**Scenario: Controlling circuits using logic**

Many physical systems need to be controlled by electronic signals. If these signals are digital signals, it implies that there are two distinct states: a signal is either **off** or **on**. Physically, this usually means there is either a low voltage (0 Volts) on that line or a high voltage (sometimes 5 Volts, sometimes 2V, sometimes 1V). We can describe a digital signal as HIGH or LOW, or we can use ON or OFF, or we can use 1 or 0.

A simple digital control is shown here: picture a push-button switch which allows a door to be locked or unlocked. For this example, if the button is pressed and a connection is made to +5V (or HIGH or ON), the door is unlocked as the lock disengages and a light turns on:



In this case, if we push the button, we change the signal going to the light and the door lock from LOW to HIGH (or from 0 V to 5 V). We could describe this by saying

Light = ButtonPush and Unlock = ButtonPush

Most control systems are far more complex. Imagine the safety system of a self-driving car. One signal could be to the brake, which would tell the braking system to apply full pressure if a scanning system in front identifies a person or animal, if the driver falls asleep, if an oncoming vehicle is detected, or if someone yells “STOP!”. Each of these signals must be processed as well: for an indicator to say that a person or animal is in front of the car requires it to identify if something is in front, then determine if that shape is a person or animal. If a plastic bag blew in front of the car, we would not want to slam on the brakes, since the risk of an accident would not be worth avoiding the plastic bag. However, if a human ran in front of the car and we saw their shape from the front, side, or rear, we would want to stop. Complex circuitry like this can have a lot of inputs and a lot of logic circuitry. In this case, we might write a description of our stopping circuit as:

ApplyBrakes = (SomethingInFrontOfCar AND (Animal OR Human)) OR (Yell) OR (Driver AND Sleep)
OR (SomethingComing AND Vehicle)

This equation can be written as a Boolean equation. A Boolean equation is one where every variable is either 1 or 0 (or TRUE or FALSE, or ON or OFF). There are three basic building blocks for Boolean equations:

|  |  |  |
| --- | --- | --- |
| *function* | *symbol* | *description* |
| AND | (dot) ∙ | TRUE when A is TRUE **and** B is TRUE |
| OR | + | TRUE when A is TRUE **or** B is TRUE |
| NOT | Overhead line | TRUE if A is not TRUE |

Every possible outcome of these operators can be seen using a **truth table**. A truth table gives the output of an equation for every combination of inputs. Simple truth tables for AND, OR and NOT include:



These operators can be combined in a way very similar to algebra. A similar order of operations applies. You might recall PEMDAS from algebra:

* Parenthesis
* (Exponents – doesn’t apply in Boolean algebra)
* Multiplication (AND in our case)
* (Division – doesn’t apply in Boolean algebra)
* Addition (OR in our case)
* (Subtraction – doesn’t apply in Boolean algebra)

To build a truth table, we can apply every possible combination of inputs to the expression and record the output. As an example of building a truth table, consider the equation:

$$Warning=Intruder ∙ \overbar{Override}, or W= I∙\overbar{O}$$

|  |  |  |
| --- | --- | --- |
| *I* | *O* | $$I∙\overbar{O}$$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

In other words, the output (Warning) is TRUE when Intruder is TRUE and Override is FALSE. This should make sense: there should only be a warning if there is an intruder and no override has been set.

Note – sometimes, we omit the dot meaning AND and draw the variables together; in other words, we write: A ˑ B as AB.

As Boolean expressions grow more complex, there is a set of rules that can be used to simplify the expressions:

 

These rules should look similar to rules in algebra with one exception: treat an unbroken overline as though there are parenthesis around that quantity.

Let’s consider a more complex example:



Let’s use Boolean Algebra to simplify:



1. Use Boolean algebra to simplify $\left(X+ \overbar{Z}\right)(X+ \overbar{Y)}$
2. $X+ \overbar{Y}\overbar{Z}$
3. $XX+X\overbar{Y}+X\overbar{Z}+ \overbar{Y}\overbar{Z} $
4. $\left(X∙ \overbar{Z}\right)(X∙ \overbar{Y)}$
5. $X+Y+Z$
6. $\left(X+ \overbar{Y}\overbar{Z}\right)+(\overbar{XY}+ \overbar{XZ+Y})$
7. Write a truth table showing the correct output for the equation A·(A+B)?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A. | A | B |  | B. | A | B |  | C. | A | B |  | D. | A | B |  | E. | A | B |  |
|  | 0 | 0 | 0 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 0 |
|  | 0 | 1 | 0 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |
|  | 1 | 0 | 1 |  | 1 | 0 | 0 |  | 1 | 0 | 0 |  | 1 | 0 | 1 |  | 1 | 0 | 1 |
|  | 1 | 1 | 1 |  | 1 | 1 | 0 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |

1. Simplify this equation using the rules of Boolean algebra: 
2. $\overbar{AB+B}$
3. $\overbar{A+ \overbar{A}B+ \overbar{A}\overbar{B}}$
4. $\overbar{A}∙B$
5. $B$
6. $\overbar{A+\overbar{AB}+ \overbar{A}B}$
7. Write a truth table showing the correct output for this equation? 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A. | A | B |  | B. | A | B |  | C. | A | B |  | D. | A | B |  | E. | A | B |  |
|  | 0 | 0 | 0 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 0 |
|  | 0 | 1 | 0 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 0 |
|  | 1 | 0 | 1 |  | 1 | 0 | 0 |  | 1 | 0 | 0 |  | 1 | 0 | 1 |  | 1 | 0 | 1 |
|  | 1 | 1 | 1 |  | 1 | 1 | 0 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |  | 1 | 1 | 0 |

1. Simplify this equation using the rules of Boolean algebra: $\overbar{AB+AC}$ + $\overbar{A}\overbar{B}C$
2. $\overbar{AB+B}$
3. $\overbar{A}+ \overbar{B}\overbar{C}$
4. $AB+ \overbar{A+BC}$
5. $\overbar{A+ \overbar{B+ \overbar{BC}}}$
6. $B$
7. A control circuit is designed for a hotel in the event of a natural disaster. The hotel has a generator (G) which should only be used if power from the city (C) is unavailable. If the generator is being used, all power to the pool should be turned off (P) since the pool uses a lot of energy. If the generator is on, emergency lighting is turned on (E); otherwise, the emergency lighting is off. Note that the emergency lighting can be turned on manually with a switch (El). If the emergency lighting is turned on, the alarm must also be turned on (A). Which set of equations accurately describes control circuitry for this situation?
8. $P=E\overbar{C} E= \overbar{P+ \overbar{El}} A= \overbar{E}P$
9. $P=E\overbar{C}+C E= \overbar{P+ \overbar{\overbar{P}∙El}} A= \overbar{C}P$
10. $P=\overbar{G}E\overbar{C} E= \overbar{P+E+ \overbar{El}} A= P$
11. $P=\overbar{G} E=El+G A=E$
12. $P=E\overbar{C} E= \overbar{E+ \overbar{El}} A= E$

32. A warning system is designed for a city located near the coast. The warning (W) is to be issued if there is a thunderstorm warning (T) and a swirl in the clouds is detected (S), since this could signify a tornado. It is also issued if there is a hurricane (H) predicted to hit the coast, or if an offshore earthquake (E) happens and a tidal wave (ω) is predicted. Otherwise, the warning must never be issued. Which of these expressions represents this scenario?

1. W = (T∙S+H) + (ω∙E)
2. W = T + S + H + E + ω
3. W = H + (T∙S) + (E ∙ω)
4. W = (H+ω) +(T∙E∙S)
5. W = (T+S) ∙H ∙ (ω+E)

33. Write a truth table showing the correct output for this equation:
TornadoWatch = (Lightning ∙ Wind).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A. | L | W | TW | B. | L | W | TW | C. | L | W | TW | D. | L | W | TW | E. | L | W | TW |
|  | 0 | 0 | 0 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 1 |  | 0 | 0 | 0 |
|  | 0 | 1 | 0 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 1 |  | 0 | 1 | 0 |
|  | 1 | 0 | 1 |  | 1 | 0 | 0 |  | 1 | 0 | 0 |  | 1 | 0 | 1 |  | 1 | 0 | 0 |
|  | 1 | 1 | 1 |  | 1 | 1 | 0 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |