Analysis of Clocked (Synchronous) Sequential Circuits

- Now that we have flip-flops and the concept of memory in our circuit, we might want to determine what a circuit is doing.

- The behavior of a clocked sequential circuit is determined from its inputs, outputs and state of the flip-flops (i.e., the output of the flip-flops).

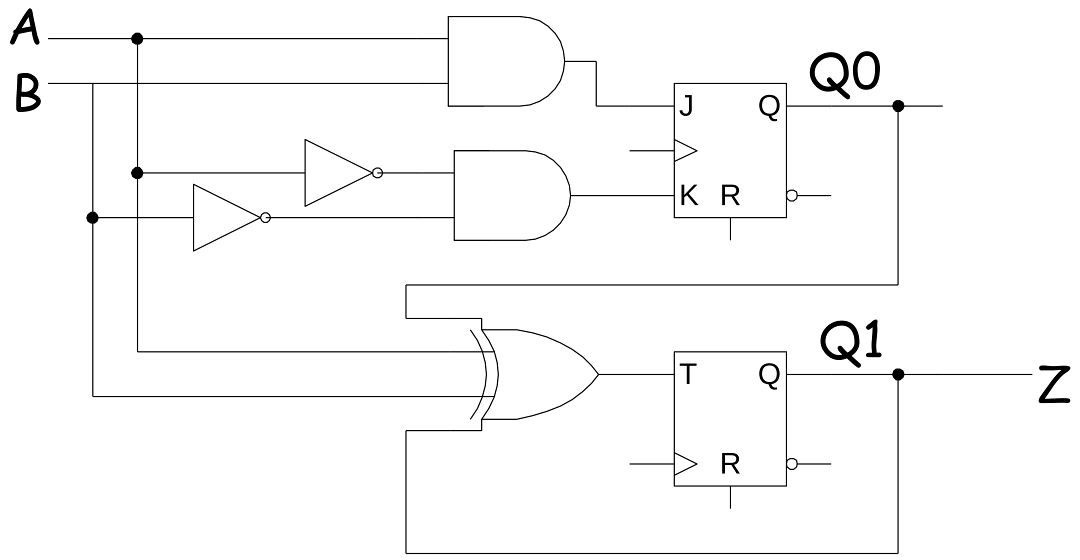
- The analysis of a clocked sequential circuit consists of obtaining a table of a diagram of the time sequences of inputs, outputs and states.
  - E.g., given a current state and current inputs, how will the state and outputs change when the next active clock edge arrives???

Analysis Procedure

- We have a basic procedure for analyzing a clocked sequential circuit:
  - Write down the **equations** for the **outputs** and the **flip-flop inputs**.
  - Using these equations, derive a **state table** which describes the next state.
  - Obtain a **state diagram** from the **state table**.
- It is the state table and/or state diagram that specifies the **behavior** of the circuit.
- Notes:
  - The **flip-flop input equations** are sometimes called the **excitation equations**.
  - The **state table** is sometimes called a **transition table**.
- We can best illustrate the procedure by doing examples…

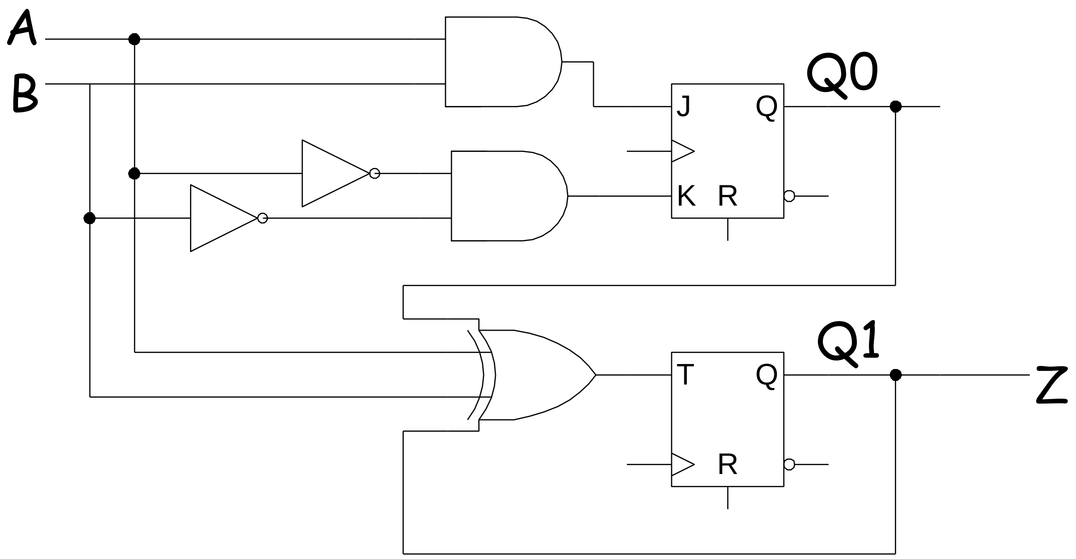
Analysis Example

- Consider the following circuit. We want to determine how it will behave.



- Observations: The circuit has two inputs **A** and **B**, one output **Z** (it happens that the output is equal to one of the flip flop outputs).

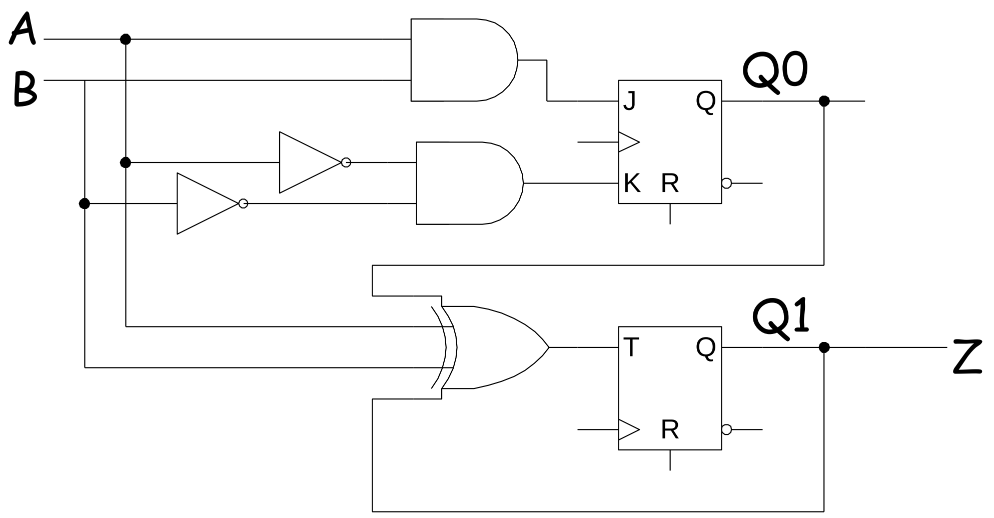
Analysis Example – More Observations



- Observations: The circuit has two flip-flops (different types) with outputs Q_0 and Q_1 **(This implies that there are as many as 4 different states in the circuit, namely Q_0Q_1 = 00, 01, 11, 10).**

- Observations: The circuit output depends on the current state (flip-flop outputs) only.

Analysis Example – Flip-Flop Input Equations and Output Equations



$$Z = Q_1$$

$$J_0 = AB$$
$$K_0 = A'B'$$

$$T_1 = A \oplus B \oplus Q_0 \oplus Q_1$$

- We write down Boolean expressions for the FF inputs and the circuit outputs.
- Done in terms of the **circuit inputs and the current state (flip-flop outputs)** of the system.

Analysis Example – State Table

- Using the FF input equations, we can build a table showing the FF inputs (this is the first step in creating the state table):

Current State	J_0K_0				T_1			
Q_0Q_1	$AB =$ 00	01	10	11	$AB =$ 00	01	10	11
00	01	00	00	10	0	1	1	0
01	01	00	00	10	1	0	0	1
10	01	00	00	10	1	0	0	1
11	01	00	00	10	0	1	1	0

$$Z = Q_1$$

$$J_0 = AB$$
$$K_0 = A'B'$$

$$T_1 = A \oplus B \oplus Q_0 \oplus Q_1$$

Analysis Example – State Table Continued

- We now have the FF input values, and know the FF behavior (for both JKFF and TFF):

Current State	J_0K_0				T_1			
Q_0Q_1	$AB = 00$	01	10	11	$AB = 00$	01	10	11
00	01	00	00	10	0	1	1	0
01	01	00	00	10	1	0	0	1
10	01	00	00	10	1	0	0	1
11	01	00	00	10	0	1	1	0

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

JKFF Behavior

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

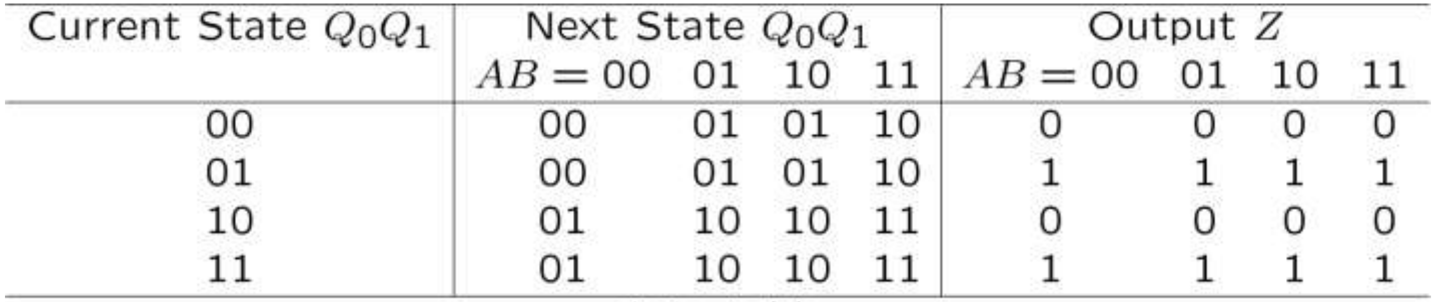
TFF Behavior

- We can then determine the next FF output values:

Current State Q_0Q_1	Next State Q_0				Next State Q_1			
	$AB = 00$	01	10	11	$AB = 00$	01	10	11
00	0	0	0	1	0	1	1	0
01	0	0	0	1	0	1	1	0
10	0	1	1	1	1	0	0	1
11	0	1	1	1	1	0	0	1

Analysis Example – State Table Summarized

- We can (finally) write the next state information and the output values (based on current state and input values) in the **state table** format:



Current State Q_0Q_1	Next State Q_0Q_1				Output Z			
	$AB = 00$	01	10	11	$AB = 00$	01	10	11
00	00	01	01	10	0	0	0	0
01	00	01	01	10	1	1	1	1
10	01	10	10	11	0	0	0	0
11	01	10	10	11	1	1	1	1

state table

- Observation: The output **Z** (in this example) is only a function of the current state; it does not depend on the inputs.

Analysis Example – State Table Alternative Representation

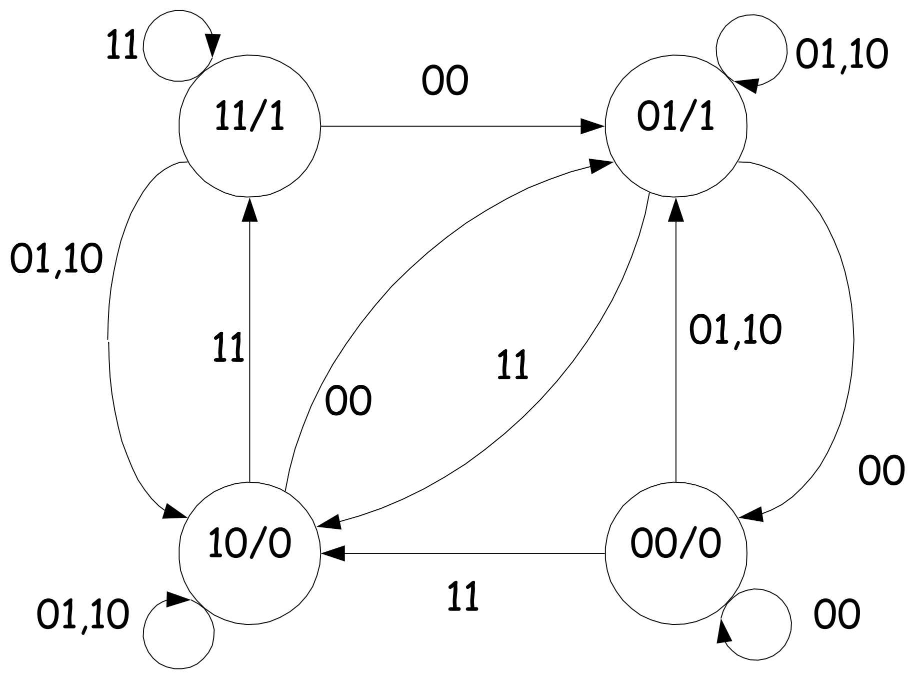
- We can also write the state table in a slightly different (tabular) format if we choose. The information presented is the same.

Current State		Input		Next State		Output
Q_0	Q_1	A	B	Q_0	Q_1	Z
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

alternative state table

State Diagram

- Another way to illustrate the behavior of a clocked sequential circuit is with a **state diagram**.

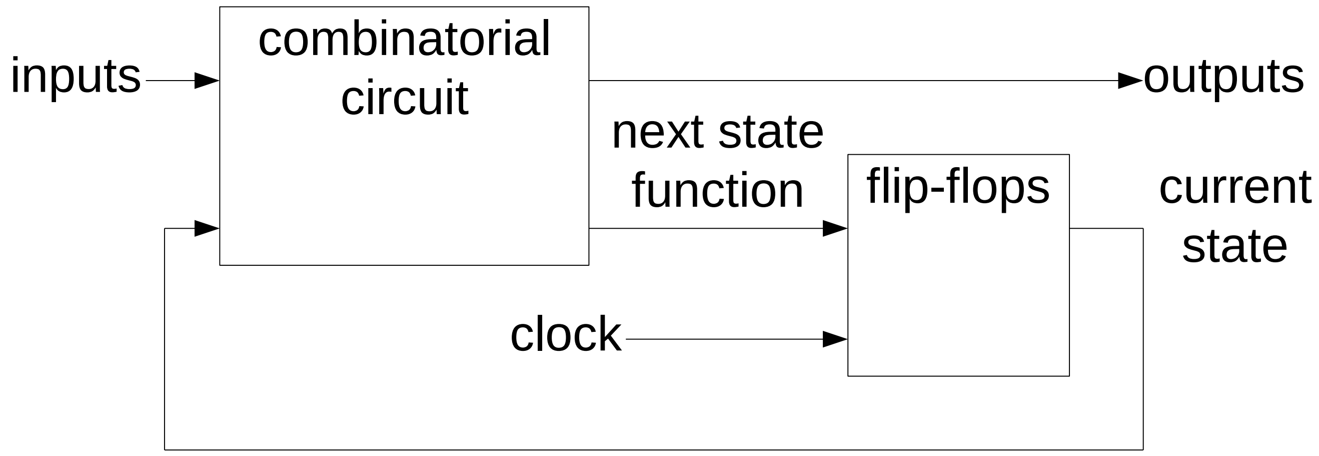


State Diagram Explained

- Each “bubble” **(state bubble)** in the state diagram represents one state of the system.
  - The flip-flop outputs that correspond to that state are **labeled** inside of the bubble.
- Each edge leaving a bubble represents a possible transition to another state once the active clock edge arrives.
  - The edges are **labeled** with the input values that cause the transition to occur.
- In this state diagram, the **output values** are **labeled** inside of the state bubbles.
  - We can do this because the outputs are only a function of the current state in this example.

Topologies of Clocked Sequential Circuits – States

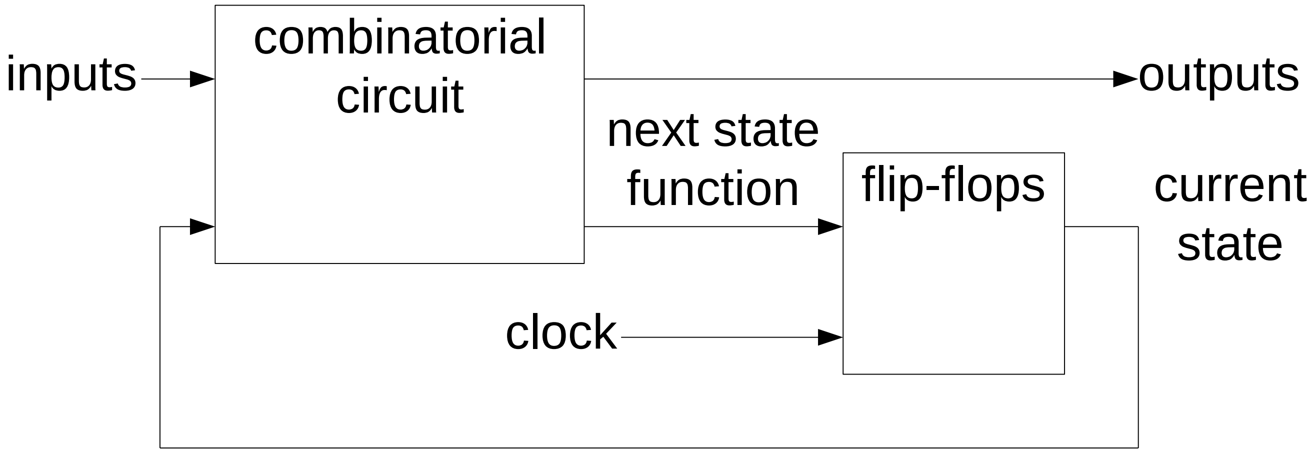
- Recall our basic block diagram of a clocked sequential circuit:



- The next state is **always** a function of the current state and the current inputs.

Topologies of Clocked Sequential Circuits - Outputs

- Recall our basic block diagram of a clocked sequential circuit:

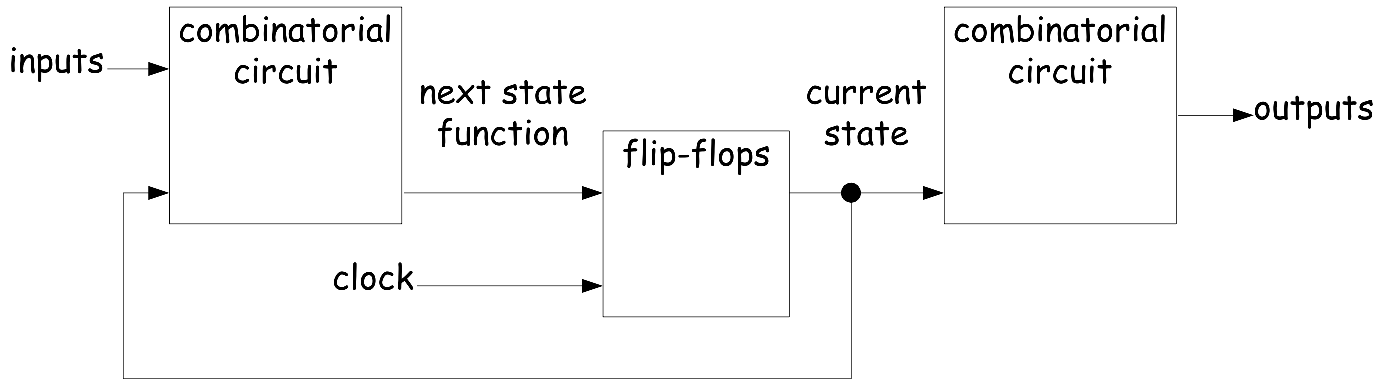


- The outputs can be a function of either:
  - The current state **only**, or
  - The current state **and** the current inputs.

- We can **categorize** the topology of the clocked sequential circuit depending on how the outputs are computed.

Moore Machines (Basic Topology)

- If outputs depend only on the current state, we have a **Moore Machine**.
- We can re-arrange the drawing of a clocked sequential circuit:

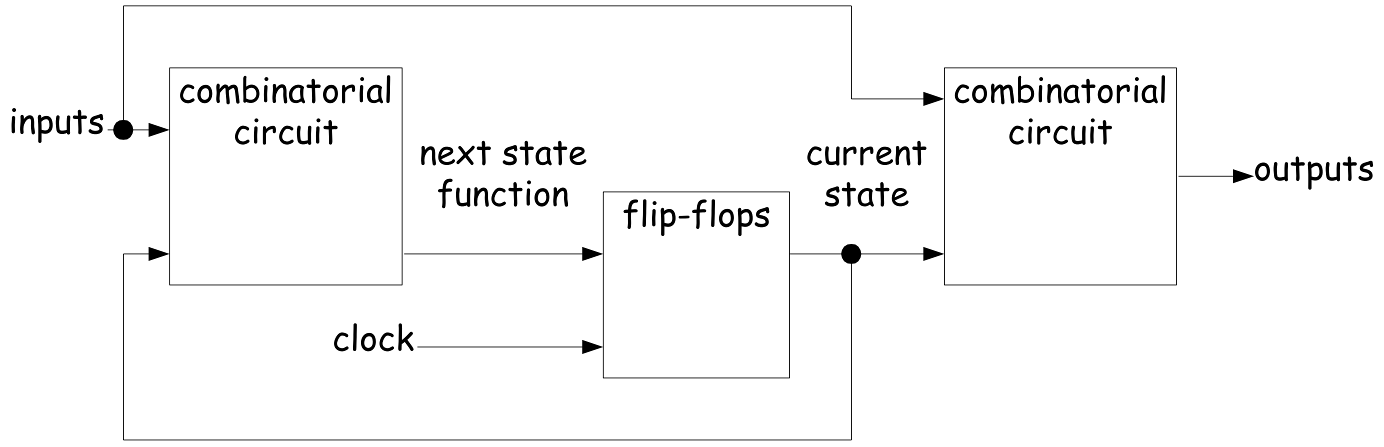


- **IMPORTANT OBSERVATION: In a Moore Machine, the outputs change synchronously with the clock as the state changes synchronously with the clock.**
  - **Important because this means outputs change at predictable times and timing analysis easy (outputs good right after clock edge).**

Mealy Machines (Basic Topology)

- If outputs depend on both the current state and the current input, we have a **Mealy Machine**.

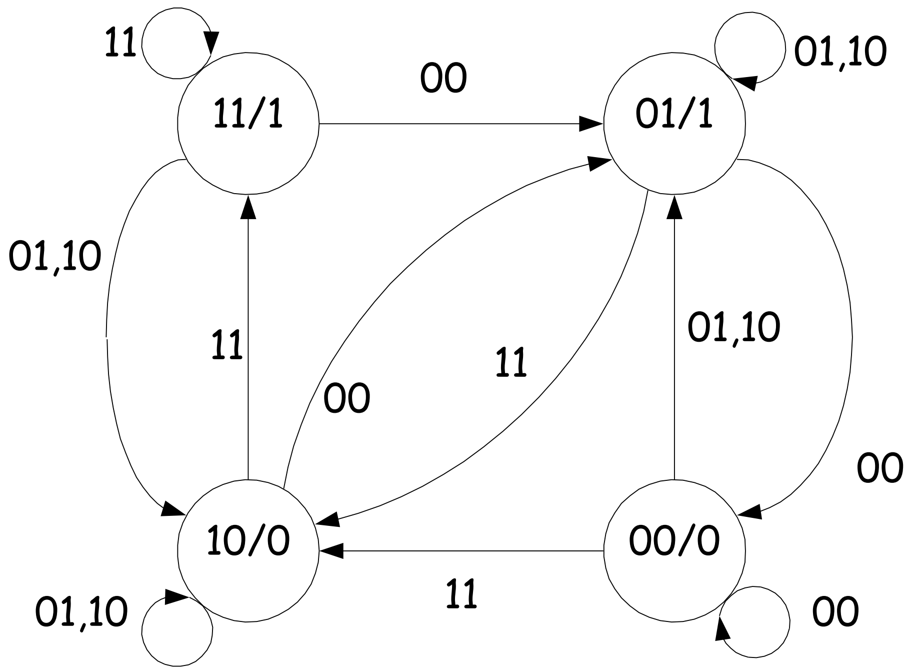
- We can re-arrange the drawing of a clocked sequential circuit:



- **IMPORTANT OBSERVATION: In a Mealy Machine, the outputs change asynchronously with respect to the clock since inputs can change at any time.**
  - **Important because this means outputs change at unpredictable times and timing analysis is harder (when are outputs valid?).**

State Diagram for a Moore Machine

- Recall our analysis example… We derived the following state diagram.



- Since we found that the outputs depended only on the current state, our analysis example was for a Moore Machine; this is why the output values were labeled **inside** to the state bubble.

State Diagram for a Mealy Machine

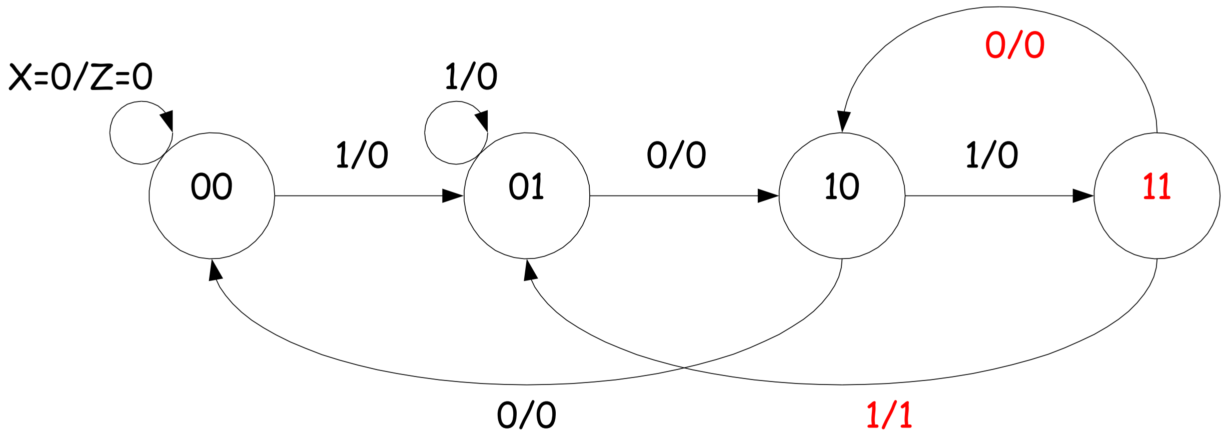
- Consider for a moment a circuit in which the output values are a function of **both** the current state and the current input values.
- E.g., if we had the following state table for a circuit with one input X, one output Z and two flip-flops (4 possible states):

Current State Q_0Q_1	Next State Q_0Q_1		Output Z	
	$X=0$	$X=1$	$X=0$	$X=1$
00	00	01	0	0
01	10	01	1	1
10	00	11	0	0
11	10	01	0	1

- Observation: The output **Z does** depend on **both** the current state and the current input; e.g., when the current state is **"11"**, the output **Z** is different for different values of **X**.
- So this state table does correspond to a Mealy Machine, and we need to draw our state diagram slightly differently.

State Diagram for a Mealy Machine Illustrated

- For a Mealy Machine, note that the outputs are labelled along the edges, indicating that the outputs depend on both the current state (where the edge begins) and the current input value (which is labelled on the edge itself).



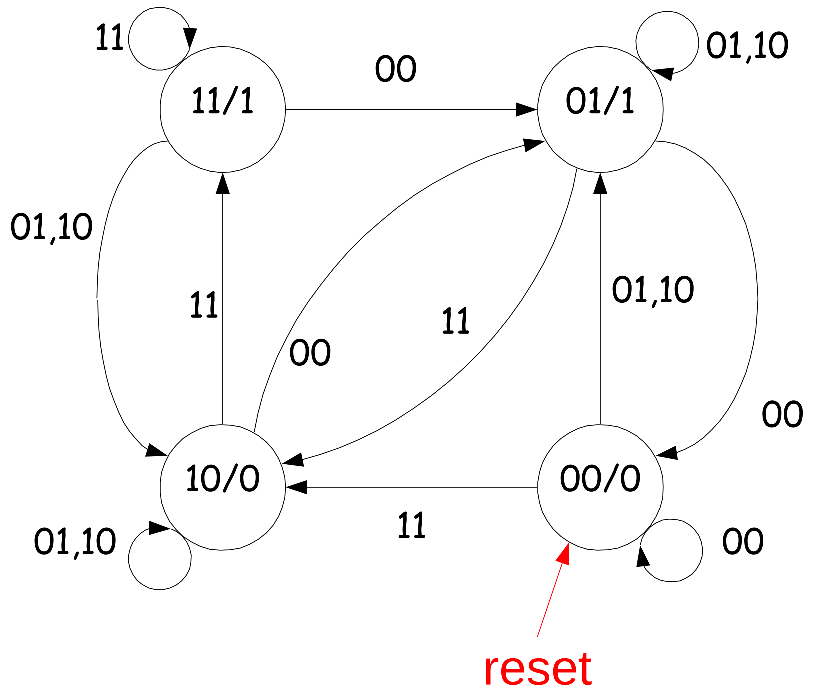
Current State Q_0Q_1	Next State Q_0Q_1 $X=0$	$X=1$	Output Z $X=0$	$X=1$
00	00	01	0	0
01	10	01	1	1
10	00	11	0	0
11	10	01	0	1

Initial State

- When we "turn on the power" or need to "start over" our clocked sequential circuits, we would like to be able to **control** the **initial state** that the circuit enters.
  - For our analysis example, both the flip-flops had **asynchronous resets**.
- We can use **resets and presets** to **force** the circuit into a particular, **initial state**.
  - For our analysis example, forcing the reset to "1" would force the flip-flop outputs to "00", and our initial state would be "00".
- We can illustrate the initial state in the state diagram if we choose…

Initial State Illustrated

- The initial state is illustrated by an incoming edge pointing at the initial state.



Textbook

- Analysis of clocked sequential circuits is described in the textbook in Chapter 5, Section 5-4. Mealy and Moore Machines are also described in Section 5-4.

Design of Clocked (Synchronous) Sequential Circuits

- Design of clocked sequential circuits is very much the opposite of the analysis and we can follow a sequence of steps… (some steps will need to be illustrated).

Design Procedure

- Understand the verbal description of the problem, and create a state diagram and/or a state table. Note that the states may have only symbolic names at this point.
- Reduce the number of states, if possible (state reduction). This may yield a circuit with fewer flip-flops.
- Perform state assignment which means assigning binary codes to the symbolic state names.
- Select a flip-flop type (e.g., DFF, TFF, JKFF, or some combination thereof…).
- Derive simplified output equations.
- Derive the next-state equations and the flip-flop input equations.
- Draw a schematic of the resulting circuit.

Design Example

- We will illustrate the design procedure by doing an example. Then, we will go back an investigate alternative choices we could have made at various steps.

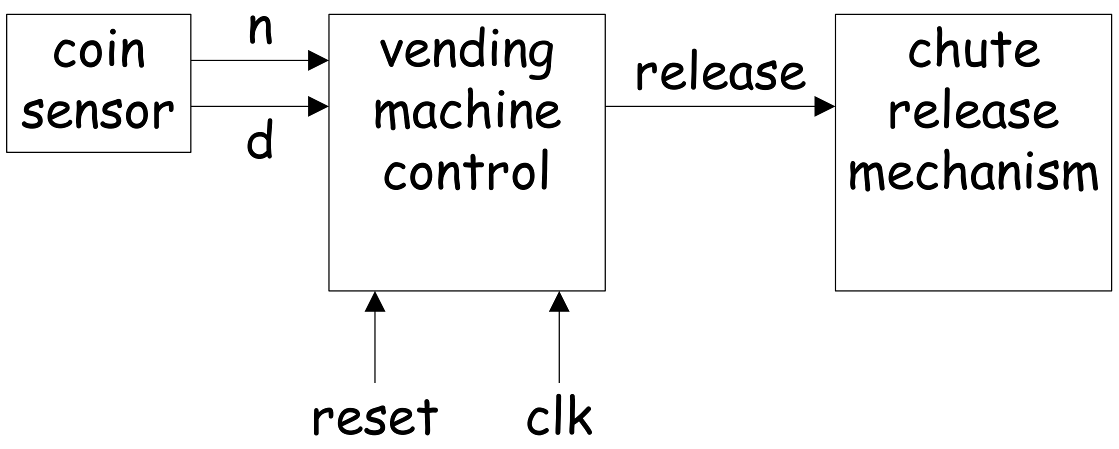
- Consider the following verbal problem description:

  - A vending machine dispenses a package of gum which costs 15 cents. The machine has a single slot that accepts nickels and dimes only and a sensor indicates the type of coin deposited. Make a controller that sends a signal to a chute release once enough change has been deposited. Note that the machine does not give change or credits.

- Getting from the verbal problem description to a state diagram and/or a state table is likely the hardest part of the design procedure.

Design Example – Block Diagram

- We can illustrate the verbal problem description as a block diagram.



- Our controller has two inputs, n and d, to indicate the coin deposited. It also has a clock and a reset input. It produces one output, release, once enough change has been deposited.

Design Example – Problem Interpretation

- We can make some observations about the problem:
  - Once 15 or more cents has been deposited we assert the release signal so the gum is dispensed.
  - When a nickel is deposited, n is set to 1.
  - When a dime is deposited, d is set to 1.
  - Inputs n or d are both 0 when no coin deposited.
  - Both n and d are never set to 1 simultaneously.

Design Example – Problem Interpretation Continued

- We need to detect sequences of deposited nickels and dimes that add up to $\geq$ **15** cents.

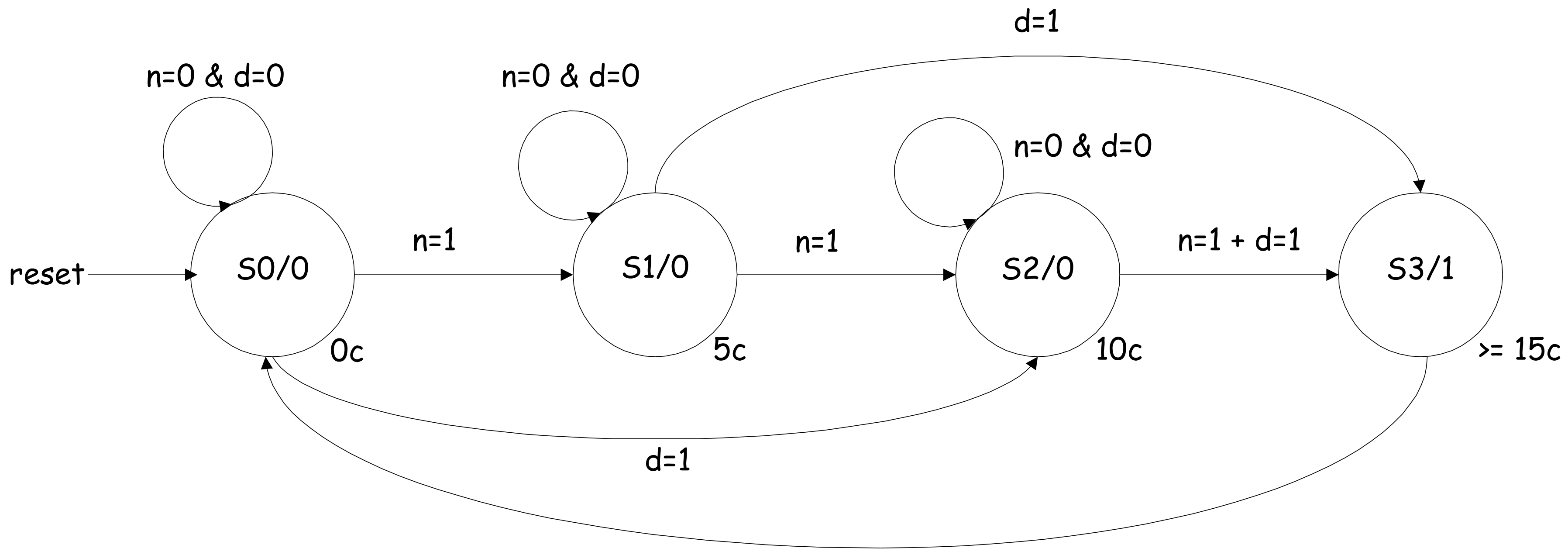
$$\begin{tikzcd}
0c \arrow[d, "n"] & 0c \arrow[d, "n"] & 0c \arrow[d, "n"] & 0c \arrow[dd, "d"] & 0c \arrow[dd, "d"] \\
5c \arrow[d, "n"] & 5c \arrow[d, "n"] & 5c \arrow[dd, "d"] & & \\
10c \arrow[d, "n"] & 10c \arrow[dd, "d"] & & 10c \arrow[d, "n"] & 10c \arrow[dd, "d"] \\
15c & & 15c & 15c & \\
& 20c & & & 20c
\end{tikzcd}$$

- It then makes sense to define a state and the amount of money deposited so far.

- We can then draw a state diagram.

Design Example – State Diagram

- We can draw the state diagram:

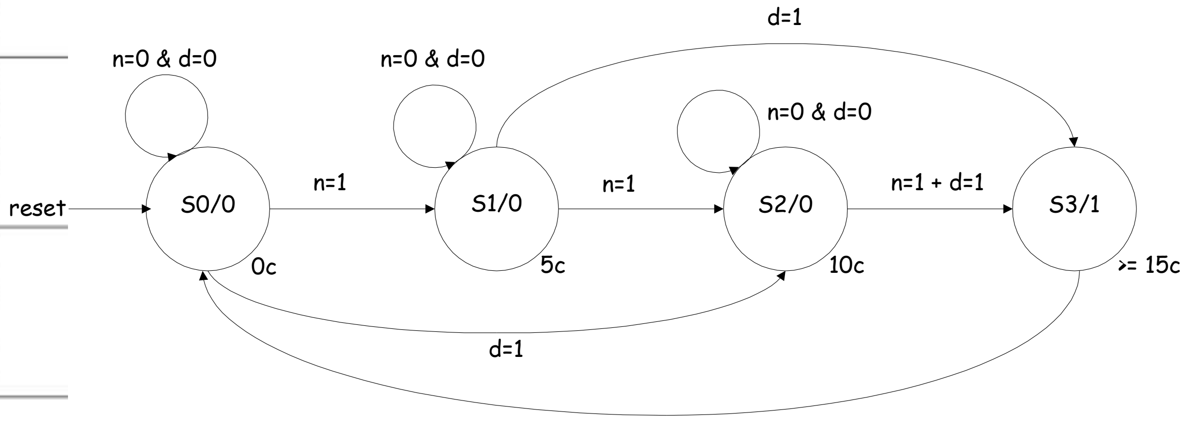


- Note that the states are symbolic (e.g., "S0" represents "0 cents deposited so far", "S1" representes "5 cents deposited so far", "S2" represents "10 cents deposited so far", and "S3" represents >= 15 cents deposited").

Design Example – State Table

- From the state diagram, we can write the state table immediately.

Current State	Input n	Input d	Next State	Output $release$
S0	0	0	S0	0
S0	0	1	S2	0
S0	1	0	S1	0
S0	1	1	-	0
S1	0	0	S1	0
S1	0	1	S3	0
S1	1	0	S2	0
S1	1	1	-	-
S2	0	0	S2	0
S2	0	1	S3	0
S2	1	0	S3	0
S2	1	1	-	-
S3	0	0	S0	1
S3	0	1	S0	1
S3	1	0	S0	1
S3	1	1	-	-



Design Example – State Assignment

- Presently, our states have symbolic names (S0, S1, etc.) and we need to encode them as bit strings.

- How should we do the encoding? Decisions, decisions, …

  - How many bits we chose for the encoding influences the number of flip-flops required.

  - How we assign binary patterns to states can influence the complexity of the flip-flop input equations.

  - Etc…

Design Example – State Assignment

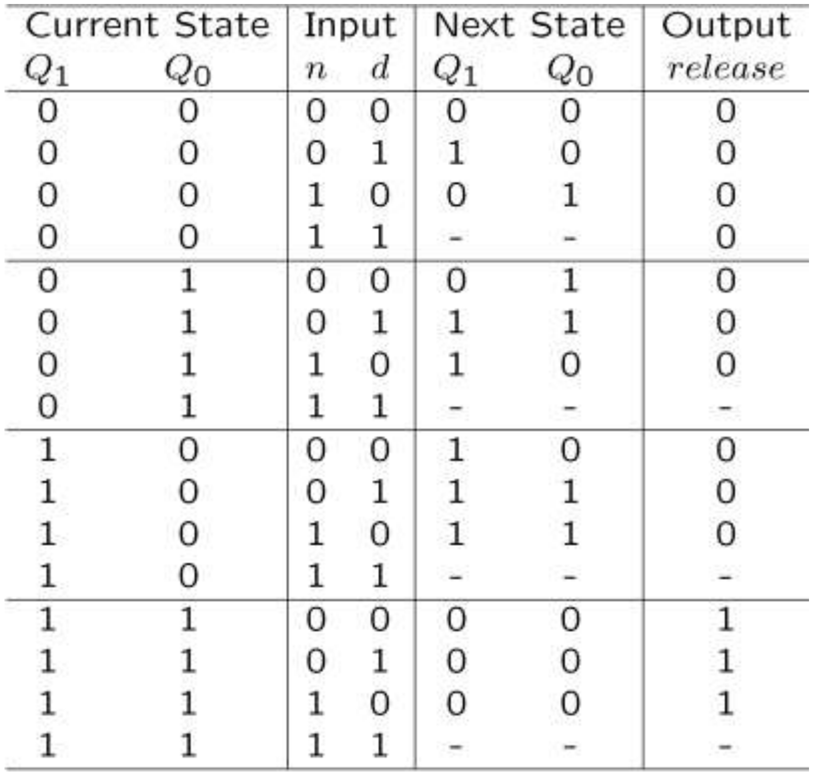
- In our example, we have 4 states, so we need a minimum of 2 bits to encode the states (and therefore a minimum of 2 flip-flops).

- We will use the following encoding:

  - **S0 $\leftarrow$ 00**
  - **S1 $\leftarrow$ 01**
  - **S2 $\leftarrow$ 10**
  - **S3 $\leftarrow$ 11**

Design Example – State Table

- With binary codes assigned, we can redo our state table with binary values instead of symbolic names:



Current State		Input		Next State		Output
Q_1	Q_0	n	d	Q_1	Q_0	$release$
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	-	-	0
0	1	0	0	0	1	0
0	1	0	1	1	1	0
0	1	1	0	1	0	0
0	1	1	1	-	-	-
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	-	-	-
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	-	-	-

Design Example – Flip-Flop Selection

- Need to select a type of flip-flop to serve as the storage element for the state information and to hold the current state information.

- We different possibilities, including DFF, TFF and JKFF.

- The selection of flip-flop type will influence the complexity of the logic required to generate the flip-flop input equations (i.e., the complexity of the next state logic).

Design Example – Flip-Flop Selection

- For our example, we will chose DFF.
- With DFF, the output Q(t+1) equals input D(t) once the active clock edge arrives.
- So, DFF are **convenient** in that the FF input equations only need to produce the encoding of the next state!!!
  - I.e., the DFF input equations should be equal to the next state information in the state table!!!

Design Example – Flip-Flop Input Equations

- Use the state table along with DFF behavior to derive FF input equations **Note: Once again… With DFF, the input equations EQUAL the next state columns of the state table.**

Q1Q0

nd	00	01	11	10
00	0	1	0	0
01	0	1	0	1
11	X	X	X	X
10	1	0	0	1

D0

Current State Q_1	Current State Q_0	Input n	Input d	Next State Q_1	Next State Q_0	Output *release*
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	-	-	0
0	1	0	0	0	1	0
0	1	0	1	1	1	0
0	1	1	0	1	0	0
0	1	1	1	-	-	-
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	-	-	-
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	-	-	-

Q1Q0

nd	00	01	11	10
00	0	0	0	1
01	1	1	0	1
11	X	X	X	X
10	0	1	0	1

D1

$$D0 = Q_1'Q_0n' + Q_0'n + Q_1Q_0'd$$

$$D1 = Q_1Q_0' + Q_1'Q_0n + Q_1'd$$

Design Example – Output Equations

- Use the state table to derive the output equations.

Q1Q0

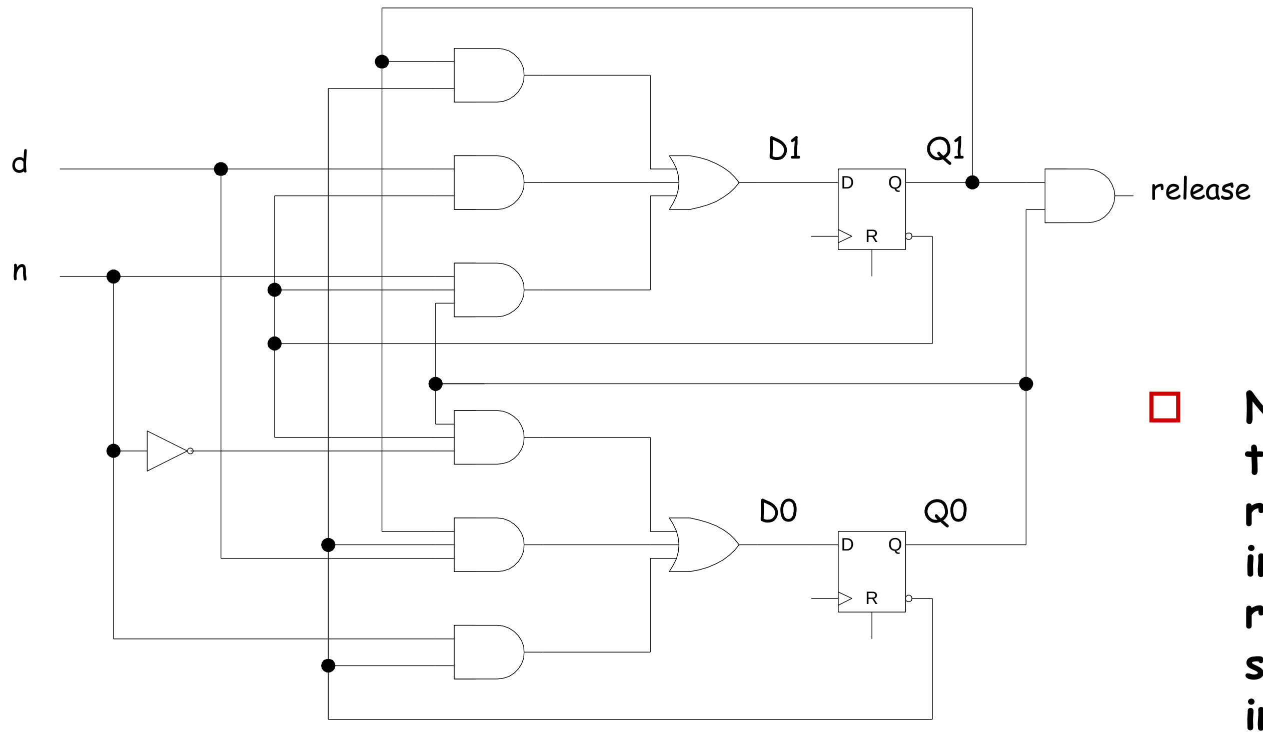
nd	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	X	X	X	X
10	0	0	1	0

release

$$release = Q_1 Q_0$$

Design Example – Circuit Implementation (Schematic)

- Finally, we can draw our schematic with two DFF, and logic equations to produce the flip-flop inputs and the output equation.



- **Note that we can use the resets on the DFF to reset the circuit into its initial state (this was a result of the encoding scheme that made the initial state 00).**

Alternatives in Flip-Flop Type Selection

- Recall that the flip-flops hold the current state information. The next state information is determined by the flip-flop input equations.
- DFF are an obvious and convenient choice to hold state information since their inputs ARE the next state equations.
- However… A different type of FF might result in input equations that are **much simpler (less gates)** compared to that obtained using DFF.
- This will **change the design procedure**, since we need to account for the different behaviors of different types of flip-flops.

Changes in Design Due To Flip-Flop Selection

- If using DFF:
  - Look at the state table and derive equations for the next state given the current state and the circuit inputs.
  - Apply (connect) the next state logic directly to the DFF inputs.
- If not using DFF:
  - Look at the state table to see how the **current state CHANGES to the next state** for a given current state and inputs.
  - Examine the **excitation table** for the FF to determine how the FF input must be set to get the DESIRED CHANGE from current state to next state.
  - Derive logic functions for the FF inputs given the current state and inputs and apply (connect) these equations directly to the FF inputs.
- EMPHASIS: When we use FF that are not DFF, the logic applied to the FF input are NOT the next state equations BUT WILL GENERATE THE CORRECT NEXT STATE IN CONJUNCTION WITH THE FF BEHAVIOR.

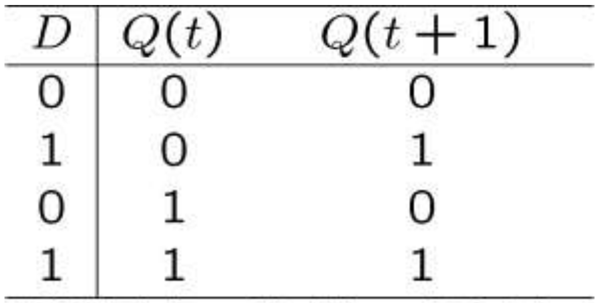
Excitation Tables

- We can rewrite the behavior of all flip-flop types in terms of how the output **changes** given the current inputs to the flip-flops.

- **Changes** in flip-flop outputs depending on flip-flop inputs is done via an **Excitation Table**.

Excitation Tables - DFF

- For DFF, the output Q(t+1) depends only on the input D(t).



D	$Q(t)$	$Q(t+1)$
0	0	0
1	0	1
0	1	0
1	1	1

DFF Excitation Table

Excitation Tables - TFF

- For a TFF, the output Q(t+1) changes to the complement of Q(t) if the input is 1.

T	$Q(t)$	$Q(t+1)$
0	0	0
1	0	1
0	1	1
1	1	0

TFF Excitation Table

Excitation Tables - JKFF

- Excitation Table for JKFF slightly more complicated, but can be determined from the characteristic table:

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

JKFF Characteristic Table

J	K	$Q(t)$	$Q(t+1)$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	0
1	1	1	0

JKFF Excitation Table

J	K	$Q(t)$	$Q(t+1)$
0	X	0	0
1	X	0	1
X	0	1	1
X	1	1	0

JKFF Excitation Table

Implementation Choices – Flip-Flop Selection

- We can now consider the changes in circuits for different flip-flop choices.
- Assume we have the following state table (no outputs shown) which was derived originally from some verbal description of a problem:

Current State		Input	Next State	
Q_1	Q_0	A	Q_1	Q_0
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

- Can implement circuit using each type of FF to illustrate the design procedure.

Implementation Choices – DFF Selection and Input Equations

- Assume we select DFF. We need two DFF and need their input equations.
- The transition table tells us that the DFF input equals the next state information.

D	$Q(t)$	$Q(t+1)$
0	0	0
1	0	1
0	1	0
1	1	1

DFF Excitation Table

Current State		Input	Next State		FF Inputs	
Q_1	Q_0	A	Q_1	Q_0	D_1	D_0
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	0	1	1	0	1	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

Implementation Choices – DFF Input Equations

- We draw K-Maps to determine DFF input equations.

Q1Q0

A	00	01	11	10
0	0	1	1	1
1	0	1	0	0

D0

Q1Q0

A	00	01	11	10
0	0	0	1	0
1	1	1	0	1

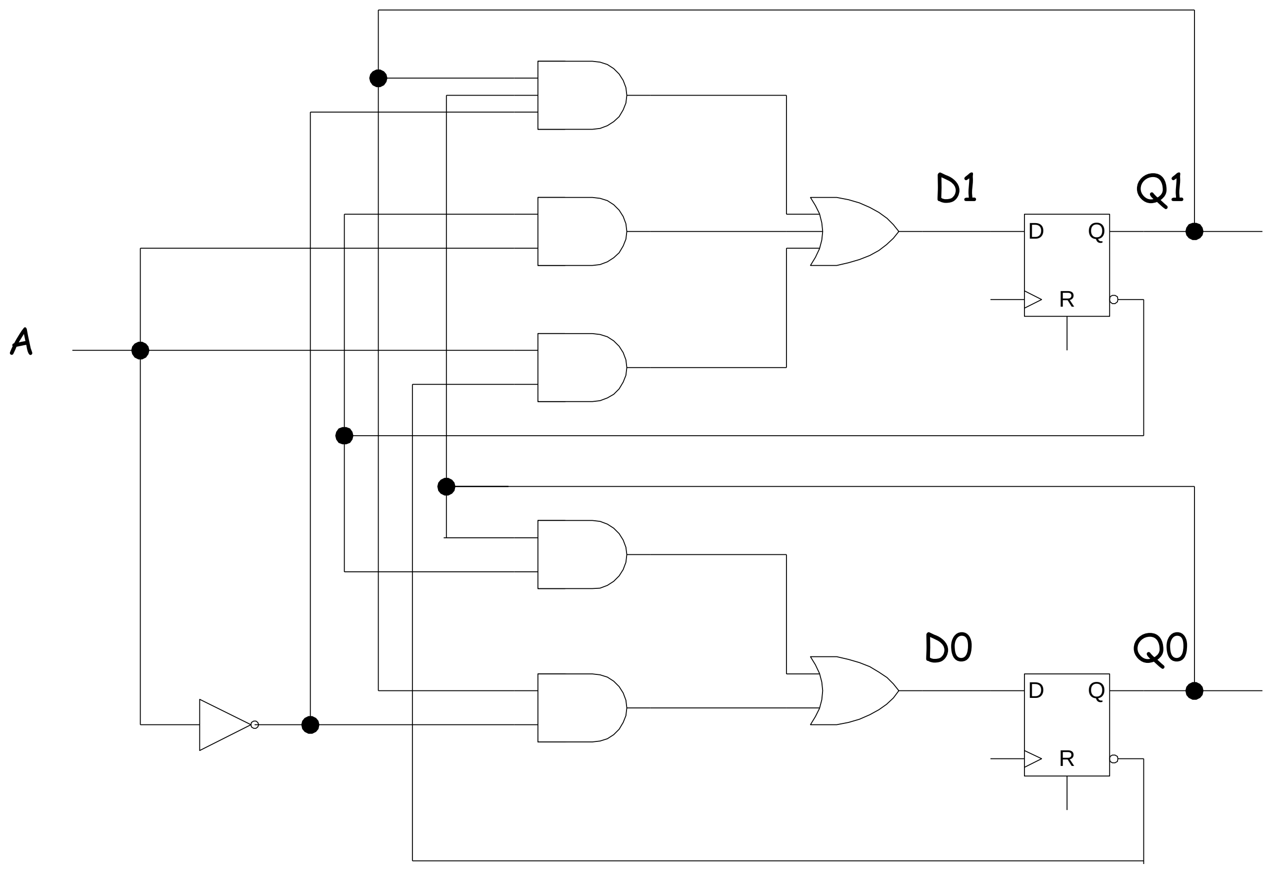
D1

$$D_0 = Q_1A' + Q_1'Q_0$$

$$D_1 = Q_1Q_0A' + Q_1'A + Q_0'A$$

Implementation Choices – DFF Schematic

- Finally, we can draw our circuit.



Implementation Choices – TFF Selection and Input Equations

- Assume we select TFF. We need two TFF and need their input equations.
- Use next state and TFF transition table to figure out TFF inputs:

T	$Q(t)$	$Q(t+1)$
0	0	0
1	0	1
0	1	1
1	1	0

TFF Excitation Table

Current State		Input	Next State		FF Inputs	
Q_1	Q_0	A	Q_1	Q_0	T_1	T_0
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	0	0
1	1	1	0	0	1	1

Implementation Choices – TFF Input Equations

- We draw K-Maps to determine TFF input equations.

Q1Q0

A	00	01	11	10
0	0	0	0	1
1	0	0	1	0

T0

Q1Q0

A	00	01	11	10
0	0	0	0	1
1	1	1	1	0

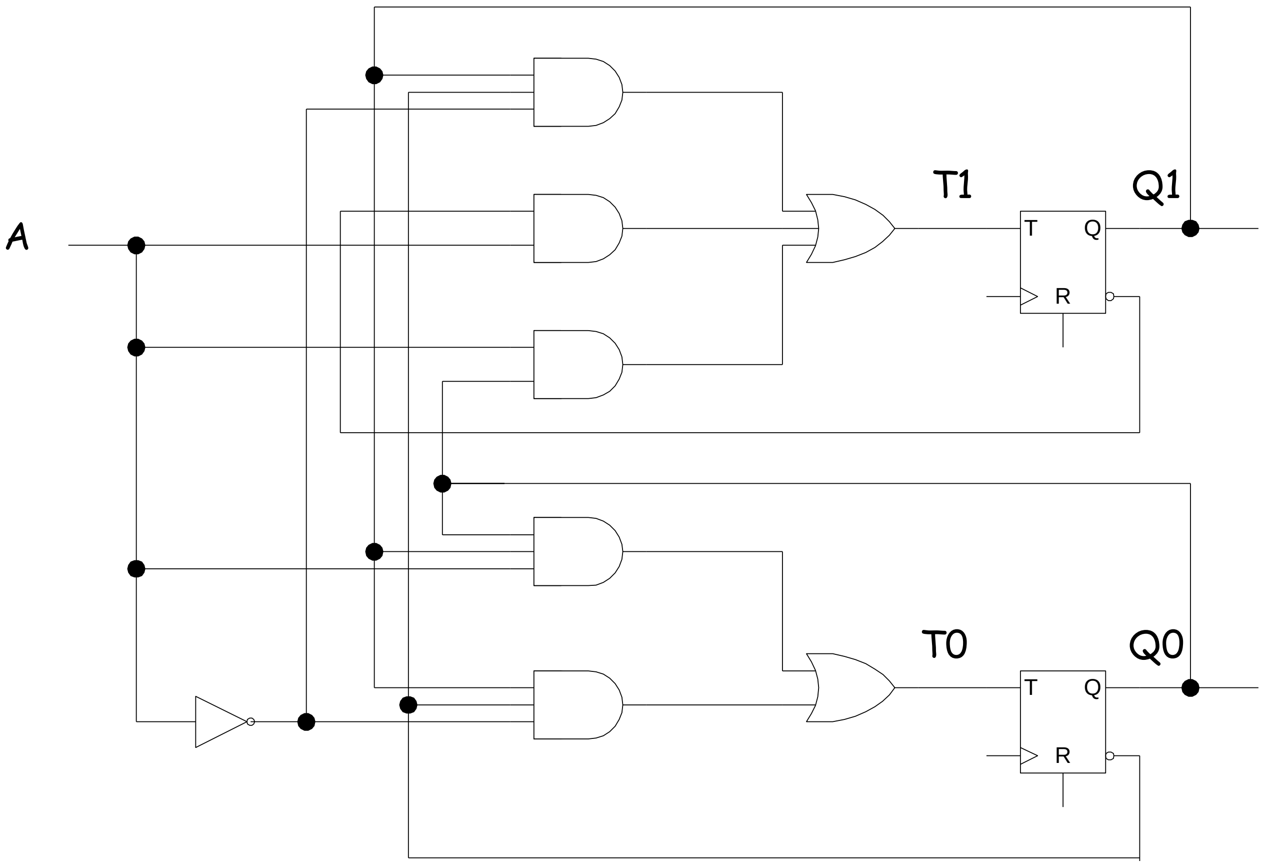
T1

$$T_0 = Q_1 Q_0' A' + Q_1 Q_0 A$$

$$T_1 = Q_1 Q_0' A' + Q_1' A + Q_0 A$$

Implementation Choices – TFF Schematic

- Finally, we can draw our circuit (not really any simpler than DFF in this example).



Implementation Choices – JKFF Selection and Input Equations

- Assume we select JKFF. We need two JKFF and need their input equations.
- Use next state and JKFF transition table to figure out JKFF inputs:

J	K	$Q(t)$	$Q(t+1)$
0	X	0	0
1	X	0	1
X	0	1	1
X	1	1	0

JKFF Excitation Table

Current State		Input	Next State		FF Inputs			
Q_1	Q_0	A	Q_1	Q_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	X	0	X
0	0	1	1	0	1	X	0	X
0	1	0	0	1	0	X	X	0
0	1	1	1	1	1	X	X	0
1	0	0	0	1	X	1	1	X
1	0	1	1	0	X	0	0	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Implementation Choices – JKFF Input Equations

- We draw K-Maps to determine JKFF input equations.

Q1Q0

A	00	01	11	10
0	0	X	X	1
1	0	X	X	0

J0

$$J_0 = Q_1 A'$$

Q1Q0

A	00	01	11	10
0	0	0	X	X
1	1	1	X	X

J1

$$J_1 = A$$

Q1Q0

A	00	01	11	10
0	X	0	0	X
1	X	0	1	X

K0

$$K_0 = Q_1 A$$

Q1Q0

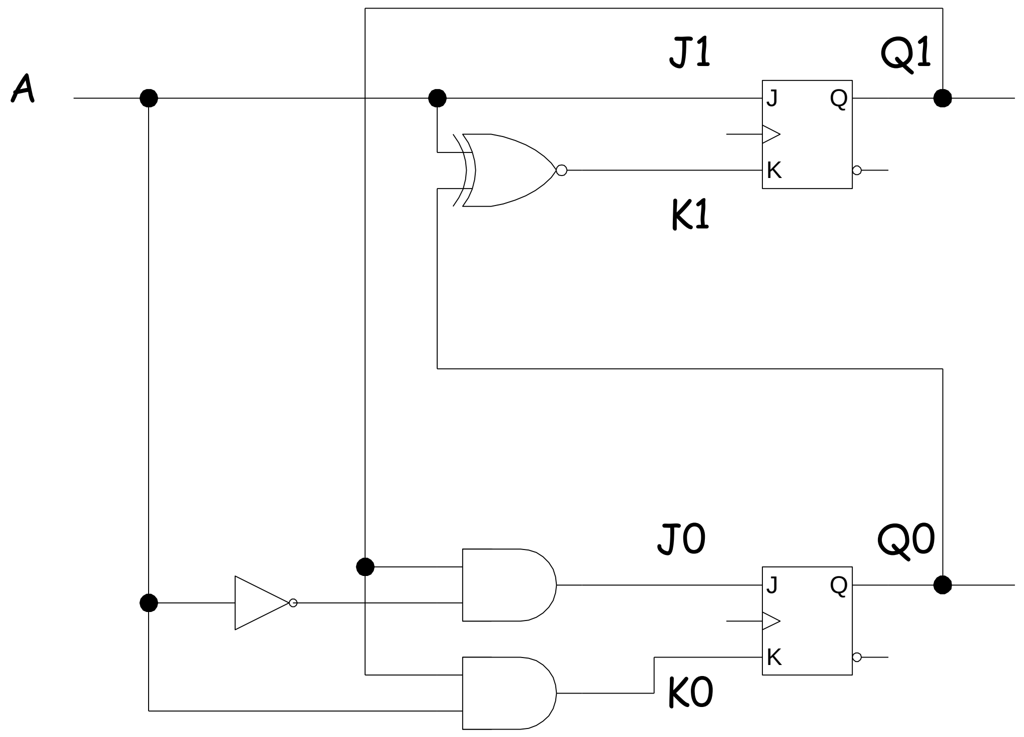
A	00	01	11	10
0	X	X	0	1
1	X	X	1	0

K1

$$K_1 = Q_0' A' + Q_0 A$$

Implementation Choices – JKFF Schematic

- Finally, we can draw our circuit (simpler than DFF in this example).



Comment on Implementation Choices

- In this example, TFF was not any simpler than DFF, but sometimes it is.
- JKFF resulted in a much simpler implementation.
  - This is often the case (but not always).
  - When JKFF are available, they have lots of don't cares "X"s in the K-Maps and this turns out (often) to yield simple input equations.

Textbook Sections

- Clocked circuit design and implementation using different types of flip-flops is covered in the textbook in Chapter 5, Section 5.7.