LECTURE 13

DIGITAL LOGIC FAMILIES

The Design Procedure

- **Specification** - Description of the Problem
- **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
  - Derive flip-flop equations from next state entries in the table
- **Output Equation Determination**
  - Derive output equations from output entries in the table
- **Optimization** - Optimize the equations
- **Technology Mapping** - Use available flip-flops and gate technology
- **Verification** - Verify correctness of final design

Formulation: Finding a State Diagram

- A State is an abstraction of the history of the past applied inputs to the sequential circuit
- A state is used to remember something about the history of input combinations applied to the circuit
  - The interpretation of past inputs is tied to the synchronous operation of the circuit
  - An input value is considered 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

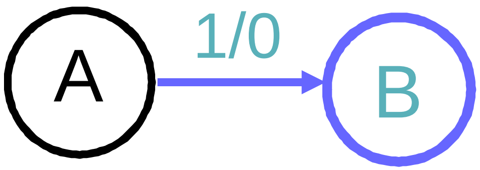
- Begin in an initial state in which NONE of the initial portion of the sequence has occurred (reset state)
- Add a state that recognizes that first symbol has occurred
- Add states that recognize each successive symbol
- The final state represents the input sequence occurence
- Add state transition arcs which specify what happens when a symbol not in the proper sequence has occurred
- Add other arcs which transition to states that represent the input subsequence that has occurred
  - The circuit must recognize the input sequence regardless of where it occurs within the overall sequence

Sequence Recognizer Example

- Example: Recognize the sequence 1101
  - Example: the sequence 1111101 contains 1101
- 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
- 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

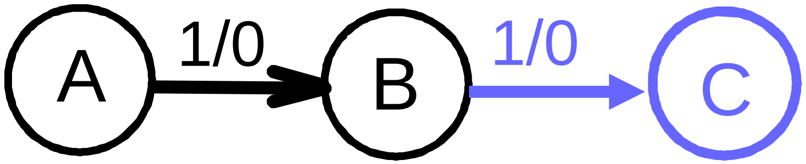
- Define states for the sequence to be recognized:
  - Assuming it starts with first symbol
  - Continues through each symbol in the sequence to be recognized
  - Uses output 1 to mean the full sequence has occurred
  - With output 0 otherwise
- Start in the initial state



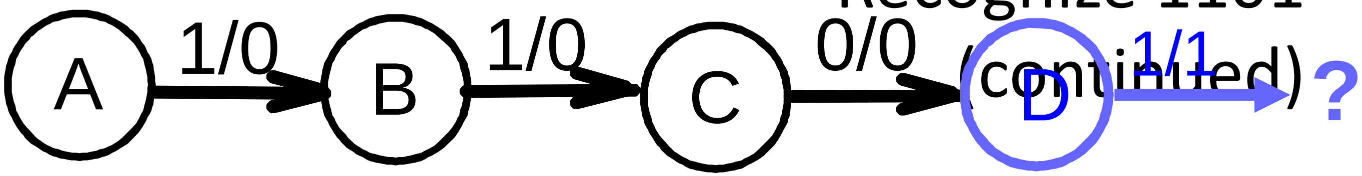
  - State ‘A’ is the initial state
  - Add a state ‘B’ that recognizes the first ‘1’
  - State ‘B’ is the state which represents the fact that the first ‘1’ in the input subsequence has occurred. The output symbol ‘0’ means that the full recognized sequence has not yet occurred

Example: Recognize 1101 (continued)

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

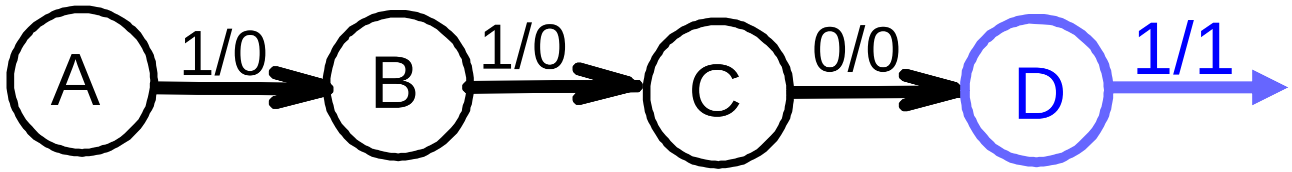


- Finally, after ‘110’ and a ‘1’, we have:

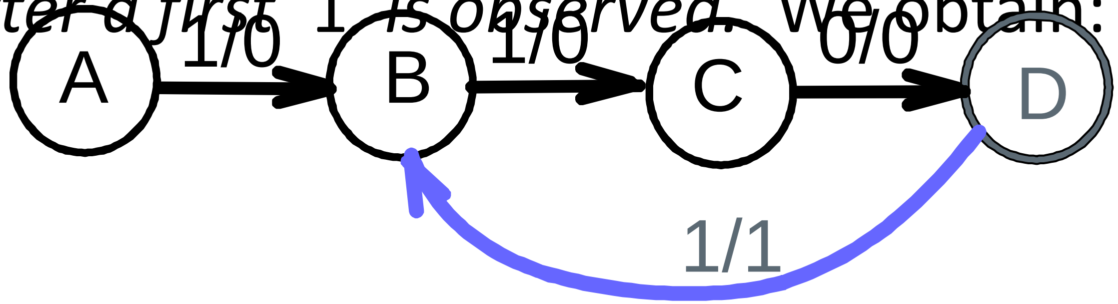


  - Transition arcs are used to denote the output function
  - Output ‘1’ on the arc from D means the sequence is recognized
  - To what state should the arc from state D go? recall 110<u>1101</u>

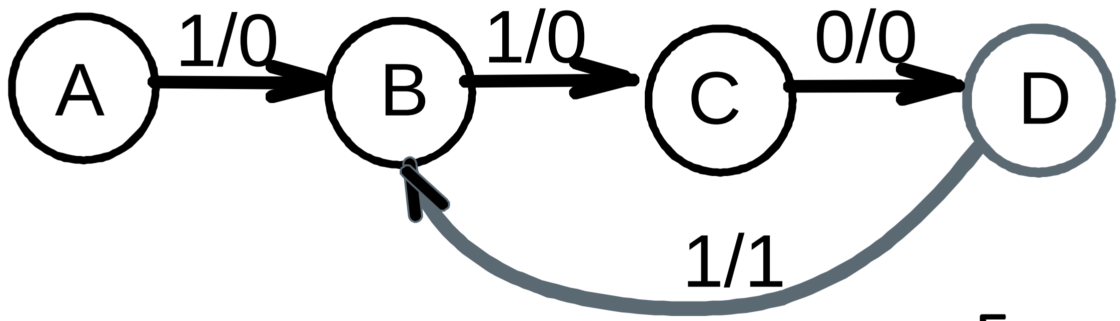
Example: Recognize 1101 (continued)



- 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:



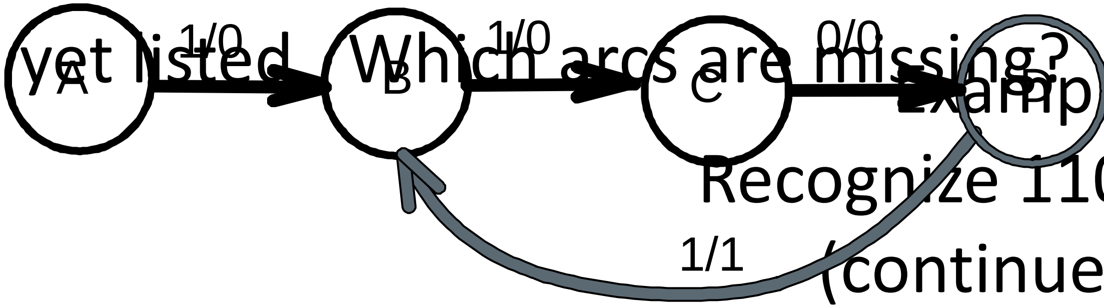
Example: Recognize 1101 (continued)



- The states have the following meanings:
  - A: Start state, no sub-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’ in 1101 has occurred and thus, the output is ‘1’

Example: Recognize 1101 (continued)

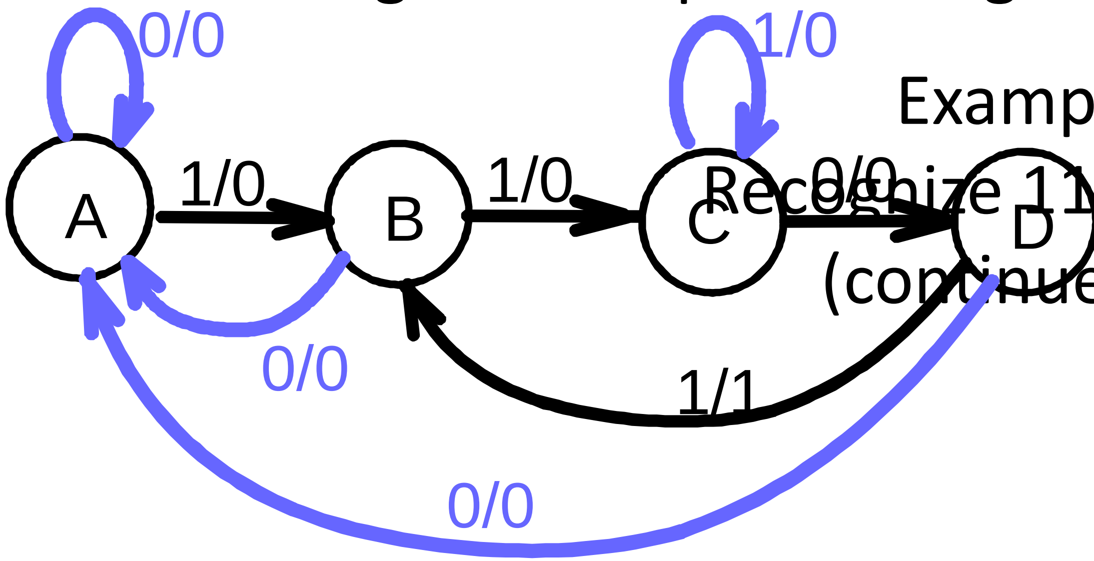
- The other arcs are added to each state for inputs are not yet listed. Which arcs are missing?



- Answer:
  - ‘0’ arc from state A
  - ‘0’ arc from state B
  - ‘1’ arc from state C
  - ‘0’ arc from state D

Example: Recognize 1101 (continued)

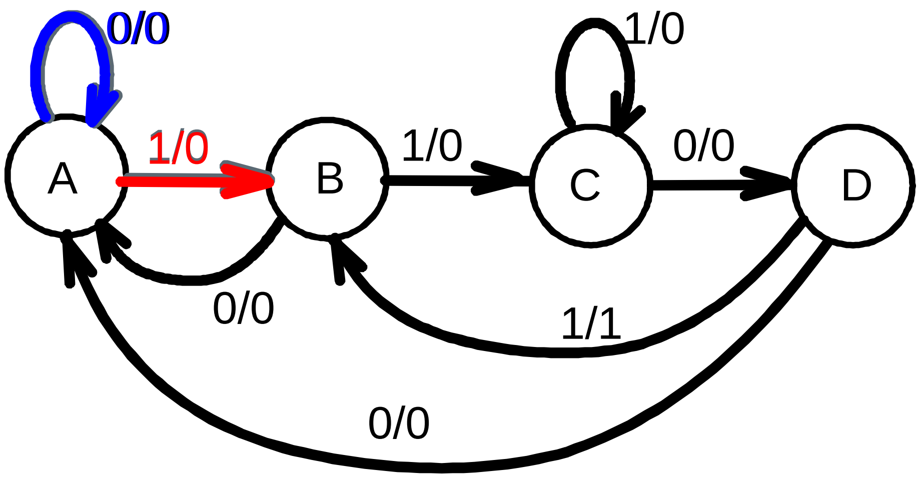
- Add the arcs for missing inputs at any state to make the state diagram complete. We get:



- The ‘1’ arc from state C to itself implies that State C means *two or more 1's have occurred*.

Formulation: Find the State Table

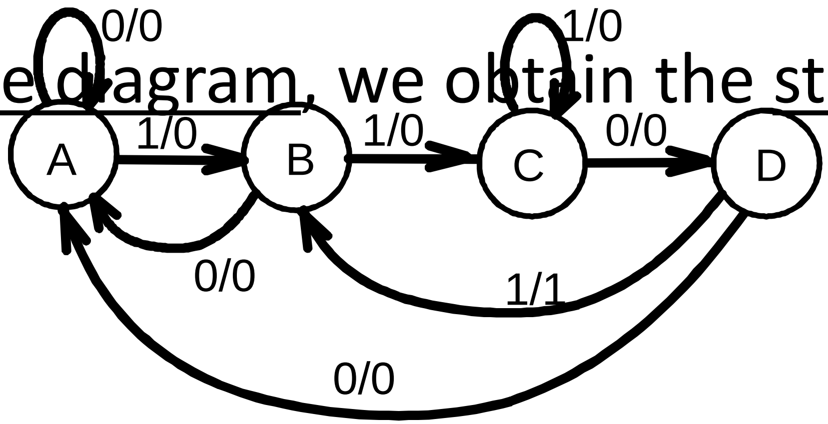
- From the State Diagram, we can fill in the State Table
- There are 4 states, one input, and one output
- We will draw a table with four rows, one for each current state
- From State A, the '0' and '1' input transitions have been filled in along with the outputs



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 obtain 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