ハードウェア構成法実験
第3回

2003/04/21

担当　千本　潤介
bonse@is.s.u-tokyo.ac.jp

内容

1. カルノー図
2. 最適AND-OR式の導出
3. QM法

1. カルノー図

カルノー図

- 論理式の入力と出力の関係を視覚化した図
- 各入力値組合せに対応した四角を並べる
- 出力が1になる入力組合せを塗りつぶす
- ハミング距離が1の入力組合せ同士は原則として隣接させる
- 図の上下辺同士と左右辺同士は連結されているものとみなす(トーラス構造)

カルノー図(2入力)

- 論理関数 IMPLY(X,Y) = -X + Y の例

X	Y	出力
0	0	1
0	1	1
1	0	0
1	1	1

真理値表

	X	
Y	11	01
	10	00

カルノー図

カルノー図(3入力)

- 論理関数 SEL(A,B,S) = A(-S) + BS の例

A	B	S	出力	A	B	S	出力
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	1

真理値表

A

S	101	111	011	001
	100	110	010	000

B

カルノー図

カルノー図(4入力)

- 論理関数 XOR(A,B,C,D)の例

A

C

1010	1110	0110	0010
1011	1111	0111	0011
1001	1101	0101	0001
1000	1100	0100	0000

D

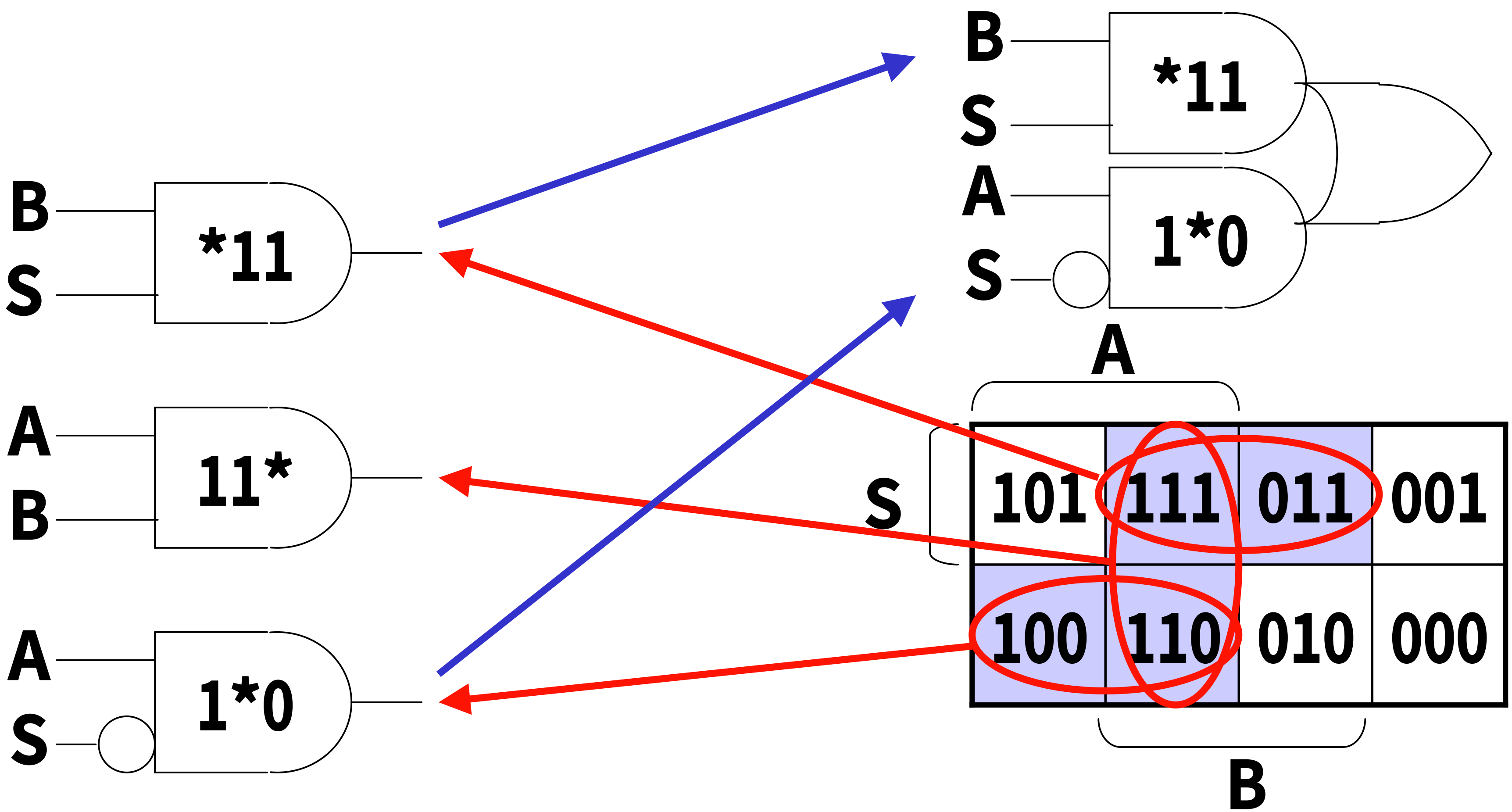
B

カルノー図による論理式簡略化

- カルノー図を用いて与えられた論理式をAND-OR形式の範囲内で単純化可能
- カルノー図で出力が1になる部分からAND項を見つける
  - AND項はカルノー図上では四角形領域となる
- 出力全体を網羅するAND項セットの中で最もコストが小さいものを選ぶ
- 選んだAND項を全てOR

論理式簡略化例

- カルノー図から発見したA(-S)、AB、BSの3つのAND項から、A(-S)とBSを選択してOR



備考

- 5入力以上のカルノー図はあまり使われない
  - 一覧性が悪くなるため視覚化のメリットが薄くなる
  - ハミング距離が1の入力組合せ同士を全て隣接させる事が不可能
- カルノー図で出力が0になる部分に着目すればOR-AND形式の論理式を求める事が可能

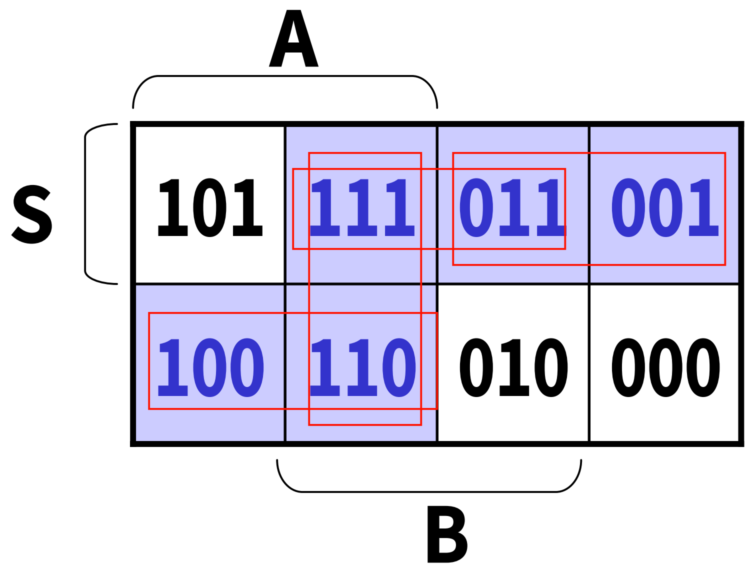
2．最適AND-OR式の導出

最適AND-OR式の求め方(1/2)

- カルノー図で1になる部分に含まれるAND項を全て列挙
- 列挙したAND項集合の部分集合を全探索
  - 出力を全て覆う部分集合の中で最もコストが低いものを選択
- 選ばれたAND項をORすれば完成

最適AND-OR式の求め方(2/2)

- 右下の例では、ABS、AB(-S)、A(-B)(-S)、(-A)BS、(-A)(-B)S、AB、A(-S)、BS、(-A)Sの9個のAND項が存在
- 計2^9=512通りの部分集合を試す
  - φ, { ABS }, { AB(-S) },
    { ABS, AB(-S) },
    { A(-B)(-S) }, . . .

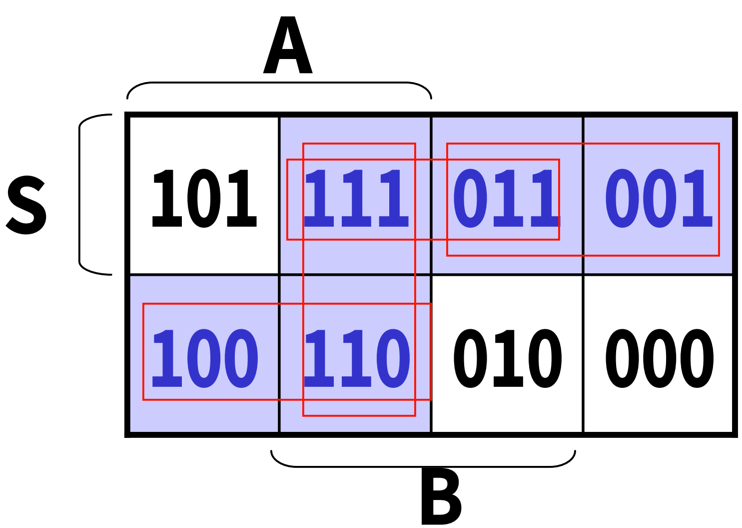


探索の効率化(1/2)

- 列挙したAND項の中に、カルノー図上で他のAND項に含まれるものが存在

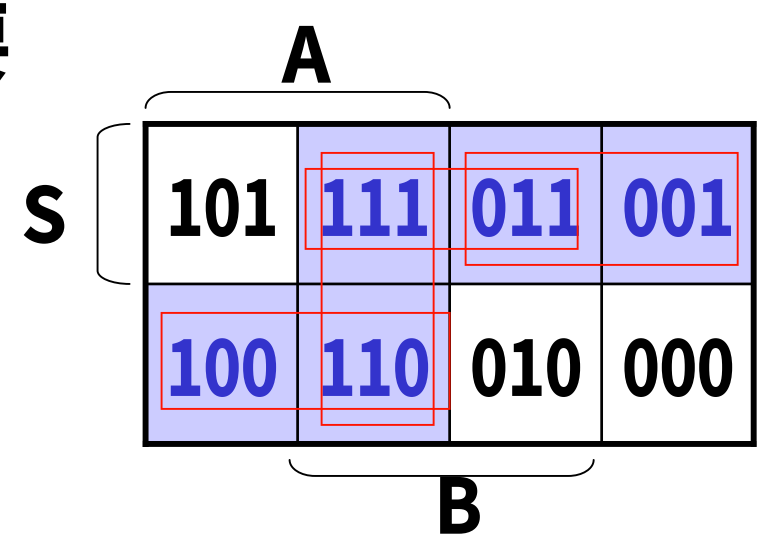
  AB(-S) ∈AB、A(-B)(-S) ∈A(-S)...

- このようなAND項は探索の対象外にして良い
  - この項を包含するAND項を使った方がコストが小



探索の効率化(2/2)

- 出力が1になる入力組合せの中に、一つのAND項にしか覆われていない物が存在
  - 下例ではA(-S)と(-A)S (前ページ手法よりAND項の組が A(-S), AB, BS, (-A)S になっている場合)
- この様な項は絶対必要なので、この項を含まない部分集合は探索不要



3．QM法

QM法

- 与えられた真理値表を満たすAND-OR式の中で最もコストが少ない物を求める手法
- 探索の無駄を減らす工夫を施してある

記法

- 以下ではAND項を、その値を1にする入力値組合せで表す
  - 例えば、A(-B)CD は 1011 と表す
- AND項に含まれない変数の部分は'*'で表す
  - 例えば、変数が A,B,C,D 4つの時、(-A)Dは 0**1 と表す

最小項の列挙(1/2)

- カルノー図で1になる部分に含まれるAND項のうち、全ての変数を含む物を**最小項**と言う
  - 主加法標準形の各AND項と同じ
- QM法では与えられた論理式の最小項を始めに全て列挙する

最小項の列挙(2/2)

- 下例では最小項は 1111, 1101, 1100, 1011, 1000, 0111, ...

	A				
C	1010	1110	0110	0010	
	1011	1111	0111	0011	D
	1001	1101	0101	0001	
	1000	1100	0100	0000	
		B			

最小項のグループ分け

- 最小項を、項に含まれる‘1’の個数毎に分ける

0個	無し
1個	0100, 1000
2個	0011, 1100
3個	0111, 1011, 1101
4個	1111

A

C

1010	1110	0110	0010
1011	1111	0111	0011
1001	1101	0101	0001
1000	1100	0100	0000

D

B

項の併合

- 恒等式 XY + X(-Y) = X を用いて項を併合
- ‘1’の個数が1異なる項同士のみ試せば良い

0個	--
1個	0100, 1000
2個	0011, 1100
3個	0111, 1011, 1101
4個	1111

--
1*00, *100
0*11, *011, 110*
*111, 1*11, 11*1

--
--
**11

--
--

--

併合に使われた項の削除

- 併合に使われた項を削除
  - この項を包含するAND項が存在するから
- 生き残った項を**主項**と言う

0個	--
1個	~~0100~~, ~~1000~~
2個	~~0011~~, ~~1100~~
3個	~~0111~~, ~~1011~~, ~~1101~~
4個	~~1111~~

--
1*00, *100
~~0*11~~, ~~*011~~, 110*
~~*111~~, ~~1*11~~, 11*1

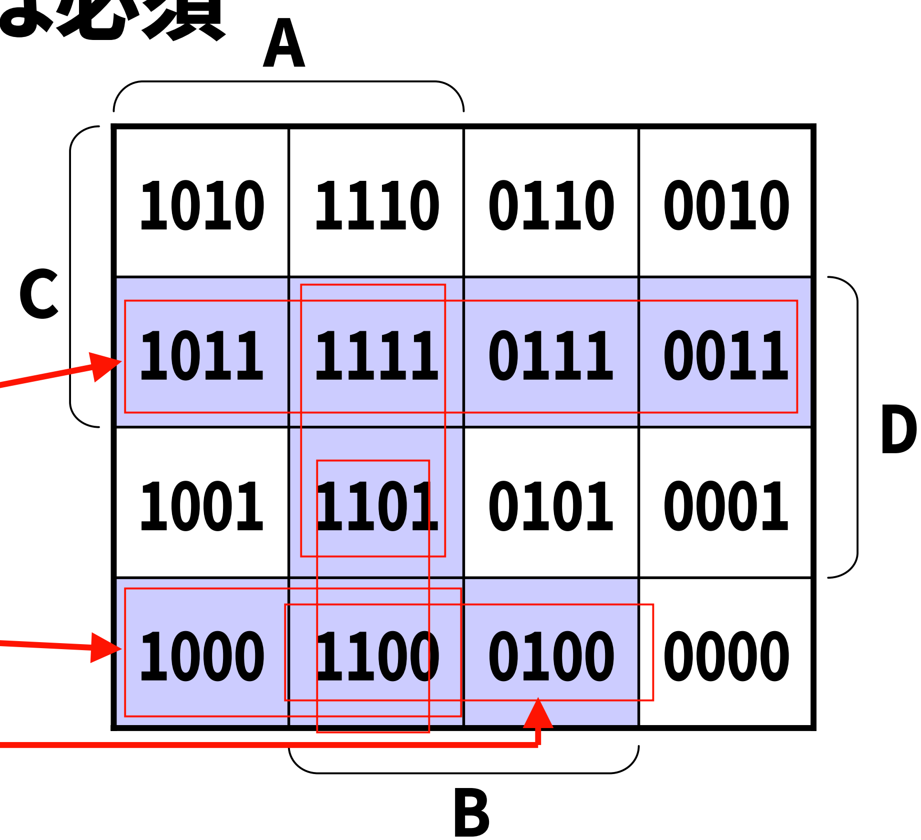
--
--
**11

--
--

--

必須項の特定

- 一つの主項にしか覆われていない箇所がある場合、その主項は必須
- そのような主項を**必須項**という
- 例では
  **11,
  1*00,
  *100



最小コスト組合せの探索(1/2)

- 必須項以外の主項の組合せを全通り探索
  - 出力を全部覆う物の中でもっとも主項の個数が少ないものを選ぶ
- 例では主項 **11, 1*00, *100, 110*, 11*1 のうち必須項が **11, 1*00, *100 なので

{ **11, 1*00, *100 }, { **11, 1*00, *100, 110* },

{ **11, 1*00, *100, 11*1 },

{ **11, 1*00, *100, 110*, 11*1 } の4通り

最小コスト組合せの探索(2/2)

- 先の例では { **11, 1*00, *100, 110* } と { **11, 1*00, *100, 11*1 } が主項個数最小解
- 通常、使う主項の個数が最も少ない組合せを選べばコストも最小になる
  - 計算速度的に合理的な方法

Don’t care のサポート

- 出力が0と1どちらでも良い入力組み合わせ (don’t care)がある場合、最適化が可能
- 主項を選ぶときはdon’t care = 1 とみなす
  - 主項のバリエーションを増やすため
- 主項集合を選ぶときは don’t care = 0とみなす
  - 覆う個所を減らし、主項の個数を節約する

備考

- 出力が0の部分に注目すれば QM法を用いてOR-AND形式の論理式を出力可能
- 実用分野では、AND-OR形式だけではなく、XOR-AND-OR形式などを対象とした論理式最適化手法が用いられている
  - 計算時間はQM法よりかかる

動作確認用入力

- プログラムが完成したら、下の例が正常に動作するか確認せよ
  - 最適解は { 11*, 0*1, *00 } と { *11, 00*, 1*0 }
  - この論理式を入力すると誤動作する例が毎年非常に多い

	A	A		
S	101	111	011	001
	100	110	010	000
		B	B	