機械知能工学科
メカトロニクス総合

MC-11/Rev 16-1.0

第11回

デジタル回路の実現

工学部 機械知能工学科
熊 谷 正 朗
kumagai@mail.tohoku-gakuin.ac.jp
東北学院大学工学部
ロボット開発工学研究室 RDE

今回の到達目標

○ロジックゲートによるデジタル回路

◇実体としてのデジタル回路を説明できる。

- デジタル回路の電圧信号
- CMOSとTTL

◇ロジックゲートを説明できる。

- AND，OR，NOT，XOR，NAND，NOR，XNOR
- ゲートの記号

◇半加算回路・1ビットの乗算回路を説明できる。

- 真理値表、ゲートによる実装

デジタル回路

○動作にかかわる電圧

◇一般的なデジタル回路

- 単一の電源(5V，3.3V，数Vの正電源等)
- 電圧の高低で01を表現

◇CMOS（Complementary MOS)型　主流

- 電源：5，3.3 他 2～6，3～18などあり
- ″0″：ほぼ0[V]　″1″：ほぼ電源電圧

◇TTL（Transistor-Transistor Logic）前の主流

- 電源：5のみ ″0″：ほほ0[V] ″1″：2.6～5[V]

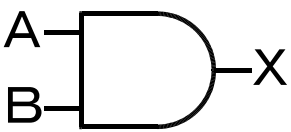
ロジックゲート

○ロジックゲート(論理ゲート)、汎用ロジックIC

※74シリーズ

◇基本的な論理演算を行う部品

- AND, OR, NOT, XOR, NAND, NOR, XNOR
- 例）2入力ANDゲート:



入力:A,B　出力:X　X ＝ A AND B

- デジタルはすべてAND OR NOTで構成可

→これらの部品の組み合わせで回路作れる

◇まとまった機能を持った部品

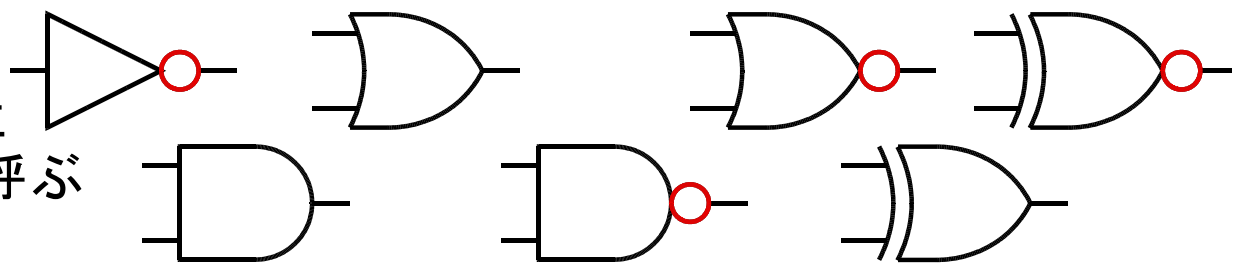
- 加算回路、フリップフロップ、デコーダなど

ロジックゲート

○基本的な論理演算を行う部品

◇真理値表と記号　一覧

A	B	NOT	AND	OR	NAND	NOR	XOR	XNOR
0	0	1	0	0	1	1	0	1
0	1		0	1	1	0	1	0
1	0	0	0	1	1	0	1	0
1	1		1	1	0	0	0	1



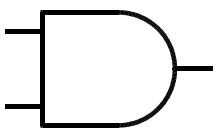
※NOTは一般に
インバータと呼ぶ

※JISでは新しい記号が制定されているが、これらが現役

ロジックゲート

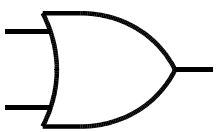
○基本の３種類

◇基本のブール代数演算に対応するゲート



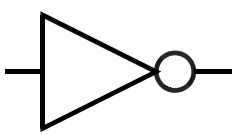
・ANDゲート：　論理積

（2本以上の入力）入力が全て1なら、出力1



・ORゲート：　論理和

（同）　入力が1本でも1なら、出力1



・インバータ：　否定

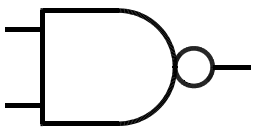
（NOTゲート）入力1なら出力0、 0なら1

◇この３種だけでも任意の回路を作れる

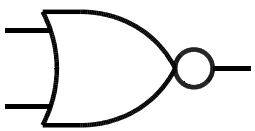
ロジックゲート

○追加の４種類のゲート

◇回路設計でこれらも標準的に用いられる



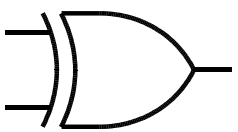
・NANDゲート　＝　NOT　AND　　（なんど）



・NORゲート　＝　NOT　OR　　（のあ）

AND　OR　の出力をNOT(01反転)

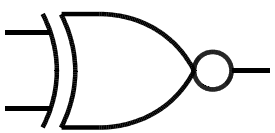
○：NOT ↑



・XORゲート　排他的論理和（Exclusive　OR）

（２入力のみ）　入力が(0,1)(1,0)のとき1

（えくすおあ）　※ORで(1,1)を0にしたもの

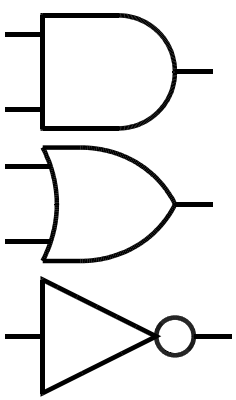


・XNORゲート　＝　NOT　XOR　　（えくすのあ）

ロジックゲート

○その他の補足

◇ロジックゲート記号の使い道



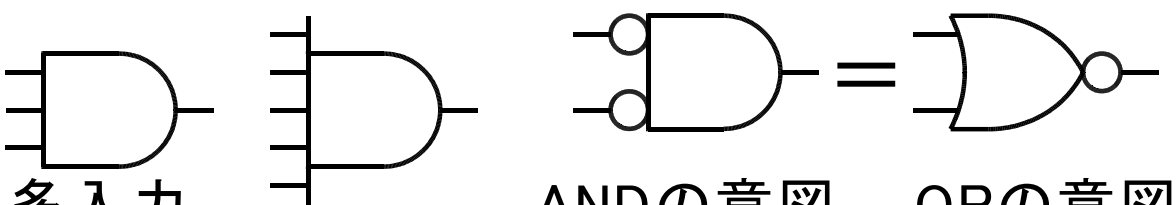
(1) デジタル回路の記述

ゲートをつないで回路を構成する

(2) 信号の処理の仕方を示す概念図

例）スイッチAとスイッチBがともにオンなら

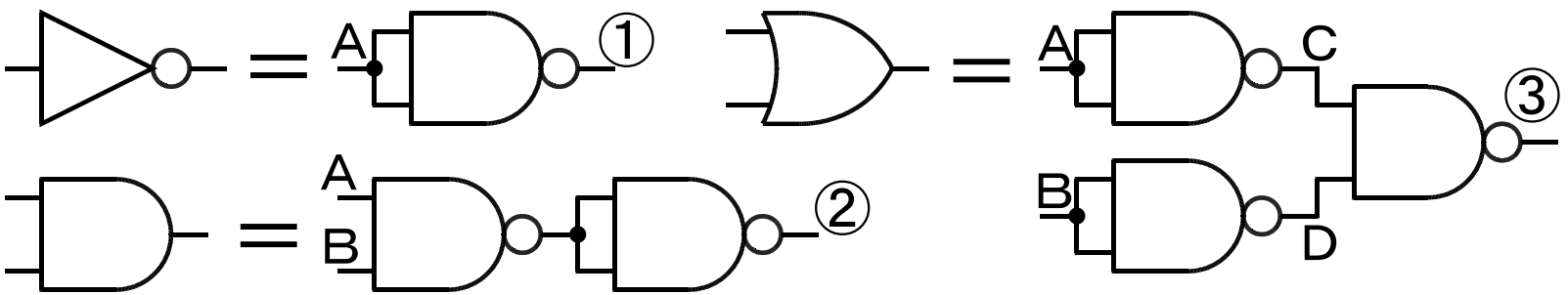
◇ゲート記号のバリエーション ※ド・モルガンの法則



ロジックゲート

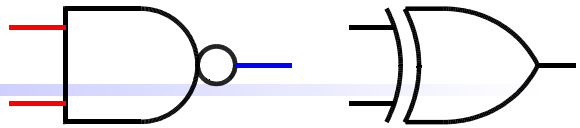
○最強のゲート NAND

◇NANDがあれば全ての回路を作れる



A	B	NAND	①	NOT	②	AND	C	D	③	OR
0	0	1	1	1	0	0	1	1	0	0
0	1	1	1		0	0	1	0	1	1
1	0	1	1	0	0	0	0	1	1	1
1	1	0	0		1	1	0	0	1	1

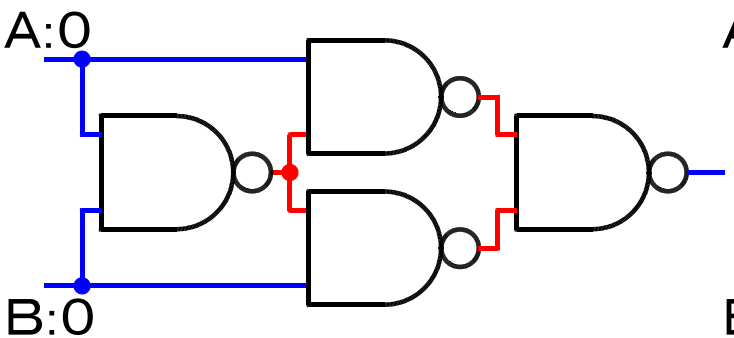
ロジックゲート

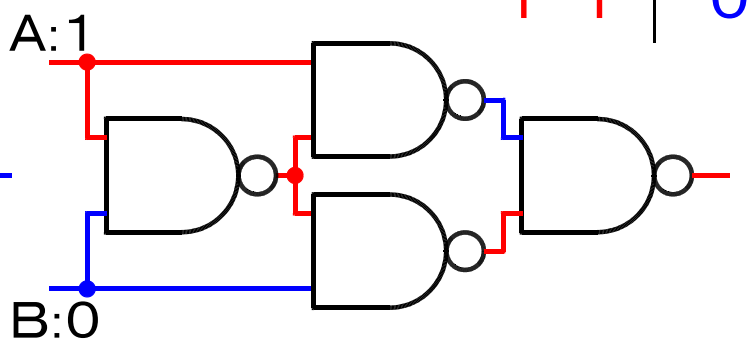


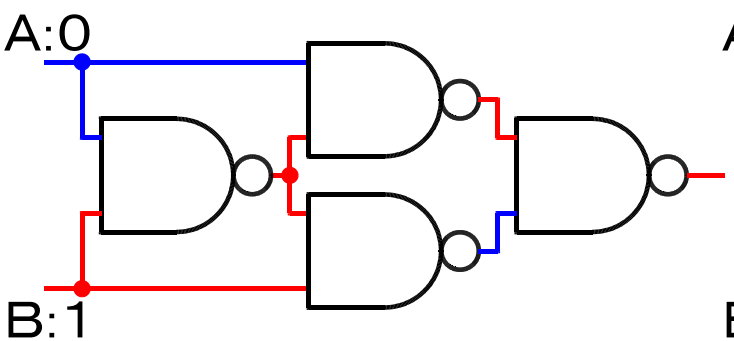
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

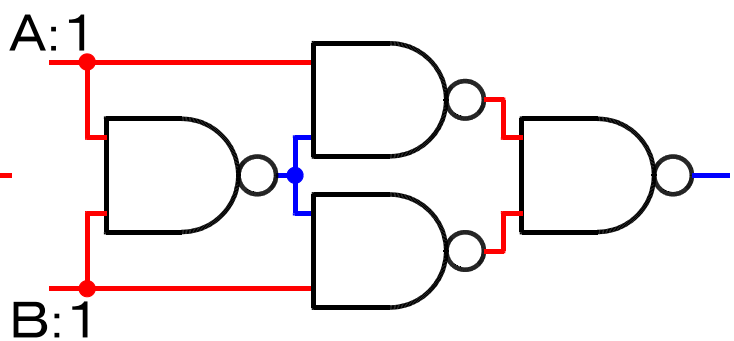
○最強のゲート NAND→XOR

◇NANDがあれば全ての回路









組み合わせ回路

○複数の入力だけに依存して出力が決まる

◇ゲート単体

◇″NANDで全て″のような例

・入力から出力にのみ流れる

・あるゲートの出力が、自身の上流に戻らない

◇組み合わせ回路の例

・(非同期の)演算回路、加算回路など

・デコーダ

例)2進の入力に対して、パターン出力

″7セグ″

組み合わせ回路

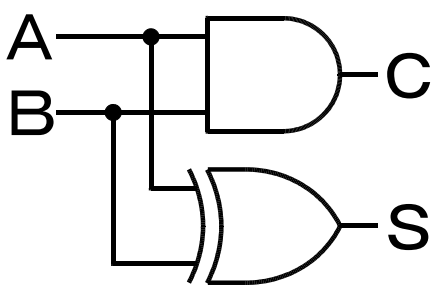
○加算回路（半加算器；ハーフアダー）

◇目的の回路を作る

（1）動作を明記する：真理値表

（2）ゲートを組み合わせて実現

目的：加算	A	B	C	S	AND	XOR
0＋0＝ 0	0	0	0	0	0	0
0＋1＝ 1	0	1	0	1	0	1
1＋0＝ 1	1	0	0	1	0	1
1＋1＝10	1	1	1	0	1	0



A＋B→CS

C：Carry：繰上

S：Sum：和

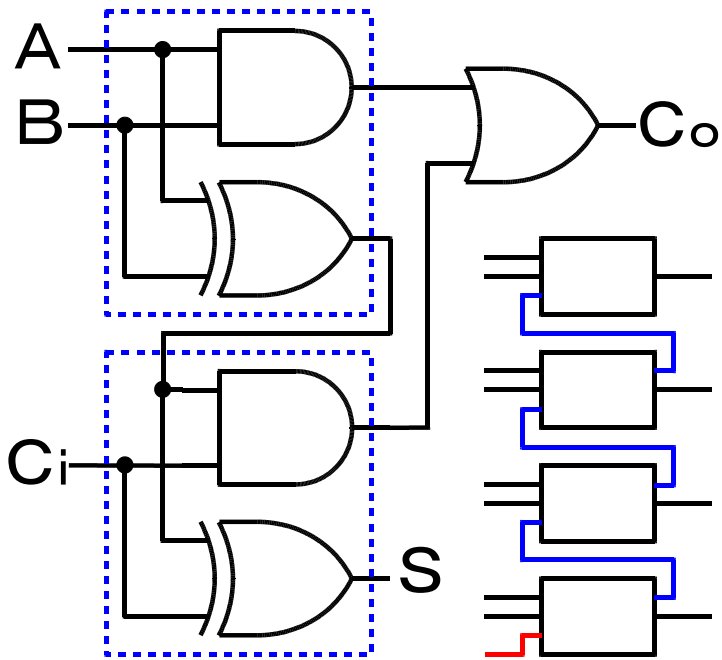
では、乗算は？

組み合わせ回路

○加算回路(全加算器;フルアダー)

◇繰り上げ入力付き

目的:加算	A	B	Ci	Co	S
0+0+0= 0	0	0	0	0	0
0+0+1= 1	0	0	1	0	1
0+1+0= 1	0	1	0	0	1
0+1+1=10	0	1	1	1	0
1+0+0= 1	1	0	0	0	1
1+0+1=10	1	0	1	1	0
1+1+0=10	1	1	0	1	0
1+1+1=11	1	1	1	1	1



Ci:繰上入力
S:和　Co:繰上出力

順序回路

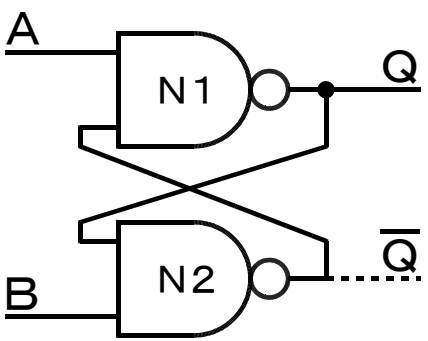
○過去の入力、出力にも依存する

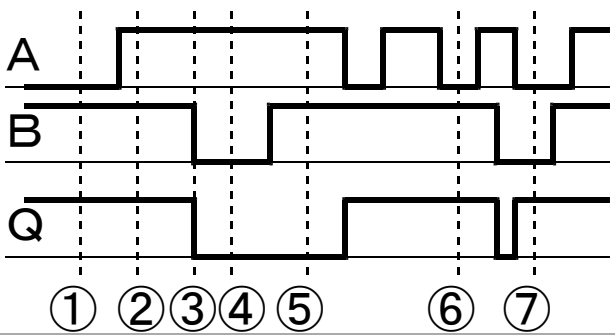
◇フリップフロップ（FF）

・RSFF、DFF、TFF、JKFF

◇RSFF（リセットーセット）

・２本の入力のうち、直近に0だったほうを覚えていて、Qに出力。





A＝0にする

→Q＝1（セット）

B＝0にする

→Q＝0

（リセット）

順序回路

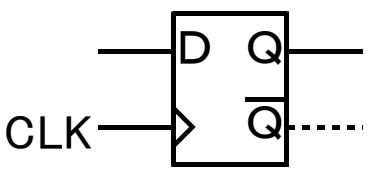
○DFF

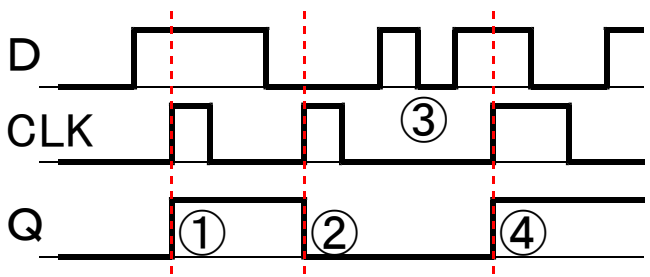
↓"立ち上がり"

◇CLK入力ののO→1のタイミングで、D入力を記憶し、それをQから出力する

◇同期式回路:

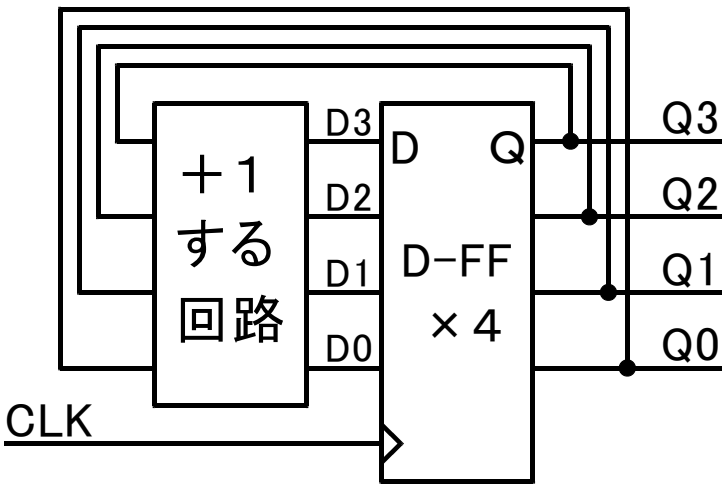
・回路内の各所にDFFを入れ、同一CLKで同時に値を固定する → 安定化、高速化

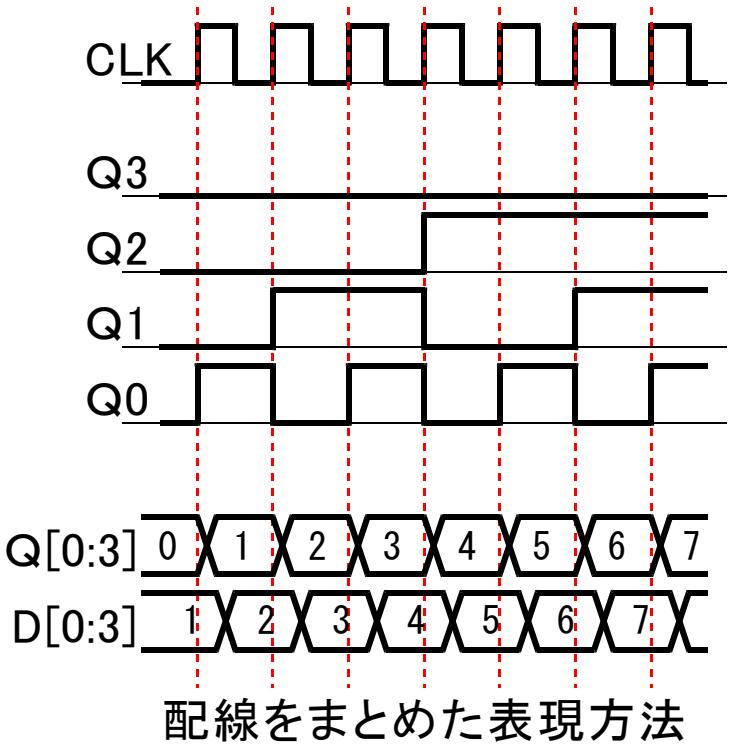




順序回路

○同期式カウンタ





配線をまとめた表現方法

CLK↑のたびに、「現在の値＋1」をD-FFが記憶し直す。

→CLKのたびに1ずつ増える計数

「＋1」する回路の工夫

→[カウントする/しない][up/down][初期設定]等可。