Attention, please!

- On March 25, I will help you prepare Midterm Exam 2
- Midterm Exam 2 is scheduled on April 6 in class
- HW2 will be due on March 25 in class
- Lab2 is due by 11:59 pm on March 20
- Midterm Exam 2 covers Chapter 3-5 to 3-12 and Chapter 4-1 to 4-5

Chapter 4 Part 2 Sequential Circuit Design

- **Part 1**
  - **Types of Sequential Circuits**
  - **Storage Elements**
    - **Latches**
    - **Flip-Flops**
  - **Sequential Circuit Analysis**
    - **State Tables**
    - **State Diagrams**
- **Part 2**
  - **Sequential Circuit Design**
    - **Specification**
    - **Formulation**
    - **State Assignment**
    - **Flip-Flop Input and Output Equation Determination**
    - **Optimization**
    - **Verification**

The Design Procedure

- **Specification**
- **Formulation - Obtain a state diagram or state table**
- **State Assignment - Assign binary codes to the states**
- **Flip-Flop Input Equation Determination - Select flip-flop types and derive flip-flop equations from next state entries in the table (Text book pp. 210 Section 4-4)**
- **Output Equation Determination - Derive output equations from output entries in the table**
- **Optimization - Optimize the equations**
- **Technology Mapping - Find circuit from equations and map to flip-flops and gate technology**
- **Verification - Verify correctness of final design**

Specification

- **Component Forms of Specification**
  - **Written description**
  - **Mathematical description**
  - **Hardware description language***
  - **Tabular description***
  - **Equation description***
  - **Diagram describing operation (not just structure)***
- **Relation to Formulation**
  - **If a specification is rigorous at the binary level (marked with * above), then all or part of formulation may be completed**

Formulation: Finding a State Diagram

- **A state is an abstraction of the history of the past applied inputs to the circuit (including power-up reset or system reset).**
  - **The interpretation of "past inputs" is tied to the synchronous operation of the circuit. E. g., an input value (other than an asynchronous reset) is measured only during the setup-hold time interval for an edge-triggered flip-flop.**
- **Examples:**
  - **State A represents the fact that a 1 input has occurred among the past inputs.**
  - **State B represents the fact that a 0 followed by a 1 have occurred as the most recent past two inputs.**

Formulation: Finding a State Diagram

- In specifying a circuit, we use states to remember meaningful properties of past input sequences that are essential to predicting future output values.
- A sequence recognizer is a sequential circuit that produces a distinct output value whenever a prescribed pattern of input symbols occur in sequence, i.e, recognizes an input sequence occurence.
- We will develop a procedure specific to sequence recognizers to convert a problem statement into a state diagram.
- Next, the state diagram, will be converted to a state table from which the circuit will be designed.

Sequence Recognizer Procedure

- **To develop a sequence recognizer state diagram:**
  - **Begin in an initial state in which NONE of the initial portion of the sequence has occurred (typically "reset" state).**
  - **Add a state that recognizes that the first symbol has occurred.**
  - **Add states that recognize each successive symbol occurring.**
  - **The final state represents the input sequence occurrence.**
  - **Add state transition arcs which specify what happens when a symbol *not* in the proper sequence has occurred.**
  - **Add other arcs on non-sequence inputs which transition to states that represent the input subsequence that has occurred.**
- **The last step is required because the circuit must recognize the input sequence *regardless of where it occurs within the overall sequence applied since "reset.".***

State Assignment

- Each of the *m* states must be assigned a unique code
- Minimum number of bits required is *n* such that

  $$n \geq \lceil \log_2 m \rceil$$

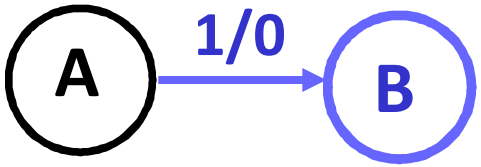
  where $\lceil x \rceil$ is the smallest integer $\geq x$
- There are useful state assignments that use more than the minimum number of bits
- There are 2^n - m unused states

Sequence Recognizer Example

- **Example: Recognize the sequence 1101**
  - **Note that the sequence 1111101 contains 1101 and "11" is a proper sub-sequence of the sequence.**
- **Thus, the sequential machine must remember that the first two one's have occurred as it receives another symbol.**
- **Also, the sequence 1101101 contains 1101 as both an initial subsequence and a final subsequence with some overlap, i. e., 1101101 or 1101101.**
- **And, the 1 in the middle, 1101101, is in both subsequences.**
- **The sequence 1101 must be recognized each time it occurs in the input sequence.**

Example: Recognize 1101

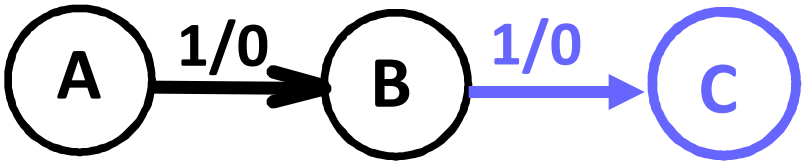
- **Define states for the sequence to be recognized:**
  - **assuming it starts with first symbol,**
  - **continues through each symbol in the sequence to be recognized, and**
  - **uses output 1 to mean the full sequence has occurred,**
  - **with output 0 otherwise.**
- **Starting in the initial state (Arbitrarily named "A"):**
  - **Add a state that recognizes the first "1."**



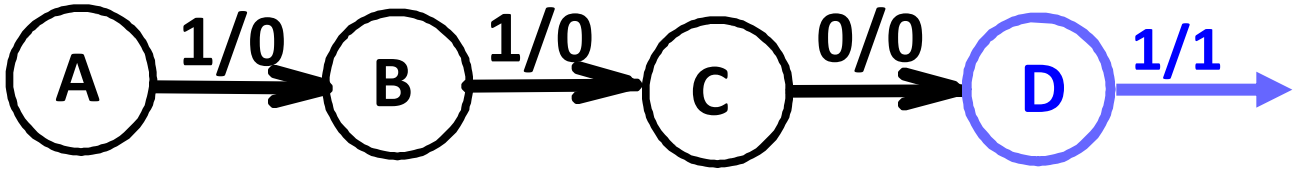
  - **State "A" is the initial state, and state "B" is the state which represents the fact that the "first" one in the input subsequence has occurred. The output symbol "0" means that the full recognized sequence has not yet occurred.**

Example: Recognize 1101 (continued)

- **After one more 1, we have:**
  - **C is the state obtained when the input sequence has two "1"s.**

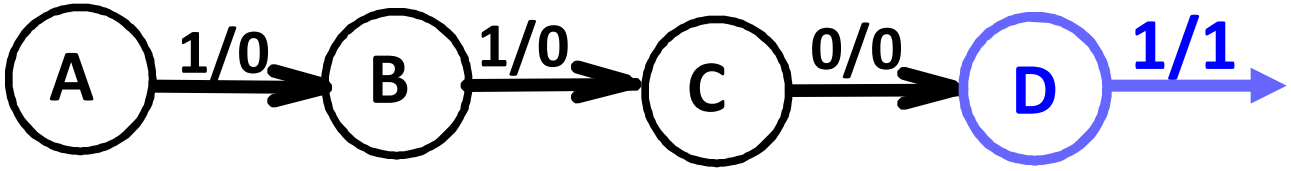


- **Finally, after 110 and a 1, we have:**

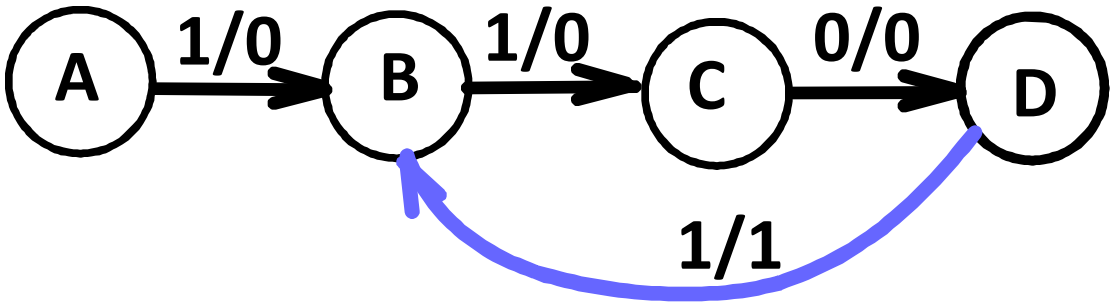


  - **Transition arcs are used to denote the output function (Mealy Model)**
  - **Output 1 on the arc from D means the sequence has been recognized**
  - **To what state should the arc from state D go? Remember: 1101<u>1101</u> ?**
  - **Note that D is the last state but the output 1 occurs for the input applied in D. This is the case when a *Mealy model* is assumed.**

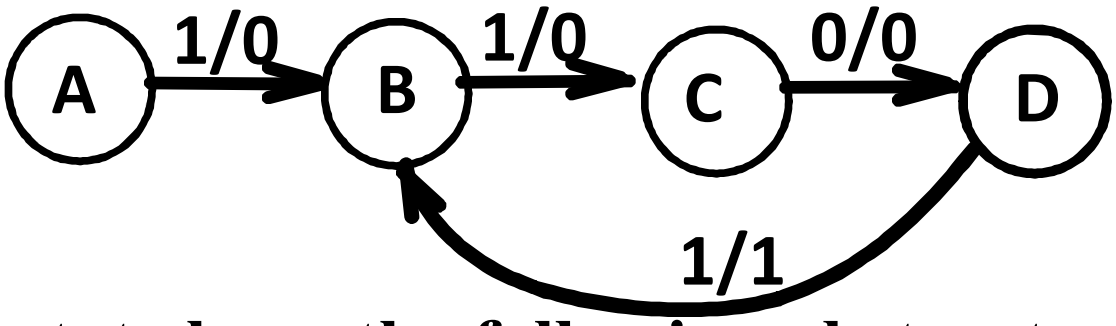
Example: Recognize 1101 (continued)



- **Clearly the final 1 in the recognized sequence 1101 is a sub-sequence of 1101. It follows a 0 which is not a sub-sequence of 1101. Thus it should represent *the same state reached from the initial state after a first 1 is observed.* We obtain:**



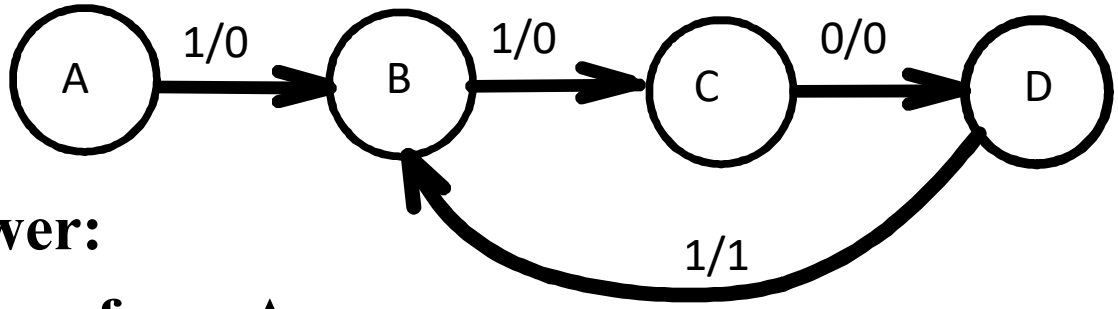
Example: Recognize 1101 (continued)



- **The state have the following abstract meanings:**
  - **A: No proper sub-sequence of the sequence has occurred.**
  - **B: The sub-sequence 1 has occurred.**
  - **C: The sub-sequence 11 has occurred.**
  - **D: The sub-sequence 110 has occurred.**
  - **The 1/1 on the arc from D to B means that the last 1 has occurred and thus, the sequence is recognized.**

Example: Recognize 1101 (continued)

- **The other arcs are added to each state for inputs not yet listed. Which arcs are missing?**



- **Answer:**

"0" arc from A

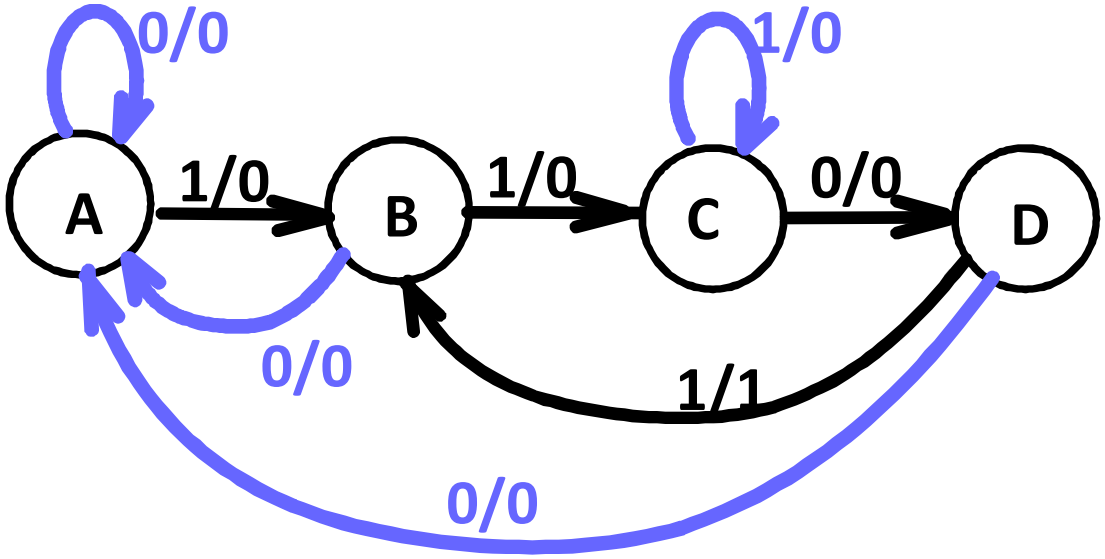
"0" arc from B

"1" arc from C

"0" arc from D.

Example: Recognize 1101 (continued)

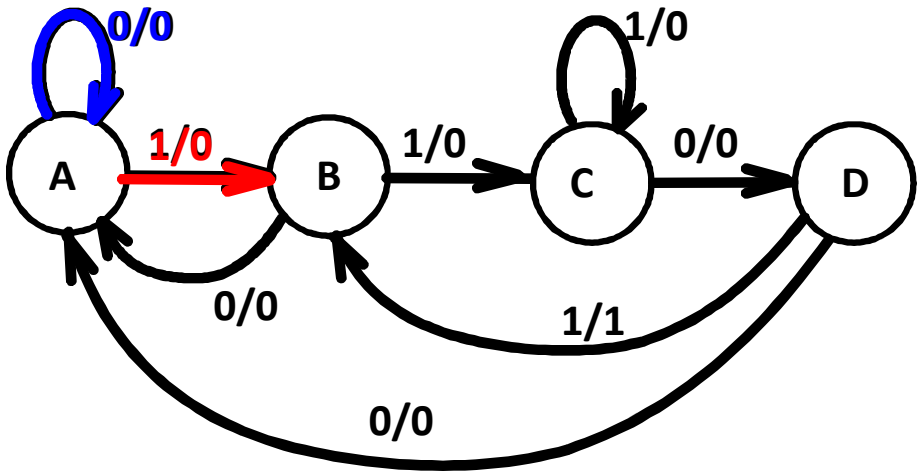
- State transition arcs must represent the fact that an input subsequence has occurred. Thus we get:



- Note that the 1 arc from state C to state C implies that State C means *two or more 1's have occurred.*

Formulation: Find State Table

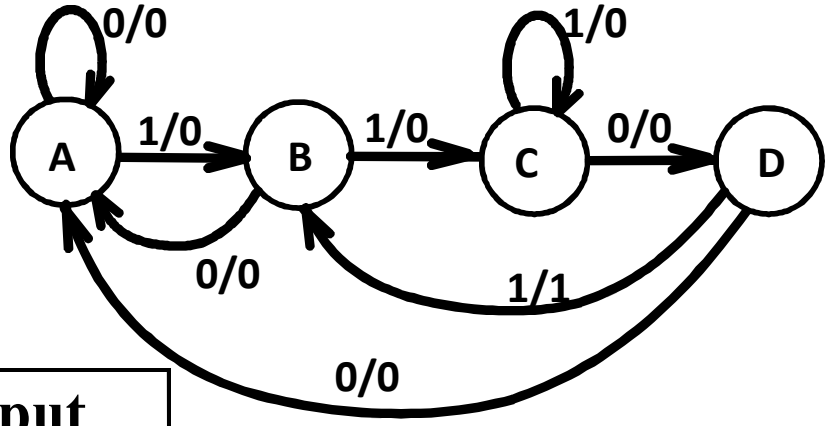
- From the State Diagram, we can fill in the State Table.
- There are 4 states, one input, and one output. We will choose the form with four rows, one for each current state.
- From State A, the 0 and 1 input transitions have been filled in along with the outputs.



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B				
C				
D				

Formulation: Find State Table

- From the state diagram, we complete the state table.



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

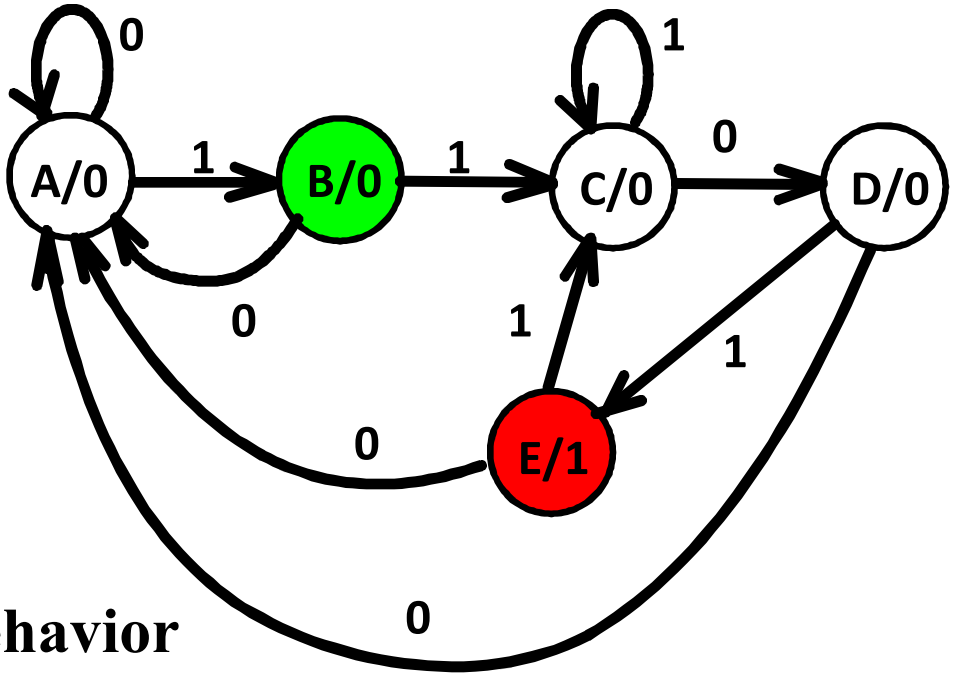
- What would the state diagram and state table look like for the Moore model?

Example: Moore Model for Sequence 1101

- For the Moore Model, outputs are associated with states.
- We need to add a state "E" with output value 1 for the final 1 in the recognized input sequence.
  - This new state E, though similar to B, would generate an output of 1 and thus be different from B.
- The Moore model for a sequence recognizer usually has *more states* than the Mealy model.

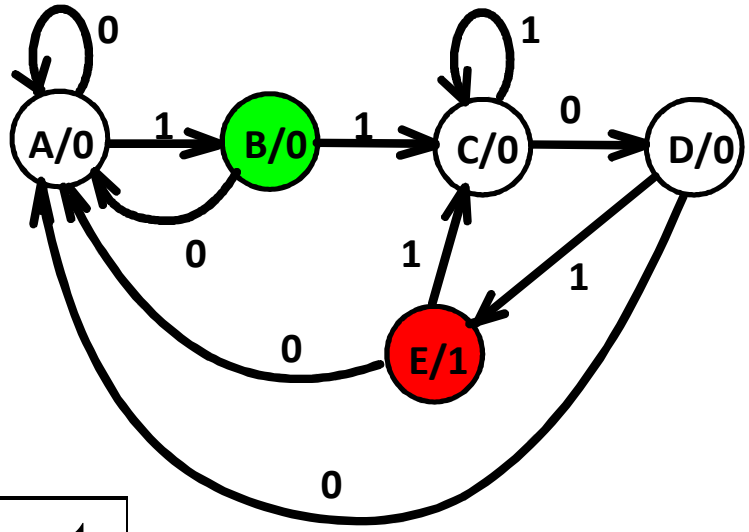
Example: Moore Model (continued)

- We mark outputs on states for Moore model
- Arcs now show only state transitions
- Add a new state E to produce the output 1
- Note that the new state, E produces the same behavior in the future as state B. But it gives a different output at the present time. Thus these states do represent a *different abstraction* of the input history.



Example: Moore Model (continued)

- The state table is shown below
- Memory needs more state in the Moore model: "Moore is More."



Present State	Next State x=0	Next State x=1	Output y
A	A	B	0
B	A	C	0
C	D	C	0
D	A	E	0
E	A	C	1

State Assignment – Example 1

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

- **Does code assignment make a difference in cost?**

• Select codes in such a way that the logic required to implement the flip-flop input equations and output equations is minimized.
• In our example, we simply assign the state codes in Gray code order because it makes it easier for the next-state and output functions to be placed on a K map.

State Assignment – Example 1 (continued)

- **Assignment 1: A = 0 0, B = 0 1, C = 1 0, D = 1 1**
- **The resulting coded state table:**

Present State Y_1Y_2	Next State D_1D_2 x = 0	Next State D_1D_2 x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1

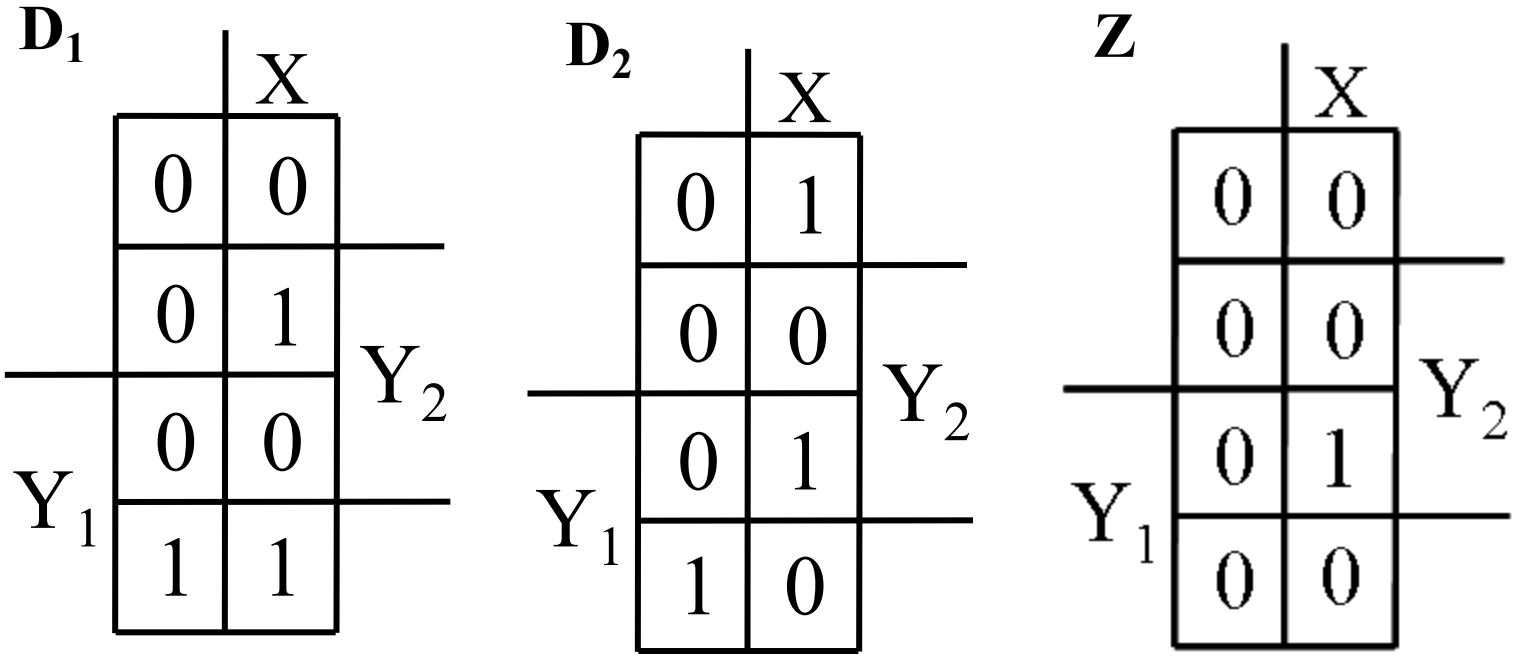
State Assignment – Example 1 (continued)

- **Assignment 2: A = 0 0, B = 0 1, C = 1 1, D = 1 0**
- **The resulting coded state table:**

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

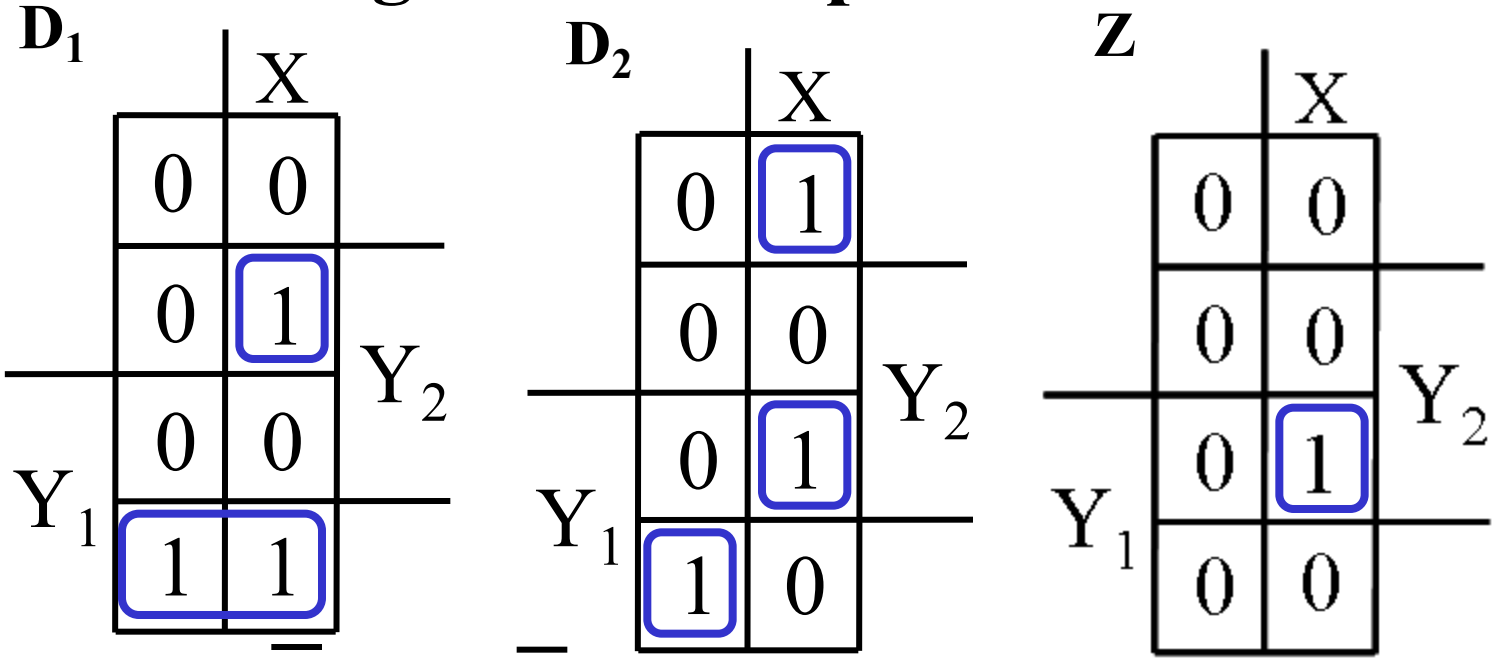
Find Flip-Flop Input and Output Equations: Example 1 - Assignment 1

- Assume D flip-flops
- Interchange the bottom two rows of the state table, to obtain K-maps for D_1, D_2, and Z:



Optimization: Example 1: Assignment 1

- **Performing two-level optimization:**



$$D_1 = Y_1\overline{Y}_2 + X\overline{Y}_1Y_2$$

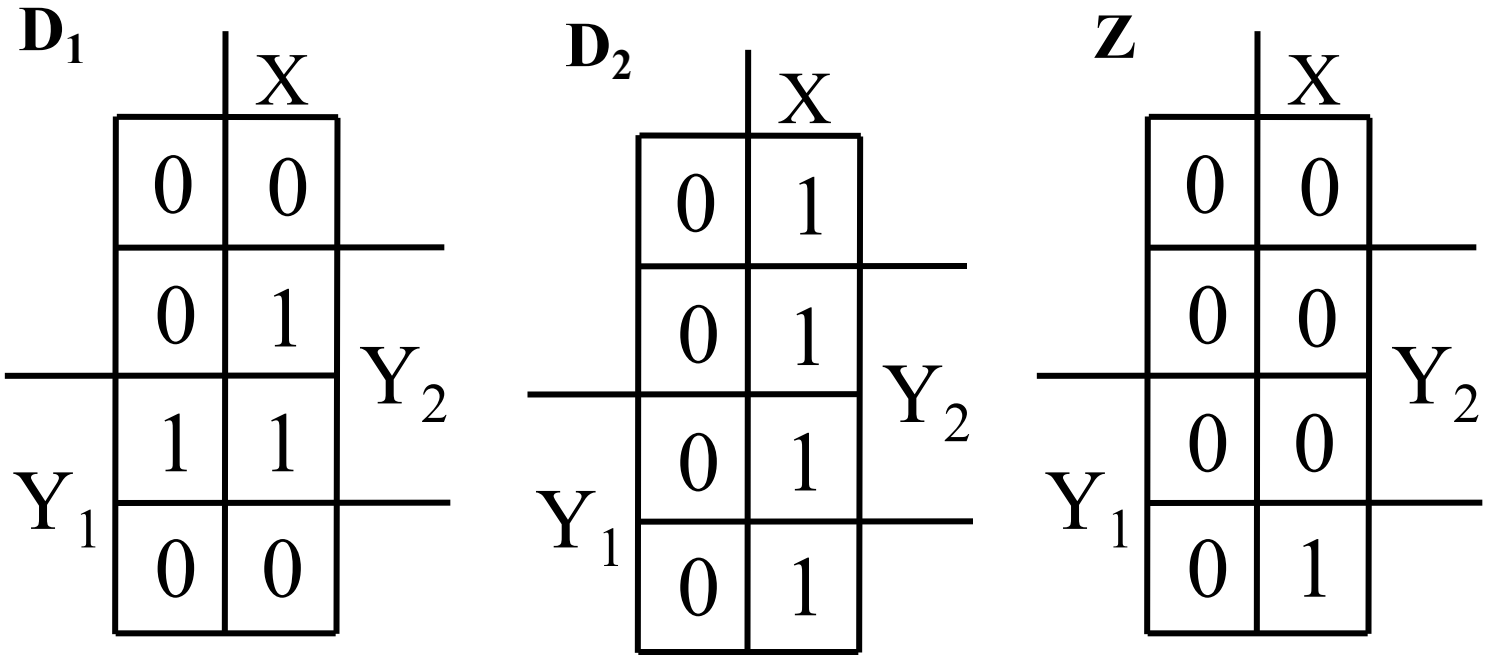
$$D_2 = \overline{X}Y_1\overline{Y}_2 + X\overline{Y}_1\overline{Y}_2 + XY_1Y_2$$

$$Z = XY_1Y_2$$

Gate Input Cost = 22

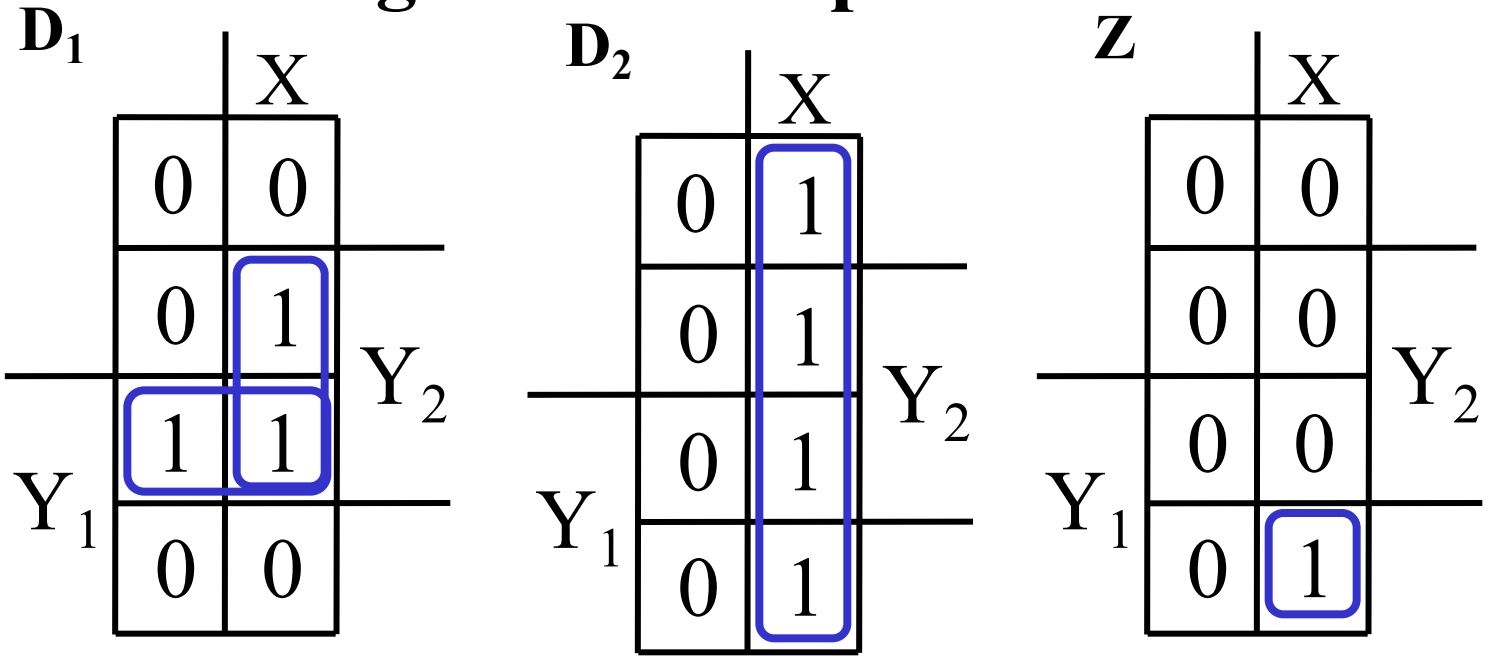
Find Flip-Flop Input and Output Equations: Example 1 - Assignment 2

- Assume D flip-flops
- Obtain K-maps for D_1, D_2, and Z:



Optimization: Example 1: Assignment 2

- **Performing two-level optimization:**



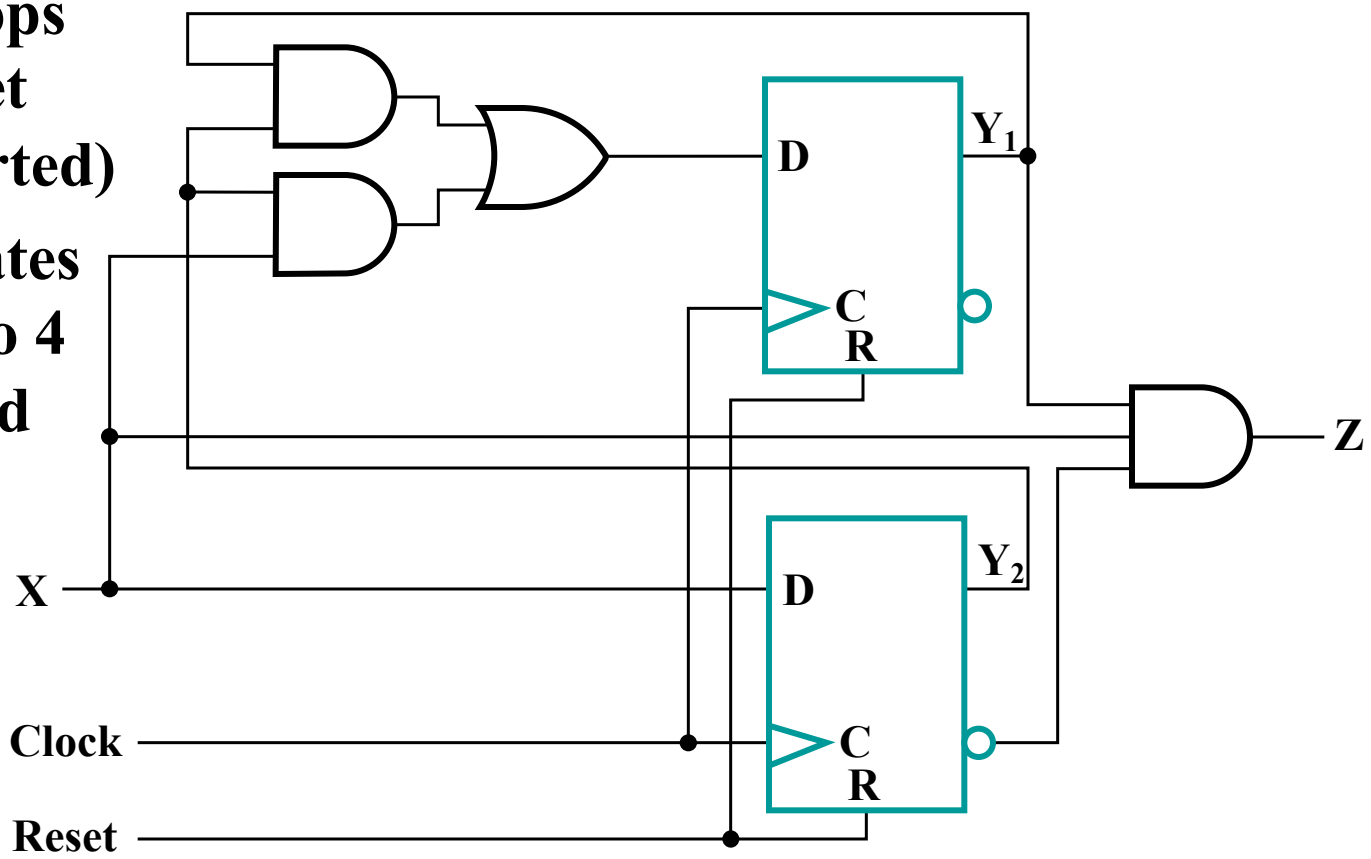
$D_1 = Y_1Y_2 + XY_2$ **Gate Input Cost = 9**

$D_2 = X$ **Select this state assignment for**

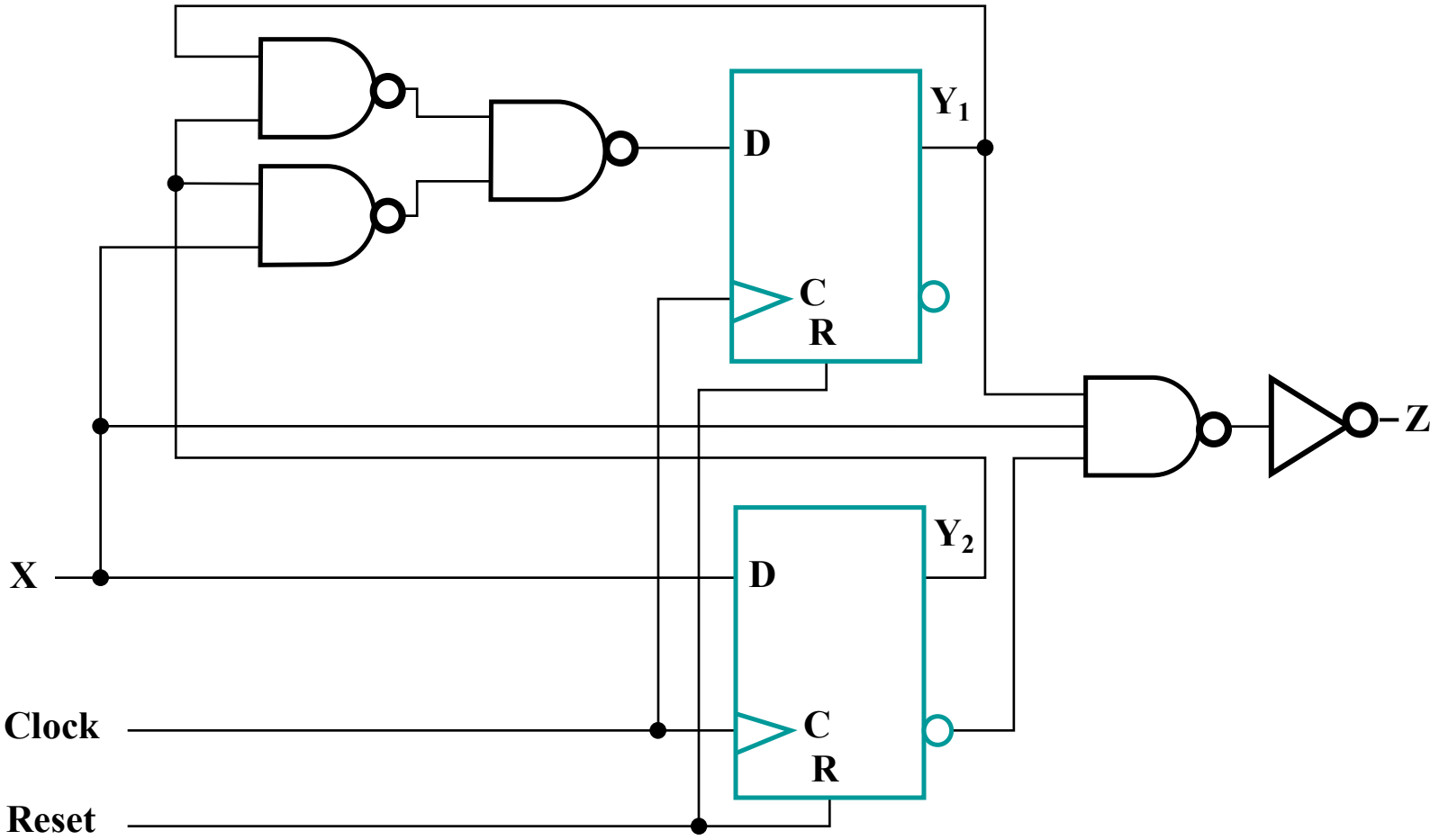
$Z = XY_1\overline{Y}_2$ **completion of the design**

Map Technology

- **Library:**
  - **D Flip-flops with Reset (not inverted)**
  - **NAND gates with up to 4 inputs and inverters**
- **Initial Circuit:**



Mapped Circuit - Final Result



Hints for HW2

- **Q 3-28: Design a 4-to-16-line decoder using two 3-to-8-line decoders and 16 2-input AND gates.**

- **Hint: You can fully use one 3-8 decoder so that you can have 3 inputs and 8 outputs. Meanwhile, you can partially use the second 3-8 decoder so that you have 1 input and 2 outputs. For this second 3-8 decoder, you only need to use one of its inputs (you can connect the other two inputs to the ground so that these two inputs are always zero) and two of its 8 outputs (the other six outputs are simply not used). Note that each of the two outputs of the second 3-8 decoder controls 8 2-input AND gates.**

Q-37

- **An 8x2 AND-OR means 8 AND gates (each of these 8 AND gates has two inputs) and one OR gate with 8 inputs. The 8 outputs of the 8 AND gates are the inputs of the OR gate.**

Weekly Exercises

- **4-13**
- **4-27**