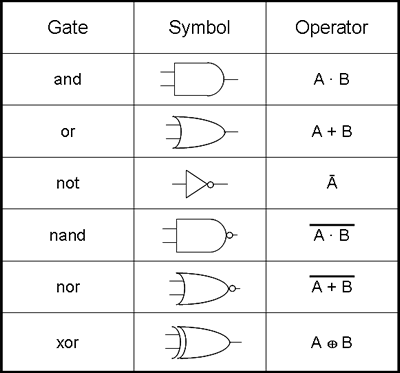
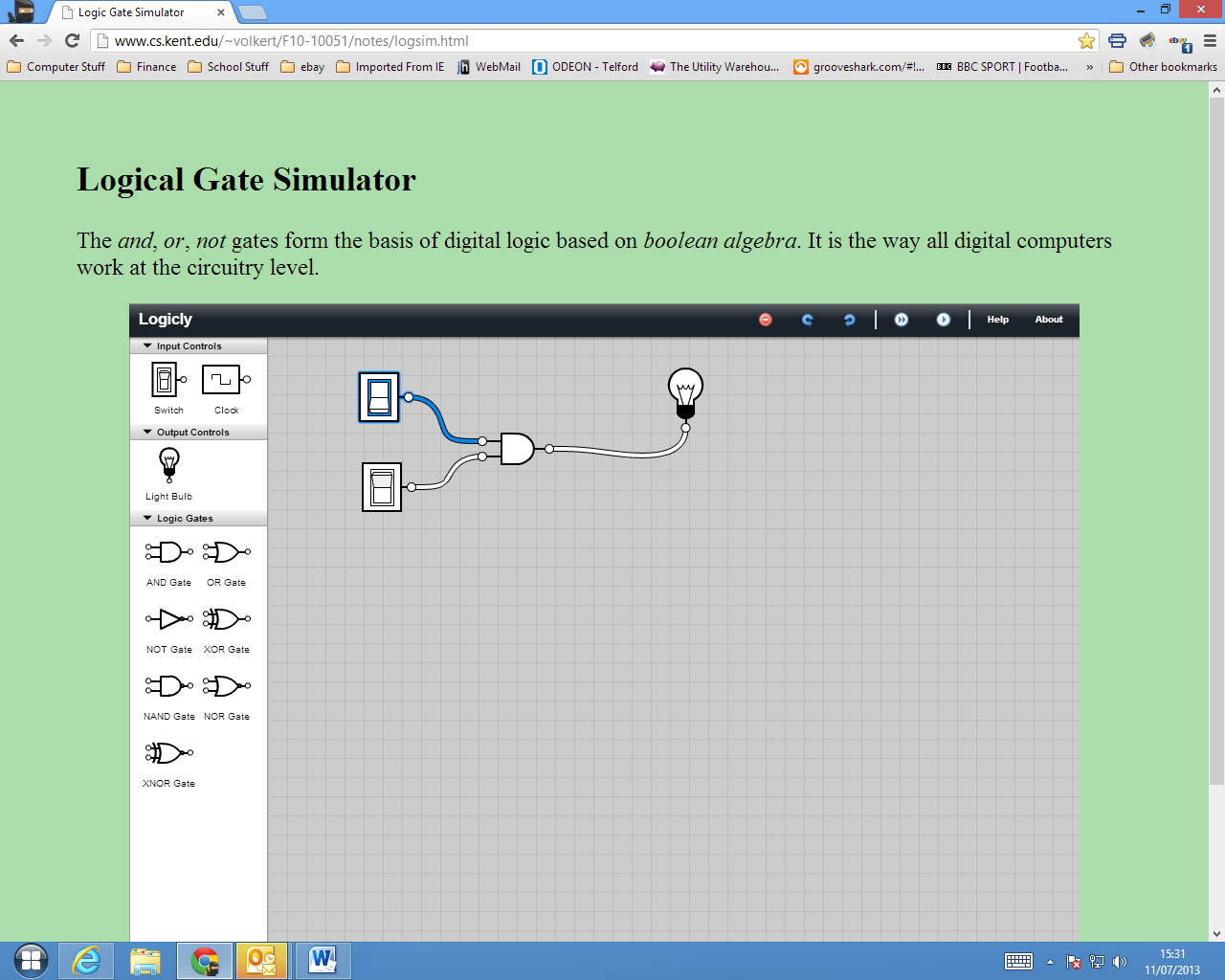
Logic Circuits

# Basics

1. Logon to the Logic Gate Simulator Website   
   <http://www.cs.kent.edu/~volkert/F10-10051/notes/logsim.html>

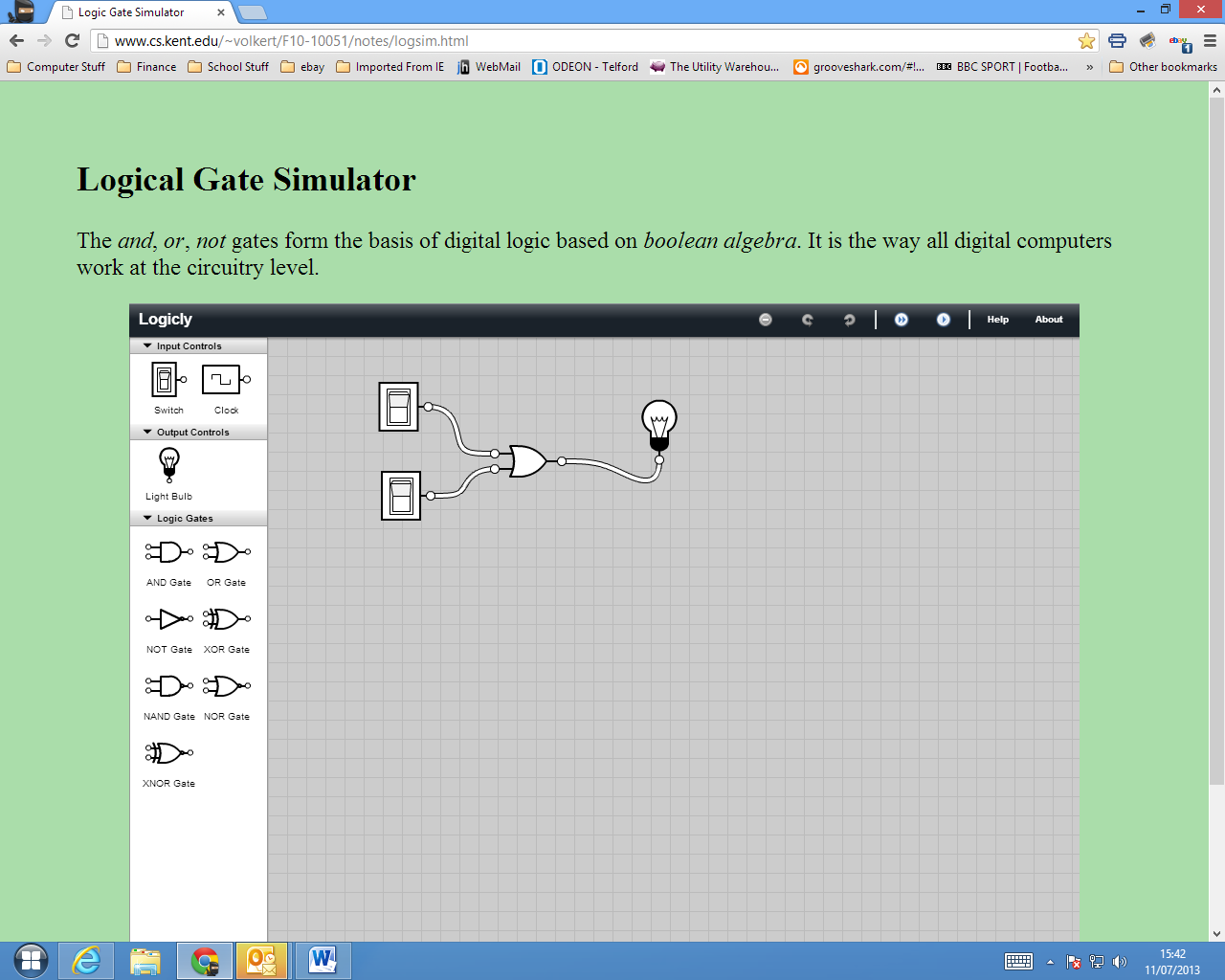
[](http://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&frm=1&source=images&cd=&cad=rja&docid=sX-y7kzWAttMnM&tbnid=wlmHz1qAnvSA4M:&ved=0CAUQjRw&url=http://www.ib-computing.com/html/program/topic_4/boolean.html&ei=yQngUcLUEtDu0gXzs4HwBQ&bvm=bv.48705608,d.d2k&psig=AFQjCNHQtkHyah6d0qsifwm7OJCVYHlFgg&ust=1373723451407374)

1. Create this Logic Circuit  
     
   

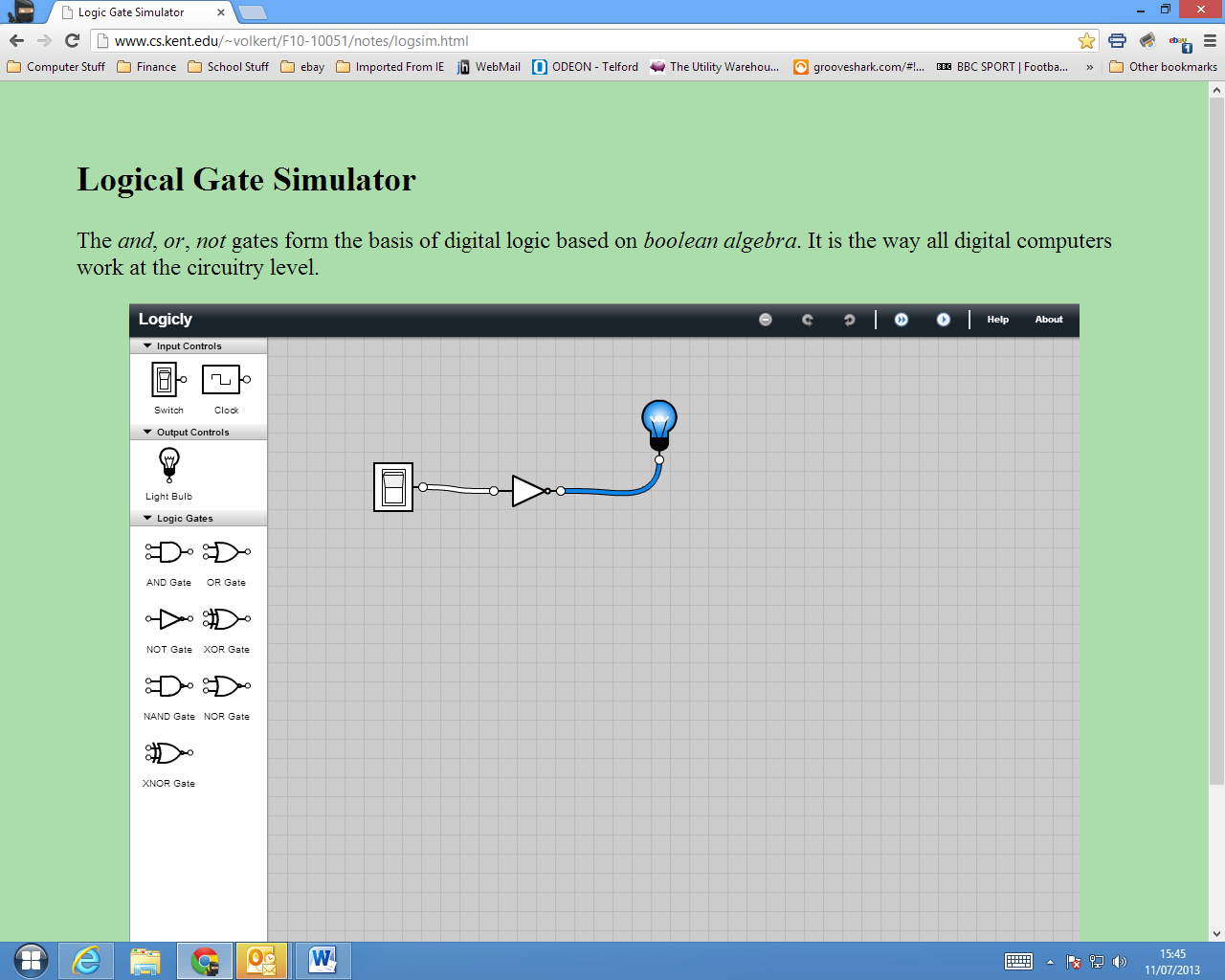
AND Gate. You get an output when switch one is on and switch two is on.

Truth Table.

|  |  |  |
| --- | --- | --- |
| X | Y | X AND Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

1. Work in a group of 2 or 3 (but no more than 3). Turn the switches on and off and see what happens. A clue is the name given to the shape in the software.  
   Write down your findings in words and devise a diagram to summarise your findings.
2. Now do the same for this circuit.  
     
      
   Or Gate. You get an output if one switch or the other is on. The light also lights if both switches are on.

|  |  |  |
| --- | --- | --- |
| X | Y | X OR Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1. And do the same for this one   
   

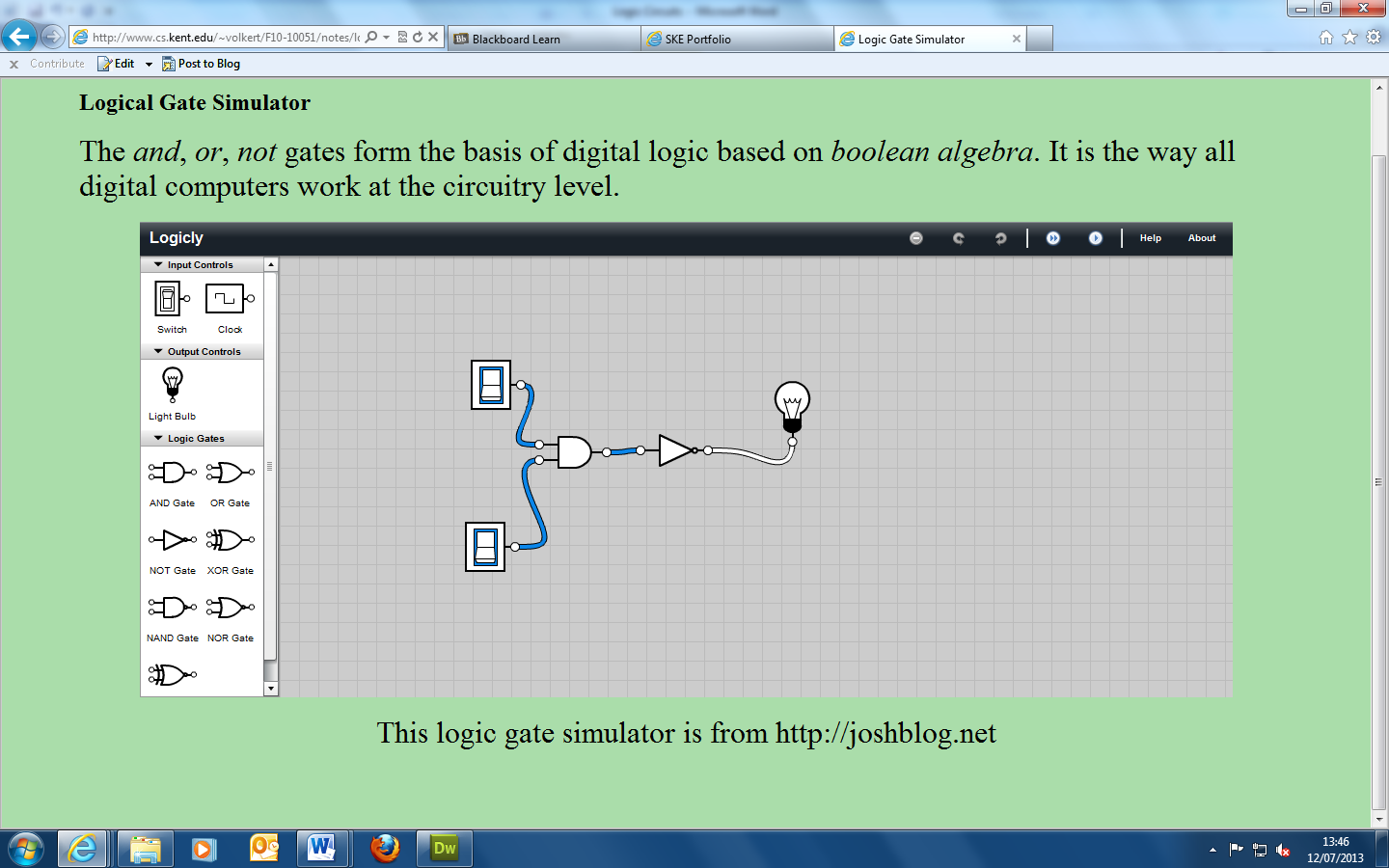
Not gate. The output is not the input.

|  |  |
| --- | --- |
| X | NOT X |
| 0 | 1 |
| 1 | 0 |

**Time for a group discussion!**

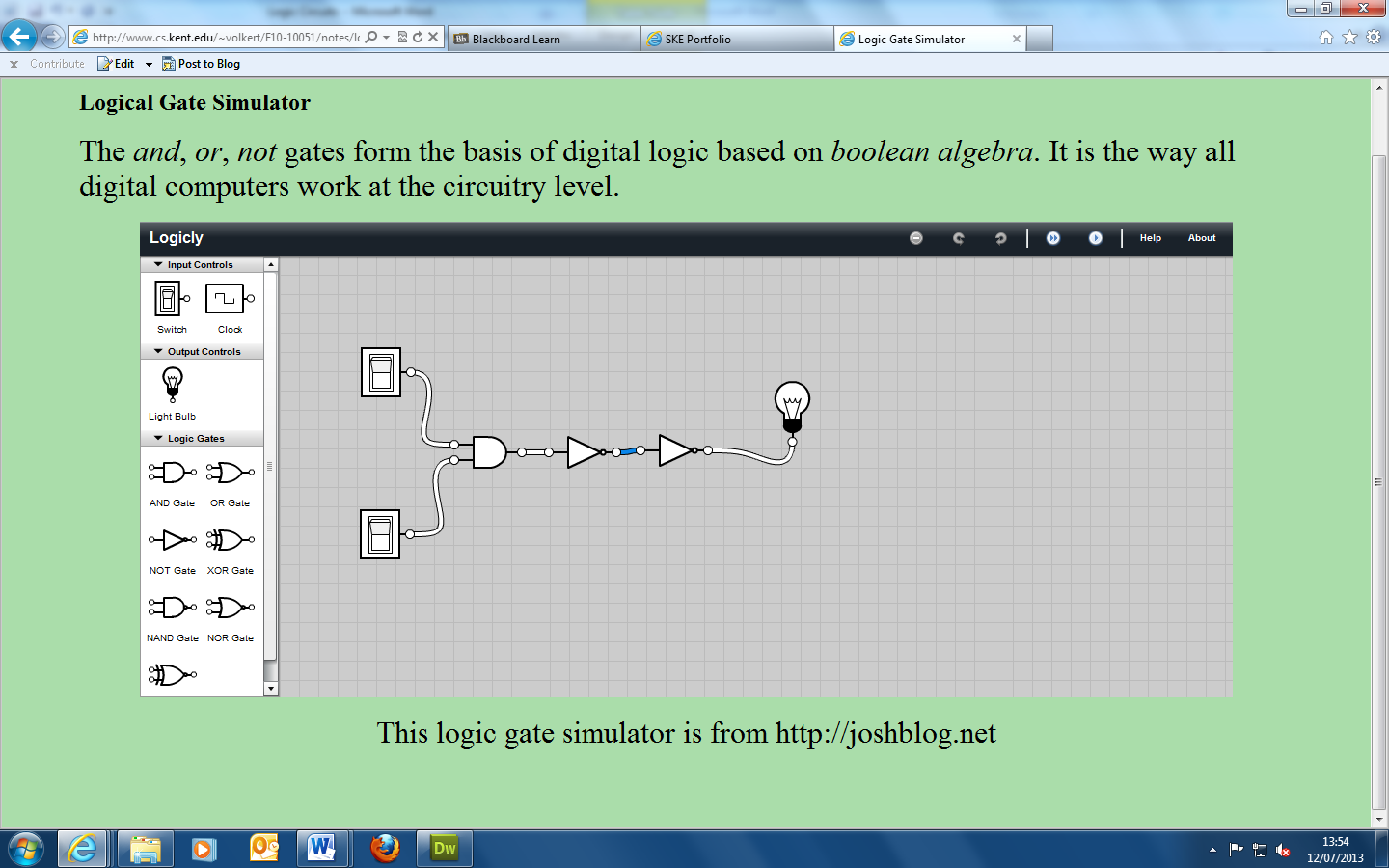
# Combining Gates

1. Now make Truth Tables for all of the other Logic Gates within the program
2. See if you can make these circuits
   1. **NOT (A AND B )**



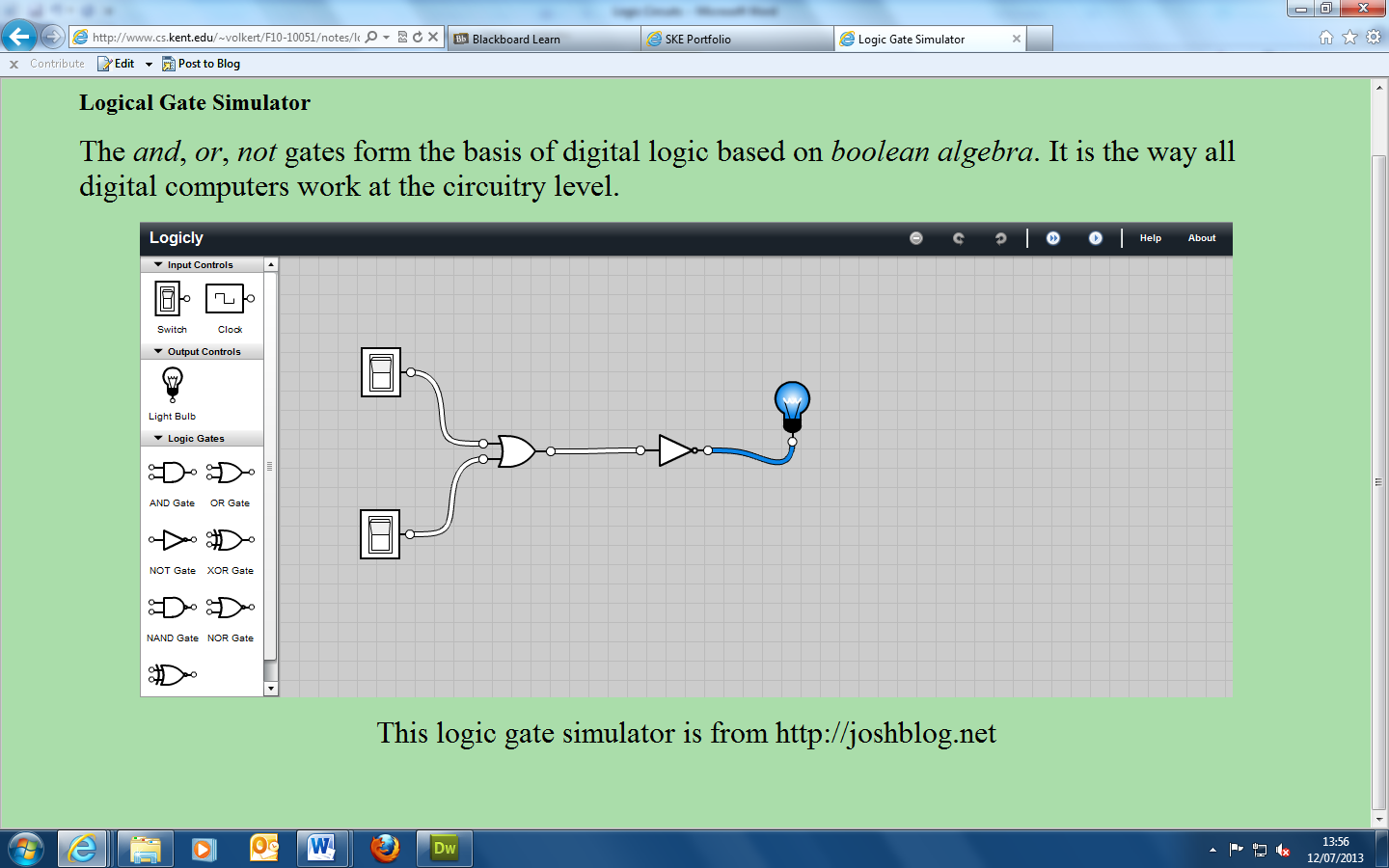
|  |  |  |
| --- | --- | --- |
| X | Y | NOT (X AND Y) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

* 1. **NOT ( NOT ( A AND B ))**



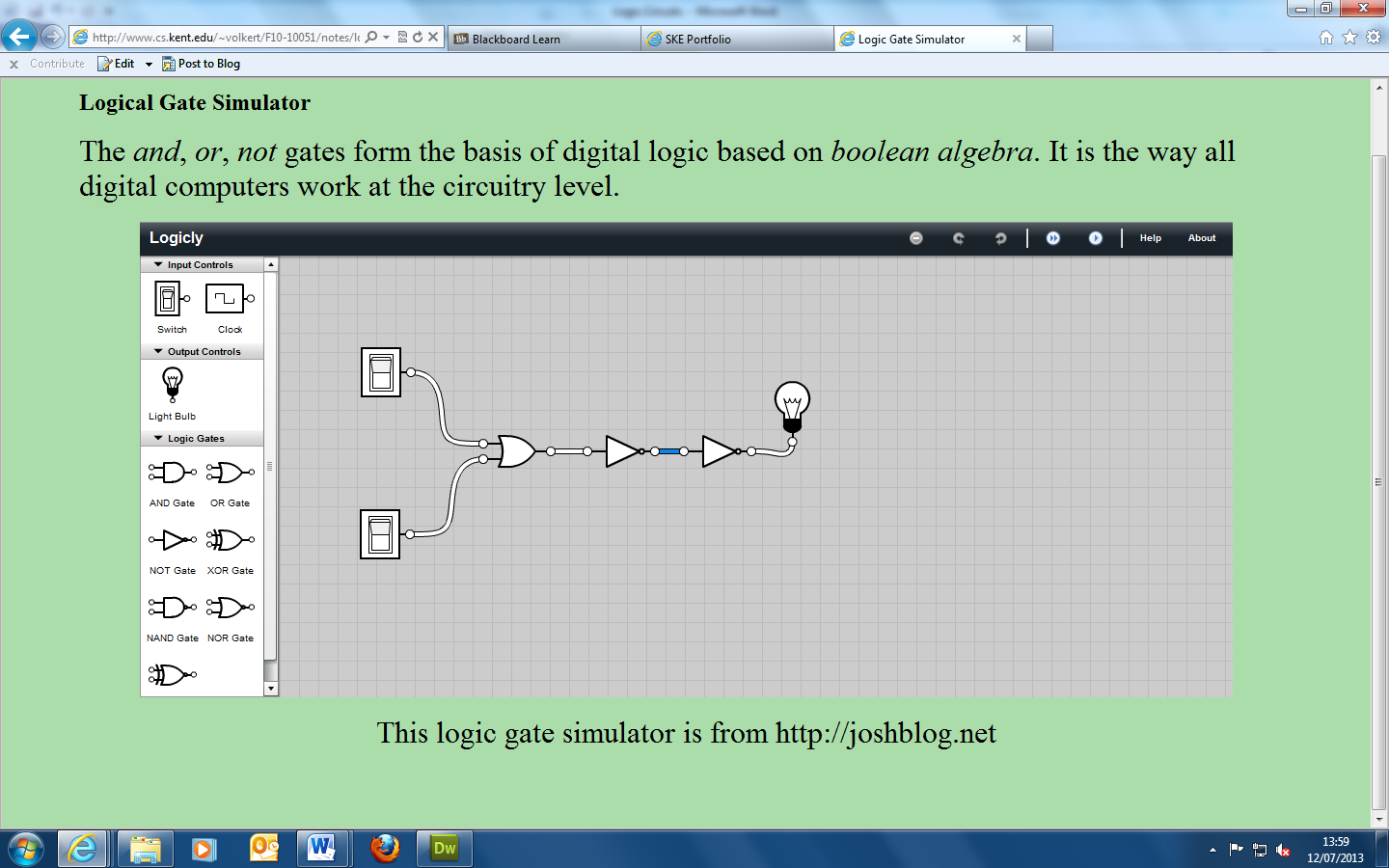
|  |  |  |
| --- | --- | --- |
| X | Y | NOT (NOT (X AND Y)) |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

* 1. **NOT ( A OR B )**



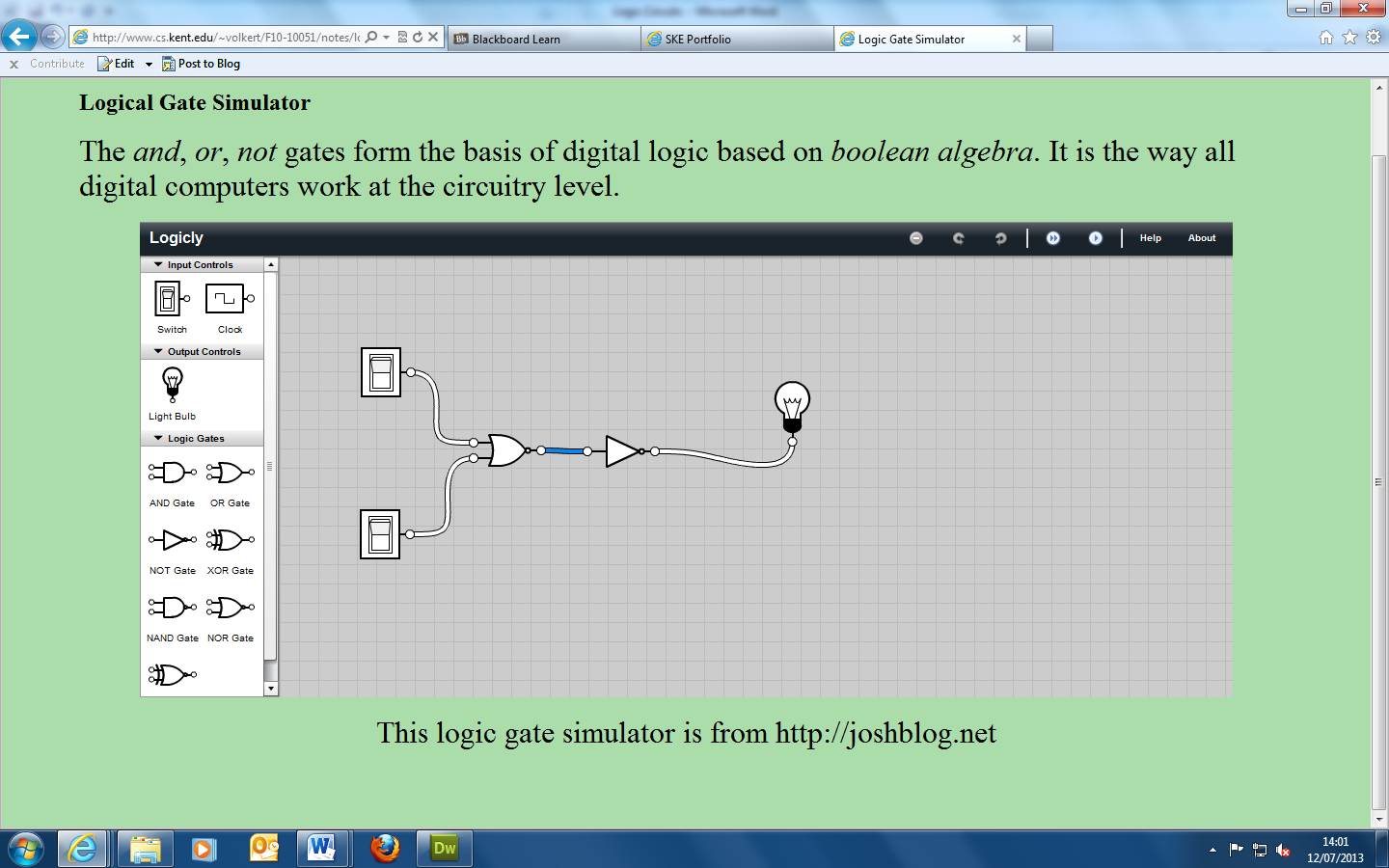
|  |  |  |
| --- | --- | --- |
| X | Y | NOT (X OR Y) |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

* 1. **NOT ( NOT ( A OR B ))**



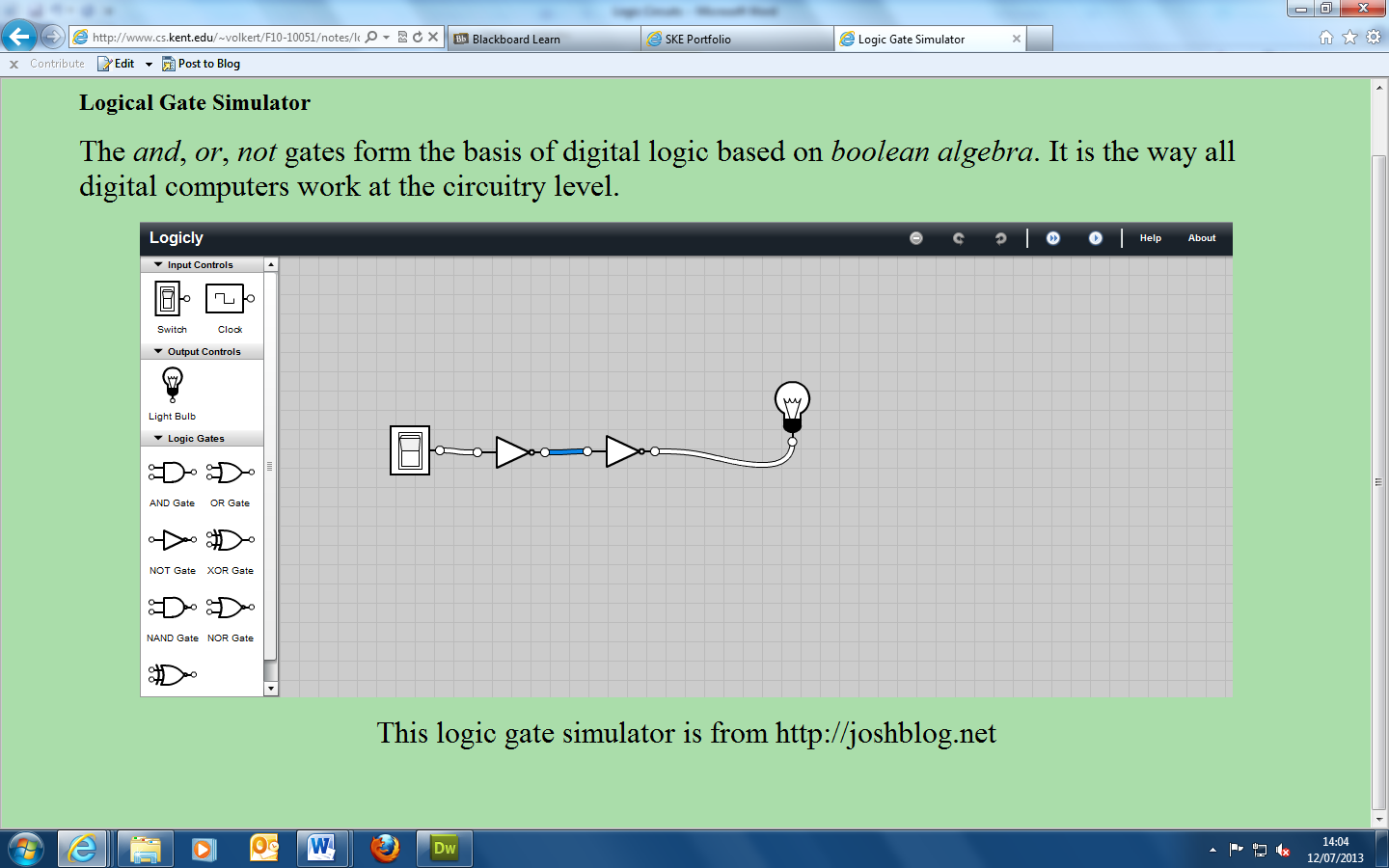
|  |  |  |
| --- | --- | --- |
| X | Y | NOT (NOT (X OR Y)) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

* 1. **NOT ( A NOR B )**

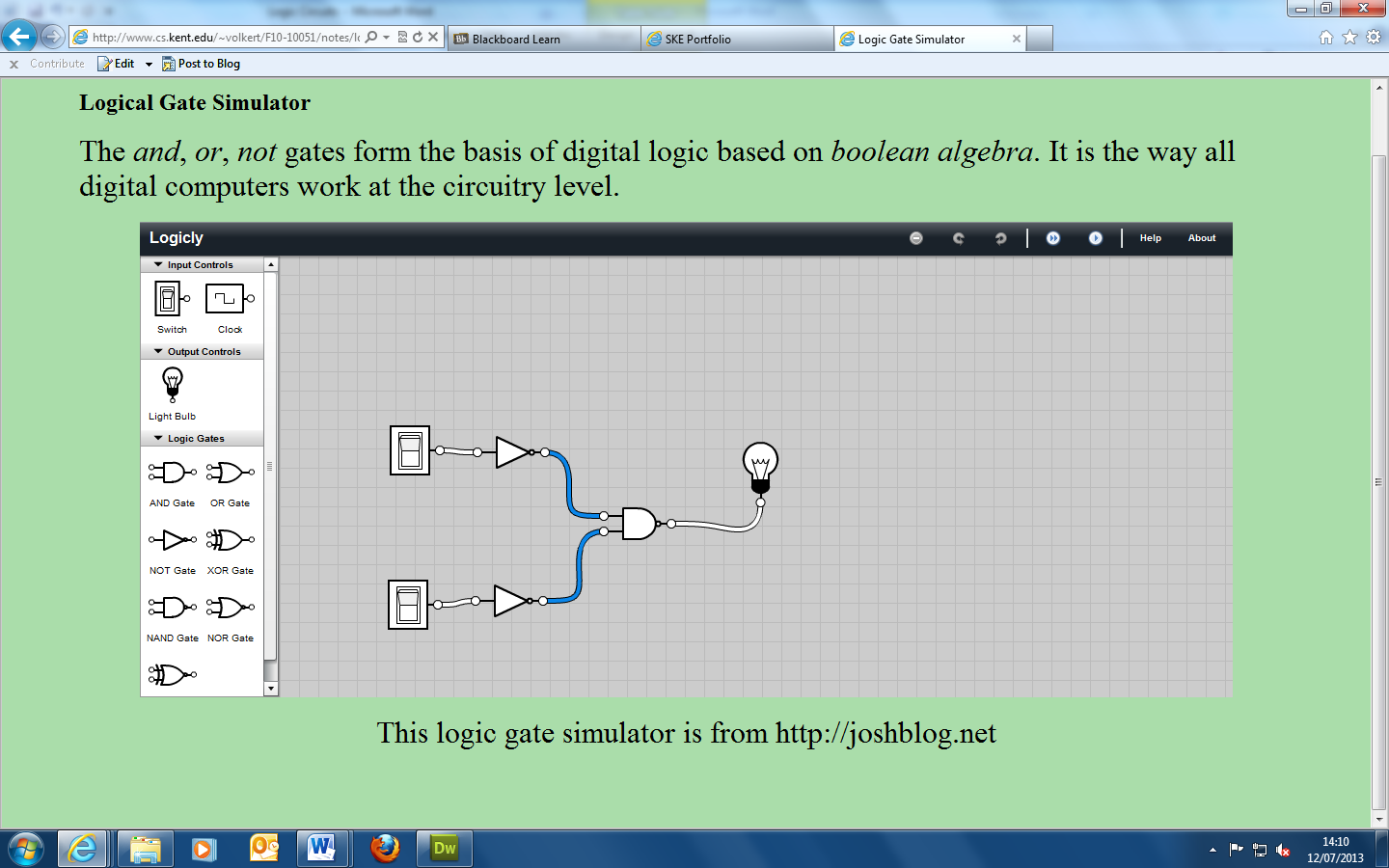


|  |  |  |
| --- | --- | --- |
| X | Y | NOT ( X NOR Y) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

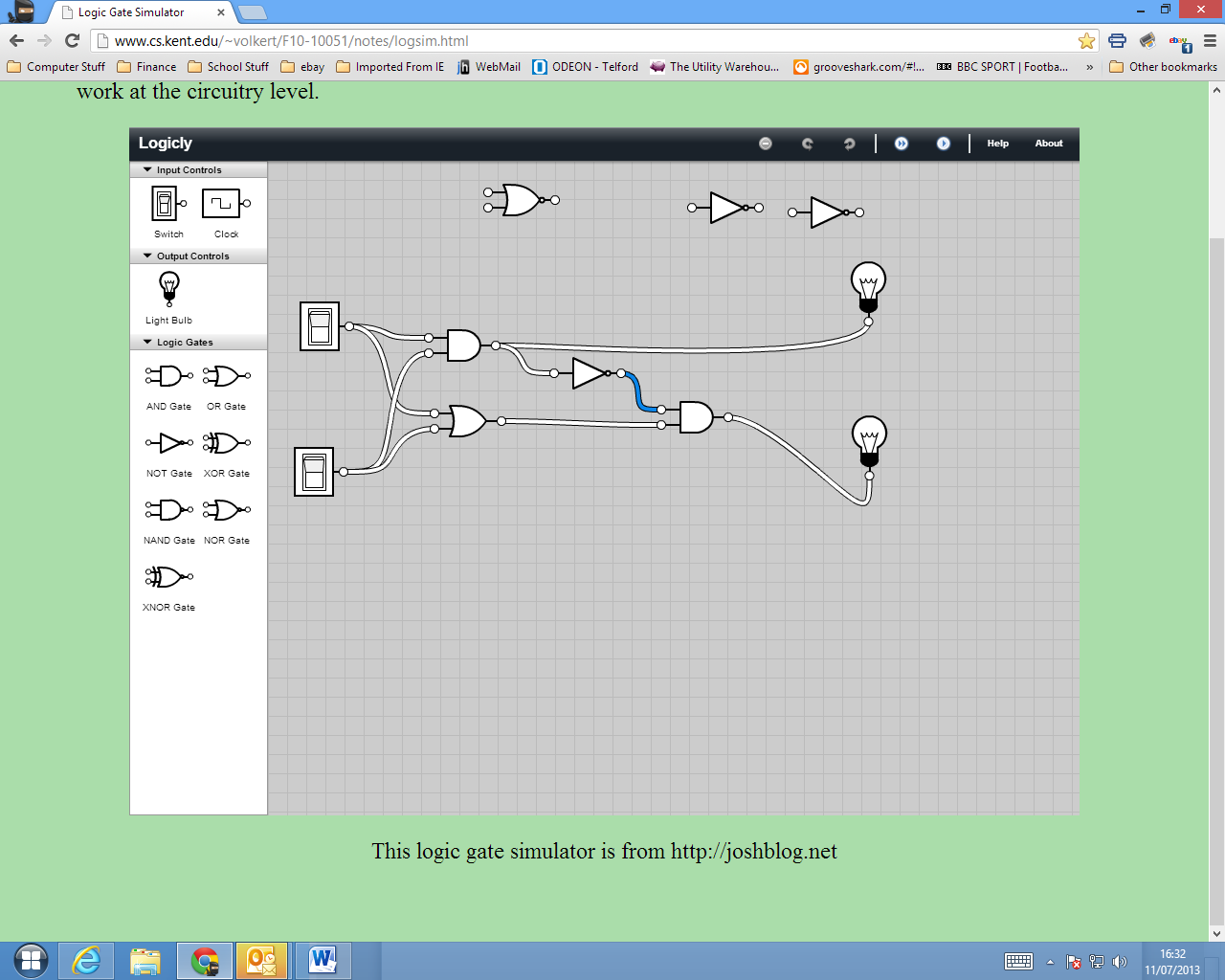
* 1. **NOT ( NOT A )**



|  |  |
| --- | --- |
| X | NOT ( NOT A ) |
| 0 | 0 |
| 1 | 1 |

* 1. **NOT A NAND NOT B**

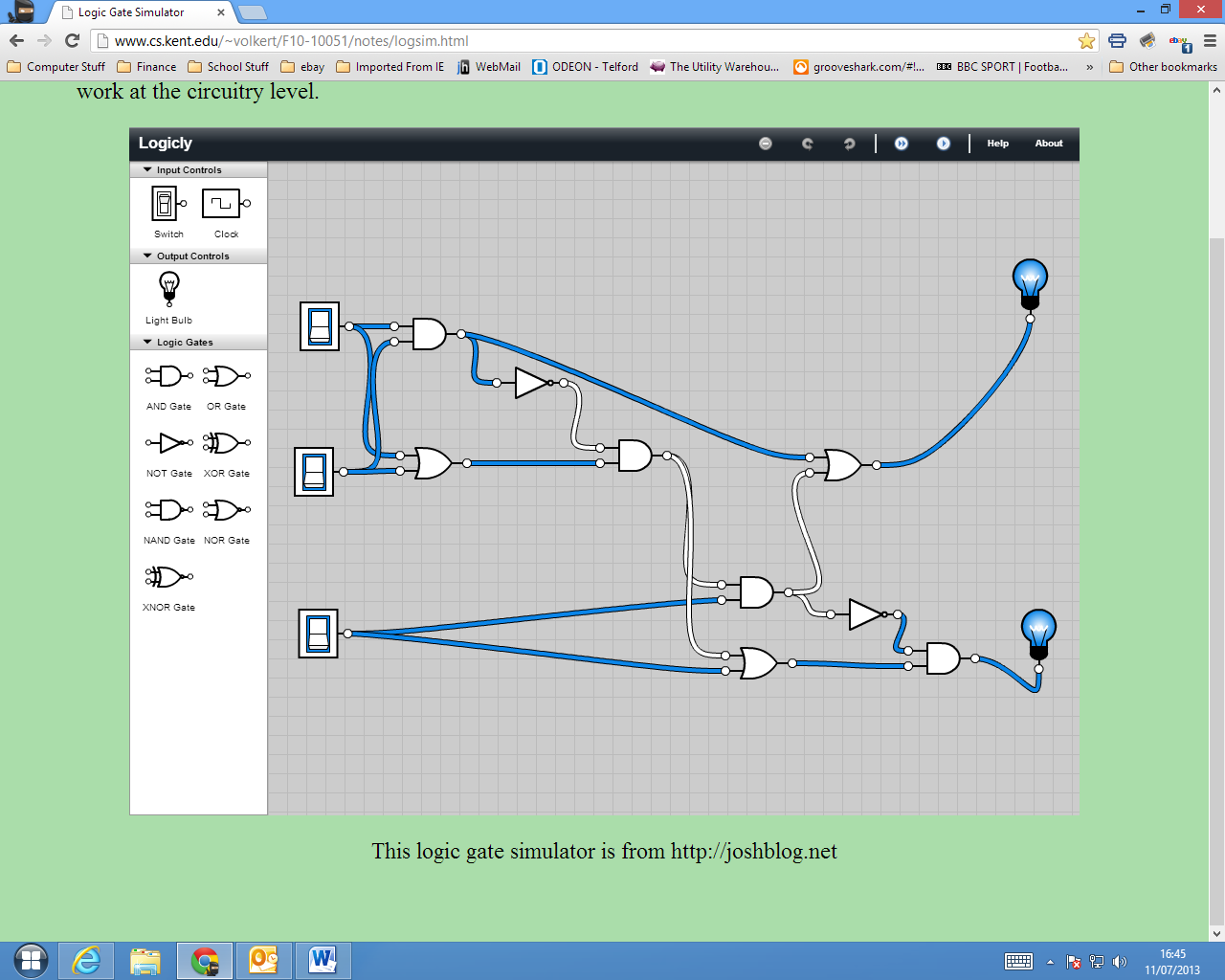
|  |  |  |
| --- | --- | --- |
| X | Y | NOT X NAND NOT Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

* 1. **This is a useful circuit. Can you think why?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | Y | X AND Y | NOT (X AND Y) | X OR Y | (NOT (X AND Y) AND (X OR Y) |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |

**This is a binary arithmetic processor. It is a half adder**

**With more switches this would be an accumulator**

* 1. **This is an extremely useful circuit!**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | Z | A | B |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

This is how an accumulator works. It can add any Binary numbers the first two switches deal with the two numbers and the third switch deals with the carried numbers in the case of 1 + 1+ 1