CSC-270 Circuits

Week 6 — Spring 2019

Dominique Thiébaut
dthiebaut@smith.edu
Feedback on Lab Reports

• Pay attention to details: RS latch implemented with NORs is activated by 1 on R or S. Implemented with NANDs, activation is with 0 on R or S.

• Use the proper units. Frequencies in Hertz (Hz, kHz, MHz, GHz). Times in seconds (ms, us, ns, ps)

• Always ask yourself "does it make sense?" or "is it the right quantity?"

• Add part numbers (74, 02, 08, 32…) to your schematics
Outline for This Week

• Moore vs Mealy FSM
  • Unwanted States
  • JK Flipflops
  • ROM Sequencers
Two Types of Controllable FSMs: Moore & Mealy
Flip-Flops

Clock

Input

Combinational

Future State

Current State

Moore Machine

Output

Combinational
Mealy Machine

Input

Clock

Flip-Flops

Combinational

Output
How to Deal with Unwanted States?

S0 → S1 → S2 → S3

D. Thiebaut, Computer Science, Smith College
Option 1
Option 2

### Truth Table

<table>
<thead>
<tr>
<th>n</th>
<th>n+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q0</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>

\[ D_0 = Q_1 \oplus Q_0 = Q + Q_2 \]
Moore Machine
With Input Command

- Input
- Combinational
- Flip-Flops
- Clock
- Output
- Combinational
An Example
2 states ⇒ 1 bit
⇒ 1 flip flop

$S_0 : Q = 0$
$S_1 : Q = 1$

<table>
<thead>
<tr>
<th>Cmd</th>
<th>$S^n$</th>
<th>$S^{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$S_0$</td>
<td>$S_0$</td>
</tr>
<tr>
<td>0</td>
<td>$S_1$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>1</td>
<td>$S_0$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>1</td>
<td>$S_1$</td>
<td>$S_0$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cmd</th>
<th>$Q^n$</th>
<th>$Q^{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$D^n = cmd \oplus Q$
• If the FSM is controlled by $n$ inputs, then there must be $2^n$ arrows leaving each state in a state diagram.

• In a Moore machine, the input affects the output only if it lasts until the next clock tick!
Create a sequencer controlled by 1 switch. When the switch outputs 0, the sequencer actives 3 lights in this sequence: Green, Yellow, Red, Green, etc.

When the switch outputs 1, the sequencer activates the lights in Red, all off, Red, all off, etc.
2 Flipflops $Q_0, Q_0$

<table>
<thead>
<tr>
<th>$\text{Cmd} \cdot Q_0 \cdot Q_0$</th>
<th>$\bar{Q}_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$1$</td>
<td>$0$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

$D_0 = \overline{\text{Cmd}} \cdot \overline{Q_0} \cdot \overline{Q_0} + \text{Cmd} \cdot Q_0 \cdot \overline{Q_0} = \overline{Q_0} \left( \overline{\text{Cmd}} \cdot \overline{Q_0} + \text{Cmd} \cdot Q_0 \right) = \overline{Q_0} \left( \overline{\text{Cmd}} \cdot Q_0 \right)$

$D_1 = \text{Cmd} + \text{Cmd} \cdot Q_0$

$= \text{Cmd} + Q_0$
# Initialize the flip-flops and the input switch
Q1 = 0
Q0 = 0
S = 0

def NOT( a ):
    return 1 - a

# simulate 20 ticks

for step in range( 20 ):
    # wait for the next clock tick (the user presses Enter)
    # get the value of the switch S from the user. User
    # must enter 0 or 1.
    while True:
        S = input( "> " ).strip()
        if S != "0" and S != "1":
            print( "Invalid input signal! Please reenter" )
        else:
            S = int( S )
            break

    # combine switch value with current outputs (present)
    # to compute what the future outputs will be.
    D1 = NOT( S ) | ( Q0 ^ Q1 )
    D0 = NOT( Q1 & Q0 )

    # show the present outputs of the flip-flops
    print( "S Q1Q0 =%d %d%d" % ( S, Q1, Q0 ) )

    # Tick! Whatever is on the D input of flip-flop
    # becomes the output.
    Q1 = D1
    Q0 = D0

http://www.science.smith.edu/dftwiki/index.php/CSC270_Python_simulator_for_Controllable_Sequencer
Exercise

Implement this FSM

S0 \rightarrow S1
0 \rightarrow 0
GY'R'B' \rightarrow G'Y'R'B'
S1 \rightarrow S2
1 \rightarrow 1
G'Y'R'B' \rightarrow G'Y'R'B'
S2 \rightarrow S3
1 \rightarrow 1
GY'R'B' \rightarrow G'Y'R'B'
S3 \rightarrow S0
0 \rightarrow 0
GY'R'B' \rightarrow G'Y'R'B'

D. Thiebaut, Computer Science, Smith College
How to deal with conditions that will never happen?

Word problem might say: “\(Y, Z\) and \(T\) are never 1 at the same time.”

<table>
<thead>
<tr>
<th></th>
<th>ZT</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td>1</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>1</td>
<td>(X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Outline for This Week

• Moore vs Mealy FSM
• Unwanted States
  • JK Flipflops
• ROM Sequencers
The JK Flipflop
Characteristic Table

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Qt+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td>Qt</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Qt'</td>
</tr>
</tbody>
</table>

- stays the same
- J follows J
- oscillates
### Characteristic Table

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Qt+1</th>
<th>Qt'</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td>Qt</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Qt'</td>
<td></td>
</tr>
</tbody>
</table>

The truth table shows how the state of the J-K flip-flop changes based on the inputs J and K.

- **Qt**: The current state of the output Q.
- **Qt'**: The next state of the output Q'.

The JK flip-flop is a type of synchronous flip-flop that has two inputs, J and K, and one output, Q.

#### Circuit Diagram

![JK flip-flop circuit diagram]

- **J** and **K** are the input signals.
- **Q** is the primary output.
- **Q'** is the secondary output.
- **Sd’** and **Cd’** are the enable inputs.

The circuit is designed to perform the operations specified in the characteristic table.

### Truth Table

<table>
<thead>
<tr>
<th>Qn</th>
<th>Qn+1</th>
<th>Jn</th>
<th>Kn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
How can we make a D flipflop with a JK flipflop?
Exercise: Redo this with JKs
Implement this FSM with JKS.
$S^n | S^{n+1}$

$S_0 | S_1$
$S_1 | S_2$
$S_2 | S_3$
$S_3 | S_0$

$J_1 = Q_1 + Q_0$
$k_1 = \overline{Q_1 \cdot Q_0}$

$K_0 = Q_0 \cdot Q_1$

$G = \overline{Q_1 + Q_0}$
$Y = \overline{Q_1 \cdot Q_0}$
$R = Q_1$
Redo this one with JKS as well!
\[ a + \overline{a} b = a b \]

\[ \begin{array}{c|cc|c|c}
\text{Cmd} & S^n & S^{n+1} & J^0 & J^1 \\
\hline
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array} \]

\[ J_0 = \overline{\text{Cmd}} (Q_1 + Q_0) + \text{Cmd} \]

\[ K_0 = \overline{\text{Cmd}} + \text{Cmd} Q_1 Q_0 \]
Where else do we find Similar FSMs and State Diagrams?
Instruction Cycle (with Interrupts) - State Diagram

http://images.slideplayer.com/18/6064420/slides/slide_3.jpg
Outline for This Week

• Moore vs Mealy FSM
• Unwanted States
• JK Flipflops

• ROM Sequencers
Implementing a ROM-Based Sequencers
Terminology

- ROM = Read-Only Memory
- PROM = Programmable Read-Only Memory
- EPROM = Erasable Programmable Read-Only Memory
- EEPROM = Electrically-Erasable Programmable ROM
NMOS 128 Kbit (16Kb x 8) UV EPROM

- FAST ACCESS TIME: 200ns
- EXTENDED TEMPERATURE RANGE
- SINGLE 5 V SUPPLY VOLTAGE
- LOW STANDBY CURRENT: 40mA max
- TTL COMPATIBLE DURING READ and PROGRAM
- FAST PROGRAMMING ALGORITHM
- ELECTRONIC SIGNATURE
- PROGRAMMING VOLTAGE: 12V

DESCRIPTION
The M27128A is a 131,072 bit UV erasable and electrically programmable memory EPROM. It is organized as 16,384 words by 8 bits.

The M27128A is housed in a 28 Pin Window Ceramic Frit-Seal Dual-in-Line package. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written to the device by following the programming procedure.

Figure 1. Logic Diagram

Figure 2. DIP Pin Connections

Read Mode
The M27128A has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. G (Enable) is the power control and should be used for device selection. Output Enable (E) is the output control and should be used to gate data to the output pins, independent of device selection.

Assuming that the addresses are stable, address access time (tAC) is equal to the delay from E to output (tCE). Data is available at the outputs after the falling edge of G, assuming that E has been low and the addresses have been stable for at least two tAC.

Standby Mode
The M27128A has a standby mode which reduces the maximum active power current from 85mA to 40mA. The M27128A is placed in the standby mode by applying a TTL high signal to the E input. When in the standby mode, the outputs are high impedance state, independent of the E input.

Two Line Output Control
Because EPROMs are usually used in larger memory arrays, this product features a 2 line control function which accommodates the use of multiple memory connection. The two line control function allows:

a. the lowest possible memory power dissipation,
b. complete assurance that output bus contention will not occur.

DEVICE OPERATION
The seven modes of operation of the M27128A are listed in the Operating Modes table. A single 5V power supply is required in each mode. All inputs are TTL levels except for VPP and 12V on A9 for Electronic Signature.
<table>
<thead>
<tr>
<th>A0</th>
<th>0000</th>
<th>0110 1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0000 1100</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>0010</td>
<td>0110 0000</td>
</tr>
<tr>
<td>0011</td>
<td>0000 0001</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>0100</td>
<td>0110 1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>1011</td>
<td>0000 1100</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>0000 0000</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>0000 0001</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>1111 1100</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>0000 0000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exercise
Exercise

Implement this FSM With a ROM-based Sequencer
What about user inputs and a ROM-based Sequencer?
Flip-flops vs ROM Sequencers: Advantages/Disadvantages
What other sequencers are out there…

See Mano, Section 6.3

Ready for Lab 6!