Common Elements in Sequential Design
Lecture 3 topics

- Registers and Register Transfer
- Shift Registers
- Counters
  - Basic Counter
  - Partial sequence counters
  - Other counters
- State Machine Basics
- Review of solution to 11.1
- Units 11 and 14
11.1 solution from text

- The Problem

Assume that the inverter in the given circuit has a propagation delay of 5 ns and the AND gate has a propagation delay of 10 ns. Draw a timing diagram for the circuit showing X, Y, and Z. Assume that X is initially 0, Y is initially 1, after 10 ns X becomes 1 for 80 ns, and then X is 0 again.

- The solution
Register and Register Transfer

- Computers, embedded systems, and other digital devices usually have a data word of a given size.
- Where is the data when it is actively being used?
- Data is typically in a register => register is the size of the data
- Registers are implemented by D F/Fs grouped together.
Loading and using registers

- May have a gated clock or a setup with enable.
- May just have a load signal.
When data needs to be moved

- Typically data is transferred between registers on tri-state busses.
Register with tri-state output

- Logic Diagram

Data transfer
Shift Registers

- Data can be shifted right, left, and sometimes both ways.
Binary Counters

- Simply count

**TABLE 12.13**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C  B  A</td>
<td>C'  B'  A'</td>
<td>T_C  T_B  T_A</td>
</tr>
<tr>
<td>0  0  0</td>
<td>0  0  1</td>
<td>0  0  1</td>
</tr>
<tr>
<td>0  0  1</td>
<td>0  1  0</td>
<td>0  1  1</td>
</tr>
<tr>
<td>0  1  0</td>
<td>0  1  1</td>
<td>0  0  1</td>
</tr>
<tr>
<td>0  1  1</td>
<td>1  0  0</td>
<td>1  1  1</td>
</tr>
<tr>
<td>1  0  0</td>
<td>1  0  1</td>
<td>0  0  1</td>
</tr>
<tr>
<td>1  0  1</td>
<td>1  1  0</td>
<td>0  1  1</td>
</tr>
<tr>
<td>1  1  0</td>
<td>1  1  1</td>
<td>0  0  1</td>
</tr>
<tr>
<td>1  1  1</td>
<td>0  0  0</td>
<td>1  1  1</td>
</tr>
</tbody>
</table>

**Figure 12.13**

Synchronous Binary Counter
Counter with D F/F
Fig 12-15 and Fig 12-16
Up/Down Counter

\[ C' \cdot B' \cdot A' \]

<table>
<thead>
<tr>
<th>CBA</th>
<th>U</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>001</td>
<td>111</td>
</tr>
<tr>
<td>001</td>
<td>010</td>
<td>000</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
<td>001</td>
</tr>
<tr>
<td>011</td>
<td>100</td>
<td>010</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>011</td>
</tr>
<tr>
<td>101</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>101</td>
</tr>
<tr>
<td>111</td>
<td>000</td>
<td>110</td>
</tr>
</tbody>
</table>

Diagram of up/down counter with state transitions and logic gates.
Other sequences

- In a computer assignment you will implement a Gray Code counter. You should familiarize yourself with Gray Code. (wikipedia)
- From it you can see how to implement any counting sequence.
- Will not be covering counters with J-K F/Fs.
State Machine Types

- In digital designs there are two fundamental ways that state machines are implemented

- Mealy Machine
  - Outputs are a function of the current state and the inputs
State Machine Types (2)

- **Moore Machine**
  - Outputs are a function of the current state only

---

Moore Machine

![Moore Machine Diagram]

Inputs → Next State Logic → Excitation (next state) → Current State Memory (F/Fs) → Output Logic → Outputs

The Moore Machine diagram shows the flow of information:
- **Inputs** to **Next State Logic**
- **Excitation (next state)** from **Current State Memory**
- **Outputs** from **Output Logic**
Some historical info


- Formal definition – A Mealy machine is a 6-tuple,
  - A finite set of states
  - A start state (initial state)
  - A finite set called the input alphabet
  - A finite set called the output alphabet
  - A transition function \((T: S \times \Sigma \rightarrow S)\) mapping pairs of a state and an input symbol to the corresponding next state.
  - An output function \((G: S \times \Sigma \rightarrow \Delta)\) mapping pairs of a state and an input symbol to a corresponding output symbol.
Formal Definition of Moore Machine

- Moore machine is named after Edward F. Moore who presented a paper in 1956, “Gedanken-experiments on Sequential Machines.” (formulated)

- Formal Definition:
  - A finite set of states
  - A start state (initial state)
  - A finite set called the input alphabet
  - A finite set called the output alphabet
  - A transition function \((T: S \times \Sigma \rightarrow S)\) mapping a state and the input alphabet to the next state.
  - An output function \((G : S \rightarrow \Delta)\) mapping each state to the output alphabet.
State Machine Design Process

- Either Traditional or Modern
- Tradition Design Methodology for creation of a state machine:
  - From a detail word specification of the problem generate a state graph or state table translating the word specification into a more formal description of the state machine.
  - If a state diagram is used, generate a state table
  - Pick the sequential element for implementation: D F/F, T T/F, J-K F/F, RS F/F
  - Select state machine type – Mealy or Moore
  - Generate the next state and output logic.
State Machine HDL Design Process

- HDL Design Methodology for creation of a state machine:
  - From a detail word specification of the problem generate a state graph or state table translating the word specification into a more formal description of the state machine. Typically it will be a state graph.
  - Select state machine type – Mealy or Moore
  - Write the 3 processes of a HDL description that capture the specification.
    - The process that specifies the F/Fs
    - The process for generation of the next state
    - The process for generation of the outputs.
Assignment

☐ Read and study textbook problems of Unit 11 and Unit 14.