電気学第一 (b) S222 講義室, 2012 June 29, July 6

高橋篤司 (S3-410, ext.2665, atsushi@lab.ss.titech.ac.jp)

1 トランジスタスイッチ

(a) 高電圧”1” でオン　(b) 低電圧”0” でオフ

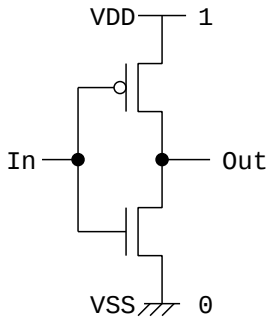
図 1: npn(nMOS) トランジスタのスイッチ

(a) 低電圧”0” でオン　(b) 高電圧”1” でオフ

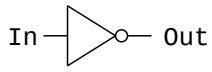
図 2: pnp(pMOS) トランジスタのスイッチ

2 論理基本ゲートとその動作

2.1 否定 (NOT) 論理回路 ($\mathbf{Out} = \overline{\mathbf{In}}$)



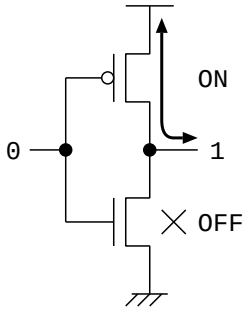
(a) NOT 論理回路



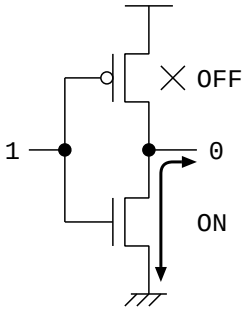
(b) NOT 論理記号

In	Out
0	1
1	0

(c) NOT 真理値表



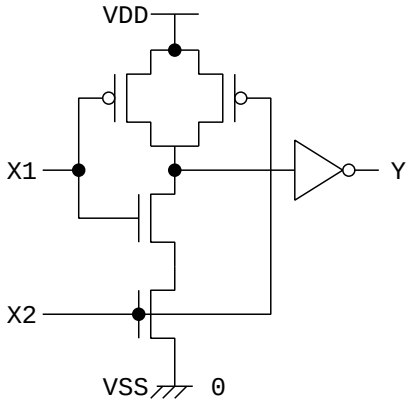
(d) 論理 0 入力



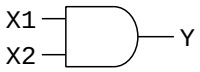
(e) 論理 1 入力

図 3: NOT ゲート

2.2 論理積 (AND) 論理回路 ($\mathbf{Y} = \mathbf{X1} \cdot \mathbf{X2}$)



(a) AND 論理回路



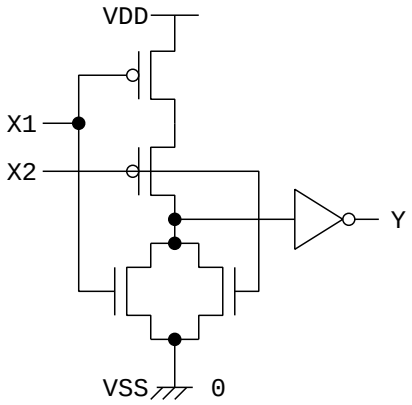
(b) AND 論理記号

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

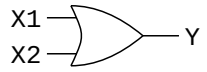
(c) AND 真理値表

図 4: AND ゲート

2.3 論理和 (OR) 論理回路 ($\mathbf{Y} = \mathbf{X1} + \mathbf{X2}$)



(a) OR 論理回路



(b) OR 論理記号

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

(3) OR 真理値表

図 5: OR ゲート

3　論理回路と論理関数

論理関数 $Y = f(X1, X2, X3)$ は，入力の論理変数の値により，論理関数値 0 または 1 の値をとる．論理変数 $X1, X2, X3$ は，0 または 1 の値をとる．

論理関数は，論理回路で実現される．図 6(a) の論理回路が実現する論理関数は，$Y = \overline{X1 \cdot X2 + X3}$ である．

論理関数は，真理値表で表現することができる．真理値表は，すべての入力変数の組合わせに対して論理関数値を示した表である．図 6(a) の論理回路の真理値表を，図 6(b) に示す．

論理関数が与えられたとき，その表現方法，論理回路による実現方法は，例えば，以下のような等式が成り立つように様々に存在する．

$0 + X = X, \quad 1 + X = 1, \quad 0 \cdot X = 0, \quad 1 \cdot X = X$

$X \cdot X = X, \quad \overline{\overline{X}} = X$

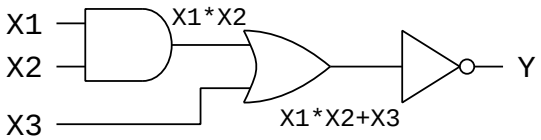
$X1 \cdot X2 = \overline{\overline{X1} + \overline{X2}}$

$X1 + X2 = \overline{\overline{X1} \cdot \overline{X2}}$

そこで，すべての入力変数の肯定形または否定形の論理積からなる項を論理和で結んだ論理和標準形で論理関数を表現することがある．図 6(a) の論理回路の論理和標準形は

$$Y = \overline{X1} \cdot \overline{X2} \cdot \overline{X3} + \overline{X1} \cdot X2 \cdot \overline{X3} + X1 \cdot \overline{X2} \cdot \overline{X3}$$

となる．論理和標準形を元に論理回路を構成することもできるが回路規模が大きくなる．論理関数を回路で実現する場合には，回路規模が小さくなるよう論理関数の表現方法を工夫する．



(a) 論理回路

X1	X2	X3	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

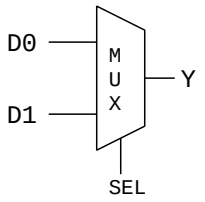
(b) 真理値表

図 6: 論理回路の例

4　組合わせ論理回路

4.1　マルチプレクサ

2 つの信号 $D0$ と $D1$ の一方を出力する回路で，選択信号によって選択を決定する．



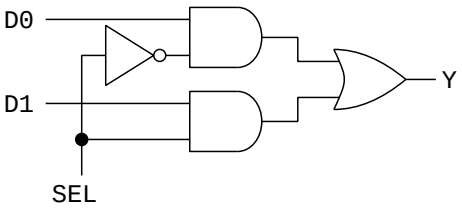
(a) 記号

SEL	Y
0	D0
1	D1

(b) 簡易真理値表

SEL	D0	D1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(c) 真理値表

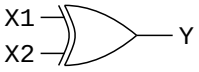


(d) 回路

図 7: マルチプレクサ回路

4.2　排他的論理和 (Exclusive OR)

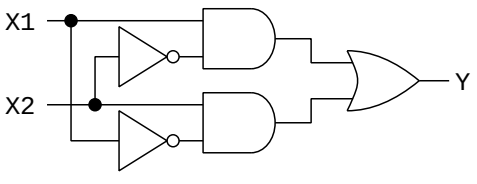
($\mathbf{Y = X1 \oplus X2}$)



(a) 記号

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

(b) 真理値表



(c) 回路

図 8: 排他的論理和 (Exclusive OR) 回路

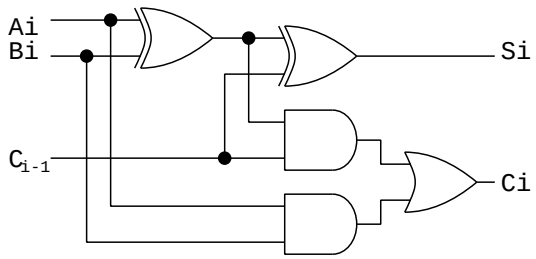
4.3 加算器

半加算器は，二つの 1 ビット入力に対して，和のビットと繰り上がりのビットを出力する．全加算器は，入力ビットの他に下位の桁からの繰り上がりのビットをも入力として用いて，その桁の和のビットと上位の桁への繰り上がりのビットを出力する．

図 9 に 1 ビットの全加算器の真理値表と回路を示す．

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) 真理値表



(b) 回路

図 9: 1 ビットの全加算器

1 ビットの全加算器を組み合わせることで多ビットの全加算器が構成できる．図 10 に 3 ビットの全加算器の構成例を示す．

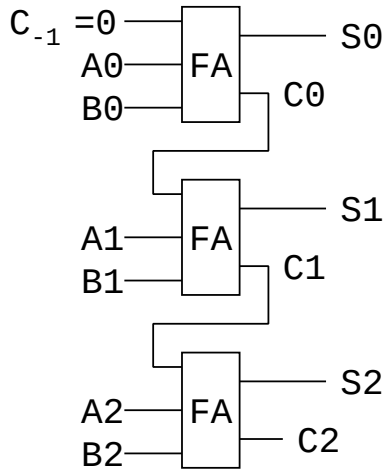


図 10: 3 ビットの全加算器

5 順序回路

組合わせ回路は，出力が入力のみに依存する論理回路である．一方，順序回路は，出力が入力だけでなく内部の状態に依存する．内部の状態は，前の状態および入力によって変化する．順序回路では，内部の状態を記憶するためにメモリ機能が必要となる．

5.1 メモリ機能

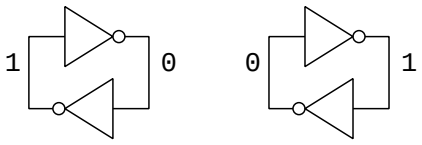
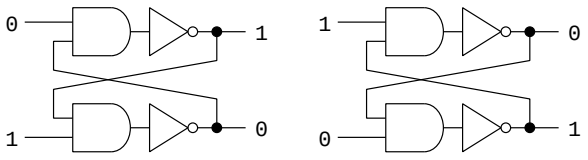
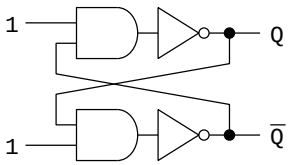


図 11: メモリセルの構造と動作

5.2 フリップフロップの基本回路



(a) データ 1 の書込み　(b) データ 0 の書込み



(c) データ Q の保持

図 12: フリップフロップ

5.3 実際のフリップフロップ

実際に用いられるフリップフロップは，入力の変化をクロックに同期させて取り込んだり，入力の取り込みタイミングを制御するための機構がついていたり，発振などの誤動作を防ぐために，メモリ機能を 2 段構成にして，入力の変化が出力にすぐに反映しないようすることが多い．図 13 に，クロック (clock) が 1 でかつロード (load) が 1 の場合に値を取り込むフリップフロップを，図 14 に，メモリ機能が 2 段構成のマスタスレーブ型 D フリップフロップのタイミングチャートを示す．

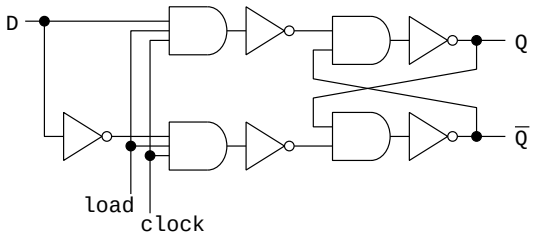
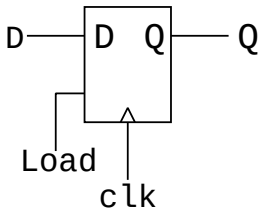
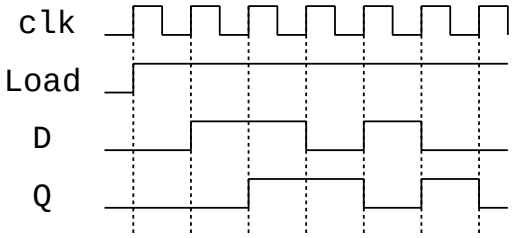


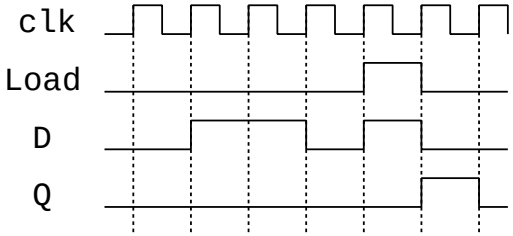
図 13: ロード，クロック付きフリップフロップ



(a) 記号



(a) 入力信号の遅延



(a) ロード信号による制御

図 14: マスタスレーブ型 D フリップフロップの動作

5.4 メモリ

1 個のフリップフロップは 1 ビットのデータを，n 個のフリップフロップで n ビットのデータを記憶することができる．メモリは，データの記憶のためにフリップフロップを規則正しくアレイ状に並べた構造をしている．アドレスを指定することで，対応するフリップフロップにデータを書き込んだり，フリップフロップから値を読み出す．

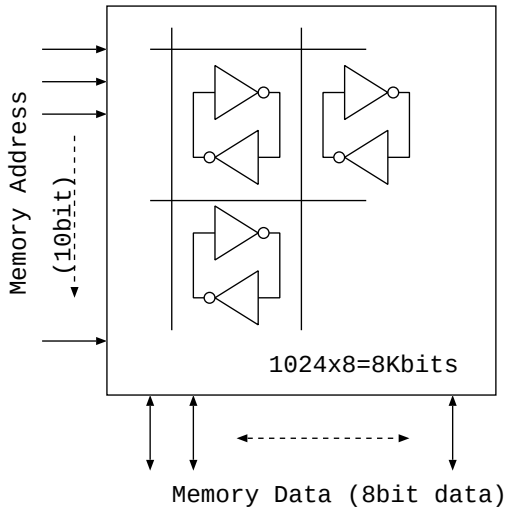


図 15: メモリアレイ

5.5 カウンタ

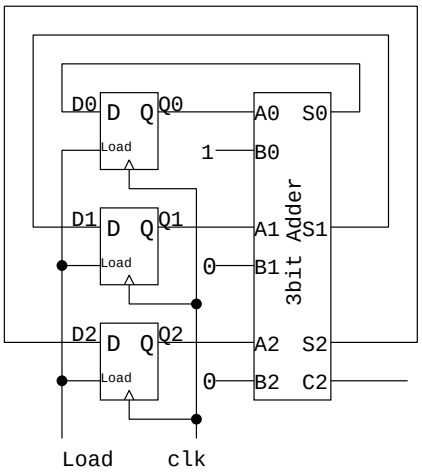
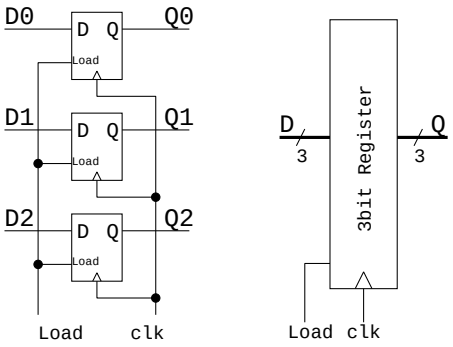


図 16: 3 ビットカウンタ

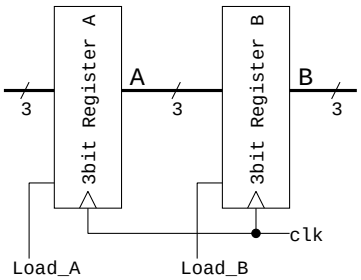
5.6 レジスタ



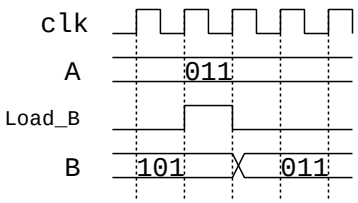
(a) 3 ビットレジスタ　　(b) 記号

図 17: 3 ビットレジスタ

5.7 レジスタ間のデータ転送

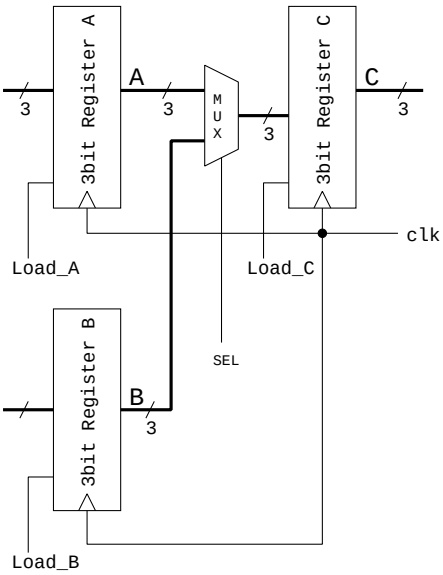


(a) A から B への転送： $B \leftarrow A$

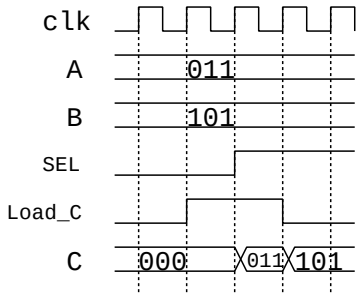


(b) タイミングチャート

図 18: レジスタ間のデータ転送



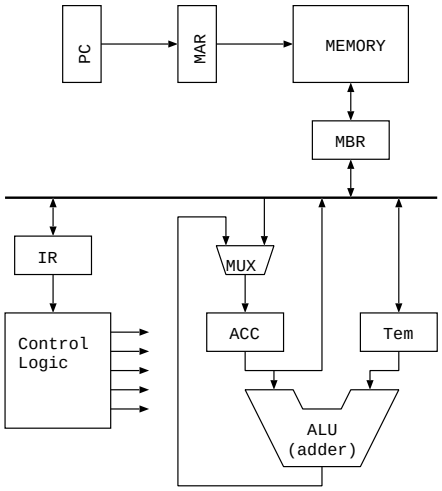
(a) A,B から C への転送：$\begin{cases} C \leftarrow A \text{ when } SEL = 0 \\ C \leftarrow B \text{ when } SEL = 1 \end{cases}$



(b) タイミングチャート

図 19: 複数のレジスタからのデータ転送

6 マイクロプロセッサ



MAR: Memory Address Register
MBR: Memory Buffer Register
PC: Program Counter Register
IR: Instruction Register
ACC: Accumulator Register
Tem: Temporary Register

図 20: プロセッサ

アセンブラ	コード	機能	説明
LDA addr	00 xxx xxx	ACC ← Mem[addr]	データの読み出し
STA addr	01 xxx xxx	Mem[addr] ← ACC	データの保存
ADD #reg	10 000 xxx	ACC ← ACC + reg(xxx)	算術加算
SUB #reg	10 001 xxx	ACC ← ACC − reg(xxx)	算術減算
BRA addr	11 xxx xxx	PC ← addr	無条件分岐

図 21: 8 ビットマイクロプロセッサの機械語命令の例

時刻	データ転送	説明
0	MAR ← PC	実行命令格納番地をメモリアドレスレジスタにセット
1	MBR ← Mem[MAR]	メモリから実行命令を読み出す
	PC ← PC+1	次の命令番地を現在の次の番地に
2	IR ← MBR	実行命令を命令レジスタに転送
3	ACC ← ACC+reg(xxx)	命令レジスタで解読に基づき命令を実行．次の命令の読み出しに戻る

図 22: 機械語命令の読み出し実行例 (ADD xxx)