UE 2-2 Electronic Engineering

Labs 1 - Digital Electronic simulation tools

Submitted by **Meldrick Reimmer, Khoi Nhat Pham and Selma Boudissa**

Contents

1 Problem 1

We create a class called Wire in python.

```
# Pham Nhat Khoi-Reimmer Meldrick and Selma Boudissa - BSCV 2016-2017
# Professor: Cedric Lemaitre
# Electrical Engineering Assignment
# Lab 1

import matplotlib.pyplot as plt
import numpy as np

# Problem 1
class Wire (object):

  # set default attributes for wire
    def __init__(self, state = False):
    # boolean state indicates if there is a current on this wire
    # this is set to false which is no current by default
        self.state = state
    # downStream stores logical gates that connect by this wire
    # but in downstream level (current is the stream)
        self.downStream = []

    def addDownStream(self, gate):
    # mutator to add a gate to downstream attribute of this wire
        self.downStream.append(gate)

    def setState(self,state):
    # mutator to set current state on this wire
        self.state = state
    # at the same time tell downstream gates to perform their responses
        for gate in self.downStream:
            gate.update()

    def getState(self):
    # accessor to get the state of current flows in this wire
        return self.state
```

lab1.py

We created the Wire class with these attributes as:

1. a constructor
2. a method addDownstream
3. methods setState and getState which allow to manipulate state

2 Problem 2

For this part we had to create class for different logic gates

```
# Problem 2
class AndGate(object):

  # set default attributes for this gate
    def __init__(self, input_a, input_b, output):
    # input and output here is the instances of class WIRE

        self.inputa = input_a
    # add this gate as downstream of input a
        input_a.addDownStream(self)

        self.inputb = input_b
    # add this gate as downstream of input a
        input_b.addDownStream(self)

        self.output = output
        self.update()

    def update(self):
    # check the current state on input wires, do the logical operation
        result = self.inputa.getState() and self.inputb.getState()
    # and come up changing the state on output wire
        self.output.setState(result)


# same idea with AndGate, only the logical is altered to OR instead of AND
class OrGate(object):

    def __init__(self, input_a, input_b, output):

        self.inputa = input_a
        input_a.addDownStream(self)

        self.inputb = input_b
        input_b.addDownStream(self)

        self.output = output
        self.update()

    def update(self):
        result = self.inputa.getState() or self.inputb.getState()
        self.output.setState(result)


# same idea with AndGate, only the logical is altered to NOT instead of AND
class NotGate(object):

    def __init__(self, input_a, output):
        self.inputa = input_a
        input_a.addDownStream(self)

        self.output = output
        self.update()

    def update(self):
        result = not self.inputa.getState()
        self.output.setState(result)

# same idea with AndGate, only the logical is altered to XOR instead of AND
class XorGate(object):

    def __init__(self, input_a, input_b, output):

        self.inputa = input_a
        input_a.addDownStream(self)

        self.inputb = input_b
        input_b.addDownStream(self)

        self.output = output
        self.update()

    def update(self):

        if self.inputa.getState() != self.inputb.getState():
            self.output.setState(True)
        else:
            self.output.setState(False)
```

prob2.py

3 Problem 3

```
# Problem 3
# this component return the current state of the input wire
class Sensor(object):

  def __init__(self, input, output):
        self.input = input
        input.addDownStream(self)
        self.output = output
        self.outputs = []
        self.update()

    def update(self):
        self.output.setState(self.input.getState())
        self.outputs.append(self.input.getState())
```

prob3.py

4 Problem 6

```
# Problem 6
# Demonstrate the usages of classes created above
def example():

  # create 4 wires
    a = Wire()
    b = Wire()
  c = Wire()
    d = Wire()

  # create gates and assign wires according to the graph
    NotGate(a,b)
  AndGate(b,c,d)

  # turn on the current
    a.setState(True)
  c.setState(True)

  # watch the current state on every wires
  print(a.getState())
    print(b.getState())
  print(c.getState())
    print(d.getState())


example()
```

prob6.py