CSC270—Circuits

Spring 2019—Week 5

Dominique Thiébaut
dthiebaut@smith.edu
Cartoon of the Day...

THIS IS WHAT LEARNING LOGIC GATES FEELS LIKE

SEE, YOU JUST CONNECT THIS 12 INPUT REVERSE
FLIP-FLOP TO THE CONTROLLED TWO-THIRDS ADDER,
WHICH RESETS THE LATCHES IN THE NOT-NAND RELAY
ARRAY, THEN LOOP BACK TO ODD-NUMBER INPUTS
AND REVERSE ALL YOUR SWITCHES!

AND WHAT’S THAT DO?
SUBTRACTION.

Sent to us
By Prof. Kristen
Dorsey

https://www.smbc-comics.com/comic/logic-gates
Oscilloscope: Video 1

https://www.youtube.com/watch?v=UHr1TxijBSM&feature=youtu.be
Oscilloscope: Video 2

https://www.youtube.com/watch?v=ta096oBzSac&feature=youtu.be
Feedback on Lab Reports

• “…as shown in Figure 1…” Use **upper case** when you refer to a figure number or table number

• “…we measured **about** 5V…” Be precise and report the exact measurement! (see next slide)

• Do not forget to add pin numbers and part numbers to your diagrams

• If you’re using a transistor explain how to identify the **base**, the **collector** and the **emitter**

• Your report should have all the information needed to redo the lab

• Be concise, thorough, and precise
Template for Lab Report

- Title
- List the **equipment used**
- Present **one section** for each experiment
  - What the **purpose** of the experiment is (1 to 2 sentences)
  - What the **setup** is (wiring diagram)
  - How the experiment is **conducted** (activate switches). You may show photos.
  - What the **observations** are. If quantities are measured, report on the values measured with the correct unit. If behavior is observed, show truth tables, or timing diagram, whatever is appropriate
- **Summarize** the findings
About Voltage Measurements

Vcc can vary quite a bit!
Minterms in Python

Adder/Subtractor: more info

Where do we find D-Flipflops?

Simplest FSM

Word Problem

Exercises

Controlled FSM
def f(a, b, c):
    '''f: boolean function of 3 variables'''
    return (a and (not b)) or c

def g(a, b, c):
    '''g: boolean function of 3 variables'''
    return (not a) or (not b) or (not c)

def h(a, b, c):
    '''h: boolean function of 3 variables'''
    return a or not b

def problem3(a, b, c, d):
    '''problem3: boolean function from Homework 2, of 4 variables'''
    return not (a or b) and (not c) and (c or d)

print("g(a,b,c) = Sigma( %s )\n      %", ", ".join([str(k) for k in mintermsOf3(g)]) )

print("h(a,b,c) = Sigma( %s )\n      %", ", ".join([str(k) for k in mintermsOf3(h)]) )

print("pb3(a,b,c,d) = Sigma( %s )\n      %", ", ".join([str(k) for k in mintermsOf4(problem3)]) )

main()
def f(a, b, c):
    """f: boolean function of 3 variables. Just pass the name of the function.""
    return (a and not b) or c

def g(a, b, c):
    """g: boolean function of 3 variables. Just pass the name of the function.""
    return not a or (b and c)

def h(a, b, c):
    """h: boolean function of 3 variables. Just pass the name of the function.""
    return a or not b

def problem3(a, b, c, d):
    """problem3: boolean function of 4 variables. Just pass the name of the function.""
    return not (a or b or c or d)

def mintermsOf4(func):
    """returns a list of minterms (as ints) for a boolean function of 4 variables. Just pass the name of the function.""
    minterms = []
    for a in [0, 1]:
        for b in [0, 1]:
            for c in [0, 1]:
                for d in [0, 1]:
                    if func(a, b, c, d) == 1:
                        minterms.append(a*8+b*4+c*2+d)
    return minterms

def main():
    print("f(a,b,c) = Sigma( %s )" % ", ".join([str(k) for k in mintermsOf3(f)]))

    print("g(a,b,c) = Sigma( %s )" % ", ".join([str(k) for k in mintermsOf3(g)]))

    print("h(a,b,c) = Sigma( %s )" % ", ".join([str(k) for k in mintermsOf3(h)]))

    print("pb3(a,b,c,d) = Sigma( %s )" % ", ".join([str(k) for k in mintermsOf4(problem3)]))

main()
Minterms in Python

```python
def f(a, b, c):
    '''f: boolean function of 3 variables.''
    return (a and b) or (not a and c)

def g(a, b, c):
    '''g: boolean function of 3 variables.''
    return (not a) or (a and b)

== RESTART: /Users/thiebaut/Desktop/MintermsOf4.py ==

f(a,b,c) = Sigma( 1, 3, 4, 5, 7 )
g(a,b,c) = Sigma( 0, 1, 2, 3, 4, 5, 6 )
h(a,b,c) = Sigma( 0, 1, 4, 5, 6, 7 )
pb3(a,b,c,d) = Sigma( 1 )

print( "g(a,b,c) = Sigma( %s )"
    % "", ".join( [str(k) for k in mintermsOf3(g)] ) )

print( "h(a,b,c) = Sigma( %s )"
    % "", ".join( [str(k) for k in mintermsOf3(h)] ) )

print( "pb3(a,b,c,d) = Sigma( %s )"
    % "", ".join( [str(k) for k in mintermsOf4(problem3)] ) )

main()
```
• Minterms in Python

• Adder/Subtractor: more info
  • Where do we find D-Flipflops?
  • Simplest FSM
  • Word Problem
  • Exercises
  • Controlled FSM
More on Adder/Subtractor

```
11 00000000 48656C6F20746865-
13 00000009 7265210A0A
14
15
16
17
18
19
20
21 00000000 B804000000
22 00000005 BB01000000
23 0000000A B9[00000000]
24 0000000F BA0E000000
25 00000014 CD80
26
27
28 00000016 BB00000000
29 0000001B B801000000
30 00000020 CD80

section .data
Hello db "Hello there!", 10, 10
HelloLen equ $-Hello

section .text
_global _start

;;; print message
mov eax, 4 ; write
mov ebx, 1 ; stdout
mov ecx, Hello ;
mov edx, HelloLen ;
int 0x80

;;; exit
mov ebx, 0
mov eax, 1
int 0x80
```
## More on Adder/Subtractor

### ADD

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Reg,Reg</td>
<td>0000001woorrrmmm</td>
</tr>
<tr>
<td>Mem,Reg</td>
<td></td>
<td>0000000woorrrmmm</td>
</tr>
<tr>
<td>Reg,Mem</td>
<td></td>
<td>0000001woorrrmmm</td>
</tr>
<tr>
<td>Acc,Imm</td>
<td></td>
<td>0000010w</td>
</tr>
<tr>
<td>Reg,Imm8</td>
<td></td>
<td>1000001wo000mmm</td>
</tr>
<tr>
<td>Mem,Imm8</td>
<td></td>
<td>1000001wo000mmm</td>
</tr>
<tr>
<td>Reg,Imm</td>
<td></td>
<td>1000000wo000mmm</td>
</tr>
<tr>
<td>Mem,Imm</td>
<td></td>
<td>1000000wo000mmm</td>
</tr>
</tbody>
</table>

### SUB

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB</td>
<td>Reg,Reg</td>
<td>0010101woorrrmmm</td>
</tr>
<tr>
<td>Mem,Reg</td>
<td></td>
<td>0010100woorrrmmm</td>
</tr>
<tr>
<td>Reg,Mem</td>
<td></td>
<td>0010101woorrrmmm</td>
</tr>
<tr>
<td>Acc,Imm</td>
<td></td>
<td>0010110w</td>
</tr>
<tr>
<td>Reg,Imm8</td>
<td></td>
<td>1000001wo101mmm</td>
</tr>
<tr>
<td>Mem,Imm8</td>
<td></td>
<td>1000001wo101mmm</td>
</tr>
<tr>
<td>Reg,Imm</td>
<td></td>
<td>1000000wo101mmm</td>
</tr>
<tr>
<td>Mem,Imm</td>
<td></td>
<td>1000000wo101mmm</td>
</tr>
</tbody>
</table>
Where Do We Find D-Flip-Flops?
• Minterms in Python

• Adder/Subtractor: more info

• Where do we find D-Flipflops?

• Simplest FSM
  • Word Problem
  • Exercises
  • Controlled FSM
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>
Informal Exploration of a Finite-State Machine (FSM)
What is the behavior of this circuit?
<table>
<thead>
<tr>
<th></th>
<th>Qn</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Minterms in Python
• Adder/Subtractor: more info
• Where do we find D-Flipflops?
• Simplest FSM
• Word Problem
• Exercises
• Controlled FSM
Can we redo this, but the other way around? That is, we 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
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>
Exercise

Generate an FSM that controls 3 lights: Green, Yellow, and Red, and cycles through all 3, staying 1 second on Green, 1 second on Yellow, and 2 seconds on Red.

(Take good notes, you will have to redo all these steps soon, for a different problem)
# GYRSequencer.py
# D. Thiebaut
# A very quick and dirty way to check a sequencer...
# This sequencer activates a G, Y, and R light system
# s.t. G is on for 1 cycle, followed by Y for 2 cycles
# followed by R for 1 cycle. Then the whole cycle repeats

Q1 = 0
Q0 = 0

def NOT( a ):
    return 1 - a

for step in range( 20 ):
    # the Q1 and Q0 outputs go through combinational
    # logic to generate the new values
    # of D1, D0, and the outputs G, Y, R...
    D1 = Q0
    D0 = NOT( Q1 )
    G = NOT( Q1 ) & NOT( Q0 )
    Y = Q0
    R = NOT( G | Y )

    # show the stable circuit signals
    print( "Q1Q0 = %d %d | GYR = %d %d %d"
           % ( Q1, Q0, G, Y, R ) )

    # wait for the next clock tick
    # (the user presses Enter)
    input( "> " )

    # as soon as the clock has ticked,
    # D1 and D0 get latched in the flipflops
    # and Q1 and Q0| reflect the values captured.
    Q1 = D1
    Q0 = D0

http://www.science.smith.edu/dftwiki/index.php/CSC270_GYRSequencer.py

<table>
<thead>
<tr>
<th>Q1Q0</th>
<th>GYR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0</td>
</tr>
</tbody>
</table>
Exercise to Do in Pairs

Generate an FSM that controls 3 lights: Green, Yellow, and Red, and cycles through all 3, staying 1 second on Green, 1 second on Yellow, and 1 second on Red.

Different from previous exercise!
Recap
Outline

- Minterms in Python
- Adder/Subtractor: more info
- Where do we find D-Flipflops?
- Simplest FSM
- Word Problem
- Exercises
- Controlled FSM
We can easily introduce input **signals** that will modify the behavior of the FSM
Controllable FSM
Flip-Flops

Combinational

Clock

Moore Machine

Input

Output
Flip-Flops → Combinational → Mealy Machine

Output

Input

Clock