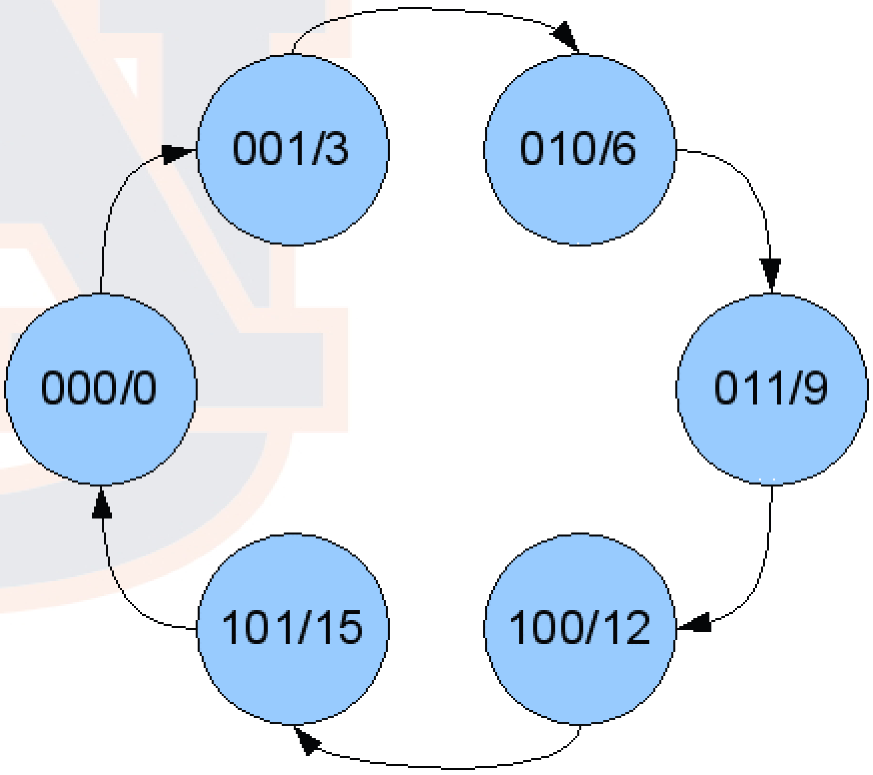
- Could use state to represent output directly (4 FFs)
- With only six states 3 bits could represent state with smaller circuit (3 FFs)





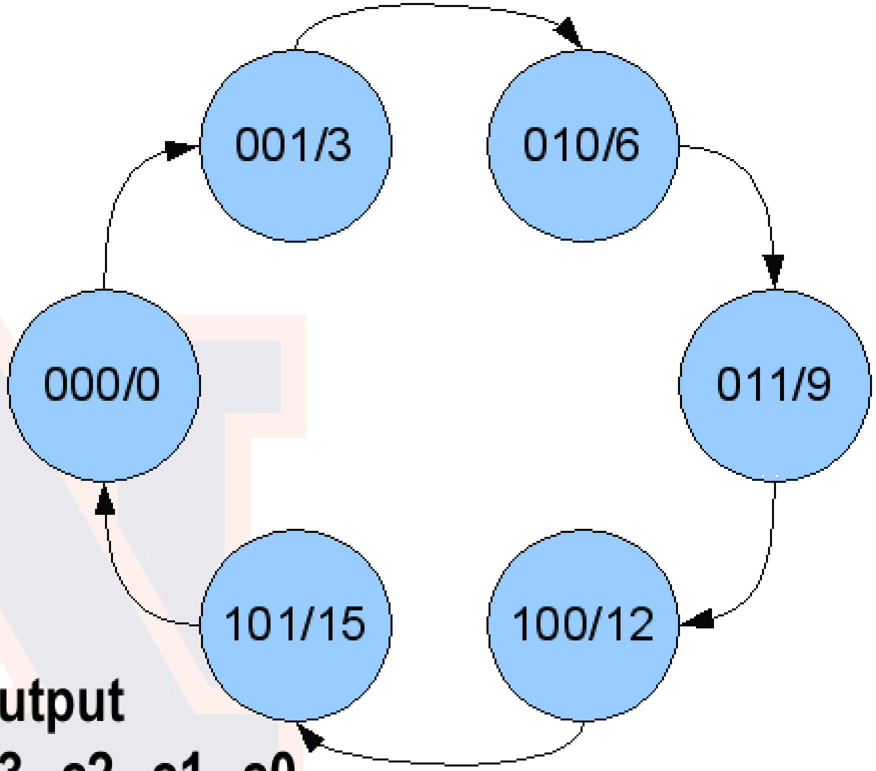
State Diagram/ Outputs

- Using a Moore model, a 3 bit value is assigned to each state
- Output of each state is not changed
- Additional expressions needed to convert from state to output



Excitation Table

- Table indicates order of states **and** output for each current state



Curr. State			Next State			Excitation			Output			
Q2	Q1	Q0	Q2	Q1	Q0	D2	D1	D0	c3	c2	c1	c0
0	0	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0	0	1	1
0	1	0	0	1	1	0	1	1	0	1	1	0
0	1	1	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	1	1	0	0
1	0	1	0	0	0	0	0	0	1	1	1	1

Next State K-maps

- [FF Inputs]=f(Current State, External Inputs)

Curr. State			Next State			Excitation			Output			
Q2	Q1	Q0	Q2	Q1	Q0	D2	D1	D0	c3	c2	c1	c0
0	0	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0	0	1	1
0	1	0	0	1	1	0	1	1	0	1	1	0
0	1	1	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	1	1	0	0
1	0	1	0	0	0	0	0	0	1	1	1	1

D2

Q2\Q1, Q0	00	01	11	10
0	0	0	1	0
1	1	0	X	X

$D2=Q2\cdot\overline{Q0}+Q1\cdot Q0$

D1

Q2\Q1, Q0	00	01	11	10
0	0	1	0	1
1	0	0	X	X

$D1=\overline{Q2}\cdot\overline{Q1}\cdot Q0+Q1\cdot\overline{Q0}$

D0

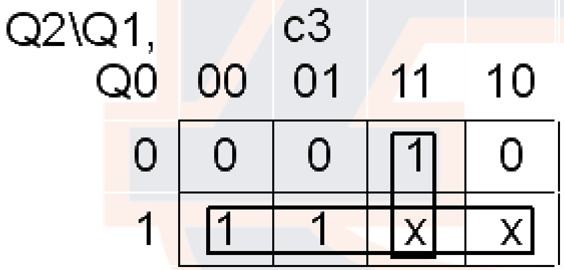
Q2\Q1, Q0	00	01	11	10
0	1	0	0	1
1	1	0	X	X

$D0=\overline{Q0}$

Output K-maps

- [Outputs]=f(Current State)

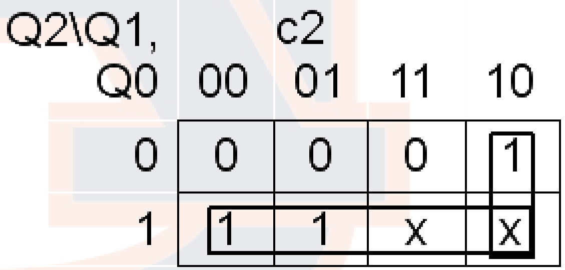
Curr. State			Next State			Excitation			Output			
Q2	Q1	Q0	Q2	Q1	Q0	D2	D1	D0	c3	c2	c1	c0
0	0	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0	0	1	1
0	1	0	0	1	1	0	1	1	0	1	1	0
0	1	1	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	1	1	0	0
1	0	1	0	0	0	0	0	0	1	1	1	1



c3

Q2\Q1, Q0	00	01	11	10
0	0	0	1	0
1	1	1	x	x

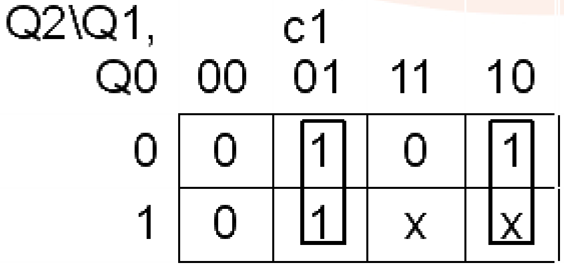
c3=Q2+Q1·Q0



c2

Q2\Q1, Q0	00	01	11	10
0	0	0	0	1
1	1	1	x	x

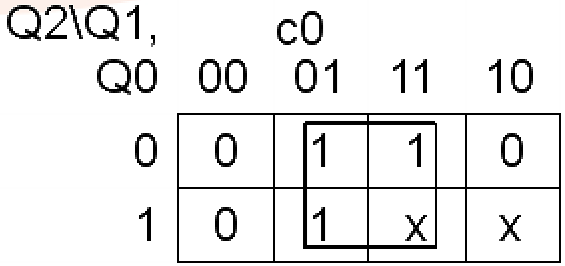
c2=Q2+Q1·$\overline{Q0}$



c1

Q2\Q1, Q0	00	01	11	10
0	0	1	0	1
1	0	1	x	x

c1=$\overline{Q1}$·Q0+Q1·$\overline{Q0}$

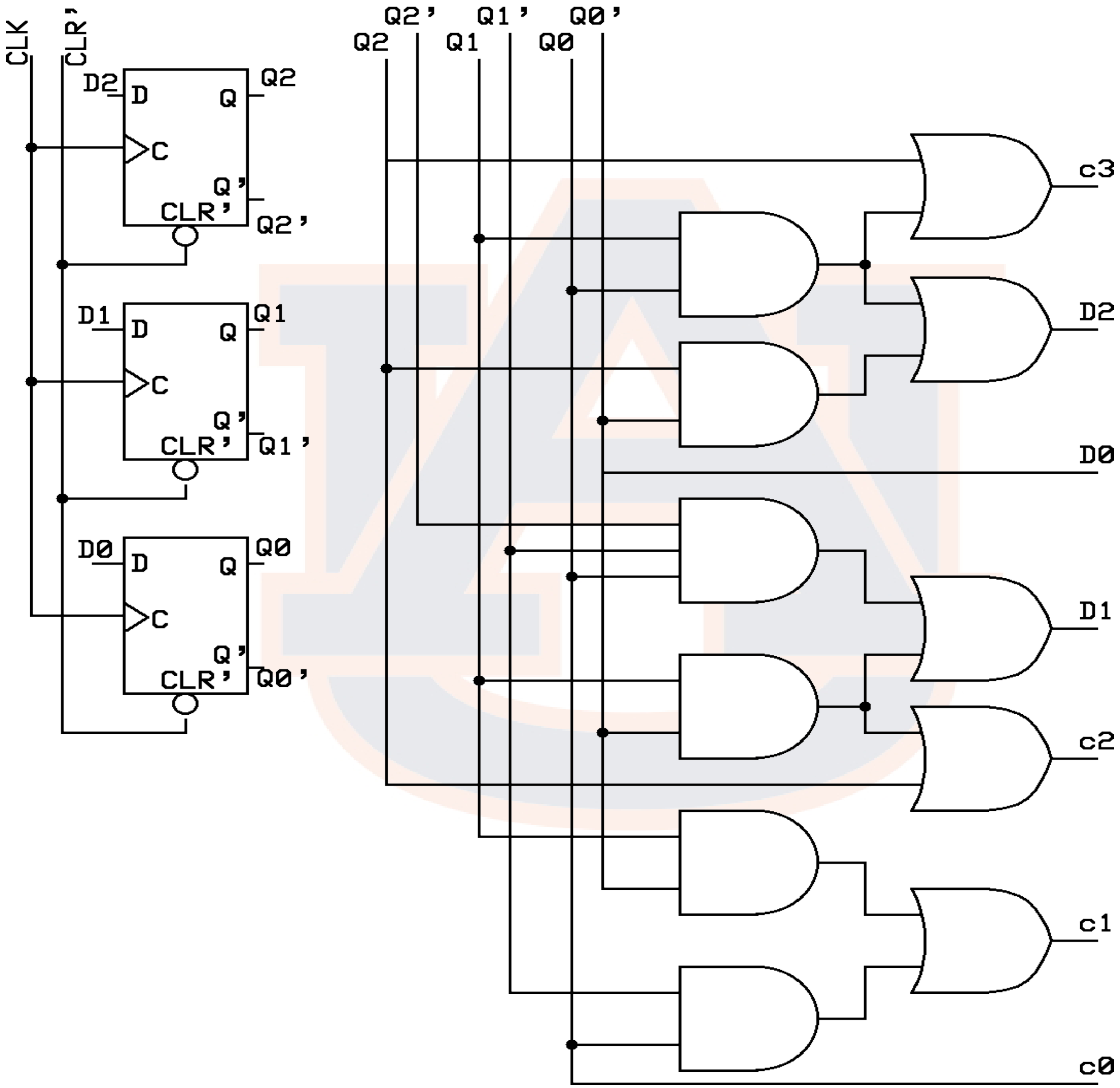


c0

Q2\Q1, Q0	00	01	11	10
0	0	1	1	0
1	0	1	x	x

c0=Q0

Schematic


CLK
CLR'
D2
D
Q
Q2
C
Q'
CLR'
Q2'
D1
D
Q
Q1
C
Q'
CLR'
Q1'
D0
D
Q
Q0
C
Q'
CLR'
Q0'
Q2'
Q1'
Q0'
Q2
Q1
Q0
c3
D2
D0
D1
c2
c1
c0

Keypad Example

Design a sequential circuit to recognize a three digit combination entered on a keypad. The keypad has four buttons labeled 0 through 3. When a key is pressed the binary value of the key is available as the two bit number, b_1b_0. Once b_1 and b_0 are stable, a clock pulse will occur (clk). The combination to open the lock will be 213_{10}.