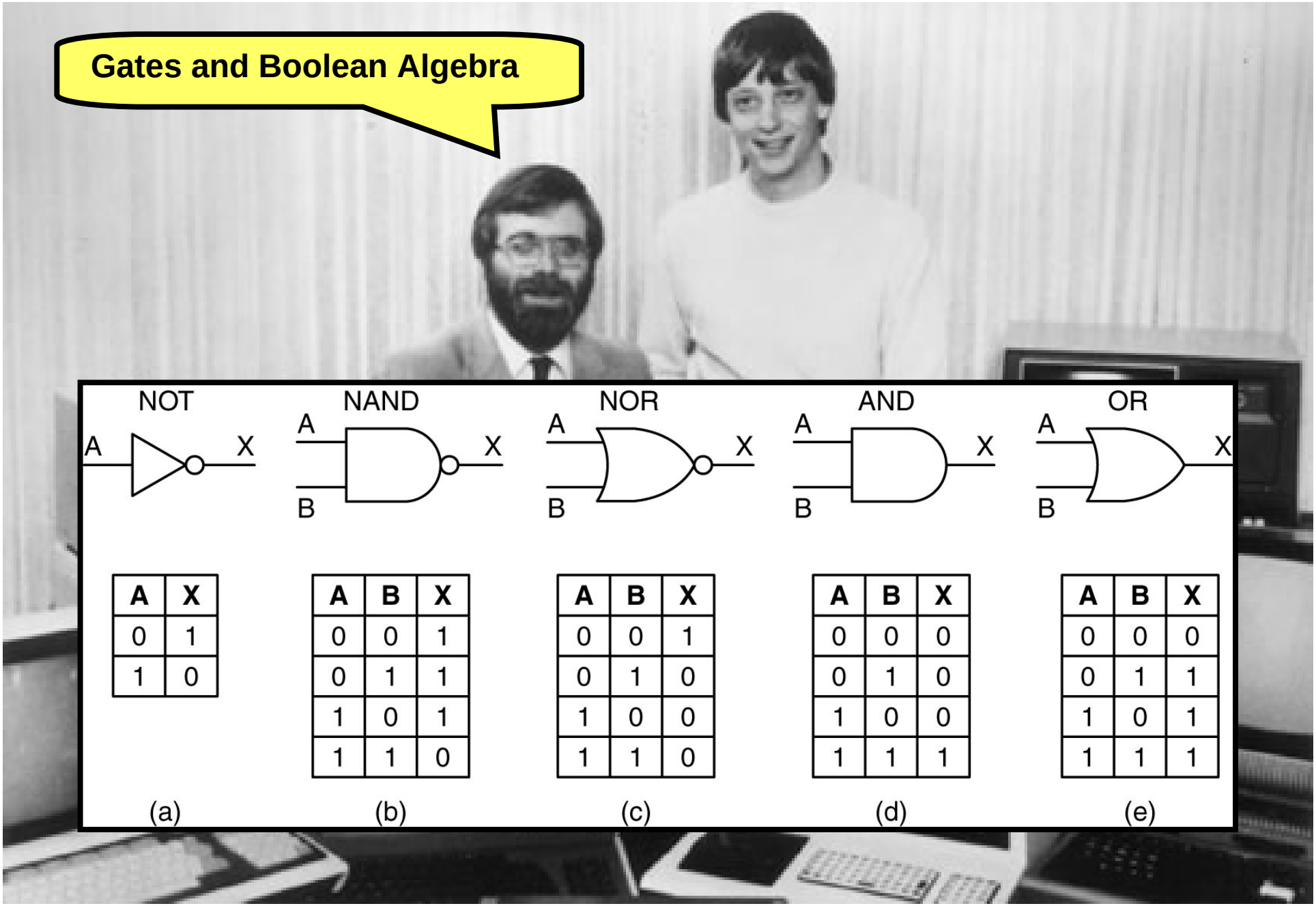
Digitalteknik & Datorarkitektur våren 2009

Räkna med grindar

Föreläsning 2 måndag 23 mars

karl.marklund@it.uu.se

Gates and Boolean Algebra



NOT

A	X
0	1
1	0

(a)

NAND

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

NOR

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Some logical functions for two variables.

x	y	x AND y	x OR y	x XOR y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

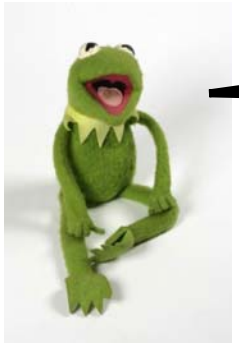
Hur många olika sådana funktioner finns det för två variabler?

Some logical functions for two variables.

x	y	x AND y	x OR y	x XOR y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Varje funktion definieras entydigt av sin kolumn i tabellen.. Varje kolumn utgörs av ett fyra bitars binärt tal...

Hur många olika fyra bitars tal finns det?



Det finns $2^4 = 16$ olika!

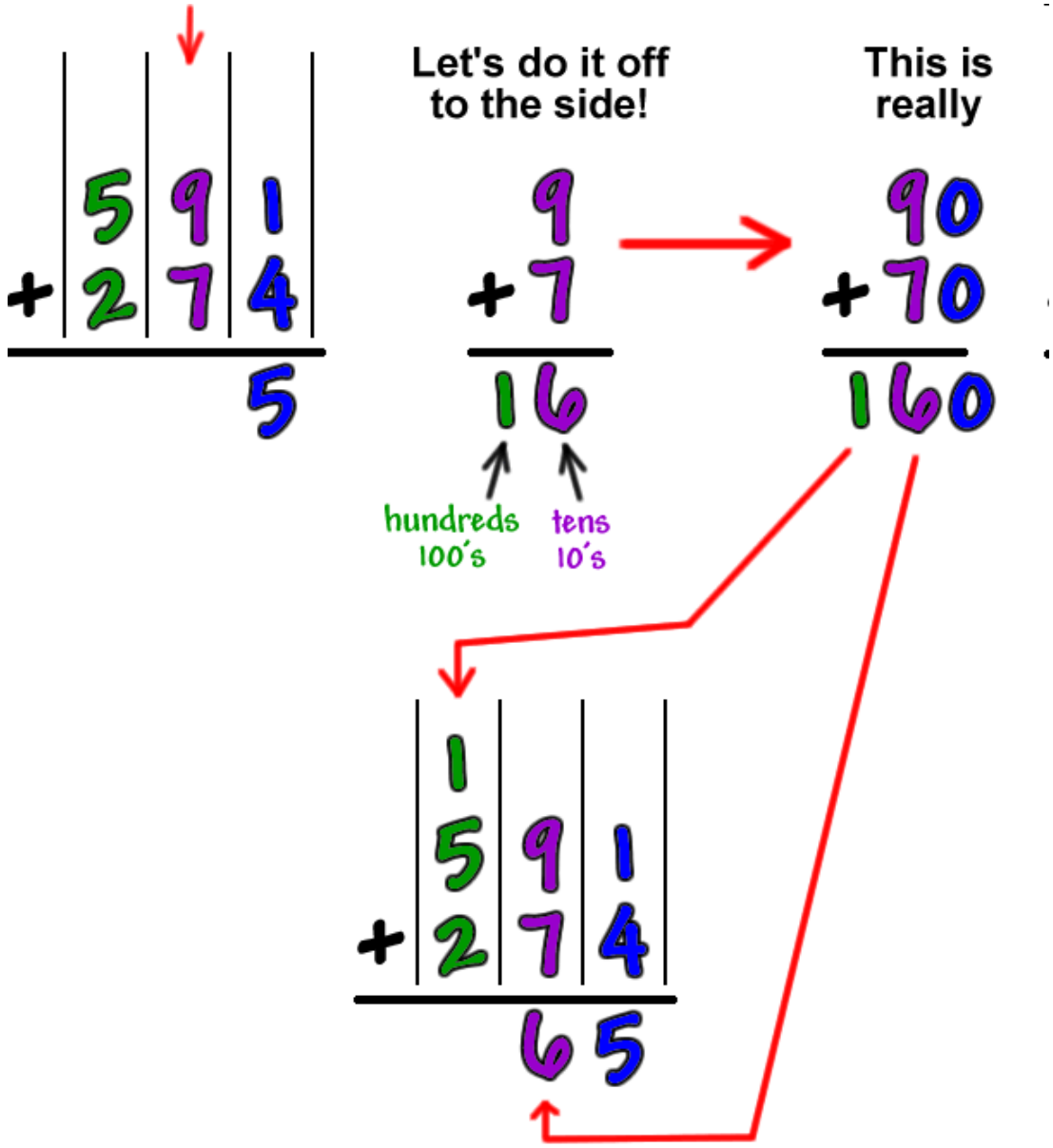
Since a truth table for 2 variables has $2^2 = 4$ lines it follows that $f(x, y)$ can have $2^{2^2} = 2^4 = 16$ possible functions.

These are

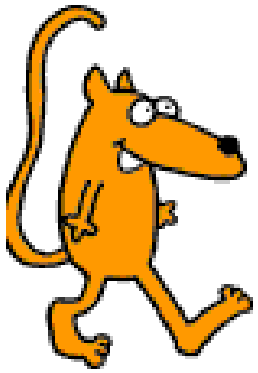
minterms	x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_A	F_B	F_C	F_D	F_E	F_F
$x'y'$	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x'y$	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
xy'	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
xy	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- $F_0 = 0$ = unconditionally false
- $F_1 = xy$ = AND
- $F_6 = x \oplus y$ = XOR = $x'y + xy'$
- $F_7 = x + y$ = OR [show that $x'y + xy' + xy = x + y$]
- $F_8 = (x + y)'$ = NOR
- $F_9 = (x \oplus y)' = x \odot y$ = XNOR (also called EQUIVALENCE)
- $F_{14} = (xy)'$ = NAND
- $F_{15} = 1$ = unconditionally true


Hur adderar vi tal?

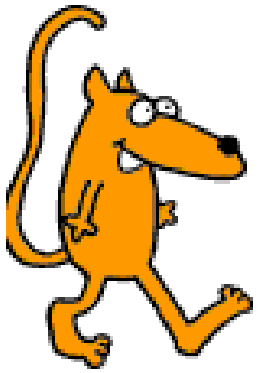


Put the hundreds guy up in the hundreds column...
Put the tens guy in the tens answer spot.



Hur adderar vi binära tal?

```
    1 0 1
+   1 1 1
-------
    ? ? ?
```



Hur adderar vi binära tal?

Minnessiffra – Carry
(överskjutande 2-tal)

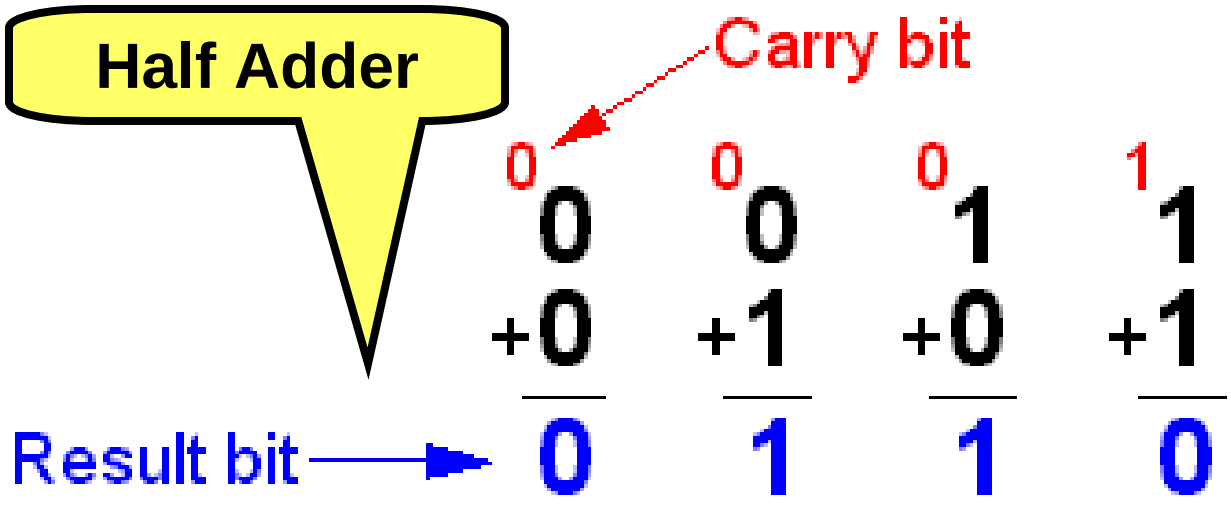
överskjutande 4-tal

överskjutande 8-tal

```
    1     (c)
  1 0 1   (x)
+ 1 1 1   (y)
-------
      0
```

Summanderna är på tre bitar och resultatet kan då bli fyra bitar.

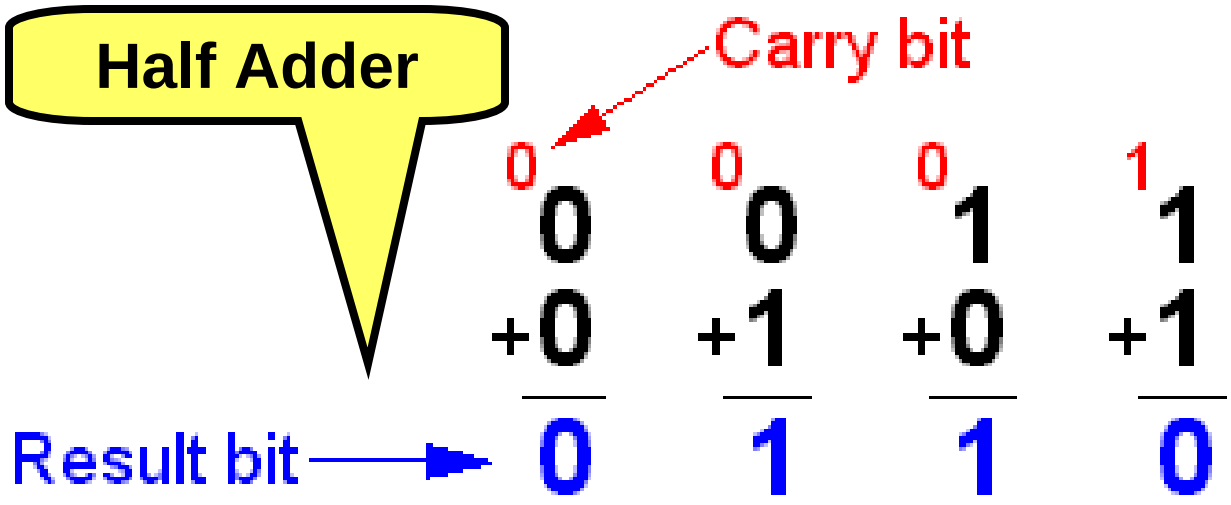
Har vi endast möjlighet att lagra tre bitar finns alltså risk för *overflow*.



Row	Inputs		Output	
	x	y	c_{out}	s
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

C_{out} = ?

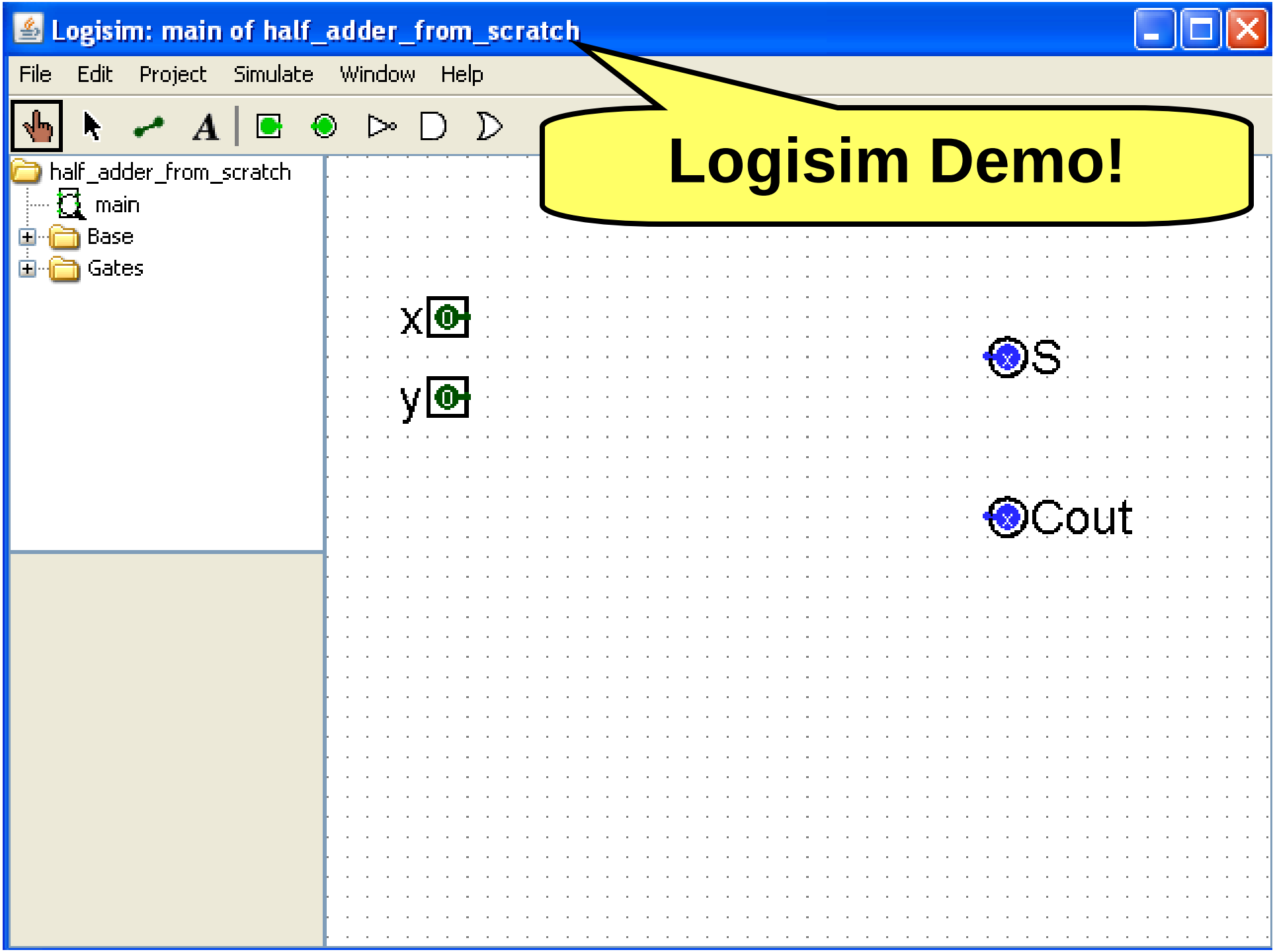
S = ?

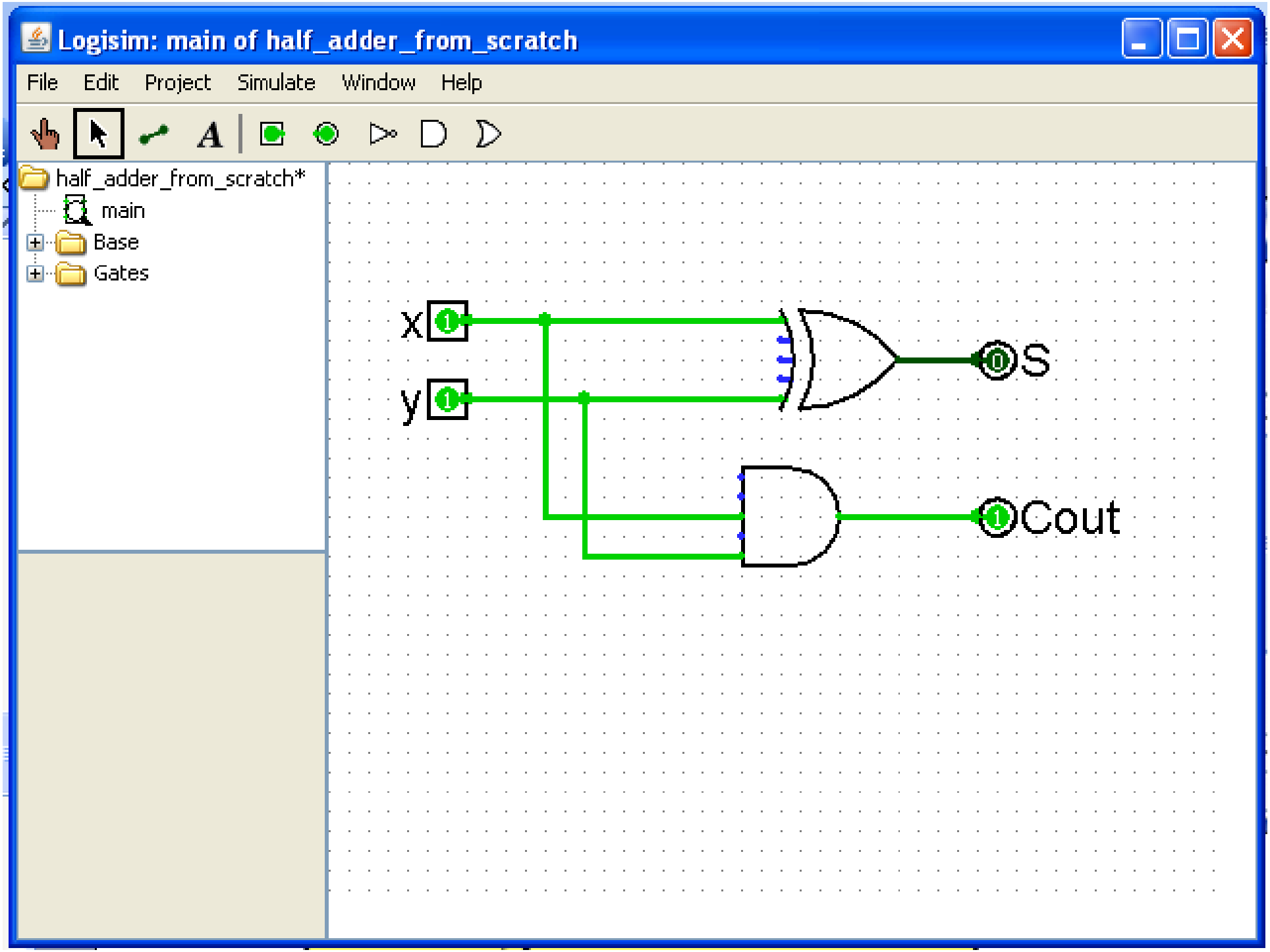


Row	Inputs		Output	
	x	y	c_{out}	s
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

C_{out} = x AND y

S = x XOR y


Logisim: main of half_adder_from_scratch
File
Edit
Project
Simulate
Window
Help
Logisim Demo!
half_adder_from_scratch
main
Base
Gates
x
y
S
Cout


Logisim: main of half_adder_from_scratch
File
Edit
Project
Simulate
Window
Help
half_adder_from_scratch*
main
Base
Gates
x
y
S
Cout

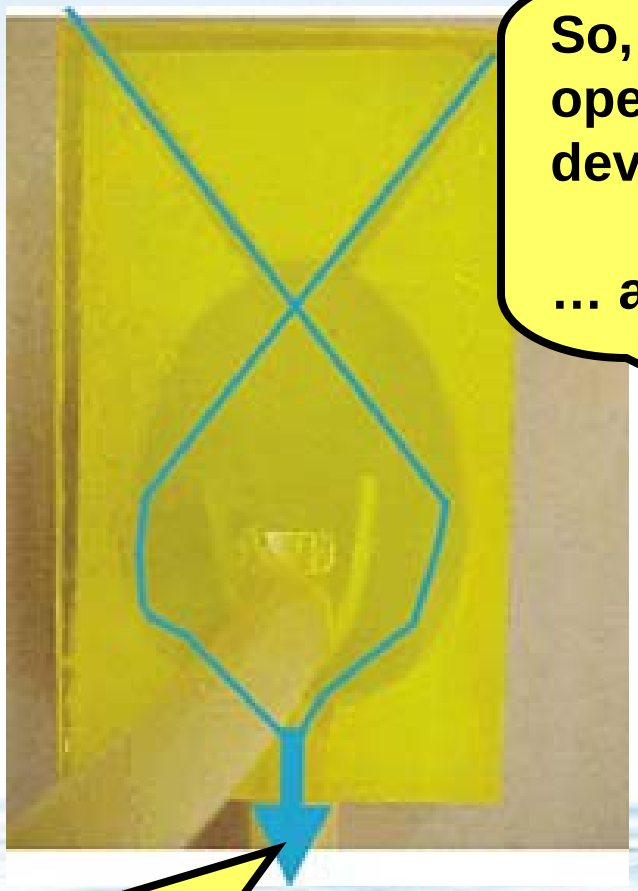
Programmable Water

When just one of them is "on"...

When both are "on", the two jets collide, going vertically down to the "U" piece, that collects the water.

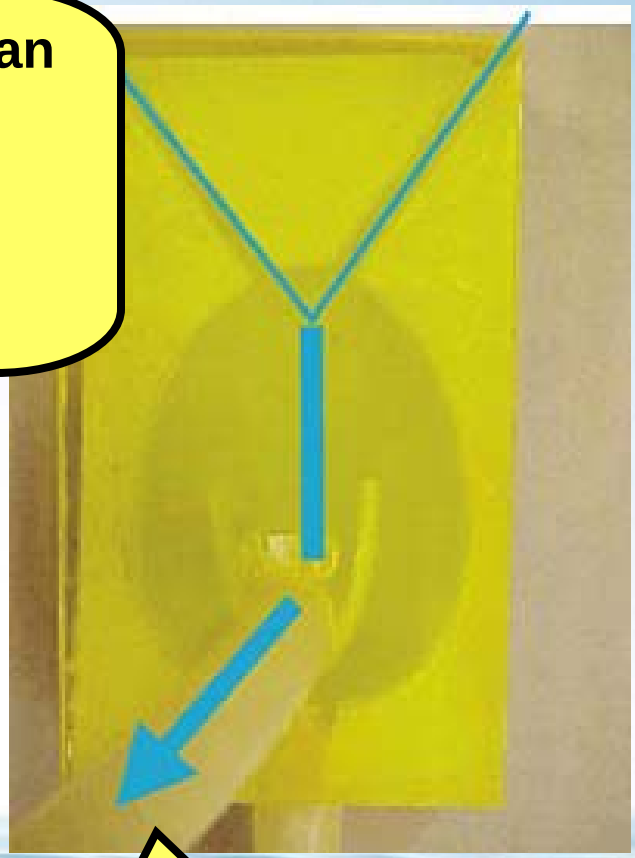
So, we have two Boolean operations in just one device…

… and it's a *half adder.*

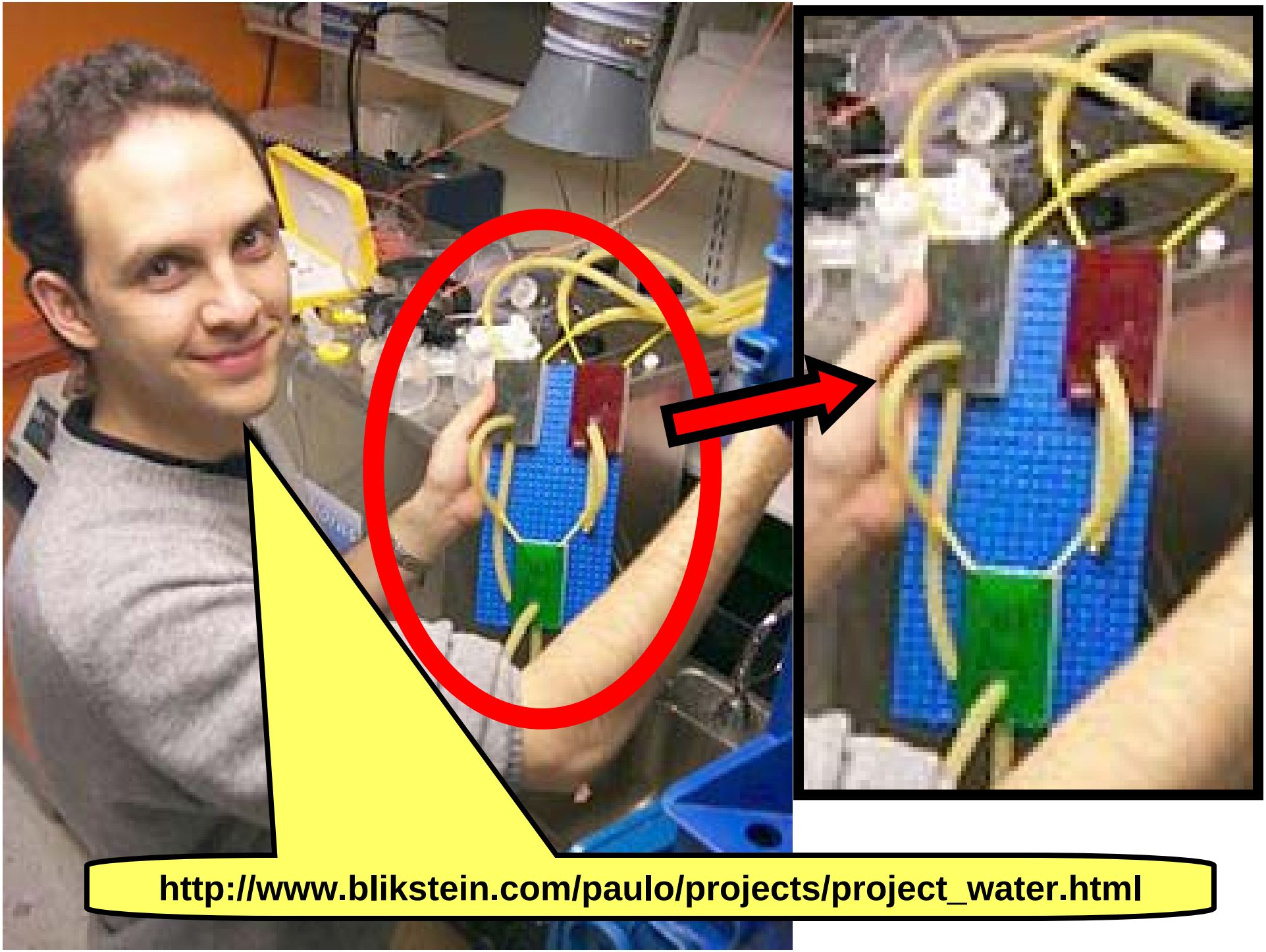


In other words, it's a ***XOR*** gate.

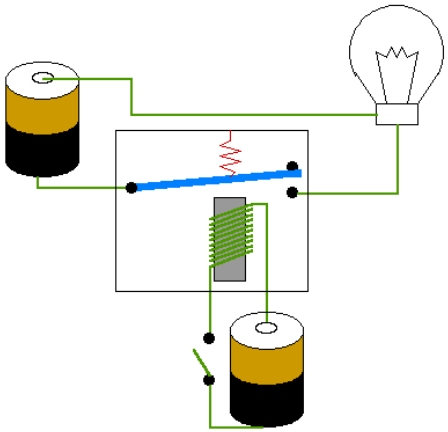
Paulo Blikstein



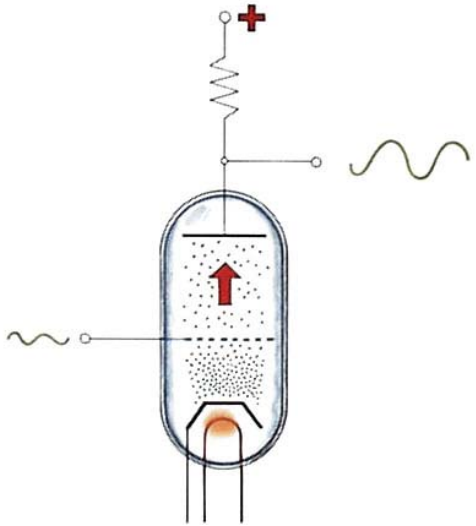
In other words, an ***AND*** gate.


http://www.blikstein.com/paulo/projects/project_water.html

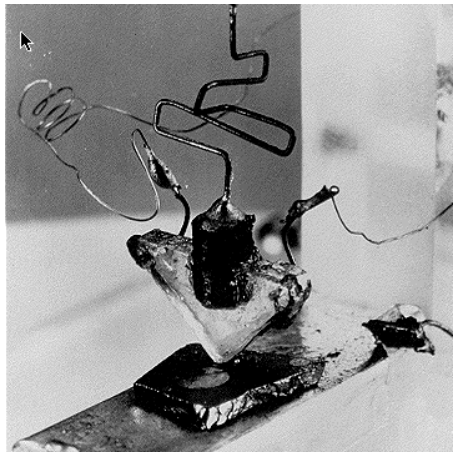

Vill man inte bygga en dator av vatten
funkar det bra med reläer.
http://web.cecs.pdx.edu/~harry/Relay/



Relay

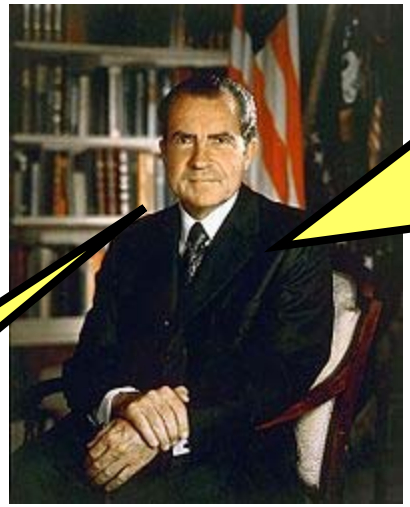


Triod

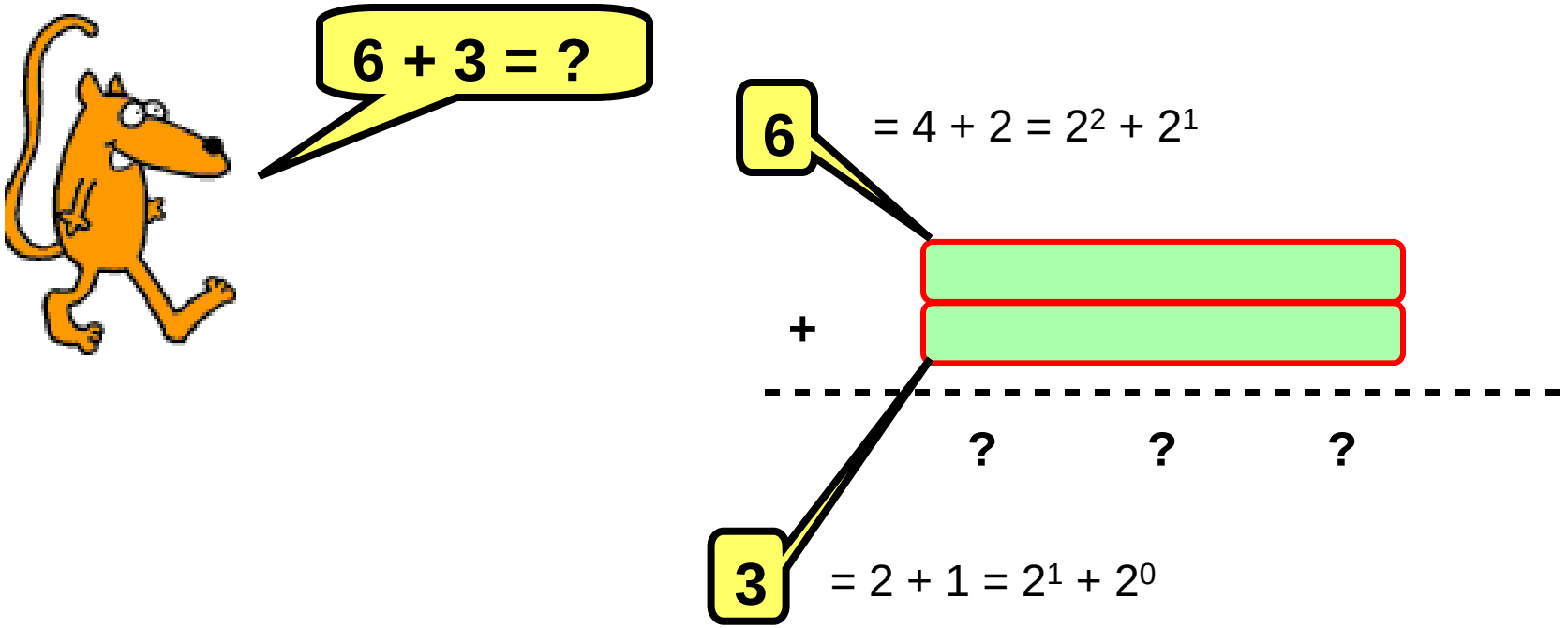


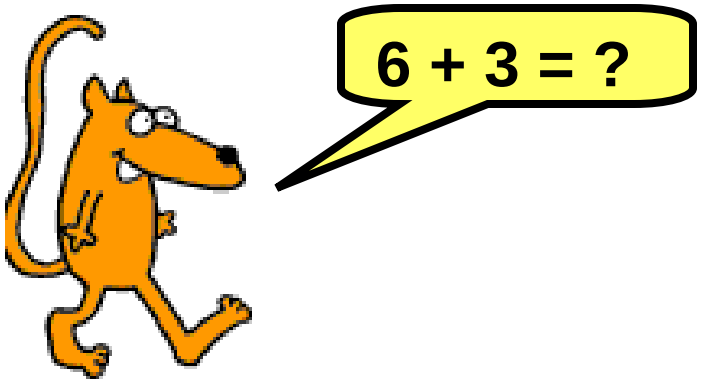
Transistor

Water Gate



Alla dessa tekniker kan användas för att konstruera logiska kretstar... till exempel för att addera två tal.


6 + 3 = ?
6
= 4 + 2 = 2^2 + 2^1
+
? ? ?
3
= 2 + 1 = 2^1 + 2^0



6 $= 4 + 2 = 2^2 + 2^1$

	1	**1**	**0**
+	**0**	**1**	**1**
	?	**?**	**?**

3 $= 2 + 1 = 2^1 + 2^0$

Carry Out från den ena enbits-additionen blir carry in för nästa…

Overflow

```
      1     0     0  (c)
      1     1     0  (x)
  +   0     1     1  (y)
 ---------------------
(1)   0  (1)0  (0)1
```

Row	Inputs			Outputs		Comment
	x	y	c_{in}	c_{out}	s	
0	0	0	0	0	0	$0 + 0 + 0 = 00_2$
1	0	0	1	0	1	$0 + 0 + 1 = 01_2$
2	0	1	0	0	1	$0 + 1 + 0 = 01_2$
3	0	1	1	1	0	$0 + 1 + 1 = 10_2$
4	1	0	0	0	1	$1 + 0 + 0 = 01_2$
5	1	0	1	1	0	$1 + 0 + 1 = 10_2$
6	1	1	0	1	0	$1 + 1 + 0 = 10_2$
7	1	1	1	1	1	$1 + 1 + 1 = 11_2$

Vi skriver det som en *sanningstabell*

Values of the inputs when Carry Out equals 1.

Row	Inputs			c_{out}
	x	y	c_{in}	
3	0	1	1	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

All inputs in a row can be represented as a ***minterm***. In a minterm each input variable appearc once, either as the variable itself or as the inverse.

Values of the inputs when Carry Out equals 1.

Row	Inputs			c_{out}	minterms
	x	y	c_{in}		
3	0	1	1	1	$!xyc_{in}$
5	1	0	1	1	$x!yc_{in}$
6	1	1	0	1	$xy!c_{in}$
7	1	1	1	1	xyc_{in}

$$C_{out} = !xyc_{in} + x!yc_{in} + xy!c_{in} + xyc_{in}$$

Row 3 5 6 7

In a ***minterm***, each input variable, A, B or C appears once, either as the variable itself or as the inverse. Each minterm ***corresponds to exactly one entry (row) in the truth table***.

Values of the inputs when Carry Out equals 1.

Row	Inputs			c_{out}	minterms
	x	y	c_{in}		
3	0	1	1	1	$!xyc_{in}$
5	1	0	1	1	$x!yc_{in}$
6	1	1	0	1	$xy!c_{in}$
7	1	1	1	1	xyc_{in}

$$C_{out} = !xyc_{in} + x!yc_{in} + xy!c_{in} + xyc_{in}$$

Row 3 5 6 7

$$C_{out} = c_{in}(!xy + x!y) + xy(!c_{in} + c_{in})$$

???

???

Values of the inputs when Carry Out equals 1.

Row	Inputs			c_{out}	minterms
	x	y	c_{in}		
3	0	1	1	1	$!xyc_{in}$
5	1	0	1	1	$x!yc_{in}$
6	1	1	0	1	$xy!c_{in}$
7	1	1	1	1	xyc_{in}

$$C_{out} = !xyc_{in} + x!yc_{in} + xy!c_{in} + xyc_{in}$$

Row 3 5 6 7

$$C_{out} = c_{in}(!xy + x!y) + xy(!c_{in} + c_{in})$$

!xy + x!y = x XOR y

$!c_{in} + c_{in} = 1$

Values of the inputs when Carry Out equals 1.

Row	Inputs			c_{out}	minterms
	x	y	c_{in}		
3	0	1	1	1	$!xyc_{in}$
5	1	0	1	1	$x!yc_{in}$
6	1	1	0	1	$xy!c_{in}$
7	1	1	1	1	xyc_{in}

$$C_{out} = !xyc_{in} + x!yc_{in} + xy!c_{in} + xyc_{in}$$

$$C_{out} = c_{in}(!xy + x!y) + xy(!c_{in} + c_{in})$$

$$C_{out} = c_{in}(x \text{ XOR } y) + xy$$

Logisim DEMO

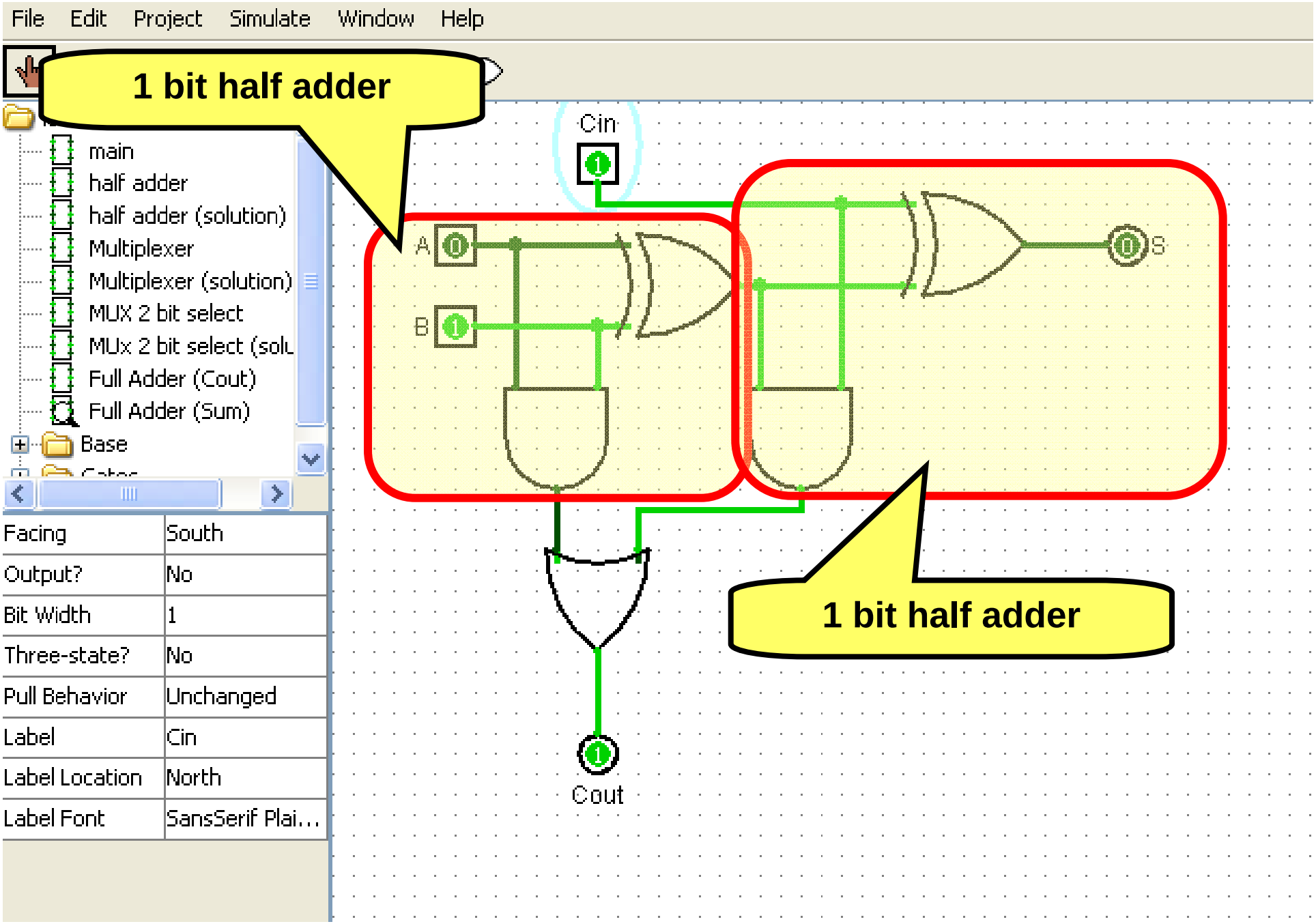
Aha!

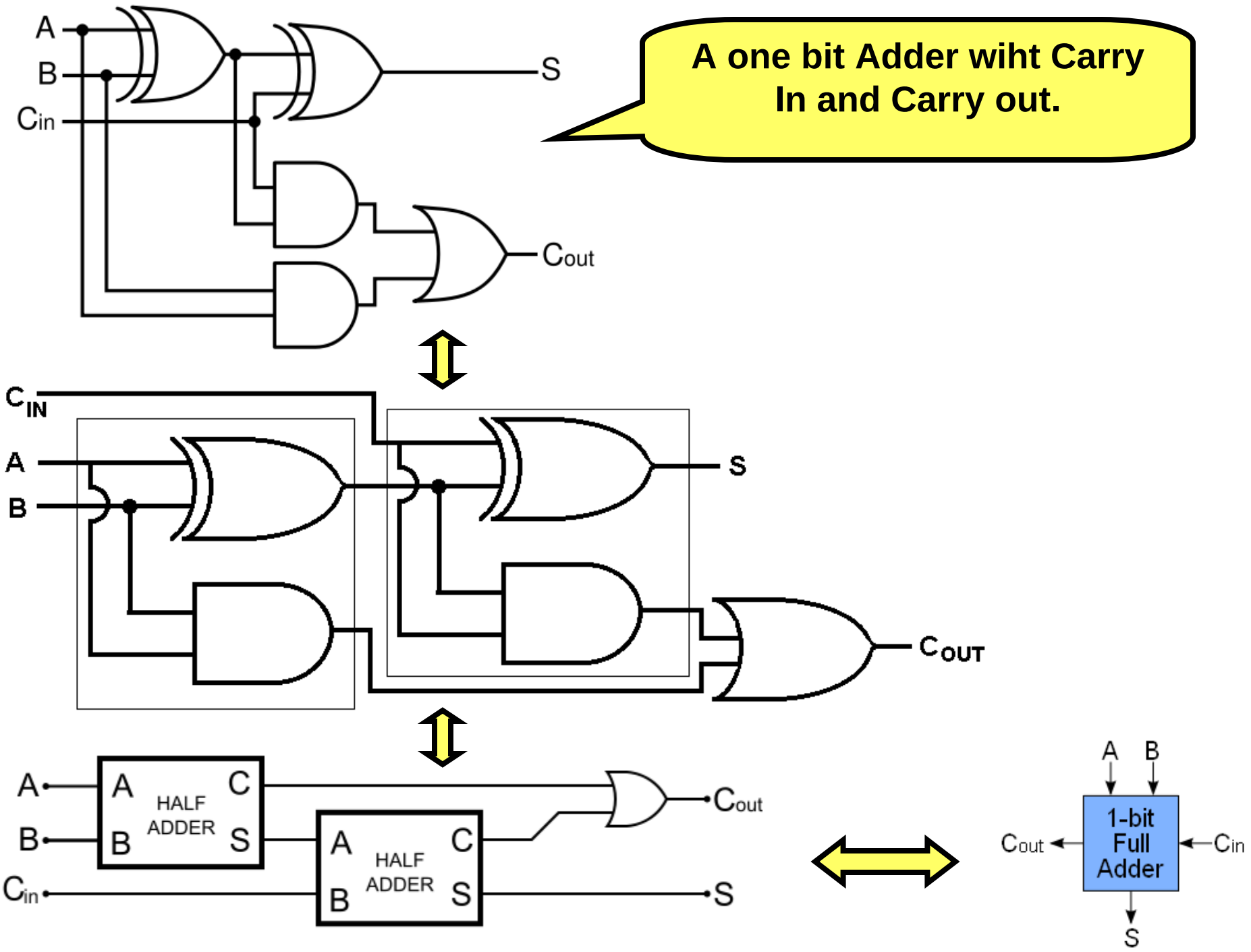
Sum is one only one when an odd number of inputs are one.

Row	Inputs			S	minterms
	x	y	c_{in}		
1	0	0	1	1	$!x!yc_{in}$
2	0	1	0	1	$!xy!c_{in}$
4	1	0	0	1	$x!y!c_{in}$
7	1	1	1	1	xyc_{in}

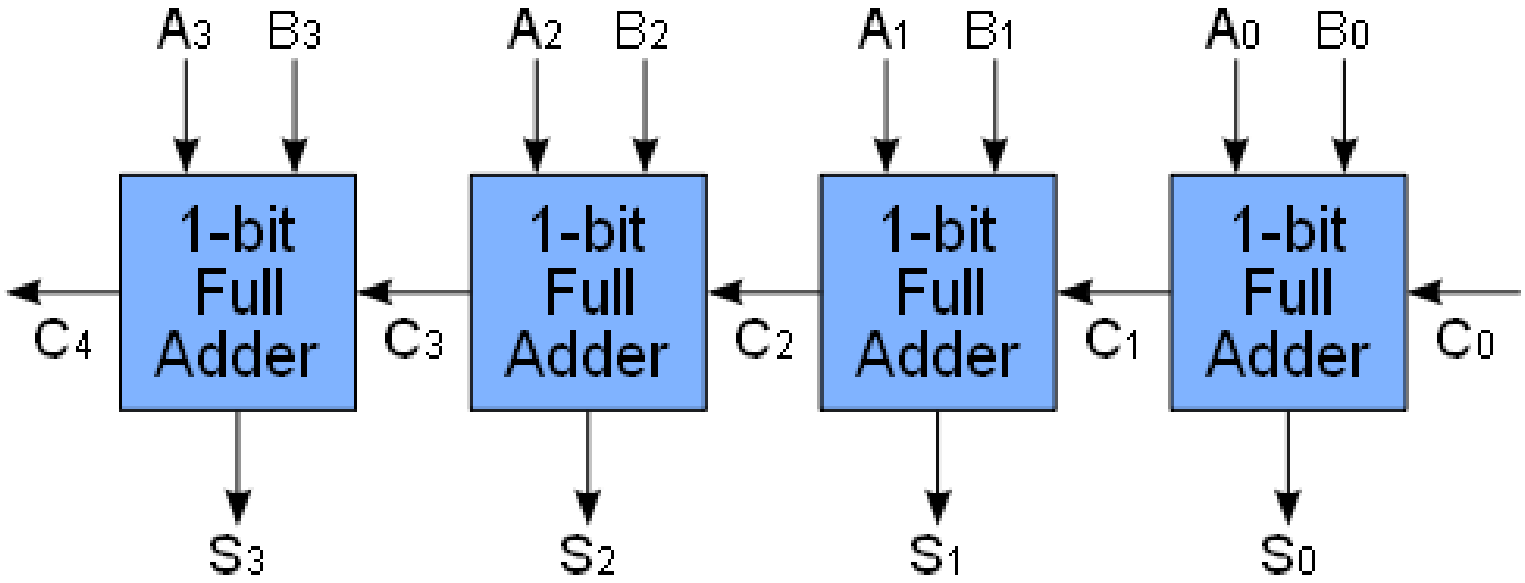
$S = x \text{ XOR } y \text{ XOR } c_{in}$

Logisim DEMO


File
Edit
Project
Simulate
Window
Help
1 bit half adder
main
half adder
half adder (solution)
Multiplexer
Multiplexer (solution)
MUX 2 bit select
MUx 2 bit select (solu
Full Adder (Cout)
Full Adder (Sum)
Base
Facing
South
Output?
No
Bit Width
1
Three-state?
No
Pull Behavior
Unchanged
Label
Cin
Label Location
North
Label Font
SansSerif Plai...
Cin
A
B
S
Cout
1 bit half adder


A one bit Adder wiht Carry In and Carry out.
A
B
Cin
S
Cout
CIN
A
B
S
COUT
A
B
Cin
A
B
HALF ADDER
C
S
A
B
HALF ADDER
C
S
Cout
S
A
B
Cout
1-bit Full Adder
Cin
S

When multiple full adders are used with the carry ins and carry outs chained together then this is called a *ripple carry adder* because the correct value of the carry bit ripples from one bit to the next.



Tutorial: **Mer om grindar och andra grundläggande komponenter.**

Uppgift: **Lär dig mer om grindar, multiplexrar, demultiplexrar adderare och register..**

Utmaning: **Konstruera ett kodlås.**

Skall kunna sätta en 4-siffrig kod.

Skall signalera när rätt kod trykts in.

Ok, om vi först sätter koden till **0127**

När vi trycker in **3, 4, 5** ... händer ingenting.

Om vi fortsätter med **0, 1** händer inget.

Men när vi fortsätter med **2, 7** så öppnas låset.