

## Sequential Circuit Design

---

### Objectives

This section deals with the design of sequential circuits including the following:

- A discussion of the construction of state/output tables or diagrams from a word description or flow chart specification of sequential behavior
- A formal synthesis technique for realizing state tables and diagrams
- A less formal technique based on transition equations

### Reading Assignment

- Sections 3.3 and 3.4.

## 1. State Table/Diagram Specification

---

- There is no algorithmic way to construct the state table from a word description of the circuit. Instead, we provide a few examples to illustrate the technique.
- It is convenient to group sequential circuits as to whether the
  - generate sequences,
  - detect sequences, or
  - transform sequences

□ Sequence Generation:

- Example: An up/down mod 6 counter: Six states and two inputs X & Y. The inputs have the following effect:

XY = 00: Do nothing (no state changes)

XY = 01: Count up

XY = 11: Reset to state 0

XY = 10: Count down

	00	01	11	10
0	0	1	0	5
1	1	2	0	0
2	2	3	0	1
3	3	4	0	2
4	4	5	0	3
5	5	0	0	4

□ Example: An arbitrary sequence generator

X = 0: 000 → 010 → 011 → 010 → 100

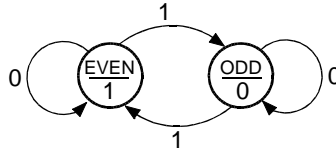
X = 1: 000 → 100 → 110 → 100

Q	X		Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>
	0	1	
0	1	5	000
1	2	0	010
2	3	0	011
3	4	0	010
4	0	0	100
5	0	6	100
6	0	7	110
7	0	0	100

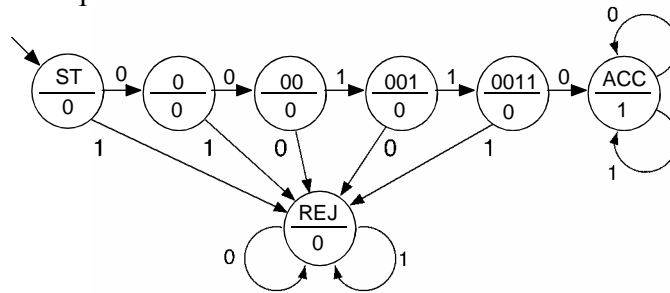
Q\*

□ Sequence detection

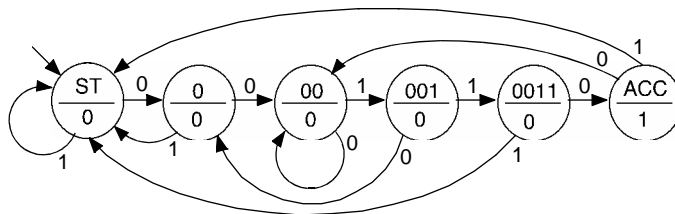
- Example: Detect an even number of 1s



- Example: Detect 00110

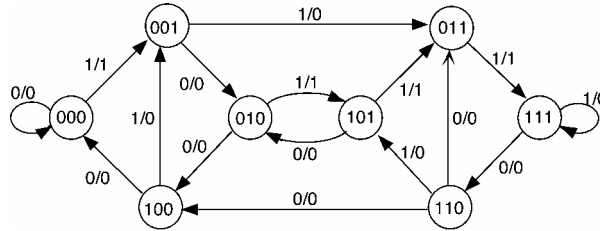


- Alternative Solution



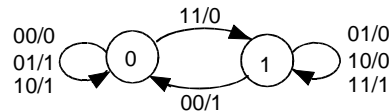
- Example: Universal length 4 sequence detector

- ◆ This one detects 1011 or 0101 or 0001 or 0111



- Sequence transformation

- Serial binary adder (arbitrary length operands)



## 2. Formal Sequential Circuit Synthesis

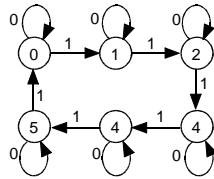
- Summary of Design Steps

- Assign state vectors to states; the state assignment problem
  - Construct the transition table
  - Pick a flip-flop type for each state variable and use the flip-flop's application tables to construct an excitation table
  - Derive excitation and output equations from the excitation table.
  - Construct a logic diagram from the excitation and output equations.

## □ State Assignment

- Any assignment of  $\lceil \log_2 n \rceil$  state variables will work, but different ones can give radically different circuits.

◆ Example: mod 6 counter



Assignment #1: Binary Counter

	0	1
0	000	000 001
1	001	001 010
2	010	010 011
3	011	011 100
4	100	100 101
5	101	101 000
110	xxx	xxx
111	xxx	xxx

Assignment #2: Johnson Counter

	0	1
0	000	000 001
1	001	001 011
2	011	011 111
3	111	111 110
4	110	110 100
5	100	100 000
101	xxx	xxx
010	xxx	xxx

Assignment #3: Ring Counter

	0	1
0	000001	000001 000010
1	000010	000010 000100
2	000100	000100 001000
3	001000	001000 010000
4	010000	010000 100000
5	100000	100000 000001
58 other rows	xxxxxx	xxxxxx

- Number of possible state assignments:

◆ Assume we have  $k$  bits to encode  $n$  states (of course  $k \geq \lceil \log_2 n \rceil$ ). Then we have  $m = 2^k$  possible state vectors.

◆ The number of ways to pick  $n$  of the  $m$  state vectors to encode the states is given by

$$\binom{n}{m} = \frac{n!}{m! \cdot (n-m)!}$$

◆ Given  $m$  state vectors, we can pick any one for the first state, any one of the remaining  $m-1$  for the second state, etc. In general we can assign the  $m$  state vectors to states in  $m!$  ways.

◆ Therefore the number of different state assignments for encoding  $n$  states with  $m$  state vectors (where  $m = 2^k$ ) is

$$\frac{n!}{m! \cdot (n-m)!} \cdot m!$$

◆ For  $n = 5$  and  $m = 8$  we have 6720 different possible state assignments.

◆ Trying all possible state assignments to find the best one is not an option.

- There is no known way to get the best assignment other than trying every possible one, which is not practical.
  - ◆ Note that if we took this approach we would have to do a complete design for each assignment and compare all the resulting designs.
- Guidelines for selecting good (but not necessarily optimal) state assignments:
  - ◆ Choose 0...00 or 1...11 for the initial reset state.
  - ◆ Minimize the number of state variables that change on each transition.
  - ◆ Maximize the number of state variables that don't change in a group of related states (a group in which most transitions stay in the group)
  - ◆ Consider using more than the minimum number of state variables. E.g., one-hot assignments.

□ Construct the transition table from the state table or state graph and the chosen state assignment.

■ Example 1.

Q	X		Y
	0	1	
A	A	B	0
B	A	C	0
C	A	C	1

State Table

A - 00  
 B - 01  
 C - 10  
 State Assignment

Q <sub>1</sub> Q <sub>0</sub>	X		Y
	0	1	
00	00	01	0
01	00	10	0
10	00	10	1
11	dd	dd	d

Transition Table

□ Example 2

Q	XY			XY			State Assignment
	00	01	10	00	01	10	
0	0	1	5	0	0	1	0 - 000
1	0	2	0	0	0	0	1 - 001
2	0	3	1	0	0	0	2 - 010
3	0	4	2	0	0	0	3 - 011
4	0	5	3	0	0	0	4 - 100
5	0	0	4	0	1	0	5 - 101

Q\* Z

State Table

Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	XY			XY		
	00	01	10	00	01	10
000	000	001	101	0	0	1
001	000	010	000	0	0	0
010	000	011	001	0	0	0
011	000	100	010	0	0	0
100	000	101	011	0	0	0
101	000	000	100	0	1	0
110	ddd	ddd	ddd	d	d	d
111	ddd	ddd	ddd	d	d	d

Q\*<sub>2</sub>Q\*<sub>1</sub>Q\*<sub>0</sub> Z

Transition Table

□ Select the Flip-Flop Type

- The four main types of flip-flops are SR, D, T and JK.

- ◆ The choice of flip-flop type can affect the complexity of the combinational logic in the resulting sequential circuit.

- ◆ Of three common types, the most versatile is the JK, since it can be easily converted into the other two.

- Any one can be converted into one of the other types, but some of these conversions take more logic than others.

- ◆ The T flip-flop is the one most suitable for counters, since it usually results in less logic than the other types.

- This is because extra logic internal to the flip-flop is exactly what is needed to implement counters.

- ◆ D flip-flops are the ones found in almost all PLDs.

- If your design is targeted for a PLD, you are usually stuck with D flip-flops.

□ Construct the Excitation Table

- Recall that the excitation table specifies the values for the flip-flop input signals needed to cause the transitions in the transition table.

■ Flip-Flop Application Table

- ◆ These tables list the flip-flop inputs signals needed for each possible transition of the flip-flop's state.

$Q \rightarrow Q^*$	D	$Q \rightarrow Q^*$	S R	$Q \rightarrow Q^*$	T	$Q \rightarrow Q^*$	J K
0 → 0	0	0 → 0	0 d	0 → 0	0	0 → 0	0 d
0 → 1	1	0 → 1	1 d	0 → 1	1	0 → 1	1 d
1 → 0	0	1 → 0	d 1	1 → 0	1	1 → 0	d 1
1 → 1	1	1 → 1	d 0	1 → 1	0	1 → 1	d 0

D Flip-Flop
SR Flip-Flop
T Flip-Flop
JK Flip-Flop

- Use these flip-flop application tables to convert a transition table into an excitation table.

■ Example 1

- ◆ Chose JK flip-flops for both state variables to get the following:

$Q_1Q_0$	X		Y
	0	1	
00	00	01	0
01	00	10	0
10	00	10	1
11	dd	dd	d

$Q^*_1Q^*_2$   
Transition Table

$Q_1Q_0$	X		Y
	0	1	
00	0d 0d	0d 1d	0
01	0d d1	1d d1	0
10	d1 0d	d0 0d	1
11	dd dd	dd dd	d

$J_1K_1J_0K_0$   
Excitation Table

- ◆ Note the rather high percentage of don't care entries. This is common with JK flip-flops.
- ◆ Note that had we used D flip-flops the transition table and excitation tables would have had the same entries.

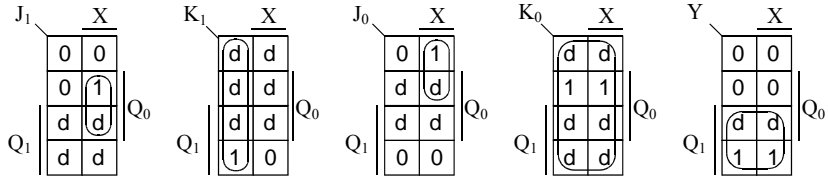


■ Example 1

$Q_1Q_0$	X		Y
	0	1	
00	0d 0d	0d 1d	0
01	0d d1	1d d1	0
10	d1 0d	d0 0d	1
11	dd dd	dd dd	d

$J_1K_1J_0K_0$

Excitation Table



$J_1 = X \cdot Q_0$

$K_1 = X'$

$J_0 = X \cdot Q_1'$

$K_0 = 1$

$Y = Q_1$

□ Construct the logic diagram

■ Example 1

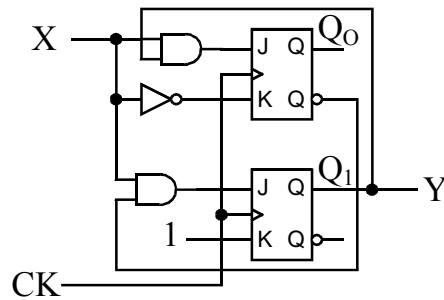
$J_1 = X \cdot Q_0$

$K_1 = X'$

$J_0 = X \cdot Q_1'$

$K_0 = 1$

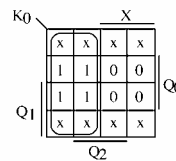
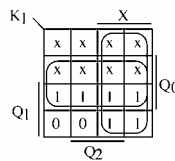
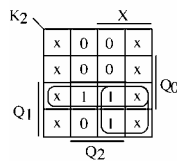
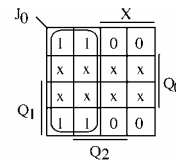
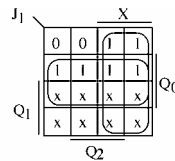
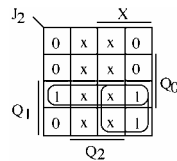
$Y = Q_1$



■ Example 2

$Q_2 Q_1 Q_0$	0	X	1
0 0 0	0x 0x 1x	0x 1x 0x	0x
0 0 1	0x 1x x1	0x 1x x0	0x
0 1 0	0x x0 1x	1x x1 0x	0x
0 1 1	1x x1 x1	1x x1 x0	0x
1 0 0	x0 0x 1x	x0 1x 0x	0x
1 0 1	x0 1x x1	x0 1x x0	0x
1 1 0	x0 x0 1x	x1 x1 0x	0x
1 1 1	x1 x1 x1	x1 x1 x0	0x

$J_2 K_2, J_1 K_1, J_0 K_0$



$$J_2 = K_2 = Q_0 \cdot Q_1 + X \cdot Q_1$$

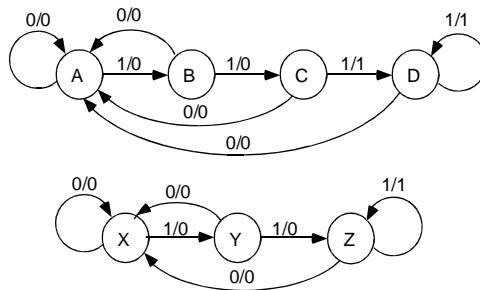
$$J_1 = K_1 = X + Q_0$$

$$J_0 = K_0 = X'$$

### 3. Sequential Circuit State Reduction

- This section presents a technique for reducing the number of states in a state table.
  - Material on this topic can be found in Section 8.6 in Brown and Vranesic
- The approach for reducing the number of states is to:
  - Define the concept of equivalent states
  - Develop a technique for identifying equivalent states
  - Construct a new reduced sequential circuit with the fewest number of states by replacing each subset of equivalent states by a single state.

## □ Example



- These two state graphs are equivalent in that they cannot be distinguished from one another by observing the inputs and outputs (assuming they start in state A and X).

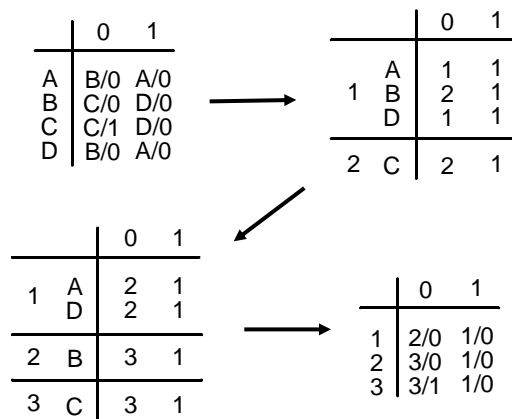
- ◆ States C and D are equivalent to state Z.

## □ Equivalent states

- Two states are equivalent if you can not tell them apart by applying input signals and observing output signals. It doesn't matter if the states are in the same state diagram or different ones.
  - ◆ An input symbol (or just input) is a configuration of input signal values: an output symbol (or just output) is a configuration of output signal values.
  - ◆ Definition: Let p be a state in state diagram G1 and q a state in state diagram G2 (G1 and G2 may be the same diagram) where both diagrams have the same set of inputs (i.e., configuration of input signal values). Then states p and q are *equivalent* if for every possible sequence of inputs, the sequence of outputs that results from applying that input sequence starting in state p is the same that would be produced starting in state q.

- If two states in the same state diagram are equivalent, then they can be replaced by a single state.
- Definition: A state diagram is *reduced* if no two of its states are equivalent.
- Let  $p$  and  $q$  be two states in a state table and  $x$  an input signal value. Then  $p$  and  $q$  are not equivalent if
  1. The output symbol in row  $p$ , column  $x$  is different from the output symbol in row  $q$ , column  $x$ , or
  2. The next state entry in row  $p$ , column  $x$  is not equivalent to the next state entry in row  $q$  column  $x$ .
- A procedure for finding when states are not equivalent:
  1. Find which states violate condition 1
  2. Find which successor states violate condition 2
  3. Repeat step 2.

### □ Example



□ Example

		AB				Z
		00	01	11	10	
INIT	A0	A0	A0	A1	A1	0
A	A0	OK00	OK00	A1	A1	0
	A1	A0	A0	OK11	OK11	0
	OK00	OK00	OK00	A001	A1	1
	OK11	A0	A110	OK11	OK11	1
	A001	A0	AE10	OK11	OK11	1
	A110	OK00	OK00	AE01	A1	1
B	AE10	OK00	OK00	AE01	A1	1
	AE01	A0	AE10	OK11	OK11	1

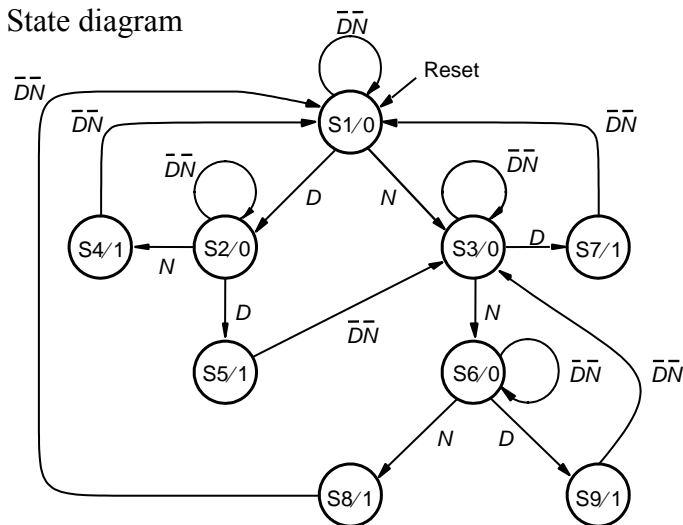
  

		AB			
		00	01	11	10
A	INIT	B	B	C	C
B	A0	D	D	C	C
C	A1	B	B	E	E
	OK00	D	D	E	C
D	A110	D	D	E	C
	AE10	D	D	E	C
	OK11	B	D	E	E
E	A001	B	D	E	E
	AE01	B	D	E	E

		AB				Z
		00	01	11	10	
A		B	B	C	C	0
B		D	D	C	C	0
C		B	B	E	E	0
D		D	D	E	C	1
E		B	D	E	E	1

■ State diagram



Note that missing edges are don't cares

### ■ State reduction

	00	01	10	11	
S1	S1	S3	S2	-	0
S2	S2	S4	S5	-	0
S3	S3	S6	S7	-	0
S4	S1	-	-	-	1
S5	S3	-	-	-	1
S6	S6	S8	S9	-	0
S7	S1	-	-	-	1
S8	S1	-	-	-	1
S9	S3	-	-	-	1

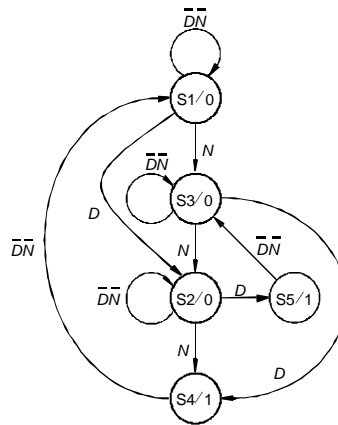
	00	01	01	11	
S1	A	A	A	-	
S2	A	B	B	-	
S3	A	A	B	-	
S6	A	B	B	-	
S4	A	-	-	-	
S5	A	-	-	-	
S7	A	-	-	-	
S8	A	-	-	-	
S9	A	-	-	-	

	00	01	10	11	
A S1	A	B	C	-	
B S3	B	C	D	-	
C S2	C	D	D	-	
C S6	C	D	D	-	
S4	A	-	-	-	
S5	B	-	-	-	
D S7	A	-	-	-	
S8	A	-	-	-	
S9	B	-	-	-	

	00	01	10	11	
A	A	B	C	-	0
B	B	C	D	-	0
C	C	D	E	-	0
D	A	-	-	-	1
E	B	-	-	-	1

Reduced State Table

### ■ Reduced Mealy Sequential Circuit



Moore State Diagram

## 4. Review

---

- How to construct a state graph, state table or ASM from a word description of a sequential circuit's behavior.
- The formal technique for designing sequential circuits consisting of the following steps:
  - State assignment and construction of the transition table
  - Selection of flip-flop type and construction of the excitation table
  - Derivation of the flip-flop input equations and output equations from the excitation table
- The use of transition lists and expressions to simplify the design of sequential circuits.
- The use of registered Moore outputs to reduce delay.