**Stevie Parris**

**Professor Popyack**

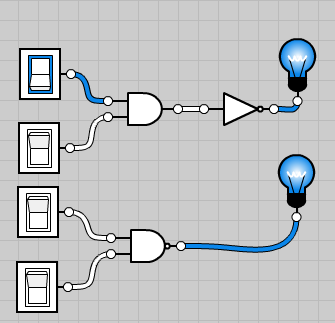
**CS 164**

**18 November 2013**

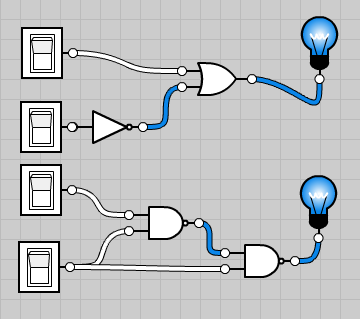
**HW 7:**

**Problems to be turned in**:

1. ***(6 points)*** Using only NAND gates, construct circuits that are equivalent to
   1. **NOT (x AND y)**



* 1. **x OR (NOT y)**



1. ***(6 points)*** Using logic tables, show that the following identities hold:
   1. **P(P + Q)= P**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Q** | **P + Q** | **P** | **P(P+Q)** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

* 1. **(PQ)' = P' + Q'**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P** | **Q** | **P’** | **Q’** | **PQ** | **(PQ)’** | **P’ + Q’** |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |

1. ***(9 points)*** Construct truth tables for the following logical expressions:
   1. **P' + Q**

|  |  |  |  |
| --- | --- | --- | --- |
| **P** | **Q** | **P’** | **P’ + Q** |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 |

* 1. **Q' + PQ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Q** | **Q’** | **PQ** | **Q’ + PQ** |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

* 1. **(PQ')'**

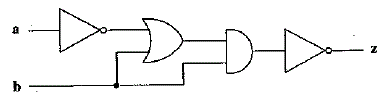
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Q** | **Q’** | **PQ’** | **(PQ’)’** |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

1. ***(10 points)*** Suppose you had three inputs, say **P**, **Q** and **R**. Write truth tables for the following logical expressions:
   1. **(P AND Q) OR R**

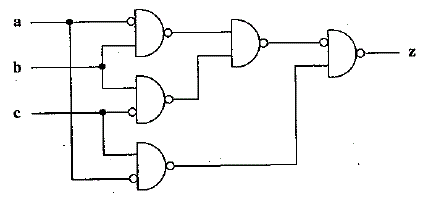
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Q** | **R** | **PQ** | **(PQ) + R** |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

* 1. **(P OR Q) AND R**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Q** | **R** | **P + Q** | **(P + Q)R** |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. ***(10 points)*** Fill in the logic table that describes the action of the following circuits and give the equivalent Boolean expressions for the output.
   1. 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **b** | **a’** | **a’ + b** | **(a’ + b)b** | **((a’ + b)b)’** |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |

* 1. .
* **(((ab)’(bc)’)’ (ac)’)’**

(Note that these are NAND gates, not AND gates.)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **ab** | **bc** | **ac** | **(ab)’** | **(bc)’** | **(ac)’** | **(ab)’**  **(bc)’** | **((ab)’**  **(bc)’)’** | **((ab)’(bc)’)’ (ac)’** | **(((ab)’(bc)’)’ (ac)’)’** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**Virtual Pests, part 2:**

USER INTERFACE:

The user interface is pretty straight forward. It uses four text boxes: pet sound, pet action, pet state, and a counter of sorts. It also has four buttons: Feed, Hit, Pat, and Click To Restart which are pretty self-explanatory.

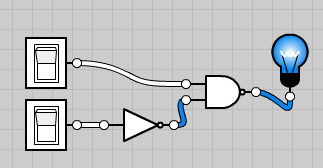
* " **initialize**" which is run automatically when the program is loaded, it lists all the attributes and assigns initial values. It is also used for resetting the page. It sets the state to "Bored", then calls "dostate".
* "**dostate**" which shows the value of the state, changes the value of "pet action" to "Chase Tail", "Wag Tail", “Claw”, “Beg”, or “Puppydog Eyes” as appropriate, changes the value of "pet sound" to "Bark", “Whine”, “Whimper”, or "Growl" as appropriate, and changes the pest's picture to go with its current state.
* "**ifhit**" which is run when the "PetHit" button is pressed, updates the state as appropriate, then calls "**running**" again using the "**setTimeout**" function.
* "**ifpat**" which is run when the "PetPat" button is pressed, updates the state as appropriate, then calls "**running**" again using the "**setTimeout**" function.
* "**iffeed**" which is run when the "PetFeed" button is pressed, updates the state as appropriate, then calls "**running**" again using the "**setTimeout**" function.
* "**running**" which examines the state, generates a random number, updates the state as appropriate, then calls "**running**" again using the "**setTimeout**" function.

<http://www.pages.drexel.edu/~spp53/cs164/Pheonix/PheonixTheDog.html>

EXTRA CREDIT:

***(5 points)***Use AND, OR and NOT gates to construct a circuit with two inputs, P and Q, and one output Z, that obey the following truth table:

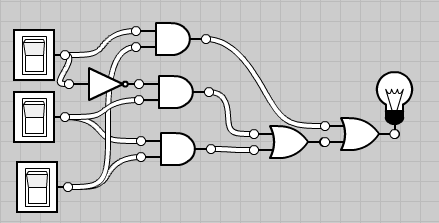
|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **Z** |
| **0** | **0** | **1** |
| **0** | **1** | **1** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |



(AB’)’

* ***(10 points)*** Use AND, OR and NOT gates to construct a circuit with three inputs, P, Q, and R, and one output Z, that obey the following truth table:

|  |  |  |  |
| --- | --- | --- | --- |
| **P** | **Q** | **R** | **Z** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** |



((ac) + ((a’b) + (bc)))