Outline – Sequential Circuits

- Introduction
- Flip-Flops
  - RS
  - JK
  - D
- Latches / Buffers
- Counters
- Shift Registers
- Other Useful Devices
  - 555 Timer
  - 7-Segment Display
- Finite State Machines
  - Examples

Introduction

- Logic circuits are classified into two categories:
  - *Combinational* logic circuits
    - Outputs solely depend on the inputs
    - Combination of basic logic gates
  - *Sequential* logic circuits
    - Outputs depend not only on the inputs but also the history of outputs and/or inputs.
    - Possess memory characteristics
    - Flip-flops (logic gates with feedback connections) are the basic building blocks

Flip-Flops

- Flip-flops are "memory" elements storing one-bit of information.
- Flip-flops play key role in
  - Storing information
  - Counting
  - Data sequencing
  - Timing
- Can be wired from basic logic gates (like NAND).
  - Available in IC form.
- They are categorized into four groups:
  - Reset-Set Flip-Flop (RS F/F)
  - Jam-Kill Flip-Flop (JK F/F)
  - Data Flip-Flop (D F/F)
  - Toggle Flip-Flop (T F/F)

RS Flip-Flop

- User can set or reset (clear) the output Q at will.
  - *Asynchronous* device
- Circuit holds the one bit of data indefinitely when both inputs are high.

<table>
<thead>
<tr>
<th>Mode</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Ñ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prohibited</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reset</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>1</td>
<td>No change!</td>
<td></td>
</tr>
</tbody>
</table>
Clocked RS Flip-Flop

- Clock coordinates the actions of independent units in a complex logic circuit.
- Clocked RS F/F operates in a synchronous fashion.
  - Inputs are in effect when the clock signal goes high.

<table>
<thead>
<tr>
<th>Mode</th>
<th>CLK</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prohibited</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>No change!</td>
</tr>
<tr>
<td>Disabled</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>No change!</td>
</tr>
</tbody>
</table>

JK Flip-Flop

- JK F/F is considered as universal flip-flop.
  - J (jam) sets the output state, Q
  - K (kill) input resets it.
- Device updates its states at the instant when the clock signal goes from either
  - Low to high logic state (positive edge triggered) OR
  - High to low logic state (negative edge triggered).

Pulse generator emits a pulse at the transition points of the clock signal allowing inputs take effect for a short duration of time.

Pulse Generators of JK F/F

Positive-Edge Trigger:
- Inverter with delay

Negative-Edge Trigger:
- Inverter with delay

Operating Modes of JK F/F

Positive-Edge Triggered:
- Mode: Toggle, Set, Reset, Hold
- Table:

<table>
<thead>
<tr>
<th>Mode</th>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>Set</td>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reset</td>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hold</td>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>No change!</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>No change!</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>No change!</td>
<td></td>
</tr>
</tbody>
</table>

Negative-Edge Triggered:
- Mode: Toggle, Set, Reset, Hold
- Table:

<table>
<thead>
<tr>
<th>Mode</th>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle</td>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>Set</td>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reset</td>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hold</td>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>No change!</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>No change!</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>No change!</td>
<td></td>
</tr>
</tbody>
</table>
**JK F/F with Clear & Preset**

<table>
<thead>
<tr>
<th>Mode</th>
<th>PR</th>
<th>CLR</th>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q</th>
<th>̅Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynch. Set</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Asynch. Clear</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Prohibited</strong></td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>No change!</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Toggle</td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>̅Q</td>
<td>̅Q</td>
</tr>
</tbody>
</table>

In **7476** (double F/Fs in chip), PR and CLR inputs are to set and reset the output states of the device in an asynchronous fashion.

---

**D Flip-Flop**

**Level Triggered:**

- Data
- Clock
- CLK
- D
- Q

**Edge Triggered:**

- Data
- Clock
- CLK
- J
- Q
- K

---

**D F/F with Clear & Preset**

<table>
<thead>
<tr>
<th>Mode</th>
<th>PR</th>
<th>CLR</th>
<th>CLK</th>
<th>D</th>
<th>Q</th>
<th>̅Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynch. Set</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Asynch. Clear</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Prohibited</strong></td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>No change!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In **7474** (double F/Fs in chip), PR and CLR inputs are to set and reset the output states of the device in an asynchronous fashion.

---

**Latch**

- Latch is a collection of level-sensitive D flip-flops.
  - Captures n-bit data.
  - Stores data which is to be later used by slower devices.

---

**7475**

<table>
<thead>
<tr>
<th>Mode</th>
<th>E</th>
<th>D</th>
<th>Q</th>
<th>̅Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Enabled</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Latch Enabled</td>
<td>0</td>
<td>0</td>
<td>No change!</td>
<td></td>
</tr>
</tbody>
</table>
Tri-state Buffers

- When several devices share the same data transfer line (bus), the connections of an inactive device must be electrically isolated during this idle period to protect all devices from short-circuiting.
- Three-state buffers or so-called tri-state buffers are employed for this purpose:
  - They have one (electrically-controlled) state called “high impedance.” (HI)
  - At that time, the connection of inactive device appears to be severed:
    - A very high impedance is observed by external circuitry.

Counters

- Counters are used to count events like clock pulses. They can be utilized to:
  - Divide frequency
  - Store data
- Digital counters are selected considering the following attributes:
  - Maximum number of counts (in bits)
  - BCD vs. binary
  - Up / down count
  - Asynchronous or synchronous operation
  - Free-running or self-stopping

4-bit BCD Counter

- 74192 is a synchronous binary-coded decimal (BCD) up/down counter.
- When the clock is connected to the UP pin, it counts up.
  - Unused clock pin (DN or UP) must be tied to Vcc.
- Initial states of the flip-flop are loaded from the data input pins when LD is 0.
- Counter can be reset when CLR = 1.
- Several counters could be cascaded for higher-order counting.
Two Decimal-Digit Counting

\[ V_{cc} = 5V \]

U1:74192
\[
\begin{array}{c}
Q_3 \\
Q_2 \\
Q_1 \\
Q_0 \\
I_3 \\
I_2 \\
I_1 \\
I_0
\end{array}
\]

U2:74192
\[
\begin{array}{c}
Q_3 \\
Q_2 \\
Q_1 \\
Q_0 \\
I_3 \\
I_2 \\
I_1 \\
I_0
\end{array}
\]

Resetting Counters

\[ V_{cc} = 5V \]

External logic elements are employed to reset the counters automatically when a certain count value is reached (say, 60).

Popular ICs

- **74279**: Quad SR Flip-Flop
- **7474**: Dual D-type Flip Flop
- **7475**: 4-bit Latch
- **7476**: Dual JK-type Flip Flop
- **74125**: Tri-state Buffer
- **74163**: 4-bit Binary Counter
- **74192**: 4-bit BCD Counter
- **7493**: 4-bit Binary Counter

Shift Registers

- Shift registers are used for **sequencing** and **interfacing** between serial and parallel communication systems.
- Binary sequence is shifted left or right at each clock pulse.
  - While shifting, the register must hold the content of the shifted data.
- The registers are classified as
  - Serial-in serial-out (SISO) shift registers
  - Parallel-in serial-out (PISO) shift registers
  - Serial-in parallel-out (SIPO) shift registers
  - Parallel-in parallel-out register (or simply "register")
Shift Registers

Serial in / Serial out:
```
<table>
<thead>
<tr>
<th>Serial in</th>
<th>Serial out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1 0 1</td>
<td>0 0 1 ...</td>
</tr>
</tbody>
</table>
```

Parallel in / Serial out:
```
<table>
<thead>
<tr>
<th>Parallel in</th>
<th>Serial out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1 0 1</td>
<td>0 0 1 ...</td>
</tr>
</tbody>
</table>
```

Serial in / Parallel out:
```
<table>
<thead>
<tr>
<th>Serial in</th>
<th>Parallel out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1 0 1</td>
<td>1 0 1 ...</td>
</tr>
</tbody>
</table>
```

4-bit SISO (Right) Shift Register

- Serial Input: \( D \)
- Clock: \( FF1 \)
- Q3 Q2 Q1 Q0
- Serial out: 1 0 1 1

4-bit SISO (Left) Shift Register

- Serial Input: \( D \)
- Clock: \( FF1 \)
- Q3 Q2 Q1 Q0
- Serial out: 1 0 1 1

Shift Register ICs

- Some commercially available shift registers are:
  - 7495: 4-bit shift register
  - 74164: 8-bit serial in, parallel out
  - 74165: 8-bit serial/parallel-in serial out
  - 74195: 4-bit parallel access
Timers

- Reliable timing source, which is to synchronize the operation of various units of the circuit, is needed in digital circuits.
- For simple logic circuits, 555 timer, which has a timing range from $\mu$s to minutes, is utilized.
- It has two operation modes:
  - Astable (free running) multivibrator
  - Mono-stable (one-shot) multivibrator
- The timing parameters of the device is adjusted via two resistances and a capacitor.

Schmitt Trigger as Clock Source

- Schmitt trigger incorporates a hysteresis loop (relay) and is frequently employed:
  - to filter glitches
  - to shape up slowly varying signals
- For time-insensitive applications, an inverting Schmitt trigger (like 7414) could be utilized to generate a clock signal.

7414 Schmitt Trigger (Cont’d)

With R and C values, one can select the oscillation frequency of output voltage waveform:

$$f_{osc} = \frac{1}{0.853 \cdot RC}$$
**7 Segment Displays**

Common Anode: (LS 5025)

![Common Anode Diagram]

Common Cathode: (LS 5015)

![Common Cathode Diagram]

---

**Seven-segment Decoder**

- 7447 decoder is frequently used to drive a 7-segment display.
- Input to the decoder is a 4-bit BCD.
- Its outputs are connected to the corresponding pins of a (common-anode) 7-segment display.
- Display is active when all control inputs are high.

---

**Example – 7 Segment Display**

- ![Example Diagram]

---

**Finite State Machines**

- Logic circuits with memory
  - Outputs = f(inputs, past inputs, past outputs)
  - Such circuits have finite states:
    - State indicates the state of a memory device (F/Fs) in circuit.
    - State is generally an input or an output to a combinational logic circuit.
  - Circuit goes thru a sequence of states in a cycle.
    - Different actions are performed at each state.
  - Example: Door combination lock
FSM Architectures

Two types of FSM architectures are commonly used:

**Mealy FSM:**
- Inputs... Outputs
- Combinational Logic Block
- Clock

**Moore FSM:**
- Inputs... Outputs
- Input Logic Block
- Output Logic Block
- Clock

---

State Transition Diagrams

- State transition diagrams (STD) are used to visualize/analyze the operation of an FSM.
  - Each state (S) is represented by a circle.
  - Transition from one state to another is indicated by an arrow.
  - Corresponding inputs (X) necessary to make that transition happen are shown above the arrows.
  - Proceeding a slash, output (Y) at the present state is also given.

---

FSM Model

- D (Data) F/F is associated with each encoded state.
- Combinational input logic generates next set of states.
- Outputs depend on present states of machine.
  - If necessary, inputs are taken into consideration.

---

Design Procedure

- Begin with circuit function specifications
  - Verbal description of the circuit’s operation
- Draw state transition diagram (STD)
- Create state transition table (STT)
  - Tabular form of state diagram
  - Very similar to truth table
- Decide on the representation of the states (a.k.a. "state encoding")
  - Use counters whenever possible
  - Simplifies the design
  - Create encoded STT
- Derive Boolean expressions for each state
- Draw circuit diagram
  - F/F for each state
  - Design combinatorial logic circuits to implement encoding
Example 1*

Design a circuit for odd-parity checking:
- Accepts a bit stream as input.
- If the number of ones in the stream is odd, the output becomes high (1).

Solution

- Before a new bit arrives, the circuit needs to remember whether the number of bits in previous “step” is even or odd.
- A natural choice for the states is:
  - Even (0) and Odd (1)

Solution (Cont’d)

State Transition Table:

<table>
<thead>
<tr>
<th>PS</th>
<th>IN</th>
<th>OUT</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even</td>
<td>0</td>
<td>0</td>
<td>Odd</td>
</tr>
<tr>
<td>Even</td>
<td>0</td>
<td>0</td>
<td>Even</td>
</tr>
<tr>
<td>Odd</td>
<td>1</td>
<td>1</td>
<td>Even</td>
</tr>
<tr>
<td>Odd</td>
<td>1</td>
<td>0</td>
<td>Odd</td>
</tr>
</tbody>
</table>

Boolean Expression:

\[ NS = PS \oplus IN \]

\[ OUT = PS \]

Example 2*

- Design the FSM of a newspaper dispenser.
- Each newspaper is, say, 15¢.
- Exact change is required to get the newspaper:
  - 5¢ (“Nickel”) + 10¢ (“Dime”)
  - 5¢ + 5¢ + 5¢
- A deposit of 2×10¢ is also accepted but no nickel is returned!
- With correct change, the lock of the dispenser is to be released.
Solution

- **Inputs:** N (Nickel) and D (Dime)
  - Lock mechanism generates RESET.
- **States:** 0¢, 5¢, 10¢, and 15¢.
- **Output:** OPEN

### Truth Table

<table>
<thead>
<tr>
<th>PS</th>
<th>D</th>
<th>N</th>
<th>NS</th>
<th>OPEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>5¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>10¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>5¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>10¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>15¢</td>
<td>0</td>
</tr>
<tr>
<td>10¢</td>
<td>0</td>
<td>0</td>
<td>10¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>15¢</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>15¢</td>
<td>1</td>
</tr>
</tbody>
</table>

Chapter 7b

Solution (Cont’d)

- Let us **uniquely** encode 4 states using two F/Fs:
  - D₁ and D₀ are their data inputs (Next states).
  - Q₁ and Q₀ are their outputs (Present states).
    - 0¢ → Q₁ = 0; Q₀ = 0
    - 5¢ → Q₁ = 0; Q₀ = 1
    - 10¢ → Q₁ = 1; Q₀ = 0
    - 15¢ → Q₁ = 1; Q₀ = 1

### Circuit Schematics:

Examining STT closely yields

\[
D_1 = Q_1 + Q_0 \cdot N + D \\
D_0 = Q_1 \cdot (N + D) + Q_0
\]

\[\text{OPEN} = Q_1 \cdot Q_0\]

Note that a FSM must be initialized correctly. Hence, **Set** and **Reset** inputs of D F/Fs are used to set the initial states of the circuit.

Chapter 7b

One-hot (State) Encoding

- Thanks to its simplicity, **one-hot state** encoding are often-time employed in digital circuit design:
  - One D F/F for each state of the FSM
  - Easy encoding
  - Simplifies derivation of Boolean expressions
- During the operation of the circuit, only one state is active (“hot”) while the remaining ones are inactive (“cool”).
- Not very efficient utilization of resources.
Example 2 - Revisited

This time, four states will be encoded using four F/Fs:

- 0¢ $\rightarrow Q_3 = 0; Q_2 = 0; Q_1 = 0; Q_0 = 1$
- 5¢ $\rightarrow Q_3 = 0; Q_2 = 0; Q_1 = 1; Q_0 = 0$
- 10¢ $\rightarrow Q_3 = 0; Q_2 = 1; Q_1 = 0; Q_0 = 0$
- 15¢ $\rightarrow Q_3 = 1; Q_2 = 0; Q_1 = 0; Q_0 = 0$

<table>
<thead>
<tr>
<th>Present States</th>
<th>Next States</th>
<th>Boolean Terms:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_0$</td>
<td>$Q_1$</td>
<td>$Q_2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$D_0 = Q_0 D' N'$
$D_1 = Q_1 N + Q_0 D' N'$
$D_2 = Q_2 D + Q_1 N + Q_0 D' N'$
$D_3 = Q_3 D + Q_2 (D + N) + Q_1$

OPEN = $Q_3$