The Building Blocks: Binary Numbers, Arithmetic, Boolean Logic and Gates

Topics:

Boolean Logic
Gates and Circuits

Prof. Dr. Slim Abdennadher

24.1.2008

Important Announcement

- **Good news: Today is your last lecture :)**
- **Bad news: It has no tutorial, so wake up**
- **Believe it or not, I will miss you after this term is over !!!**
- **Next week is a revision lecture, show up if you are interested (you should be!!)**

Expressions and Circuits

- A **circuit** is a network of gates that implements one or more boolean functions.
- We can build a circuit for any Boolean expression by **connecting primitive logic gates** in the correct order.
- Notice that the order of operations is explicit in the circuit.

Primitive Logic Gates

- A **gate** is an electronic device that operates on a collection of binary inputs to produce a binary output.
- Each basic operation can be implemented in hardware with a logic gate.

Operation:	AND (product) of two inputs	Or (sum of) two inputs	NOT (complement) of one input
Expression:	XY or $X * Y$	$X + Y$	X' or $\bar{X}$ or $-X$

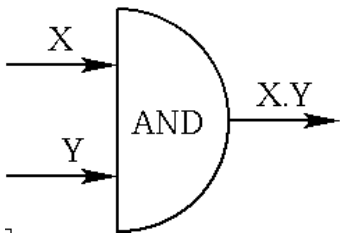
Truth Table:

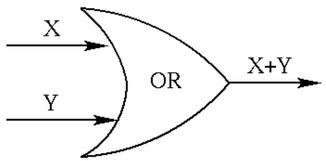
X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

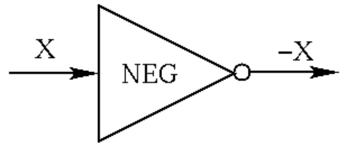
X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

X	X'
0	1
1	0

Logic Gate:



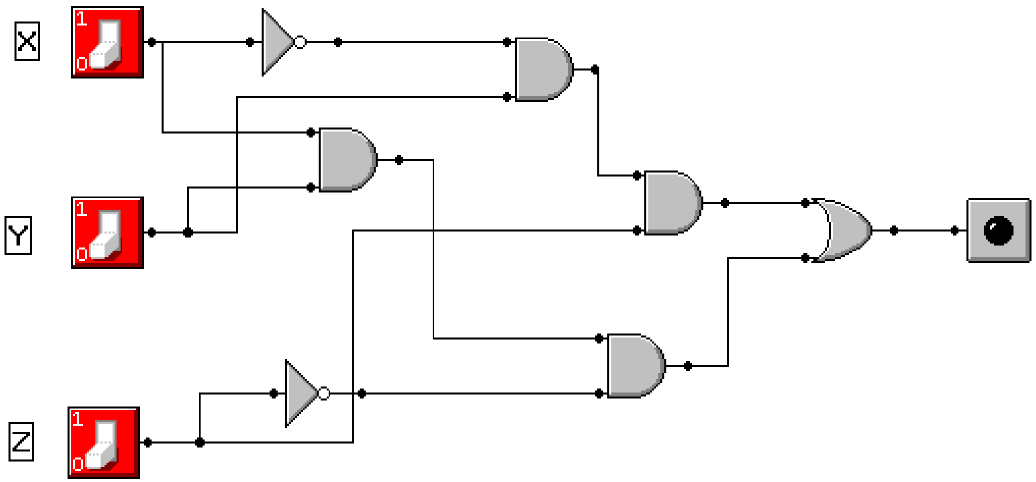




Expressions and Circuits – Example

- **Truth table:** **Circuit:**

X	Y	Z	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



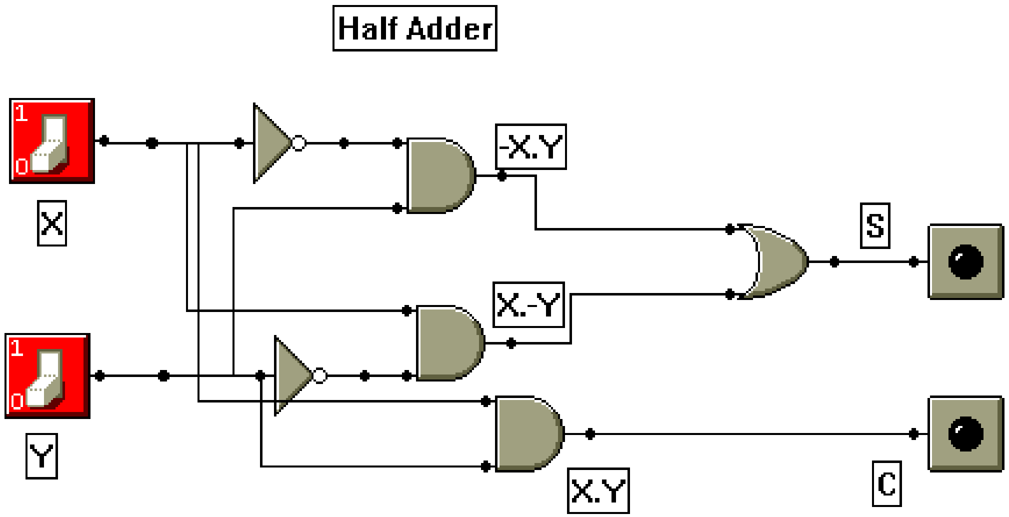
- **Boolean Expressions:**

$$S = X' * Y * Z + X * Y * Z'$$

Functionality of Circuits – Example

- **Task:** Adding two numbers consisting of one digit each.
- **Truth table:** Two inputs X and Y and two outputs S (Sum) and C (Carry)

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- **Boolean Expressions:**

  $S = X' * Y + X * Y'$

  $C = X * Y$

Equivalence Proof with Truth Tables

- Two expressions are **equivalent** by showing that they always produce the same results for the same inputs
- **Example:** $(x + y)' = x'y'$

x	y	$x + y$	$(x + y)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

x	y	x'	y'	$x'y'$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Simplifying Circuits

- Simpler hardware is almost always better.
  - In many cases, simpler circuits are faster.
  - Less hardware means lower costs.
  - A smaller circuit also consumes less power.
- So, how could circuits be simplified?

Boolean Algebra

- The secret is Boolean Algebra which let's us simplify Boolean functions just as regular algebra allows us to manipulate arithmetic functions.
- A **Boolean Algebra** requires
  - A set of values with at least two elements, denoted 0 and 1
  - Two binary operations $+$ and $*$
  - A unary operation $'$
- These values and operations must satisfy the axioms shown below.

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

Satisfying the Axioms

- The AND, OR and NOT operations do satisfy all of the axioms.

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

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

x	x'
0	1
1	0

- For example, the axiom $x + x' = 1$ always holds.
  - There are only two possible values for x, 0 or 1.
  - The complement of these values is 1 and 0 by definition of NOT.
  - According to the definition of OR, $0 + 1 = 1$ and $1 + 0 = 1$

x	x'	$x + x'$
0	1	1
1	0	1

Similarities with Regular Algebras

- The axioms in blue look just like regular algebraic rules.
- Associative laws show that there is no ambiguity in an expression like xyz or $x + y + z$. So, multi-input primitive gates can be used.

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

The Complement Operation

- The magenta axioms deal with the complement operation.
- First three axioms: Think about English examples
  - "It is snowing or it is not snowing" is always true ($x + x' = 1$).
  - "It is snowing and it is not snowing" can never be true ($x * x' = 0$).
  - "I am not not handsome" means that "I am handsome" ($(x')' = x$).

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

The Complement Operation

- The magenta axioms deal with the complement operation.
- DeMorgan's laws explain how to complement arbitrary expressions.
  - "I am not rich-or-famous" means that "I am not rich and I am not famous"
  - "I am not old-and-bald" means that "I am not old or I am not bald". But I could be (1) young and bald, or (2) young and hairy or (3) old and hairy.

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

Simplification of Boolean Expressions – Example (I)

$abc + abc' + a'b$

$$
\begin{aligned}
&= ab(c + c') + a'b \quad \text{(Distributivity)} \\
&= ab * 1 + a'b \quad [x + x' = 1] \\
&= ab + a'b \quad [x * 1 = x] \\
&= ba + ba' \quad \text{(Commutativity)} \\
&= b(a + a') \quad \text{(Distributivity)} \\
&= b * 1 \quad [x + x' = 1] \\
&= b \quad [x * 1 = x]
\end{aligned}
$$

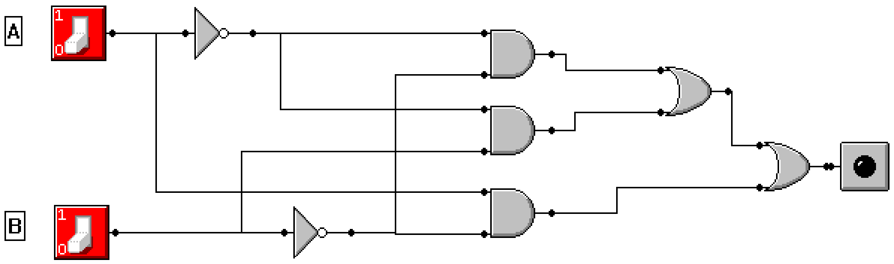
Simplification of Boolean Expressions – Example(II)

$$
\begin{aligned}
& x'y' + xyz + x'y \\
&= x'y' + x'y + xyz \quad \text{(Commutativity)} \\
&= x'(y' + y) + xyz \quad \text{(Distributivity)} \\
&= x'(y + y') + xyz \quad \text{(Commutativity)} \\
&= (x' * 1) + xyz \quad [y + y' = 1] \\
&= x' + xyz \quad [x' * 1 = x'] \\
&= (x' + x)(x' + yz) \quad [\text{Distributivity}] \\
&= (x + x')(x' + yz) \quad [\text{Commutativity}] \\
&= 1 * (x' + yz) \quad [x' + x = 1] \\
&= (x' + yz) * 1 \quad [\text{Commutativity}] \\
&= (x' + yz) \quad [x' * 1 = x']
\end{aligned}
$$

- **Truth Table**:

A	B	P
0	0	1
0	1	1
1	0	1
1	1	0

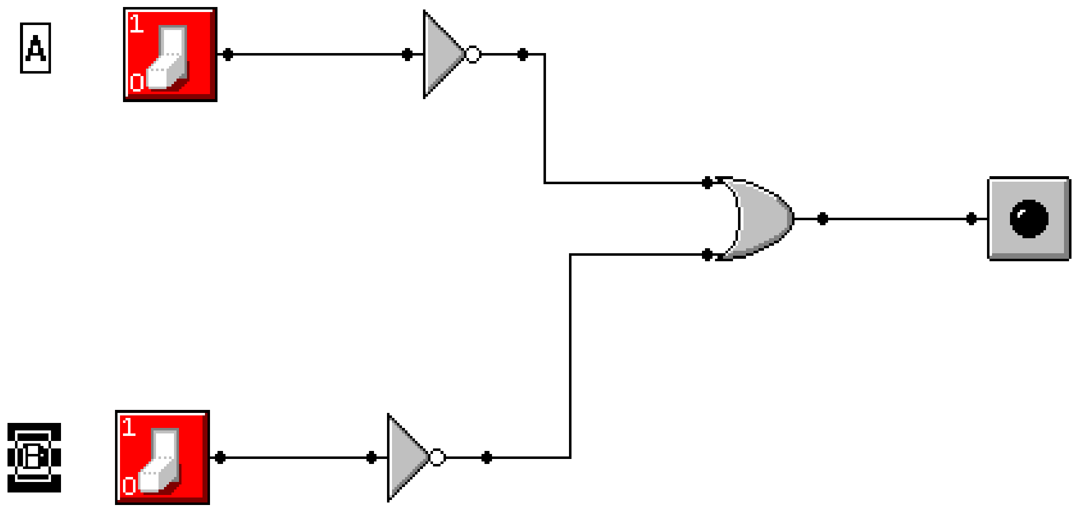
- **Boolean Expression:** $P = A' * B' + A'B + A * B'$
- Three AND gates and two OR gates and two NOT gates: **7 gates**
- **Logical Circuit:**



Simplification of Circuits

$$
\begin{aligned}
A' * B' + A'B + A * B' & \\
&= A'(B' + B) + A * B' \quad (Distributivity) \\
&= A' * (B + B') + A * B' \quad (Commutativity) \\
&= A' * 1 + A * B' \quad [x + x' = 1 \\
&= A' + AB' \quad [x * 1 = x] \\
&= (A' + A)(A' + B') \quad (distributivity) \\
&= (A + A')(A' + B') \quad (commutativity) \\
&= 1 * (A' + B') \quad [x + x' = 1] \\
&= (A' + B') * 1 \quad (commutativity) \\
&= A' + B' \quad [x * 1 = x]
\end{aligned}
$$

- $P = A' * B' + A'B + A * B' = A' + B'$
- **Number of gates:** Instead of 7 gates only 3 gates
- **Simplified Circuit:**



Transistors versus Gates

- Computers can be built out of any bistable device.
- Today's "microprocessor revolution" was made possible by the (1956 Nobel prize-winning) invention of the transistor.
- A **transistor** is an electronic "switch" that can be set to either "on" (representing 1) or "off" (representing 0).
- Today, transistors are extremely small 1cm^2 (and even smaller when etched on a chip) and fast (switch speed of nanoseconds).
- Gates can be built using transistors. Thus, we can design higher-level constructs using gates, without thinking about transistors anymore.
- **Gates are a useful abstraction.**

Summary

- Boolean Algebra invented by George Boole way back in the 1850s.
- Claude Shanon got the idea to apply Boolean Algebra to circuit design.
  - Boolean expressions are expressions that evaluate to either true or false
  - Can use the operators AND, OR, and NOT
- Learned about gates and circuits
- Learned how to design circuits from truth tables
- Learned how to simplify expressions (circuits)
- Learned how to prove the equivalence of two expressions (circuits)