CSC270 — Circuits

Spring 2020—Week #4

Dominique Thiébaut
dthiebaut@smith.edu
Happy (belated) Valentine's Day!
Lessons Learned
The Transistor
What is f(a)?

Transistor = Switch
Practical Design Rules for Discrete Parts
Input Switch

1 KΩ
Logic Indicator
Logic Indicator
Exercise

What is $Y(a)$?

(Typical Midterm Question)
Variables that are inverted in the expression of the inputs must go through an odd number of inversions…
Lab #2 Reports
• Decoders, cascading decoders
• Multiplexers
• Journey into mystery!
Python Decoder

def decoder( enable, a, b, c):
    L = 8*[0]
    if enable==1:
        decimal = c + b*2 + a*4
        L[decimal] = 1
    return L

http://www.science.smith.edu/dftwiki/index.php/3-to-8_Decoder_in_Python
3-to-8 with 2-to-4s?

<table>
<thead>
<tr>
<th>a b c</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e b c</th>
<th>3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 x x</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 0 0 0</td>
</tr>
</tbody>
</table>
3-to-8 with 2-to-4s?
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e</th>
<th>b</th>
<th>c</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```python
def decoder2to4( enable, a, b ):
    L = 4*[0]
    if enable==1:
        decimal= b+a*2
        L[decimal] = 1
    return L

def decoder3to8( enable, a, b, c ):
    Y7_Y4 = decoder2to4( a, b, c )
    Y3_Y0 = decoder2to4( not a, b, c )
    return Y3_Y0+Y7_Y4

def main():
    print( "a  b  c  | Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7" )
    print( "-----------------------------------" )
    for a in [0, 1]:
        for b in [0, 1]:
            for c in [0, 1]:
                Y = decoder3to8(1, a, b, c)
                print( "%-3d%-3d%-3d%-3d%-3d%-3d%-3d%-3d" % (a,b,c, Y[0],Y[1],Y[2],
                                                    Y[3], Y[4], Y[5], Y[6], Y[7] ) )
    main()
```
def decoder2to4(enable, a, b):
    L = 4*[0]
    if enable==1:
        decimal= b+a*2
        L[decimal] = 1
    return L

def decoder3to8(enable, a, b, c):
    Y7_Y4 = decoder2to4(a, b, c)
    Y3_Y0 = decoder2to4(not a, b, c)
    return Y3_Y0+Y7_Y4

def main():
    print("a b c | Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7")
    print("----------------")
    for a in [0, 1]:
        for b in [0, 1]:
            for c in [0, 1]:
                Y = decoder3to8(1, a, b, c)
                print("%-3d%-3d%-3d%-3d%-3d%-3d%-3d%-3d"% (a,b,c, Y[0],Y[1],Y[2],
                        Y[3], Y[4], Y[5], Y[6], Y[7] ))
    main()
https://destination-nouvellezelande.com/plus-belles-cascades-nouvelle-zelande/
Multiplexers
4-to-1 Mux

I_0  I_1  I_2  I_3  \[ \rightarrow \]  Y

C_1  C_0  \[ \leftarrow \]
4-to-1 Mux

(Mechanical Analogy)
4-to-1 Mux

(Mechanical Analogy)
4-to-1 Mux

(Mechanical Analogy)
4-to-1 Mux

(Mechanical Analogy)
What logic circuit will we find inside the Mux?

4-to-1 Mux

\[ Y = I_0 \cdot C_0 + I_1 \cdot C_0 + I_2 \cdot C_1 + I_3 \cdot C_1 \]

\( I_0, I_1, I_2, I_3 \) are the inputs, and \( C_0, C_1 \) are the select lines.
8-INPUT MULTIPLEXER

The TTL/MSI SN54/74LS151 is a high speed 8-input Digital Multiplexer. It provides, in one package, the ability to select one bit of data from up to eight sources. The LS151 can be used as a universal function generator to generate any logic function of four variables. Both assertion and negation outputs are provided.

- Schottky Process for High Speed
- Multifunction Capability
- On-Chip Select Logic Decoding
- Fully Buffered Complementary Outputs
- Input Clamp Diodes Limit High Speed Termination Effects

CONNECTION DIAGRAM DIP (TOP VIEW)

J SUFFIX
CERAMIC
CASE 620-09

N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-03

PIN NAMES

VCC = PIN 16
GND = PIN 8
○ = PIN NUMBERS
A Memory Chip

Figure  Organization of a 1K x 1 memory chip.

http://www.slideshare.net/binbinno/memory-systems-n
Exercise 1: Mux as a Generic Building Block

<table>
<thead>
<tr>
<th>AB</th>
<th>(C_0)</th>
<th>(C_1)</th>
<th>(O_0)</th>
<th>(O_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_{D0})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(C_{D1})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(C_{D11})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(C_{D10})</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise 1: Mux as a Generic Building Block

<table>
<thead>
<tr>
<th>AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- Inputs: \( C_0 \) and \( C_1 \)
- Outputs: \( Y \)
- Signal lines: \( I_0, I_1, I_2, I_3 \)
What does **Combinational** Mean?

Is \( f(a,b) \) always predictable? Always deterministic?
We stopped here last time...
FOOL!! DID YOU THINK YOU COULD ESCAPE US??

YOU'VE NEVER READ A TALE LIKE "THE SORCERER!"

GO BACK! BACK TO THE DARK WORLD FROM WHENCE YOU COME! BEGONE! I COMMAND YOU!
How does this work?
How does this work?

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q1</th>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Timing Diagrams

\[ a \rightarrow f \rightarrow b \]

\[ \text{time} \]

\[ \text{time} \]

\[ \text{time} \]
(Demo starts around 1min 52sec)

https://www.youtube.com/watch?v=mo4Lq0DvJ68
Timing Diagrams
The RS Latch

- $R = \textbf{Reset}$, $S = \textbf{Set}$, $Q =$ usual name for the output of a circuit that has a “state”
- The circuit has a feedback loop
- The circuit has a state
The RS Latch Explained
The RS Latch Explained
Ouroboros

From Wikipedia, the free encyclopedia

For other uses, see Ouroboros (disambiguation).

The ouroboros or uroboros (/ˈjoʊərəˈbɔːrəs, joʊˈroʊ-/,[3] from the Greek οὐροβόρος ὀφίς tail-devouring snake) is an ancient symbol depicting a serpent or dragon eating its own tail.
The RS Latch Explained

\[ S \rightarrow Q_1 \rightarrow S \]

\[ Q_2 \rightarrow \neg Q_2 \rightarrow R \]
The RS Latch Explained

The RS latch is a type of memory circuit used in digital electronics. It consists of two inputs, R (Reset) and S (Set), and two outputs, Q1 and Q2. The diagram shows the basic operation of the RS latch.

When R is 0 and S is 1, the output Q1 is 1 and Q2 is 0. Conversely, when R is 1 and S is 0, Q1 is 0 and Q2 is 1. The RS latch is often used in digital systems to store information temporarily.
The RS Latch Explained
The RS Latch Explained
The RS Latch Explained

S → Q1
R → Q2

0 → Q2
1 → Q1
The RS Latch Explained

\[ S \quad Q_1 \quad 0 \]
\[ Q_2 \quad R \quad 1 \]
The RS Latch Explained

S

Q_1

Q_2

R

1

0
The RS Latch Explained
The RS Latch Explained

\[ \begin{align*}
S & \quad Q_1 \quad 0 \\
Q_2 & \quad R \\
\end{align*} \]
The RS Latch Explained

\[ S \quad Q_1 \quad 0 \]
\[ Q_2 \quad R \quad 1 \]
The RS Latch Explained
The RS Latch Explained
The RS Latch Explained
Latch = Sequential Circuit
Important Properties of the Latch

• Gates must have ability to *pass* and to *block*

• There must be a *loop*

• There must be *two inversions* in the loop
Concepts of Passing and Blocking
Concepts of Passing and Blocking

\[ \begin{align*}
1 & \rightarrow a \quad a' \\
1 & \rightarrow 0 \quad 1 \\
1 & \rightarrow 1 \quad 1 \\
1 & \rightarrow 0 \quad 0
\end{align*} \]
The D-Flip-flop

D

Clk

Q'

Q
The D-Flip-flop

D

0

Clk

Q'

Q
The D-Flip-flop

D

Clk

Q'

Q

1
The D-Flip-flop
Meet the D-Flip-flop!

The SN54/74LS74A dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and Q outputs.

Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.

### MODE SELECT — TRUTH TABLE

<table>
<thead>
<tr>
<th>OPERATING MODE</th>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S_D</td>
<td>S_D</td>
</tr>
<tr>
<td>Set</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>Reset (Clear)</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>Undetermined</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Load “1” (Set)</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Load “0” (Reset)</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

* Both outputs will be HIGH while both S_D and C_D are LOW, but the output states are unpredictable if S_D and C_D go HIGH simultaneously. If the levels at the set and clear are near V_{IL} maximum then we cannot guarantee to meet the minimum level for V_{OH}.

H, h = HIGH Voltage Level
L, l = LOW Voltage Level
X = Don’t Care
i, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

### ORDERING INFORMATION

- **J SUFFIX**
  - CERAMIC: CASE 632-08
- **N SUFFIX**
  - PLASTIC: CASE 646-06
- **D SUFFIX**
  - SOIC: CASE 751A-02

**SN54LSXXJ**
- Ceramic

**SN74LSXXN**
- Plastic

**SN74LSXXD**
- SOIC
Meet the D-Flip-flop!

When a latch becomes "edge-triggered" we call it a Flip-flop.
Definitions

- **Period** ($T$) (measured in ms, us, ns)
- **Frequency** $= \frac{1}{T}$ \hspace{1em} (1/ms=kHz, 1/us=MHz, 1/ns=GHz)

Amplitude (Volts)
Outline

D-Flip-Flops in Computers

Exploring Finite State Machine with D-Flip-Flop

From Word Problem to FSM

Exercise

Moore vs. Mealy Machines
Where Do We Find D-Flip-Flops?
Outline

D-Flip-Flops in Computers

Exploring Finite State Machine with D-Flip-Flop

From Word Problem to FSM

Exercise

Moore vs. Mealy Machines
Characteristic Table

Characteristic Tables

A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form.

D Flip-flop

<table>
<thead>
<tr>
<th>D</th>
<th>Qt+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

D  
---

Qt+1 
---

Clk  
---

Q  
---

time  
---
Informal Exploration of an FSM (Automata)
What is the behavior of this circuit?
<table>
<thead>
<tr>
<th>Qn</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

D-Flip-Flops in Computers

Exploring Finite State Machine with D-Flip-Flop

From Word Problem to FSM

Exercise

Moore vs. Mealy Machines
Can we redo this, the other way around? Start with the state diagram and figure out the circuit?
Word Problem

- Design a Finite State Machine (FSM) that oscillates between 2 states, and outputs 1 in State 1 and 0 in State 0