コンピュータシステムA - ハードウェアを中心に -

#5 機械計算、二値論理、論理回路

Yutaka Yasuda

アナログ表現・デジタル表現

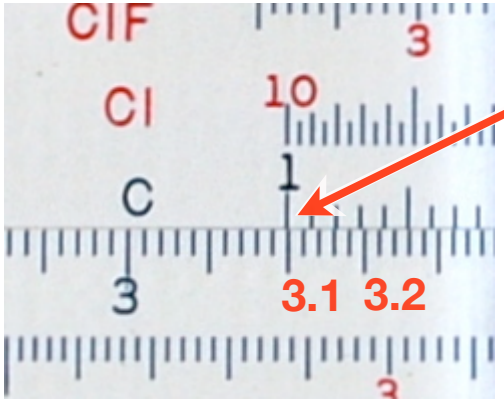
- アナログ情報

  連続的に変化する情報としてとらえ、連続的に変化する何かに置換して表現する

- デジタル情報

  一定の精度での数値によって表現（刻みのある離散的な数値の列として表現）

（計算尺）



これをAnalogに扱うか、Digitalに扱うかがポイント

アナログ表現・デジタル表現

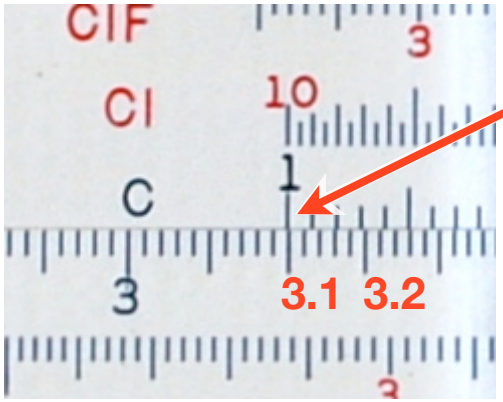
- アナログ情報

  連続的に変化する情報としてとらえ、連続的に変化する何かに置換して表現する

- デジタル情報

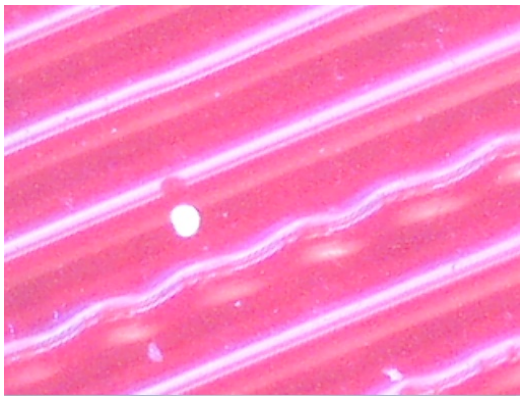
  一定の精度での数値によって表現（刻みのある離散的な数値の列として表現）

（計算尺）



これをAnalogに扱うか、Digitalに扱うかがポイント

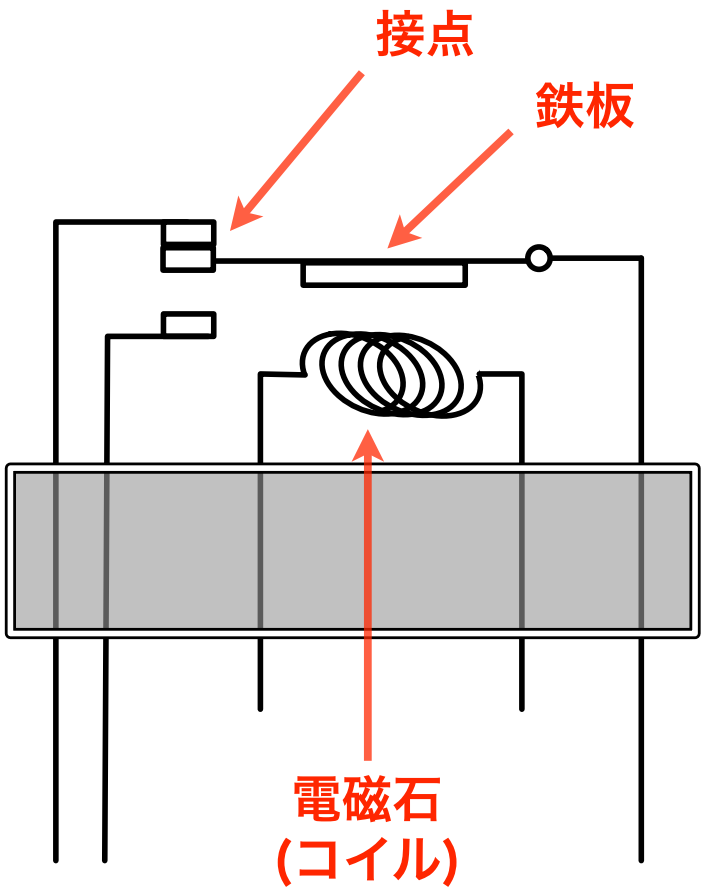
（音）

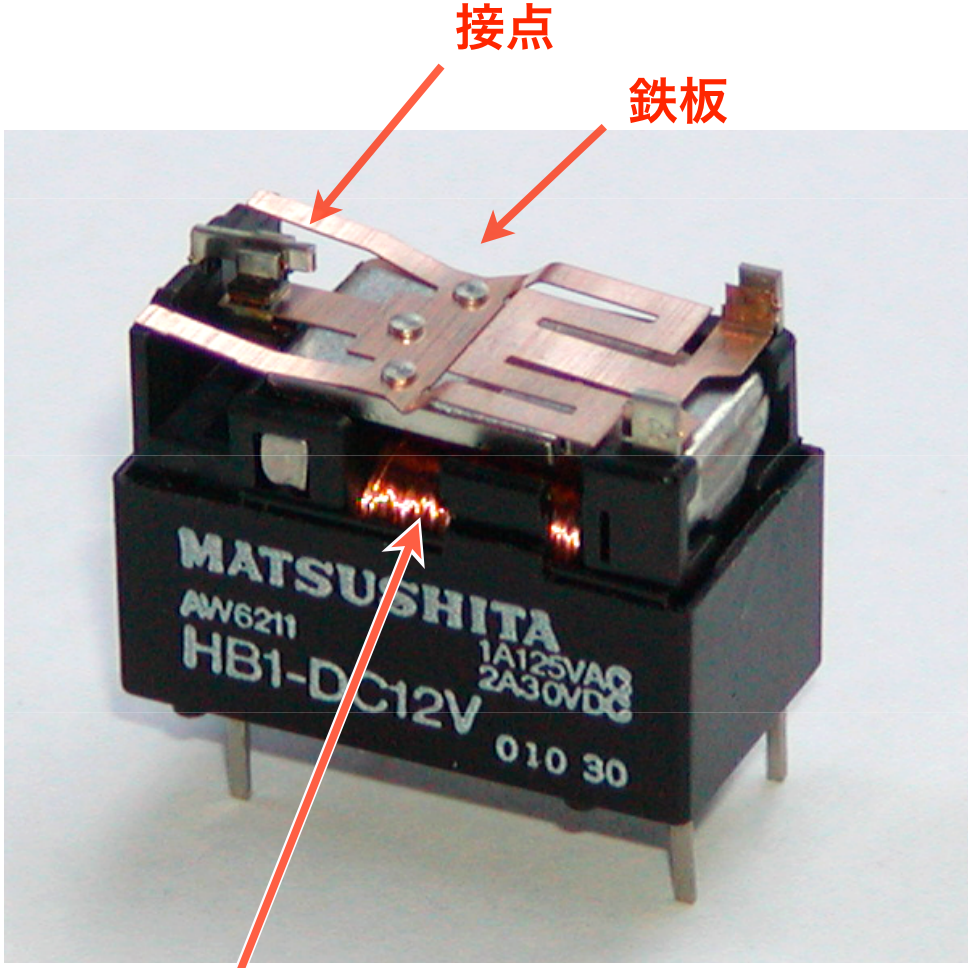


連続な波として記録するか、数値化して扱うか

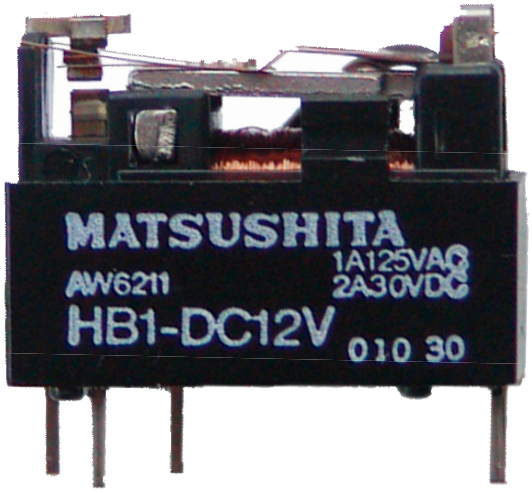

機械式計算機
（手回し）
テキスト

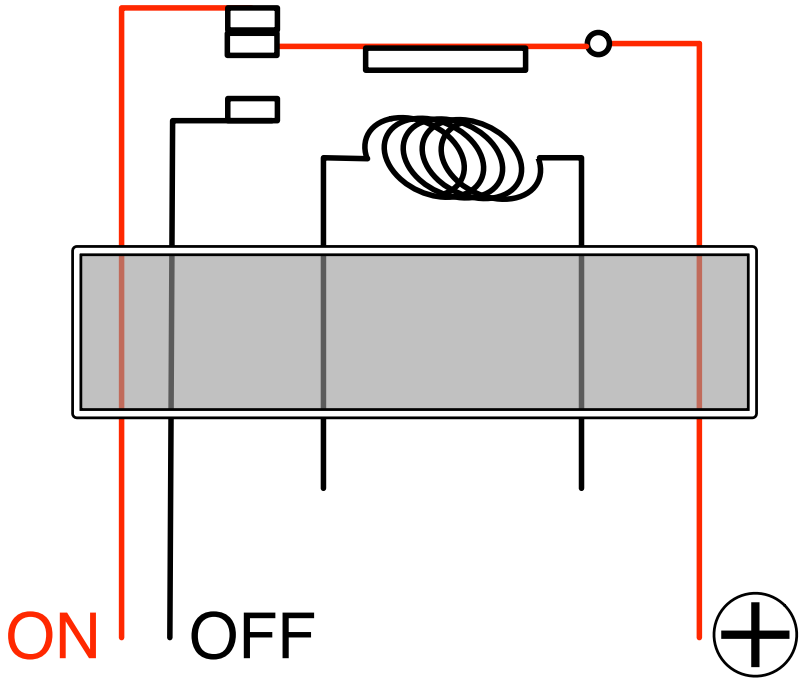
リレー


接点
鉄板
電磁石
(コイル)

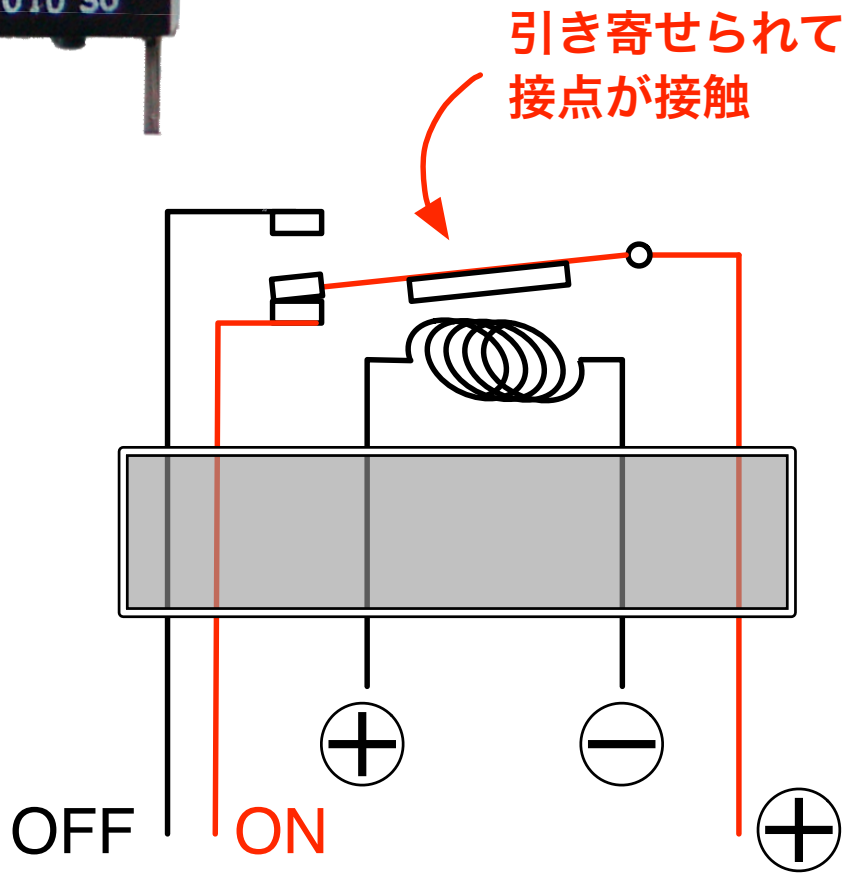

接点
鉄板
MATSUSHITA
AW6211
HB1-DC12V
1A125VAC
2A30VDC
010 30
電磁石
(コイル)

リレー



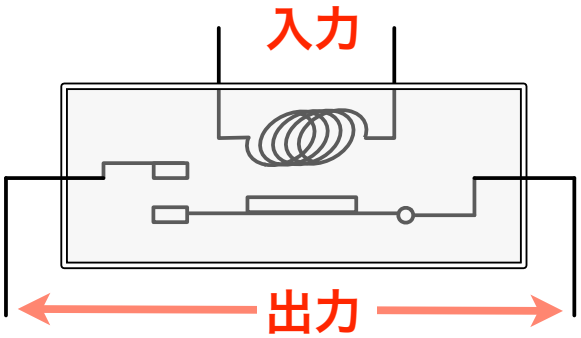


電磁石に電圧を
掛けない場合



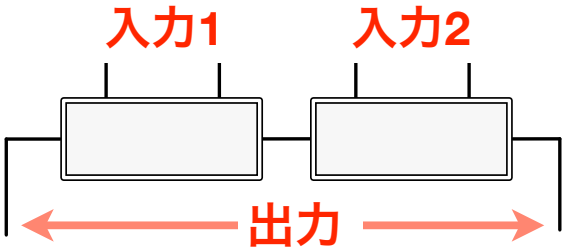
電磁石に電圧を
掛けた場合

リレーによる AND / OR



電磁石に電圧を掛けるとONになる単純なリレー

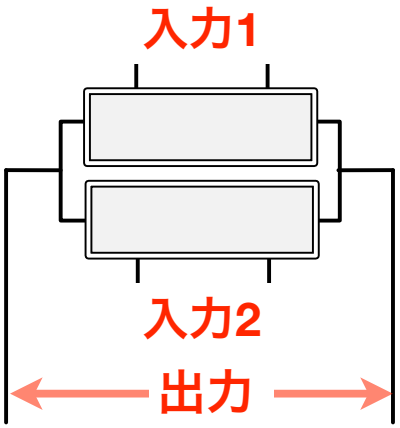
入力	出力
ON	**ON**
OFF	OFF



二つ直列にすると「両方ON」の時に「ON」になる

＝AND

入力1	入力2	出力
ON	**ON**	**ON**
OFF	**ON**	OFF
ON	OFF	OFF
OFF	OFF	OFF



二つ並列にすると「どちらかON」の時に「ON」になる

＝OR

入力1	入力2	出力
ON	**ON**	**ON**
OFF	**ON**	**ON**
ON	OFF	**ON**
OFF	OFF	OFF

パターンと計算

- パターン処理による加算の実現
- 組み合わせの記憶・再現
- 機械処理可能

$3 + 8 = ?$

10x10通りのパターンを
機械処理できればよい

それ自体が難しい

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

二進法の利用

- 二進（二値）であれば 4 通りで済む

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

VS

0+0= 0
1+0= 1
0+1= 1
1+1=10

	0	1
0	0	1
1	1	10

進数

- 一桁を幾つの記号で回すか、を意味する
- 右端のドラムが一周まわると、左隣のダイヤルが一つ進む

10進数はドラムに10種の記号がある数え方。

2進数は2種しか記号がない。短い周期で桁があがるだけで、回り方は同じ。

二進法による足し算

234+456=690 は?

```
 2     3     4
 +     +     +
 4     5     6
 =     =     =
 6     8    10
      +1
 6     9     0
```

234 (11101010) +456 (111001000) =690

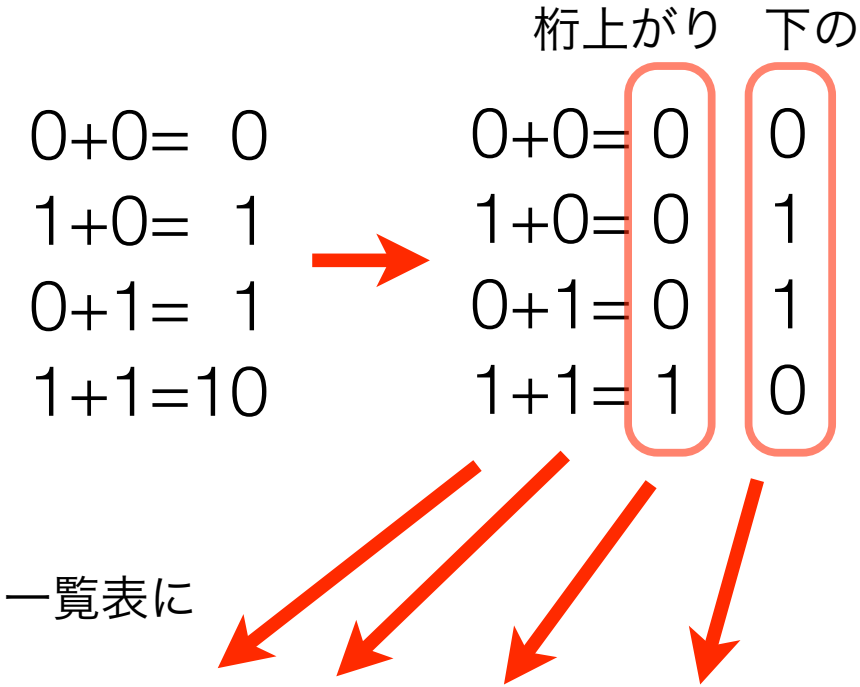
```
          1     1     1     0     1     0     1     0
          +     +     +     +     +     +     +     +
    1     1     1     0     0     1     0     0     0
    =     =     =     =     =     =     =     =     =
    1    10    10     1     0    10     0     1     0
   +1    +1                 +1
 1  0     1     0     1     1     0     0     1     0
```

多数桁の加算は筆算で分解。
1 桁の加算さえできれば良い。

2進でも同様に 1 桁の加算ができればよい。
つまり4 通りの加算パターンを機械で実現できれば良い。

0+0= 0
1+0= 1
0+1= 1
1+1=10

二進法での足し算をリレーで実現する（上の桁）



一覧表に

入力1	入力2	桁上がり	下の桁
0	0	0	0
1	0	0	**1**
0	**1**	0	**1**
1	**1**	**1**	0

1/0 を
ON/OFFに

桁上がり（上の桁）
のパターンだけとりあえず

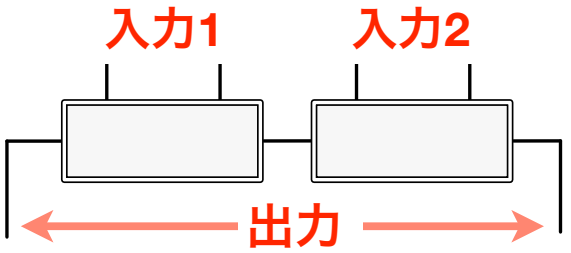
入力1	入力2	桁上がり
OFF	OFF	OFF
ON	OFF	OFF
OFF	**ON**	OFF
ON	**ON**	**ON**

二進法での足し算をリレーで実現する（上の桁）

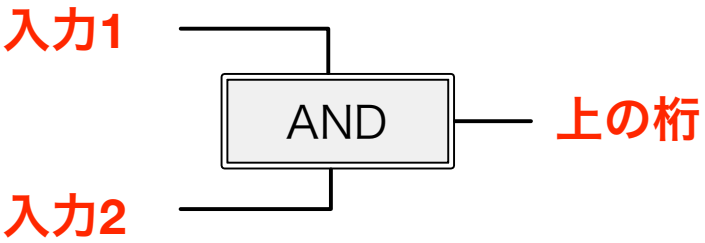
入力1	入力2	桁上がり	下の桁
0	0	0	0
1	0	0	**1**
0	**1**	0	**1**
1	**1**	**1**	0

1/0 を
ON/OFFに

入力1	入力2	桁上がり
OFF	OFF	OFF
ON	OFF	OFF
OFF	**ON**	OFF
ON	**ON**	**ON**



抽象化して
こう描く

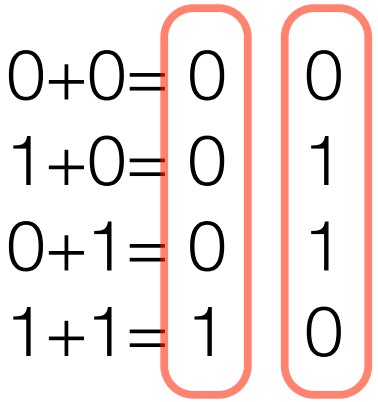


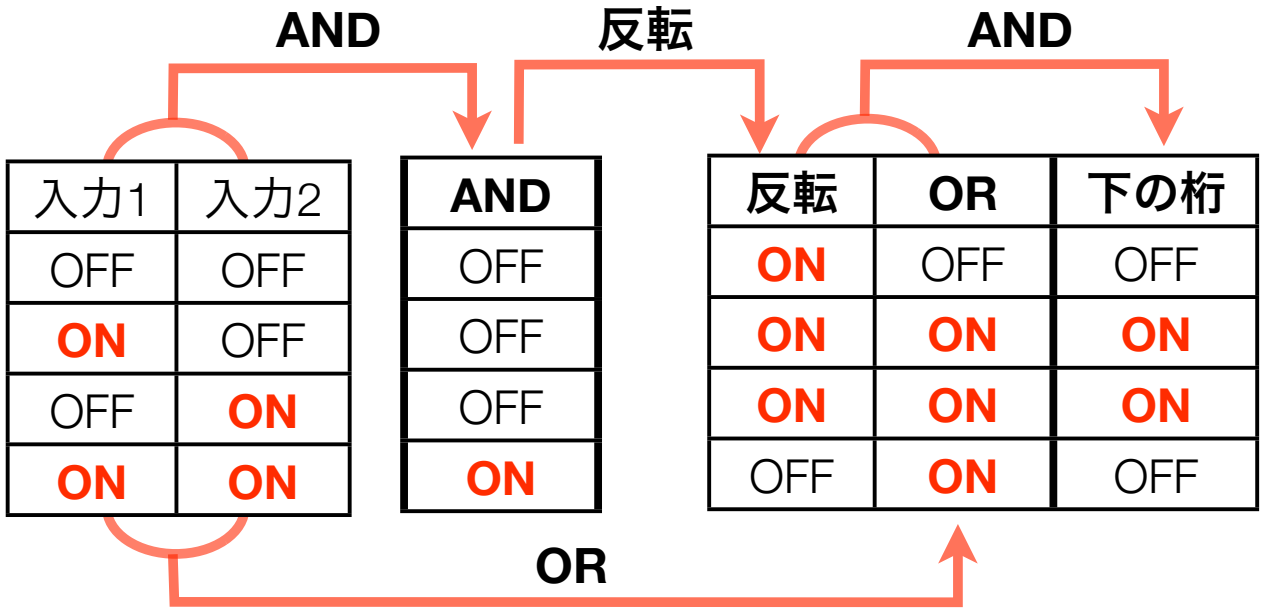
AND 回路

**2入力1接点
リレーの図**

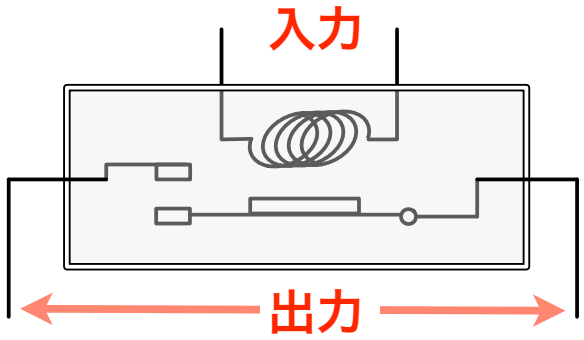
二つ直列にすると「両方ON」
の時に「ON」になる

二進法での足し算をリレーで実現する（下の桁）





リレーによる反転操作



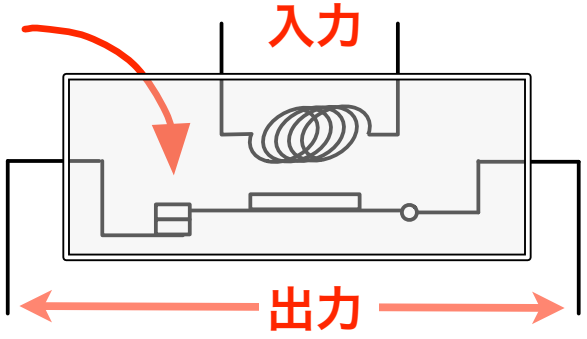
電磁石に電圧を掛けると

ON

になるリレー

入力	出力
ON	**ON**
OFF	OFF

引き寄せられると
接点が離れる



電磁石に電圧を掛けると

OFF

になるリレー

入力	出力
ON	OFF
OFF	**ON**

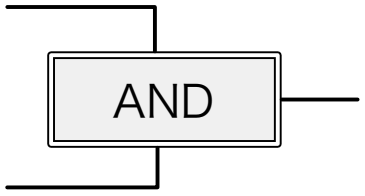
二進法での足し算をリレーで実現する（下の桁）

	桁上がり	下の桁
0+0=	0	0
1+0=	0	1
0+1=	0	1
1+1=	1	0

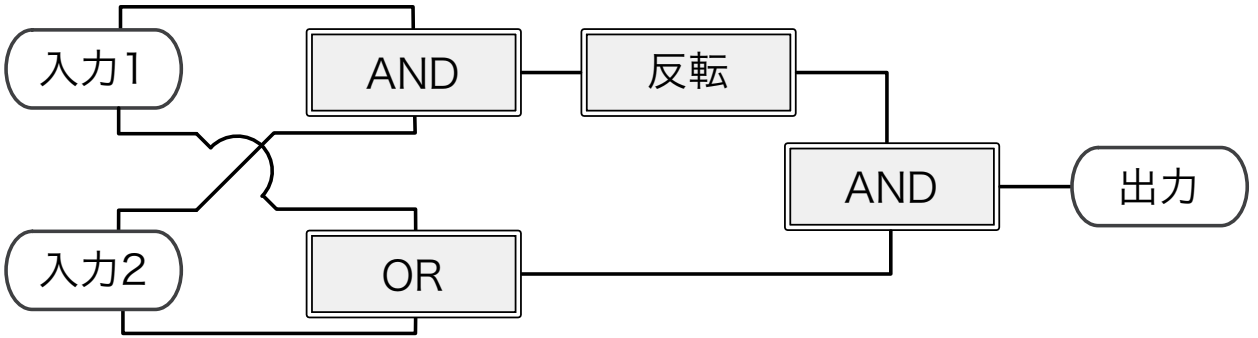
AND　反転　AND

入力1	入力2	AND	反転	OR	下の桁
OFF	OFF	OFF	ON	OFF	OFF
ON	OFF	OFF	ON	ON	ON
OFF	ON	OFF	ON	ON	ON
ON	ON	ON	OFF	ON	OFF

OR



2入力1接点
リレーの図

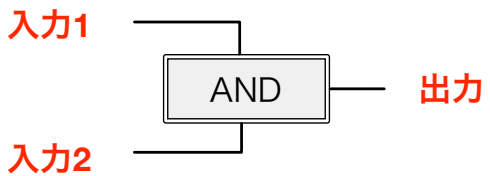


二進法での足し算をリレーで実現する

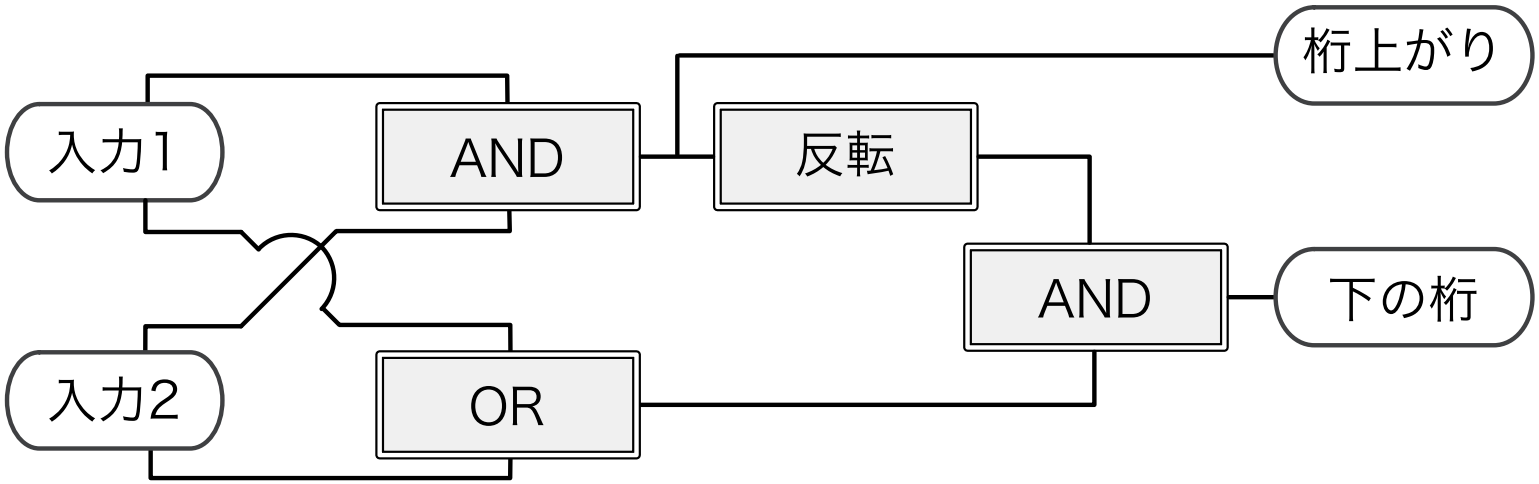
0+0= 0 0
1+0= 0 1
0+1= 0 1
1+1= 1 0

桁上がり　下の桁

入力1	入力2	桁上がり	下の桁
OFF	OFF	OFF	OFF
ON	OFF	OFF	**ON**
OFF	**ON**	OFF	**ON**
ON	**ON**	**ON**	OFF



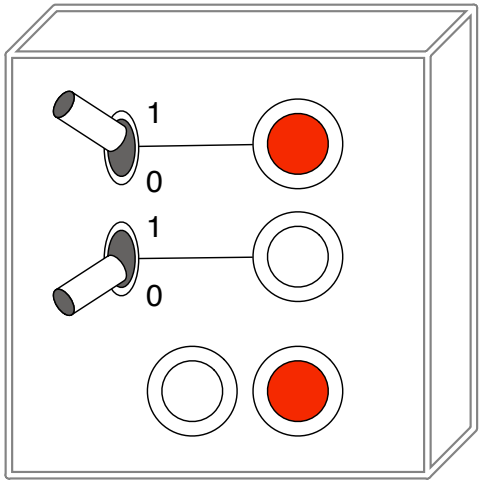
上の桁の回路



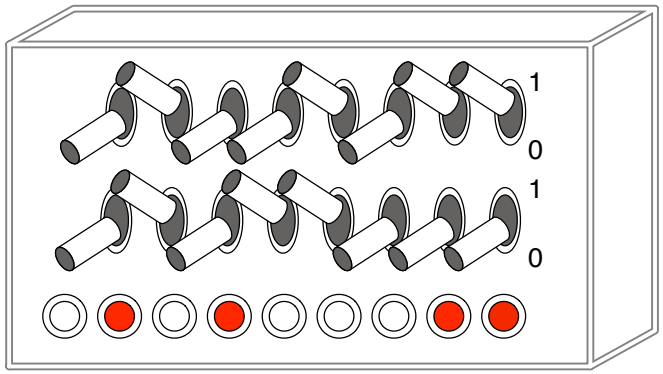
二進法での足し算をリレーで実現する

足し算のパターン

入力1	入力2	桁上がり	下の桁
OFF	OFF	OFF	OFF
ON	OFF	OFF	**ON**
OFF	**ON**	OFF	**ON**
ON	**ON**	**ON**	OFF



リレーを 4 つ使う
「計算機」
（1桁・加算専用）



多数桁へ...

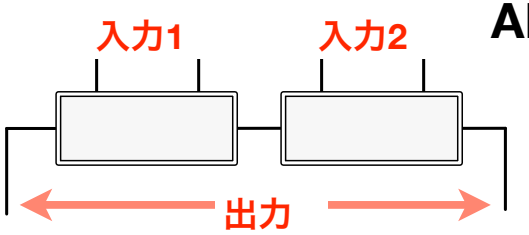
TOSBAC 3400 (1967)

TOSBAC 3400, at Kyoto Sangyo University

TOSBAC 3400 (1967)

TOSBAC 3400, at Kyoto Sangyo University

ゲート記号による表記

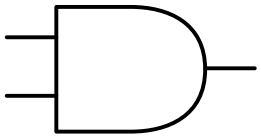


AND

真理値表

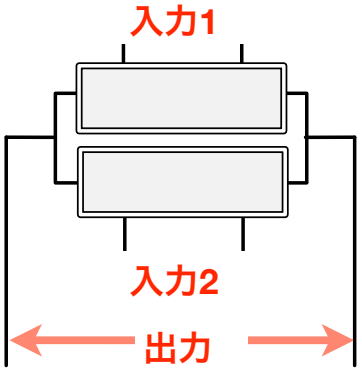
入力1	入力2	出力
ON	**ON**	**ON**
OFF	**ON**	OFF
ON	OFF	OFF
OFF	OFF	OFF

ゲートのシンボル



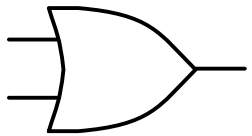
入出力表

入力1	入力2	出力
1	**1**	**1**
0	**1**	0
1	0	0
0	0	0

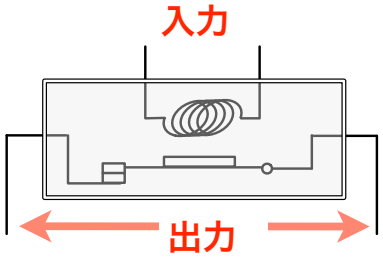


OR

入力1	入力2	出力
ON	**ON**	**ON**
OFF	**ON**	**ON**
ON	OFF	**ON**
OFF	OFF	OFF

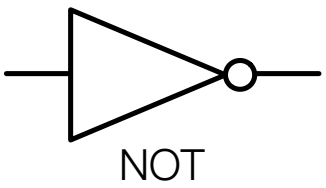


入力1	入力2	出力
1	**1**	**1**
0	**1**	**1**
1	0	**1**
0	0	0



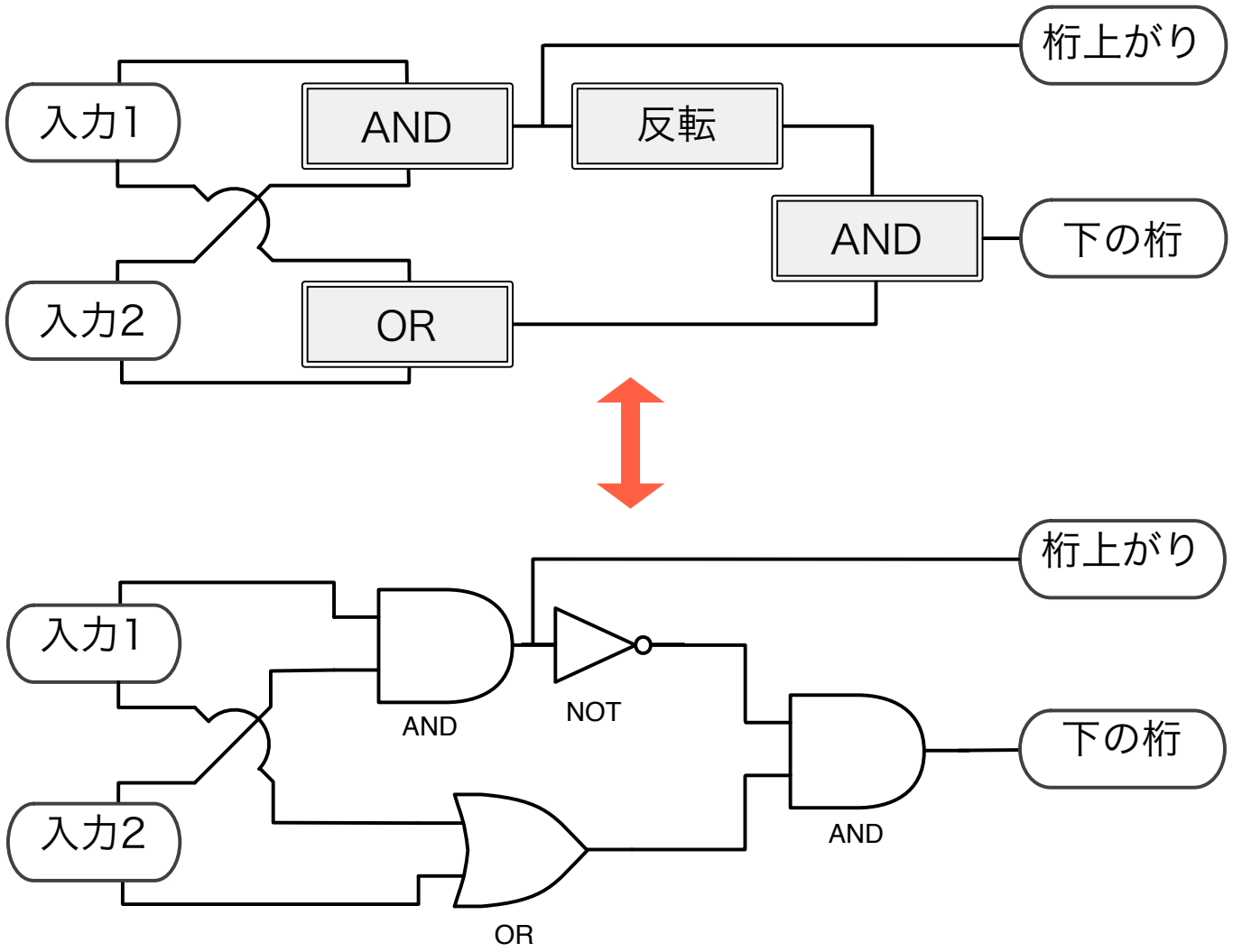
NOT（反転）

入力	出力
ON	OFF
OFF	**ON**



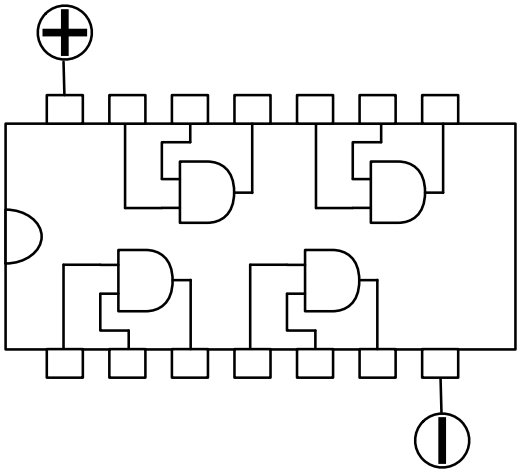
入力	出力
1	0
0	**1**

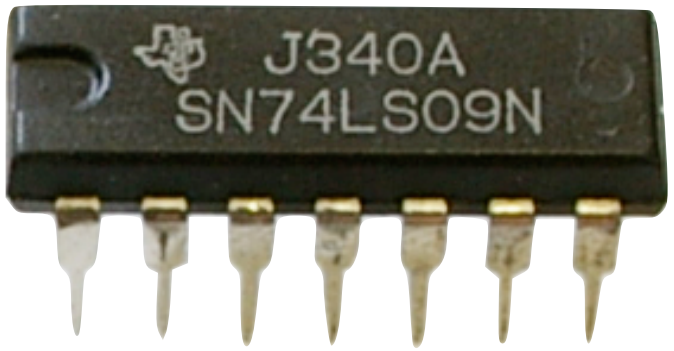
ゲート記号による表記



ゲート IC

SN7409 (2-in AND x 4)





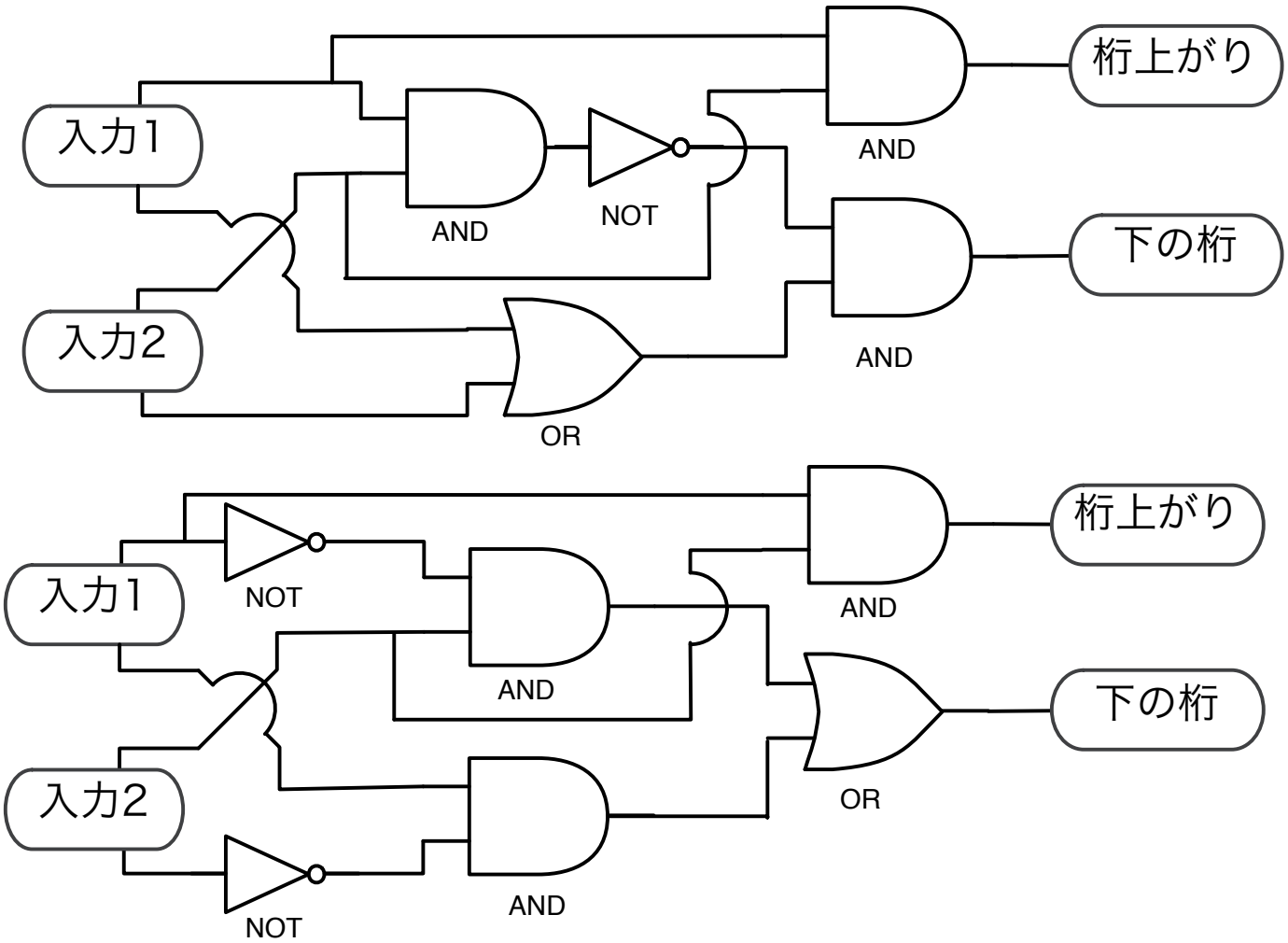
ICの反応速度は数十ナノ秒

ゲートが幾つか集積されて一つのパッケージに入っているため、回路全体がさらに小さくなる。

故障も減り、配線などの工数が減って製造コストも下がる。

他の回路の可能性

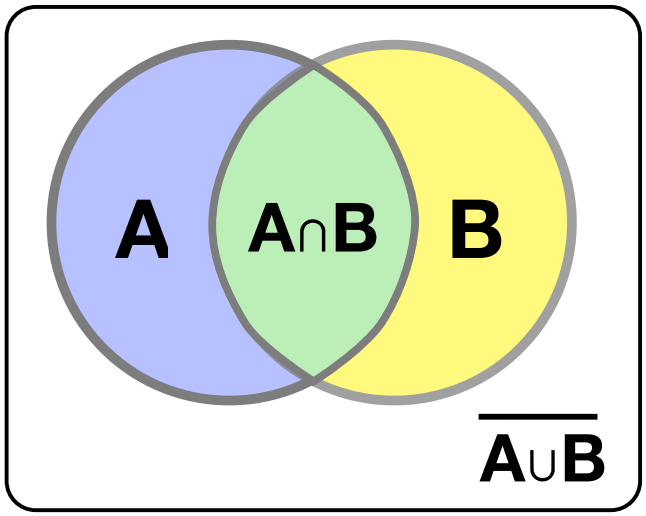
論理的に等価な回路は幾らもあり得る



論理回路

- リレー回路による加算：実体はパターン処理
- それを論理処理（and, or, not の組み合わせ）で実現
- 論理を処理する回路が実現できた

ベン図



真理値表

A	B	A and B
偽	偽	偽
真	偽	偽
偽	真	偽
真	真	真

論理回路

- 真をON, 偽をOFFとすれば、リレーで論理関数(and, or, not)を実現する電気回路を実装できる

  → 論理関数は直接的に電気回路に翻訳（変換）できる

- ONを1、OFFを0とすれば、二値（二進法）の計算は論理関数の組み合わせによって実現できる

  → 二値の計算は論理回路でハードウェア化できる

- 我々は数値を電気回路によって計算する方法をみつけた

ブール代数

- 真と偽による論理を代数的に処理する
- 真偽はそれぞれ 1 と 0 に置換
- 論理関数： and, or, not

真理値表

A	B	A and B
偽	偽	偽
真	偽	偽
偽	**真**	偽
真	**真**	**真**

入出力表

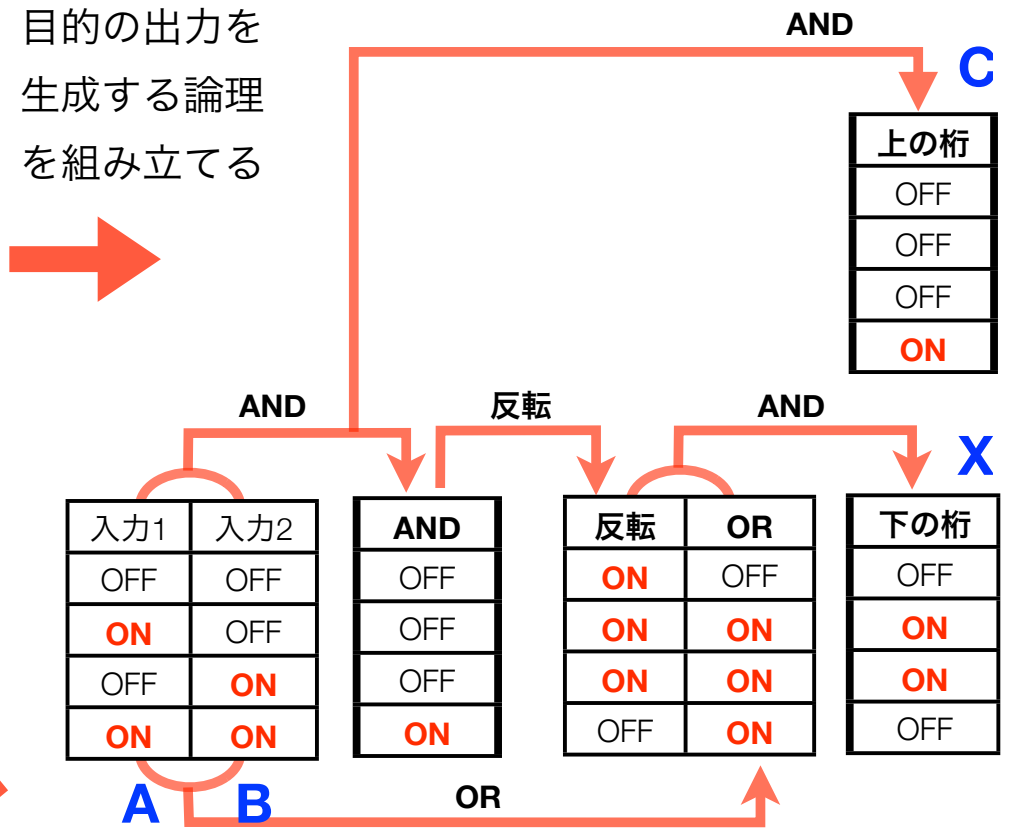
A	B	A and B
0	0	0
1	0	0
0	**1**	0
1	**1**	**1**

入出力表から論理式へ

入出力表

入力1	入力2	桁上がり	下の桁
0	0	0	0
1	0	0	**1**
0	**1**	0	**1**
1	**1**	**1**	0
A	**B**	**C**	**X**

目的の出力を
生成する論理
を組み立てる



論理式に書き直してみる

C = A AND B

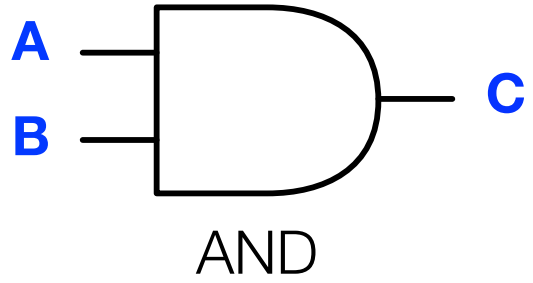
X = (NOT (A AND B)) AND (A OR B)

（一般的な論理式は AND を積、ORを和として表現するがここでは分かりやすく AND/OR を用いた）

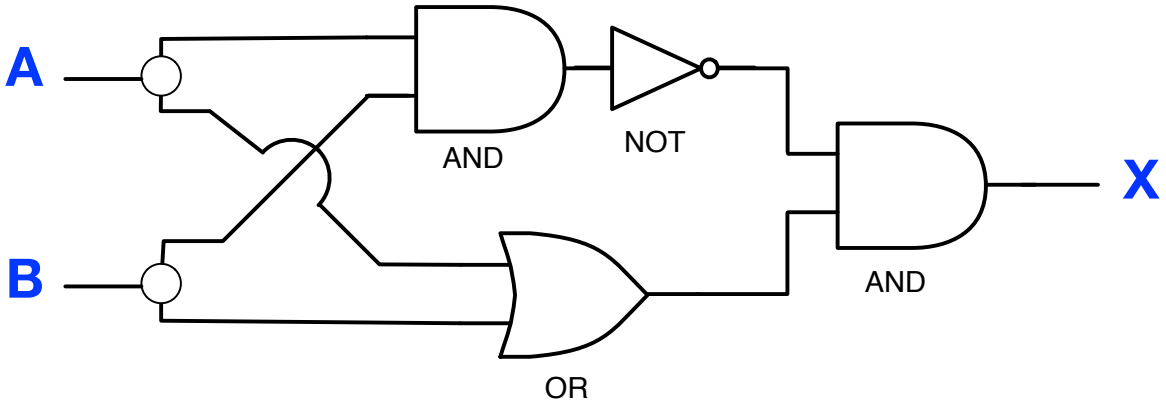
論理式と論理回路

A	B		
0+0=	0	0	
1+0=	0	1	
0+1=	0	1	
1+1=	1	0	
	上の桁	下の桁	
	C	X	

C = A AND B



X = (NOT (A AND B)) AND (A OR B)



ここまでのまとめ

- 電気の流れを制御して計算をする方法が見つかった
- 実体は計算処理ではなく論理処理である

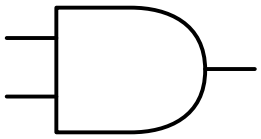
  二値の計算を二値論理によって実現する
- 論理関数は回路によって容易に実現できる

  計算する機械（ハードウェア）ができた

復習

- 各論理回路（ゲート）の入出力表を埋めてみよ

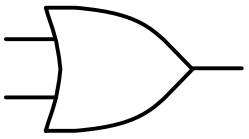
ゲートの
シンボル



AND

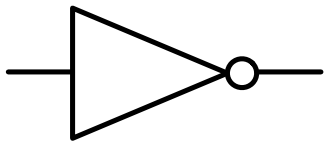
入出力表

入力1	入力2	出力
1	1	
0	1	
1	0	
0	0	



OR

入力1	入力2	出力
1	1	
0	1	
1	0	
0	0	



NOT

入力	出力
1	
0	

半加算回路

- 先の加算回路は完全ではない

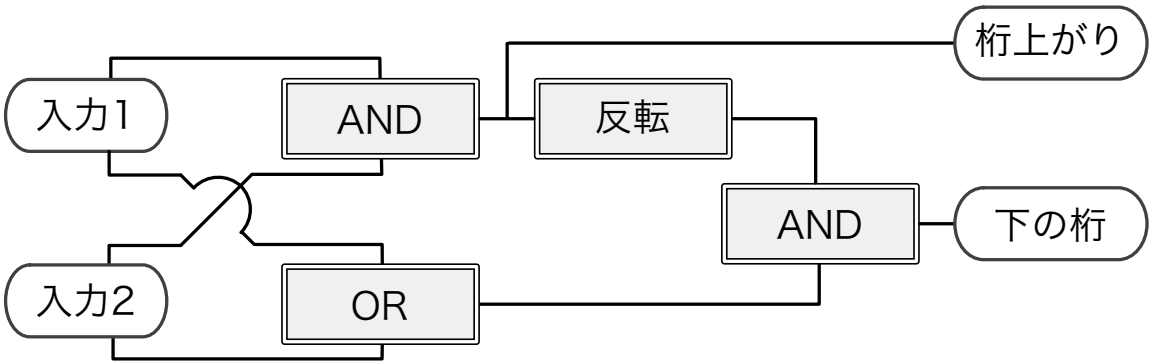
桁上がりを出力するが入力にそれがない

	桁上がり	下の桁
0+0=	0	0
1+0=	0	1
0+1=	0	1
1+1=	1	0

AND　反転　AND

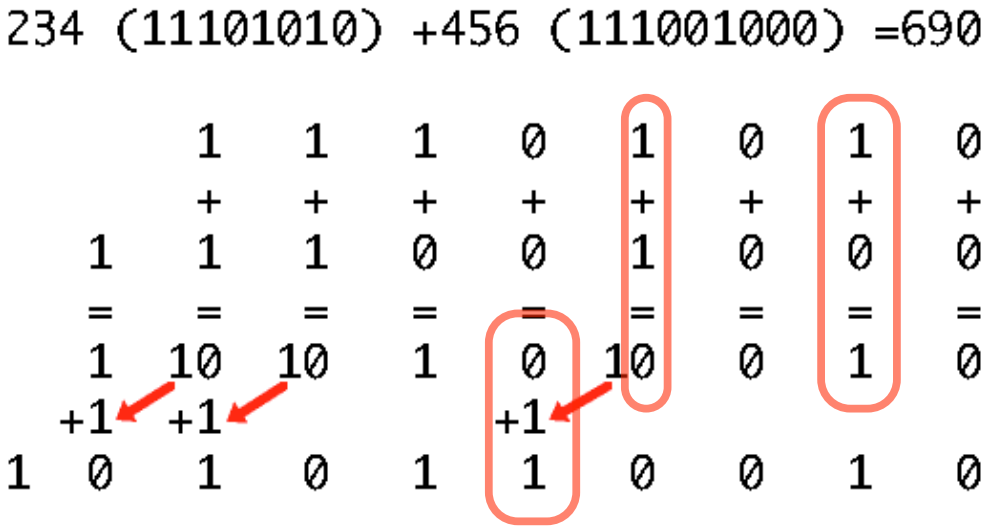
入力1	入力2	AND	反転	OR	下の桁
OFF	OFF	OFF	**ON**	OFF	OFF
ON	OFF	OFF	**ON**	**ON**	**ON**
OFF	**ON**	OFF	**ON**	**ON**	**ON**
ON	**ON**	**ON**	OFF	**ON**	OFF

OR

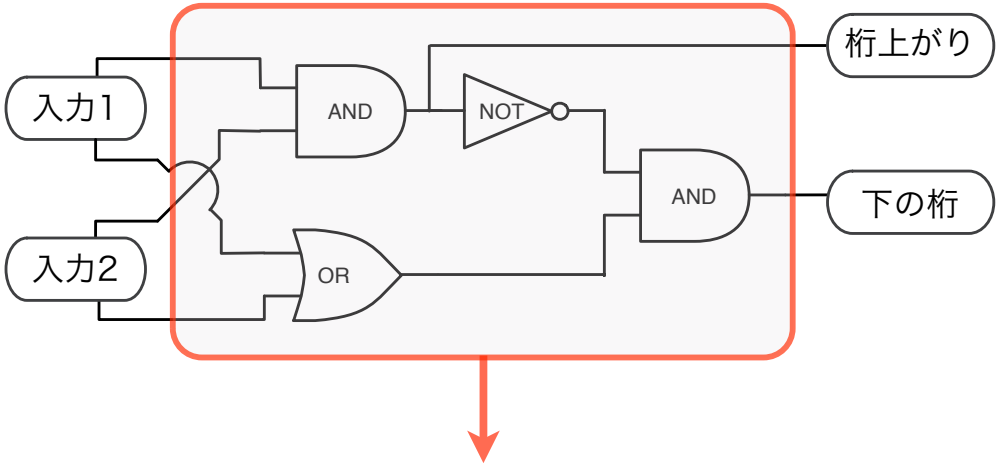


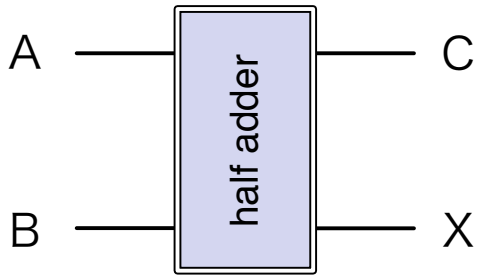
半加算回路

- 完全な加算には一桁について二回の加算処理が必要



全加算回路

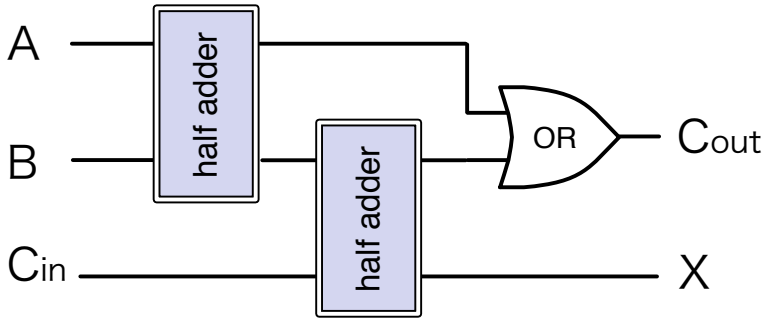




半加算器をこのように抽象化する

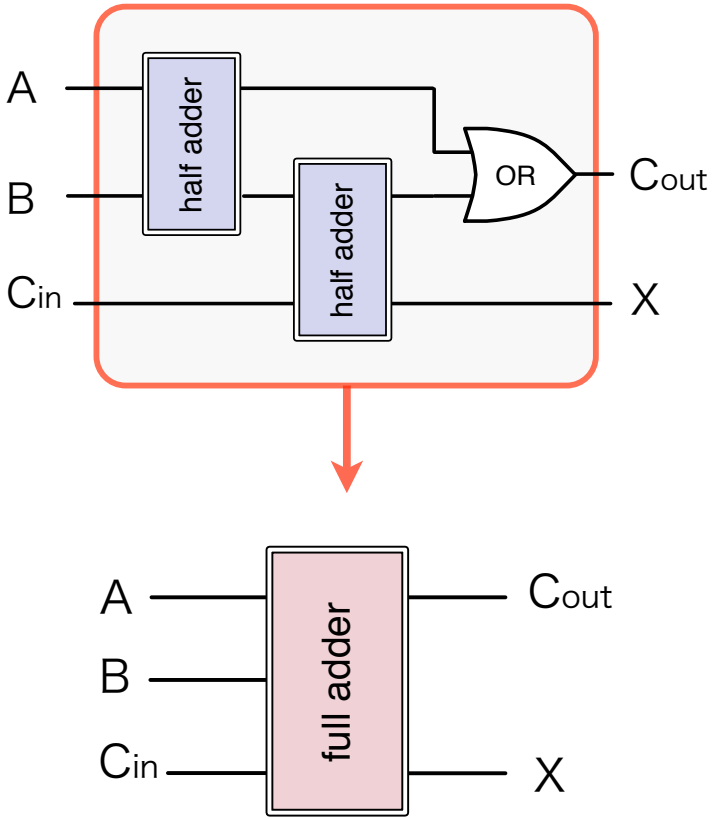
```
234 (11101010) +456 (111001000) =690

      1   1   1   0   1   0   1   0
      +   +   +   +   +   +   +   +
  1   1   1   0   0   1   0   0   0
  =   =   =   =   =   =   =   =   =
  1  10  10   1   0  10   0   1   0
 +1  +1          +1
1  0   1   0   1   1   0   0   1   0
```

まず A, B を加算し、その下の桁と前の桁からの桁上がりを加算する。

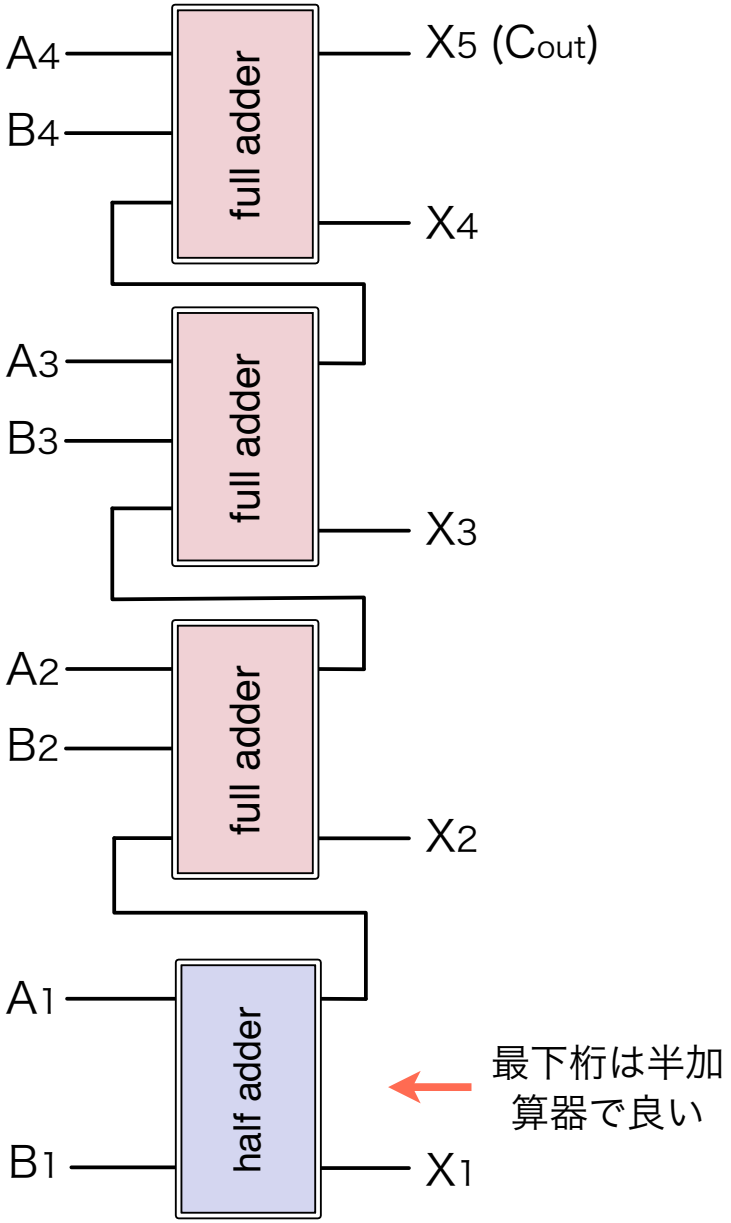


前半の加算と、後半の加算両方で桁上がりが生じることはない。

全加算回路



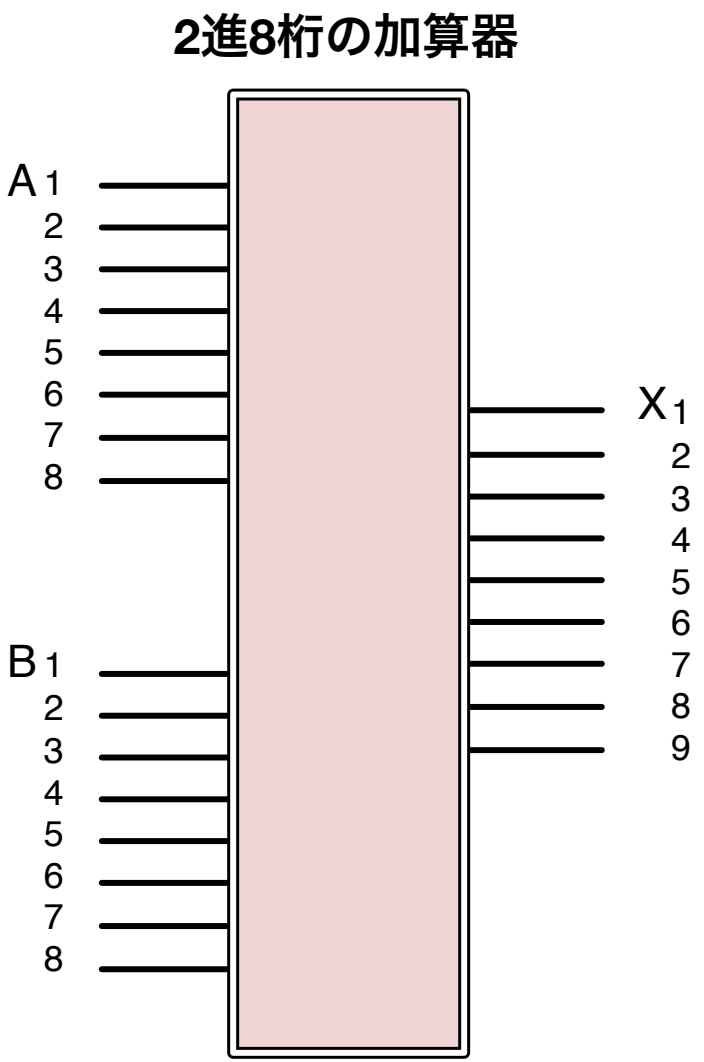
全加算器をこのよ
うに抽象化する



4桁の加算器を1桁の
加算器で構成する

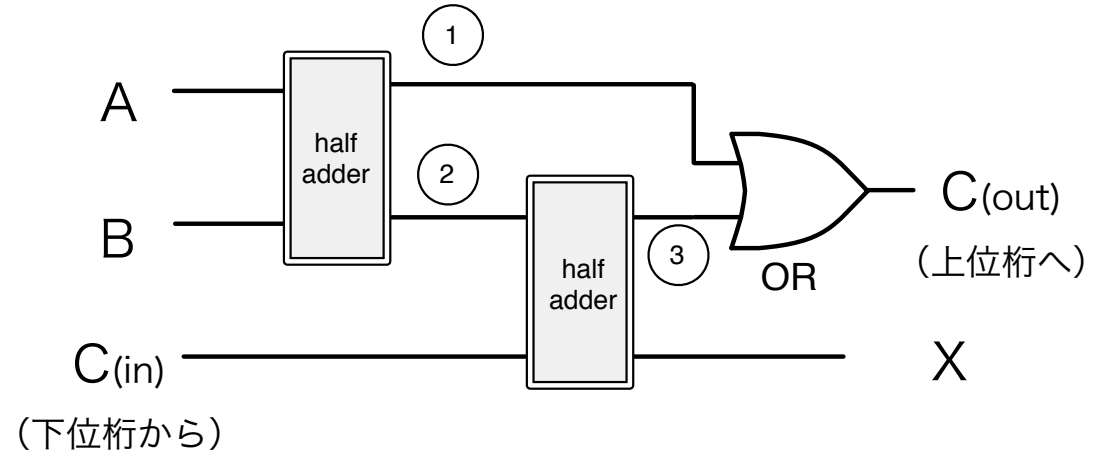
まとめ

- 2進8桁の加算器
- 全加算器7＋半加算器1で実現可能
- 全加算器は半加算器 2 + OR
- 半加算器は AND x 2, OR, NOT
- AND, OR はリレー二つ
  NOT は一つで実現できる
- この構造で作れる、と確信
  がもてましたか？

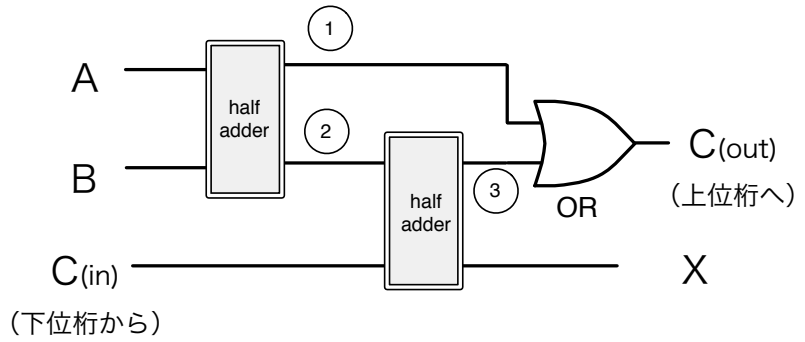


課題

- 設問：下記の入出力表の空白の部分を埋めよ。



A	B	C (in)	1	2	3	C (out)	X
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0					
1	1	0					
0	0	1					
0	1	1					
1	0	1					
1	1	1	1	0	0	1	1



命題：全加算器の入出力表

A	B	C (in)	1	2	3	C (out)	X
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0					
1	1	0					
0	0	1					
0	1	1					
1	0	1					
1	1	1	1	0	0	1	1

参考：半加算器の入出力表

入力1	入力2	桁上がり	下の桁
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

1 の出力はA,B を入力とする半加算器の桁上がりなので、そのまま半加算器の入出力表を見ながらあてはめる

桁上がり

A	B	1
0	0	0
0	1	0
1	0	??
1	1	
0	0	
0	1	
1	0	
1	1	1

桁上がり

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