Introduction to Programming in Python

Building a Computer: Logic Circuits

Outline

Boolean Functions

Boolean Functions

A boolean variable is a variable that has the value 1 (True) or 0 (False)

Boolean Functions

A boolean variable is a variable that has the value 1 (True) or 0 (False)

A boolean function is an algebraic expression consisting of boolean variables and logical operations

Boolean Functions

A boolean variable is a variable that has the value 1 (True) or 0 (False)

A boolean function is an algebraic expression consisting of boolean variables and logical operations

The three basic boolean functions: $\texttt{not}(x) = \bar{x}$, $\texttt{or}(x, y) = x + y$, and $\texttt{and}(x, y) = x \cdot y$

The truth table for a boolean function is a listing of all possible input values, together with the output value

The truth table for a boolean function is a listing of all possible input values, together with the output value

Truth tables for `not`, `or`, and `and` functions

x	$\bar{x}$
0	1
1	0

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function
2. Delete all rows from the truth table where the value of the function is 0

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function
2. Delete all rows from the truth table where the value of the function is 0
3. For each remaining row, create a "minterm" as follows

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function
2. Delete all rows from the truth table where the value of the function is 0
3. For each remaining row, create a "minterm" as follows
    a. For each variable x: if its value in that row is 1, write x; otherwise, write $\bar{x}$

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function
2. Delete all rows from the truth table where the value of the function is 0
3. For each remaining row, create a "minterm" as follows
   a. For each variable x: if its value in that row is 1, write x; otherwise, write $\bar{x}$
   b. Combine all of the variables using $\cdot$

Any boolean function can be expressed in terms of the basic boolean functions using the Minterm Expansion Algorithm

The algorithm

1. Write down the truth table for the boolean function
2. Delete all rows from the truth table where the value of the function is 0
3. For each remaining row, create a "minterm" as follows
   a. For each variable x: if its value in that row is 1, write x; otherwise, write $\bar{x}$
   b. Combine all of the variables using $\cdot$
4. Combine all of the minterms using $+$

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function $(x \implies y)$

x	y	$x \implies y$	minterm
0	0	1	
0	1	1	
1	0	0	
1	1	1	

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function ($x \implies y$)

x	y	$x \implies y$	minterm
0	0	1	
0	1	1	
1	0	0	
1	1	1	

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function ($x \implies y$)

x	y	x $\implies$ y	minterm
0	0	1	$\bar{x} \cdot \bar{y}$
0	1	1	
1	0	0	
1	1	1	

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function ($x \implies y$)

x	y	x $\implies$ y	minterm
0	0	1	$\bar{x} \cdot \bar{y}$
0	1	1	$\bar{x} \cdot y$
1	0	0	
1	1	1	

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function ($x \implies y$)

x	y	x $\implies$ y	minterm
0	0	1	$\bar{x} \cdot \bar{y}$
0	1	1	$\bar{x} \cdot y$
1	0	0	
1	1	1	$x \cdot y$

Example: consider the proposition "if you score over 93% in this course, then you will get an A"

The proposition is described by the `implication` function ($x \implies y$)

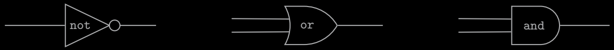
x	**y**	**x $\implies$ y**	**minterm**
0	0	1	$\bar{x} \cdot \bar{y}$
0	1	1	$\bar{x} \cdot y$
1	0	0	
1	1	1	$x \cdot y$

Therefore, $\texttt{implication}(x, y) = \bar{x} \cdot \bar{y} + \bar{x} \cdot y + x \cdot y$

Logic Circuits

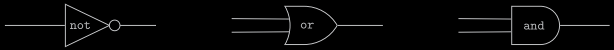
Logic Circuits

The circuits (called gates) that implement the `not`, `or`, and `and` functions

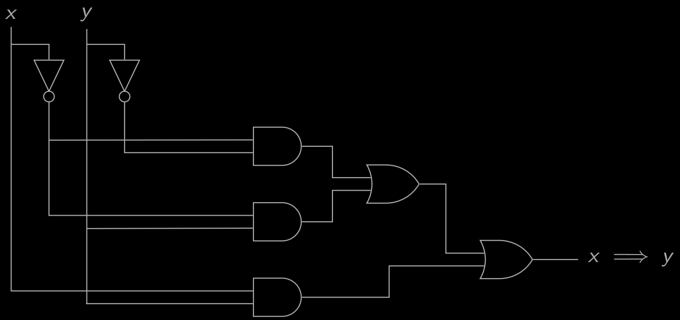


Logic Circuits

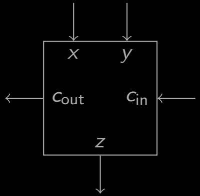
The circuits (called gates) that implement the `not`, `or`, and `and` functions



The circuit for the `implication` function $\bar{x} \cdot \bar{y} + \bar{x} \cdot y + x \cdot y$



A full adder (FA) circuit can add two 1-bit numbers (with carry) to produce a 2-bit result

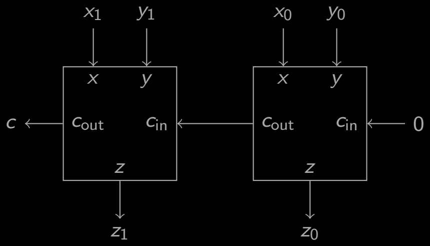


x	y	c_{in}	z	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

An n-bit ripple-carry adder for adding two n-bit numbers is n FA circuits chained together

An n-bit ripple-carry adder for adding two n-bit numbers is n FA circuits chained together

Example (2-bit ripple-carry adder for adding two 2-bit numbers)



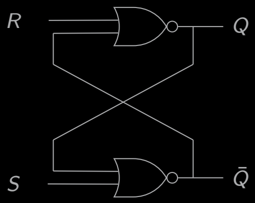
Truth table for a `nor` gate (`or` followed by `not`)

x	y	$\overline{x + y}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth table for a `nor` gate (`or` followed by `not`)

x	y	$\overline{x + y}$
0	0	1
0	1	0
1	0	0
1	1	0

A 1-bit memory circuit, called a latch, built using two `nor` gates

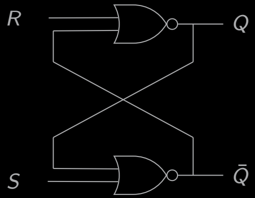


S	R	Q	$\bar{Q}$
0	0	0	1
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1

Truth table for a nor gate (or followed by not)

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

A 1-bit memory circuit, called a latch, built using two nor gates



S	R	Q	$\bar{Q}$
0	0	0	1
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1

Billion latches can be combined to produce a 1GB Random Access Memory (RAM) module