Combinational circuits
- output is simply dependent on the current input

Sequential circuits
- output may depend on the input sequence
- The effect of the input sequence can be memorized as a state of the system
- So a sequential circuit is also called a State Machine
- Memory elements (usually D flop-flips) are used to store the state
- System state changes with input
- A different input sequence produces different final state and different output sequence

Example:
- A very simple machine to remember which building I am at
- The only input is the clock signal
- The state machine is represented as a state transition diagram (or called state diagram) below
- One step (i.e., transition) can be taken whenever there is a clock signal

State Transition Diagram (or State Diagram)

Sequential Circuit and State Machine

State Transition Table (State Table)

Counter state machine

Another counter
- Counter need not have number of states that is equal to a power of 2
- Here is a five state counter
- Is it simpler?
In a state transition diagram, state may change with time.
A clock signal represents passage of time.
Each time a clock arrives, state changes to next state.
Clock is an implicit input.
There may or may not be other explicit inputs.

For the previous example, let say we also have an explicit input \( i \).
Next state depends on current state and the value of input \( i \).
When the next state depends upon the inputs, the inputs are examined at the clock edges.

State Machine with Explicit Inputs

State Transition Table with Explicit Inputs

- State transition table will have two sets of inputs:
  - Current state variable and explicit input variables
- Total number of row in table is \( 2^{n+m} \)
  - \( n \) is number of variables representing states
  - \( m \) is number of input variables

<table>
<thead>
<tr>
<th>Current State (( x_0, y_0 ))</th>
<th>Input ( i )</th>
<th>Next State (( x_n, y_n ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ x_n = x_0' y_0' i + x_0 y_0 i' \]
\[ y_n = y_0' i' + y_0 i \]

Output of state machine

- Output of a state machine may depend on state, or state & input:
  - Mealy machine: Output depends on both current state and current input (i.e., depends on transition)
  - Moore machine: Output depends on current state
- Thus we have two different circuits to implement:
  - 1. Decides what is the next state
  - 2. Decides what is the output
- Both circuits are combinational
- States are remembered by memory elements
  - Usually D flips-flops are used to remember states

Overall structure of a State machine

- Moore machine (outputs depend on current state, but not current inputs)
- Mealy machine (outputs depend on both current state and current inputs)

Steps in designing a state machine

- Start writing a state transition diagram
  - It has an initial state
  - It has other states to keep track of various activities
  - It has some transitions
- Generate a state transition table and output table
  - Write state transition table and output table in binary
    - Needs state assignment, i.e., the code used for each state
    - State assignment is a complex process
    - For the time being assume straightforward combinations
- Derive canonical sum-of-product expressions
  - You can simplify the expressions
Determining number of states

- Identify how many different things we need to keep track of
- This is critical to know
- Otherwise the number of states (and their meaning) may get out of hand very quickly
- This is different from what is the output of interest (in each state we may have some outputs)
  - If we need to know how many 1's there are, we need states corresponding to the count
  - If we need to know if we have even or odd number of 1's, we may need only two states

Example

- Design a state machine that will repeatedly display in binary values 1, 3, 5, and 7

Example (contd.)

- We need four states: S0, S1, S2, S3
  - State transition table
  - Implementation level state transition table
  - Output table
  - Implementation level output tables
  - State transitions are
    - S0 -> S1
    - S1 -> S2
    - S2 -> S3
    - S3 -> S4
    - S4 -> S0

Another Example for State Machine

- Design a state machine to display the characters in the string HELLO using a seven segment display
  - How many states do we need?
    - Five, one for each character
  - State transitions are
    - S0 -> S1
    - S1 -> S2
    - S2 -> S3
    - S3 -> S4
    - S4 -> S0

To Detect if # of 1’s in Input is Divisible by 3

- Design a state machine with 1 bit of input and 1 bit of output
  - The output bit will be 1 whenever the number of bits in input sequence is divisible by 3
  - How many states do we need?
  - What are the meaning of the states?
    - In state S0 (00), remainder = 0 (i.e., divisible by 3)
    - In state S1 (01), remainder = 1
    - In state S2 (10), remainder = 2
  - Choose to design a Moore machine
    - Output is 1 whenever in state S0
State machines as sequence detector

- State machine by nature are ideally suited to track state and detect specific sequence of events
- For example, we may design specific machines to track certain pattern in an input sequence
- Examples:
  - to count 1’s in a sequence and produce an output if a specific situation occurs like 3rd one, or every 2nd one, or nth one
  - to generate an output or stop if a specific pattern in the sequence (such as 011 or 0101 or 1111) is observed
- In each of these cases, it is to create a relationship between input and output sequence
- We will review input and output relations for such operations

Example input/output sequences

- n-th one detector, n=2
  - Input: 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1
  - Output: 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0

- n-th one detector, n=3
  - Input: 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1
  - Output: 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0

- 011 pattern detector
  - Input: 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1
  - Output: 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

- 1010 pattern detector
  - Input: 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1
  - Output: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0

How to design sequence detector

- Our goal is to be able to identify minimum number of states
- It is very easy to miss that goal (in terms of number of states)
- Sometimes CAD tools may identify redundant states
- We first discuss the number of possible states to track
  - For example in sequence detection, for 011,
    - we need states representing we have not seen the first zero, we have seen only the first 0, we have seen 01, and finally we have seen 011
    - So a four state system will work
  - 1010 has a pattern that also repeats part of the sequence
    - So we need states that represent starting state, received first 1, first 10, first 101, and finally 1010 (a total of five state)
    - However after we see 1010, we have already seen 10 pattern for the next output (i.e., if we have 101010 repeating)

3rd One Detector

- Use a Mealy machine design
- 3 states are enough
- Have a similar structure to the Moore machine to detect if # of 1’s in Input is Divisible by 3

Design of a sequence detector for 011

- Four states and state transitions are shown in the figure
- Output: 1 for State S3, 0 for all others

Design of a sequence detector for 1010

- Four states and state transitions are shown in the figure
- Output: 1 for State S4, 0 for all others
Another example: a complex vending machine

- Vending Machine
  - Collect money, deliver product and change
  - Vending machine may get three inputs, n, d, q
    - Inputs are nickel (5c), dime (10c), and quarter (25c)
    - Only one coin input at a time
    - Product cost is 40c
    - Does not accept more than 50c (blocks the coin slot)
    - Returns 5c or 10c back
    - Exact change appreciated
- How many states?
- What are the output signals?

Design of Complex Vending Machine

- We are designing a Mealy state machine (i.e., output depends on both current state and inputs).
- Suppose we ask the machine to directly return the coin if it cannot accept an input coin.
- The following two-bit code is used:
  - 00 -- no coin, 01 -- nickel, 10 -- dime, and 11 -- quarter
- Inputs: I1, I2 which represent the coin inserted
- Outputs: C1, C2, P where C1, C2 represent the coin returned and P indicates whether to deliver product
- States: S00, S05, S10, S15, S20, S25, S30, S35
  - 3 bits are enough to encode the states
  - Notice the names (they need not be S0, S1, ...)
- State assignment: S00 -- 000, S05 -- 001, S10 -- 010, S15 -- 011, S20 -- 100, S25 -- 101, S30 -- 110, S35 -- 111

State Diagram for Vending Machine

Algorithmic State Machine (ASM) Charts

- Another way to represent a state machine
- State diagrams are useful when the machine has only a few inputs and outputs
- ASM charts may be more convenient for larger machines

Example: Moore Machine

Example: Mealy Machine