論理設計
東野担当７回目
授業スライド
１１月１８日

基礎工学部情報科学科　東野輝夫

授業時間変更＆中間試験のお知らせ

- 長谷川先生のご都合で，長谷川先生担当の水曜３限の計算機言語の授業と水曜４限の東野の論理設計の授業を下記のように交換して実施します．
- １１月２５日（水）
  - ３限　計算機言語　→　論理設計　の授業に変更
  - ４限　論理設計　　（この日は２コマ続けて論理設計の授業を実施）
- １２月　９日（水）
  - ３限　計算機言語　（この日は２コマ続けて計算機言語の授業を実施）
  - ４限　論理設計　→　計算機言語　の授業に変更

- 東野担当の論理設計の中間試験について
  - １１月２５日（水）の４限（15:10 開始）に東野担当の論理設計の中間試験を実施します．
  - この日は３限に普通の授業を実施し，４限に試験を実施します．試験はCLE上で実施します．詳細は次のスライドで説明します．
    - 15:10　試験の方法を説明（CLEビデオ or Zoom）
    - 15:20　CLEに試験問題を掲示し，制限時間を決めて時間内にレポート課題提出と同じ方法で答案をuploadしてもらうことで答案回収します．

中間試験の試験範囲

- 試験範囲
  - 論理回路の教科書の１章～１１章（基本的に中間試験の対象は，論理設計の１回目から６回目の授業で説明した６章から１１章の内容）
- 理解してほしい項目
  - カルノー図（ドント・ケアを含むカルノー図）の簡単化
  - クワイン・マクラスキー法を用いた主項や最簡積和形の生成
  - 多出力論理関数の簡単化
  - 同期式順序回路の動きを状態遷移図（Mealy型とMoore型の両方）で表現
  - Dフリップ・フロップを用いて与えられた状態遷移図を実現する同期式順序回路を生成

中間試験の実施方法について

- 試験は解答時間35分間＋CLEにアップするための時間10分の合計45分で，問題は４題です.
- 11月25日(水)15:00過ぎに出欠確認と質問用にZoomを立ち上げますので、15:10までにログインして下さい．Zoomのミーティングや IDやパスワードなどの情報は当日の午前９時にCLE上に表示します.
- 15:10から試験の実施方法などをもう一度Zoomで説明し，試験を15:20に開始します.
- 試験問題は15:20にCLE上で閲覧できるようになり，15:55頃を目安に解答を終了し，16:05までに解答をCLEにアップして下さい.
- 試験の方法の詳細については当日の午前９時にCLE上に表示します.
- 配布したテキストやスライドの閲覧は自由です（持ち込み可と条件は同じ）.

授業計画の変更

長谷川先生との授業の入れ替えに伴い，章の説明の順番を変更します

- 授業計画：東野担当の授業計画を下記のように変更します．
  1. ドントケアを含む論理関数の簡単化（６章）
  2. フリップフロップとレジスタ（１０章）
  3. 同期式順序回路（Mealy型, Moore型順序回路）（１１章）
  4. カルノー図を用いた論理関数の簡単化（１章から５章の復習）
  5. 組合せ論理回路設計、よく用いられる組み合わせ回路（７章，８章）
  6. 加減算器とALU、順序回路の簡単化（９章，１２章前半）
  7. 演習
  8. 順序回路の簡単化、カウンタ（１２章後半，１３章）
  9. 中間試験（１章～１１章）
  10. ＩＣを用いた順序回路の実現（１５章）
  11. 演習
  12. ＣＰＵの設計（付録）
  13. ＣＰＵの設計，演習
  14. 乗算器と除算器（１４章）
  15. 期末試験（１２章～１５章，付録）

８コマ目の授業を11/25の３限に実施
９コマ目の試験を11/25の４限に実施
中間試験の範囲を１１章までに変更
期末試験の範囲を１２章以降に変更

第９章
加減算器とALU

O(log n)の比較器の話

～桁上げ先見加算器と同じような考え方を用いた
比較器の高速化～

nビットの２進数の比較器

- 2^n-1 以下の非負整数 $A=(a_{n-1},a_{n-2},\cdots,a_1,a_0)$，$B=(b_{n-1},b_{n-2},\cdots,b_1,b_0)$ の２進数表現が入力として与えられたとき，
  - A＞B ならば（x，y）＝（1，0）
  - A＝B ならば（x，y）＝（0，0）または（1，1）
  - A＜B ならば（x，y）＝（0，1）

  を満たすようにx，yを出力する比較器を作りたい．

1. 同一の基本回路M_1を下の図１のように接続することにより，比較器を作りたい．M_1の各入力変数を図２のように名付けるとき，出力p′，q′を入力a，b，p，qの最簡積和形で表せ．

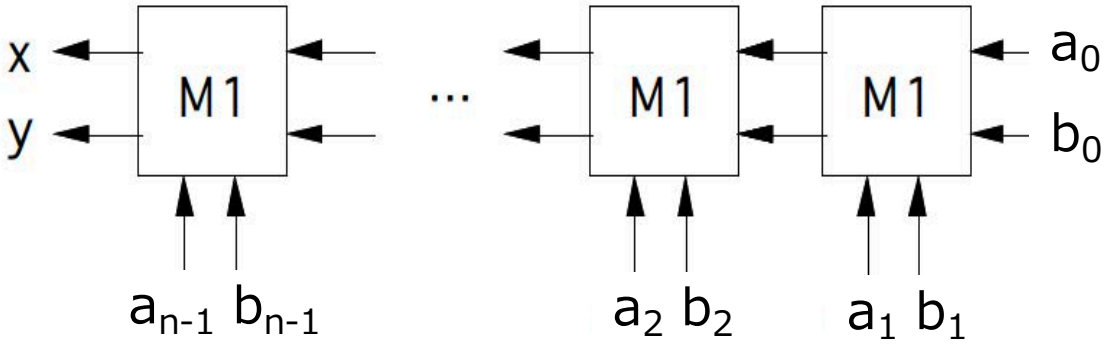


図１

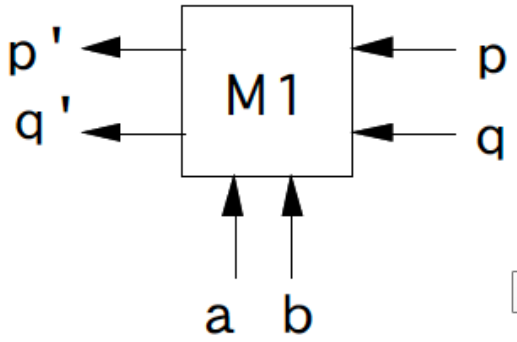


図２

nビットの２進数の比較器

1. 同一の基本回路M_1を下の図１のように接続することにより，比較器を作りたい．M_1の各入力変数を図２のように名付けるとき，出力ｐ′，ｑ′を入力ａ，ｂ，ｐ，ｑの最簡積和形で表せ.

（ヒント）

- ａ＞ｂのときｐ′＝１，ａ＝ｂのとき ｐ′＝ｐ，ａ＜ｂのときｐ′＝０となるように出力ｐ′を決めればよい.
- 同様に，ａ<ｂのとき ｑ′＝１，ａ＝ｂのときｑ′＝ｑ，ａ＞ｂのときｑ′＝０となるように出力ｑ′を決めればよい.

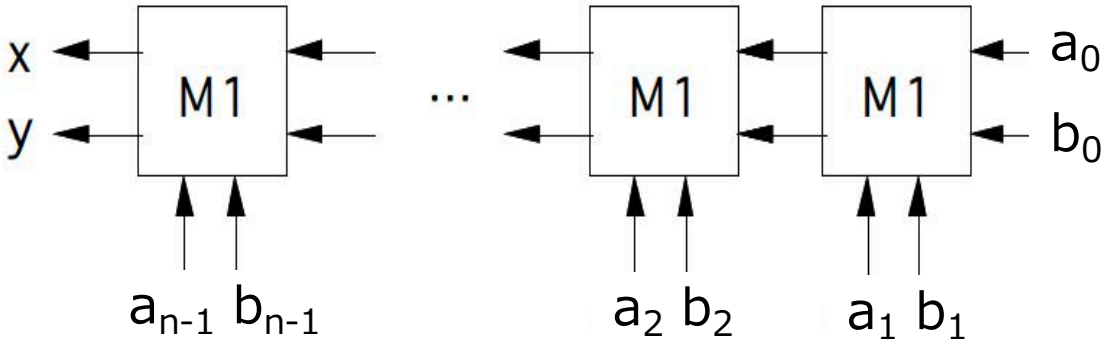


図１

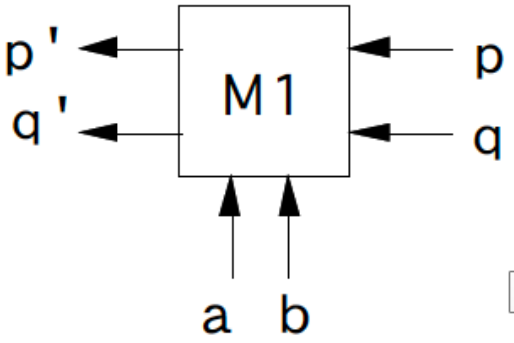


図２

O(n)の遅延時間の比較器

1. 同一の基本回路M_1を下の図１のように接続することにより，比較器を作りたい．M_1の各入力変数を図２のように名付けるとき，出力ｐ′，ｑ′を入力ａ，ｂ，ｐ，ｑの最簡積和形で表せ．

（解答）

- ａ＞ｂのときｐ′＝１，ａ＝ｂのとき ｐ′＝ｐ，ａ＜ｂのときｐ′＝０となるように出力ｐ′を決めればよい．同様に，ａ＜ｂ のとき ｑ′＝１，ａ＝ｂのときｑ′＝ｑ，ａ＞ｂのときｑ′＝０となるように出力ｑ′を決めればよい．

  – $p' = a \cdot \overline{b} \vee a \cdot b \cdot p \vee \overline{a} \cdot \overline{b} \cdot p = a \cdot \overline{b} \vee a \cdot p \vee \overline{b} \cdot p$

  – $q' = \overline{a} \cdot b \vee a \cdot b \cdot q \vee \overline{a} \cdot \overline{b} \cdot q = \overline{a} \cdot b \vee b \cdot q \vee \overline{a} \cdot q$

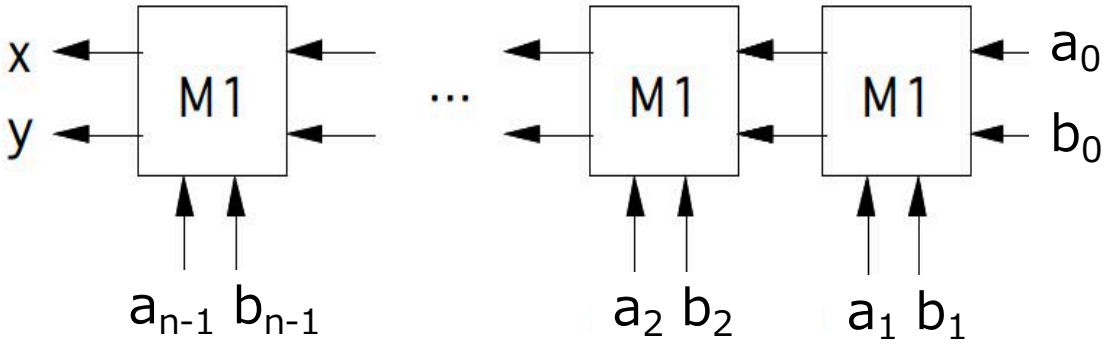


図１

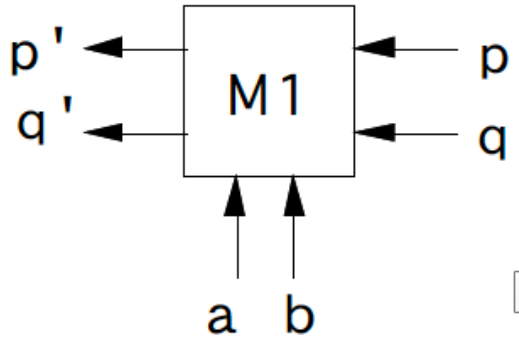


図２

O(n)の遅延時間の比較器

1. 同一の基本回路M_1を下の図１のように接続することにより，比較器を作りたい．M_1の各入力変数を図２のように名付けるとき，出力ｐ′，ｑ′を入力ａ，ｂ，ｐ，ｑの最簡積和形で表せ．

（解答）

- 図１は下記の図３のように接続しても良い．この場合，Ａ＝Ｂ なら，最下位の桁の p, q に入力した０又は１の値が x, y に出力される．
- この比較器でｎビットの数の比較をする場合，遅延時間は O(n) になる．

  – $p' = a \cdot \overline{b} \vee a \cdot b \cdot p \vee \overline{a} \cdot \overline{b} \cdot p = a \cdot \overline{b} \vee a \cdot p \vee \overline{b} \cdot p$

  – $q' = \overline{a} \cdot b \vee a \cdot b \cdot q \vee \overline{a} \cdot \overline{b} \cdot q = \overline{a} \cdot b \vee b \cdot q \vee \overline{a} \cdot q$

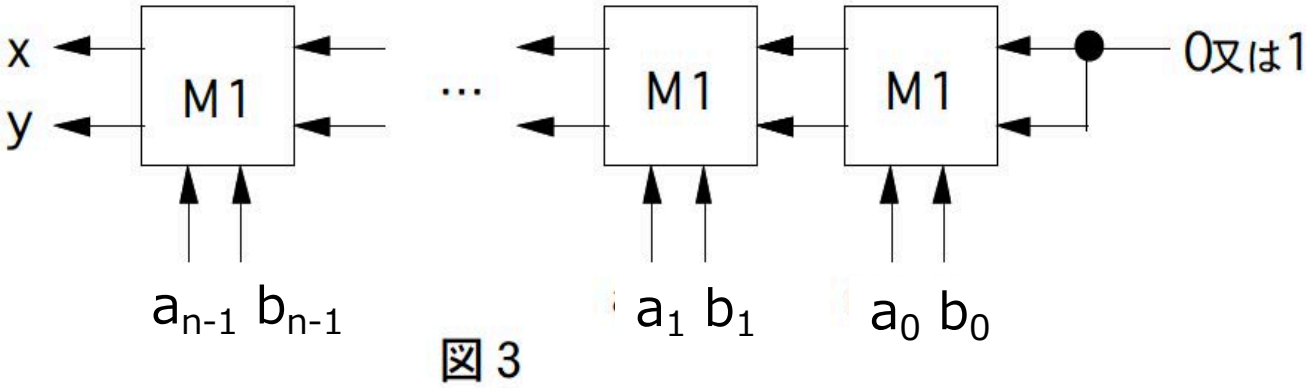


図３

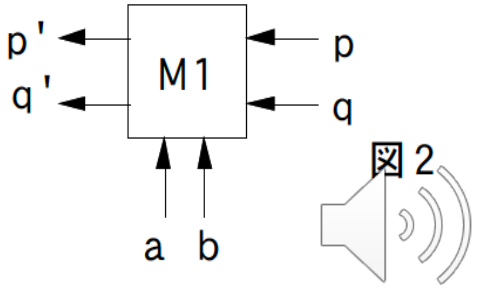


図２

加減算回路を用いた比較器

- 同一の基本回路M_2を図４のように接続することで，別の比較器を作りたい．

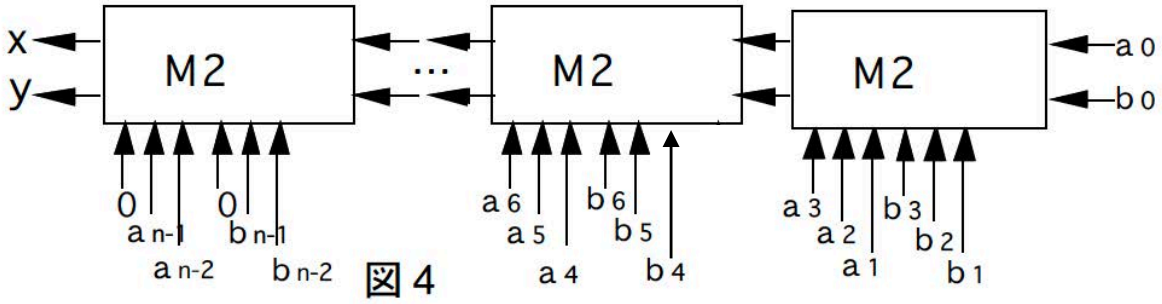


図４

- M_2では，右端にa_0, b_0 を入力し，$a_{n-1},\cdots,a_2,a_1$ 及び $b_{n-1},\cdots,b_2,b_1$については，下位の桁から３ビットづつを右の基本回路から順次入力する．上位の余った入力には０を入力する．M_2を４ビット全加算器１個及び幾つかの AND, OR, NOT ゲートを用いて実現せよ．M_2の入力変数を図５のように表し，４ビット全加算器を図６のように表すとする．

（ヒント）A＞Bと A－B ＞０は同じ．２の補数表現で考えてみよ．

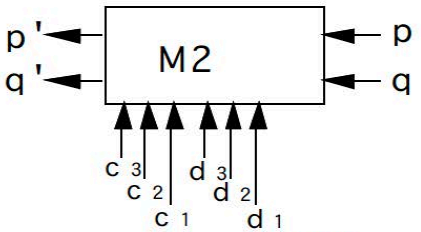


図５

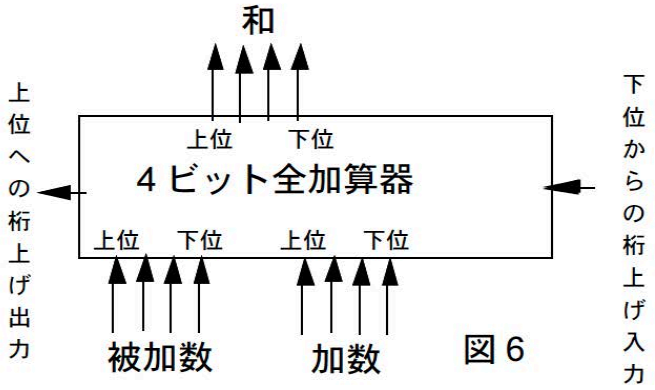


図６

加減算回路を用いた比較器

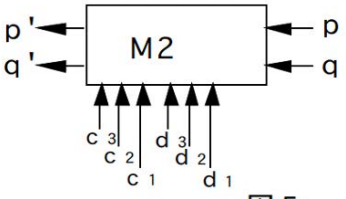


図5

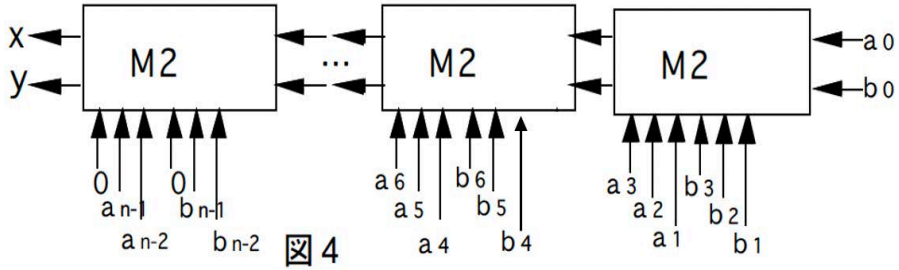


図4

- 同一の基本回路M_2を図４のように接続することで，別の比較器を作りたい．

（ヒント）

- ４ビットの２進数 $c=c_3c_2c_1p$ 及び $d=d_3d_2d_1q$ に対して $c>d$ であることと，$c-d>0$ は等価．同様に，$c=d$ と$c-d=0$，$c<d$ と$c-d<0$ も等価．よって$c-d$の値を計算し，その値が正なら $(p',q')=(1,0)$，負なら$(p',q')=(0,1)$，ゼロなら $(p',q')=(p,q)$ となるように回路を組めばよい．
- ２の補数表現で $S=(c_3c_2c_1p)-(d_3d_2d_1q) = (c_3c_2c_1p)+(\bar{d}_3\bar{d}_2\bar{d}_1\bar{q})+1$
- もし $(c_3c_2c_1p)-(d_3d_2d_1q)=0$ なら，$p'=p$, $q'=q$ とすればよい．よって，M_2は図７のような回路で実現できる（これ以外の回路も色々考えられる）．

（解答）

- p', q' は次のように求められる．
  - $p'=$ (Sが正)∨{(Sが0)・p}
    $= (\bar{s}_3\cdot(s_2\vee s_1\vee s_0))\vee\{(\bar{s}_3\bar{s}_2\bar{s}_1\bar{s}_0)\cdot p\}$
  - $q'=$ (Sが負)∨{(Sが0)・q}
    $= (s_3)\vee\{(\bar{s}_3\bar{s}_2\bar{s}_1\bar{s}_0)\cdot q\}$

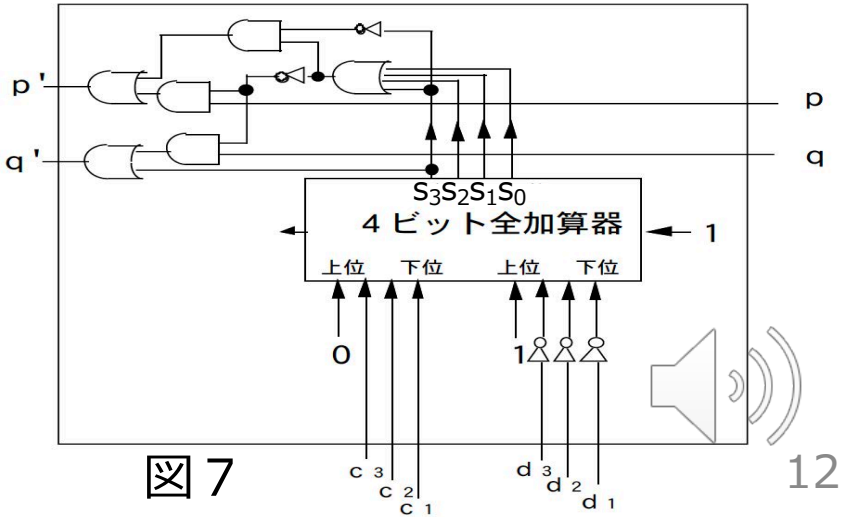


図７

O(log n)の遅延時間の比較器

- M_2を幾つか用いて，遅延時間が入力ビット数 n に対して，O(log n) となる比較器を16ビットの２進数（すなわち n ＝16）を例に実現せよ.

(ヒント)

- 先週説明した桁上げ先見加算器と同じような考え方を用いれば，M_2 を５個用いて木状に組み合わせれば実現できる.

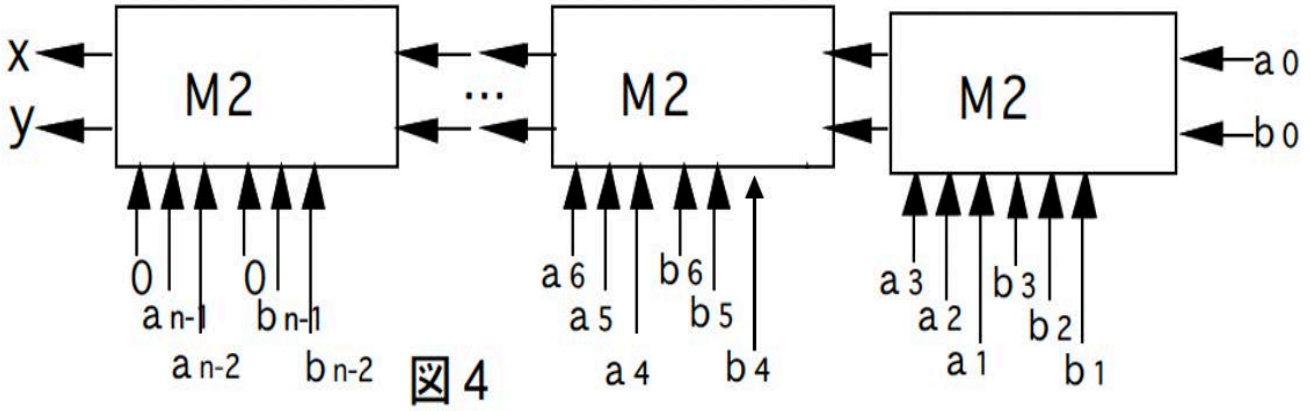


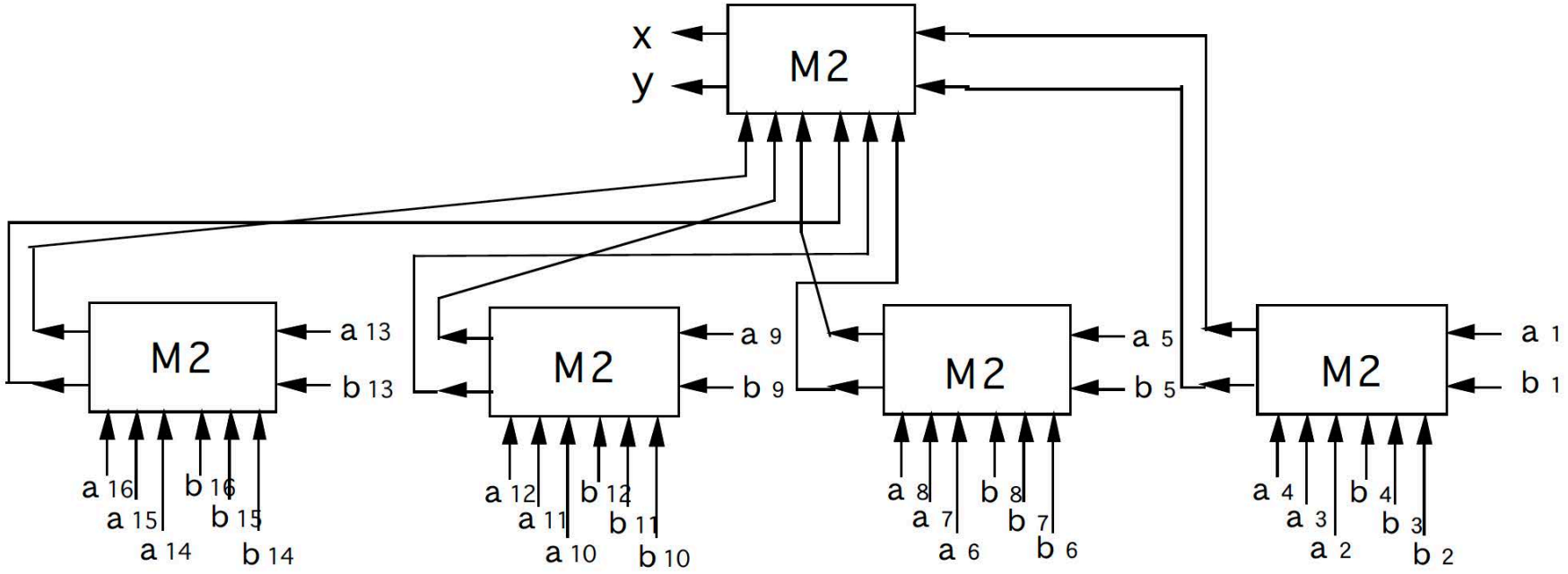
図４

O(log n)の遅延時間の比較器

- M_2を幾つか用いて，遅延時間が入力ビット数 n に対して，O(log n)となる比較器を16ビットの２進数（すなわち n ＝16）を例に実現せよ.

(解答)

- M_2 を５個次のように木状に組み合わせれば実現できる.



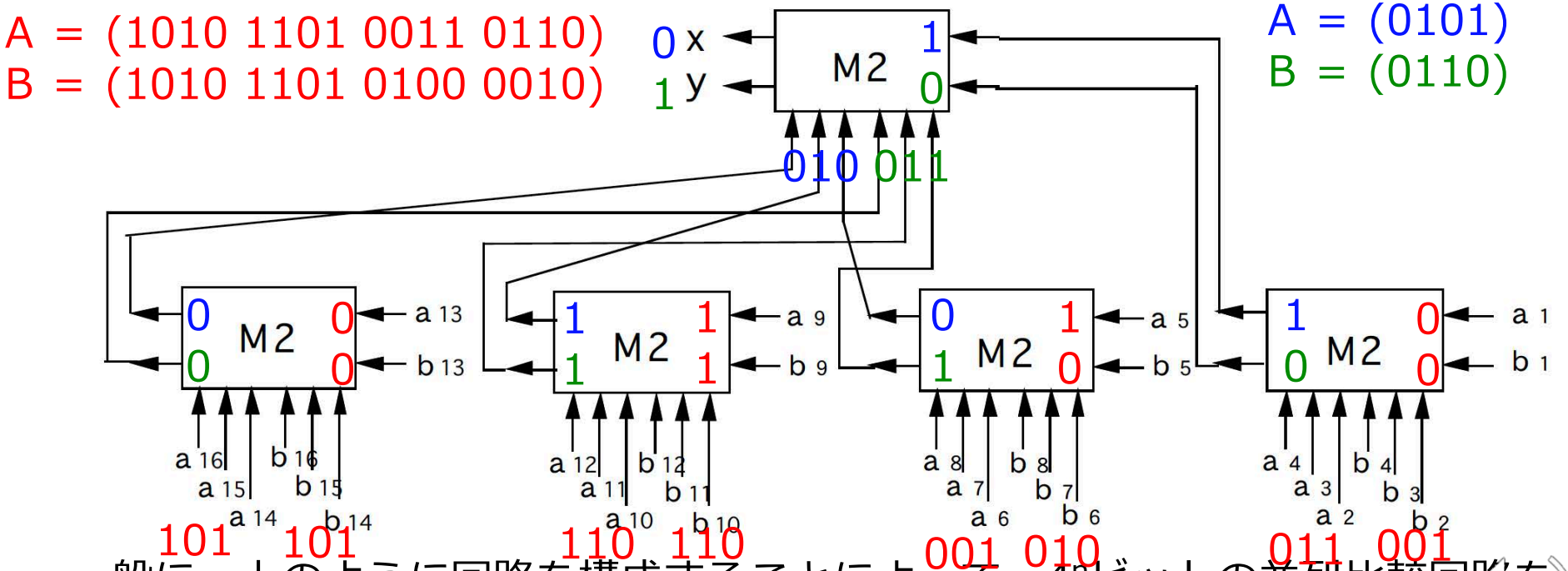
- 一般に，上のように回路を構成することによって，4^nビットの並列比較回路を作ることができる． この並列比較回路の遅延時間も桁上げ先見加算器と同様，O(log n) である.

O(log n)の遅延時間の比較器

- M_2を幾つか用いて，遅延時間が入力ビット数 n に対して，O(log n)となる比較器を16ビットの２進数（すなわち n＝16）を例に実現せよ.

(解答)

- M_2 を５個次のように木状に組み合わせれば実現できる.



- 一般に，上のように回路を構成することによって，4^nビットの並列比較回路を作ることができる． この並列比較回路の遅延時間も桁上げ先見加算器と同様，O(log n) である.

第６章から第１１章の復習

中間試験の試験範囲

- 試験範囲
  - 論理回路の教科書の１章～１１章（基本的に中間試験の対象は，論理設計の１回目から６回目の授業で説明した６章から１１章の内容）
- 理解してほしい項目
  - カルノー図（ドント・ケアを含むカルノー図）の簡単化
  - クワイン・マクラスキー法を用いた主項や最簡積和形の生成
  - 多出力論理関数の簡単化
  - 同期式順序回路の動きを状態遷移図（Mealy型とMoore型の両方）で表現
  - Dフリップ・フロップを用いて与えられた状態遷移図を実現する同期式順序回路を生成

ドント・ケアを含む カルノー図の簡単化

ドントケアを含む論理関数の簡単化

- ドントケアを含む論理式の簡単化では，ドントケアを都合よく解釈して，より簡単な最簡積和形を求める．
- ドントケアは 0, 1 のいずれでもよいので，より好ましい最簡積和形が求まるように 0, 1 を割り当てることになる．これを実現するための簡単化の手続きは以下のようになる．
  - ステップ1：論理関数 *f* のすべての主項を求める
    - ポイント1：ドントケア X を 1 と考えて主項を生成する．より大きなループ，つまりリテラル数の少ない積項が作れることを期待
  - ステップ2：論理関数 *f* のすべての最小項を最小数の主項で覆う（最小被覆）
    - ポイント2：最小項の 1 は必ず被覆するが，X は被覆してもしなくてもよい． 被覆した部分は 1 が割り当てられ，被覆しなかった部分は 0 が割り当てられることになる
  - ただし，X のみを被覆するループは主項の被覆に選択しない

例題6・1

- 図6·1 のカルノー図で示されている論理関数 *f* を最小化せよ
  - ステップ１
  - ステップ２

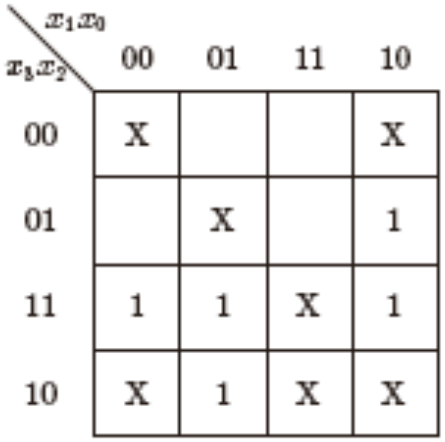
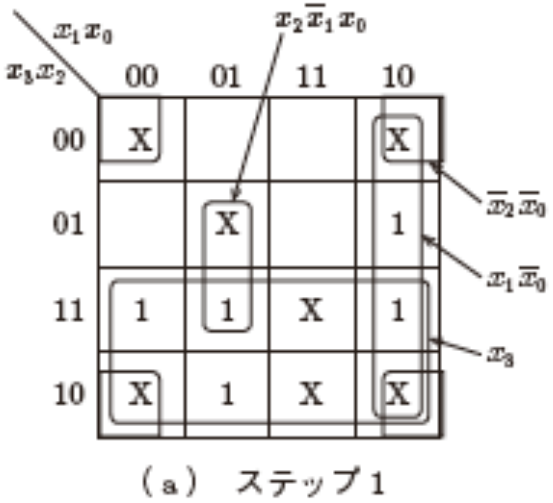


図 6・1 例題 6·1 のカルノー図



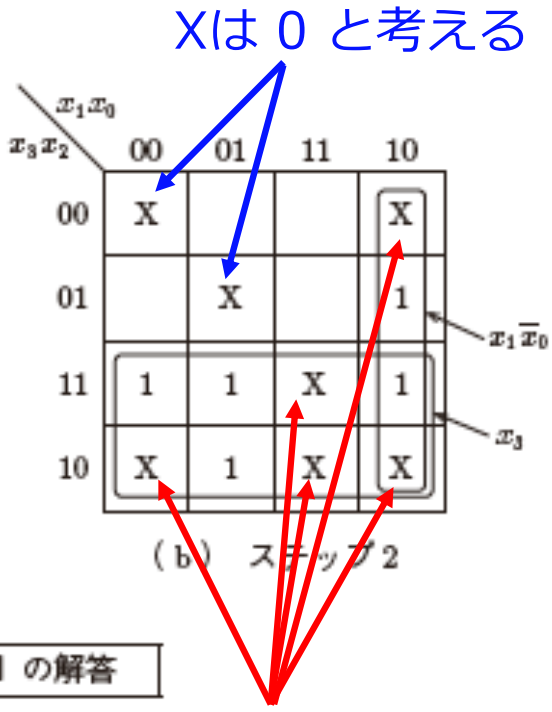


図 6・2 例題 6·1 の解答

最小被覆の求め方

- 最小被覆：論理関数 f のすべての最小項を最小数の主項で覆うこと
- ある最小項を被覆している主項が一つしかないとき，その主項を 必須主項（あるいは必須項，essential prime implicant）と呼び，その最小項を 特異最小項 と呼ぶ.
- 主項 A と B の間に次の条件が成立するとき，B は A に支配されているという
  - 条件1：A に含まれる最小項の集合 ⊇ B に含まれる最小項の集合
  - 条件2：A のリテラル数 ≤ B のリテラル数（ A のループ ≧ B のループ）

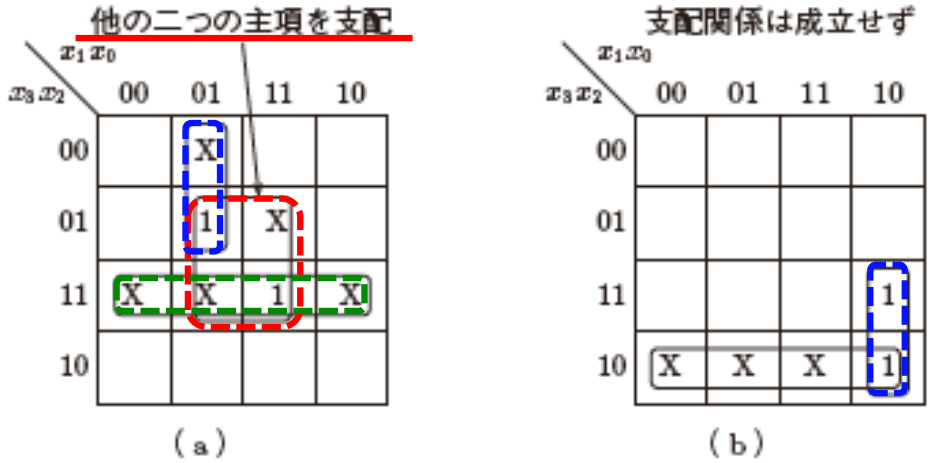


図 6・3 主項の支配関係の例

最小被覆の求め方

- 必須主項と主項の支配関係を用いると，最小被覆を求める手続きは以下のようになる
  - ステップ1：必須主項をすべて探し出し，最小被覆に含める
  - ステップ2：必須主項が被覆していた部分をすべてドントケアにする
  - ステップ3：他の主項に支配されている主項をすべて探し出して除去する
  - ステッ プ1 に戻る（変化がなくなったとき終了する）

例題6・2

- 図6·4 のカルノー図に対して，必須主項と主項の支配関係に注目して最小被覆を求め，最簡積和形を示せ

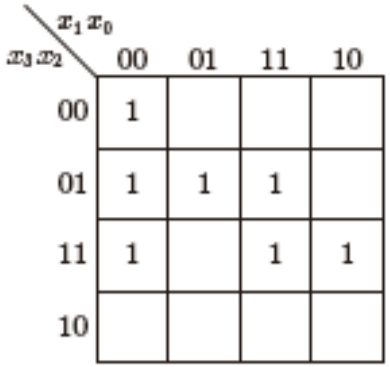


図 6・4 例題 6·2 のカルノー図

例題6・2

- 図6·5(a)に主項をすべて列挙したカルノー図を示す
- カルノー図に含まれる最小項の中で (x_3, x_2, x_1, x_0) = (0, 0, 0, 0) が特異最小項．この特異最小項を含む主項が必須主項．最小被覆に含める（ステップ1）
- 次に図6·5(b)のように必須主項が被覆していた最小項をドントケアに置き換える（ステップ2）
- 残された主項の中で他の主項に支配されている主項を見つける．この例では，点線のループで示した二つの主項が他の主項に支配されており，主項から取り除く（ステップ3）
- 再び必須主項を探すと図6·5(c)のように 2 つ見つかる（ステップ1）．必須主項をドントケアに置き換える（ステップ 2）．図6·5(d)で，残された二つの主項はいずれを選択してももう一方の主項を支配するので，一方を選択することで最小被覆が求まる．
- 最簡積和形は，

$$\bar{x}_3\bar{x}_1\bar{x}_0 \vee \bar{x}_3x_2x_0 \vee x_3x_2\bar{x}_0 \vee x_2x_1x_0$$
$$\text{または } \bar{x}_3\bar{x}_1\bar{x}_0 \vee \bar{x}_3x_2x_0 \vee x_3x_2\bar{x}_0 \vee x_3x_2x_1$$

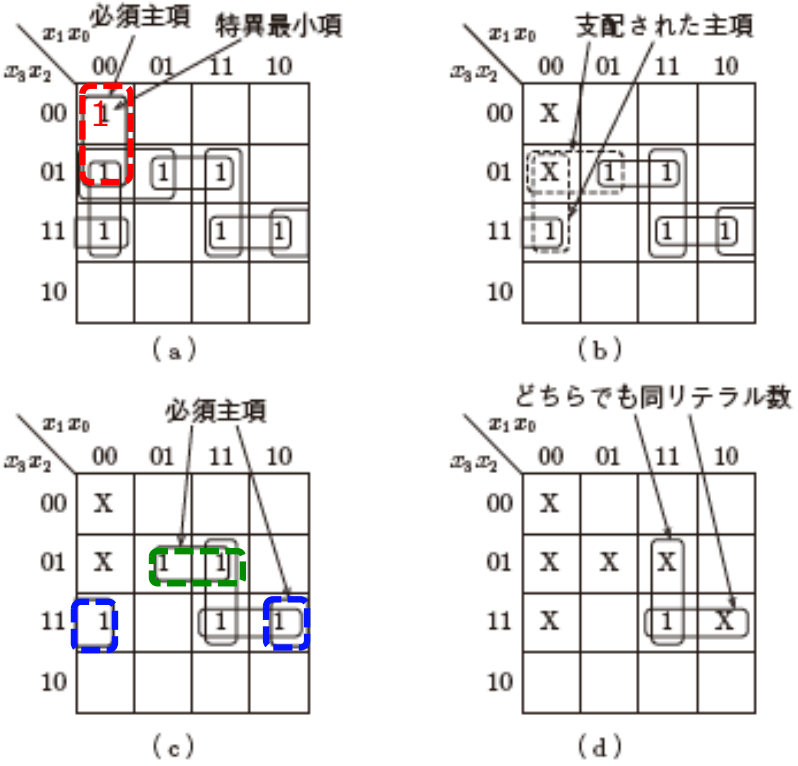


図 6・5　例題 6·2 の解答

クワイン・マクラスキー法を用いた主項や最簡積和形の生成

クワイン・マクラスキー法

キューブ表現

- 主項の生成に積項のキューブ表現を用いる．論理変数 x_i に対する係数a_i を考え，a_i =1, 0, - のときにそれぞれ x_i, $\bar{x}_i$, 1 を表現するものとする．積項に対する係数の列挙$a_1a_2,\cdots,a_n$ を積項のキューブ表現と呼ぶ（n は論理変数の数）．
- 例えば 4 変数の論理関数において積項 $x_3x_2\bar{x}_1x_0$ のキューブ表現は 1101，x_2x_1 のキューブ表現は -11- である．
- 二つの積項のキューブ表現において (n−1) か所が同じで 1 か所だけが 0/1 と異なっているとき，これらの積項はカルノー図上で隣接している．これらの隣接した積項の論理和のキューブ表現は，異なっていた箇所を - に置き換えたものになる．この操作をキューブの併合という．
- 例えば，1-10 と1-00 を併合すると，1--0 となる．これは

  $x_3x_1\bar{x}_0 \vee x_3\bar{x}_1\bar{x}_0 = x_3\bar{x}_0$

  に相当する．

クワイン・マクラスキー法

すべての主項を求める手続き

- ステップ1：すべての最小項に対してキューブ表現を求める．キューブに含まれる 1 の個数によって昇順にグループ分けし，第一段階のリストを生成する．
- ステップ2：リストの第一グループに含まれるキューブと第二グループに含まれるキューブを比較し 1 か所だけ異なる場合には，それらのキューブにチェックを付ける．それらを併合して得られるキューブを次の段階のリストの第一グループに入れる．
- 同様に第二グループと第三グループに対して比較し，併合結果は次の段階のリストの第二グループに入れる．この隣どうしのグループに対する比較・併合を最後のグループまで行う．
- ステップ3：次の段階のリストにグループが二つ以上あれば，ステップ2 に戻る．
- ステップ4：チェックのついていないキューブが主項である．

x_3	x_2	x_1	x_0	f_a
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

表 5・2　7 セグメントデコーダの真理値表

第一段階のリスト

	x_3	x_2	x_1	x_0	最小項	
第一グループ	0	0	0	0	(0)	v
第二グループ	0	0	1	0	(2)	v
	1	0	0	0	(8)	v
第三グループ	0	0	1	1	(3)	v
	0	1	0	1	(5)	v
	0	1	1	0	(6)	v
	1	0	0	1	(9)	v
第四グループ	0	1	1	1	(7)	v

クワイン・マクラスキー法（例題6・3）

すべての主項を求める手続き

- ステップ1：すべての最小項に対してキューブ表現を求める．キューブに含まれる 1 の個数によって昇順にグループ分けし，第一段階のリストを生成する．
- ステップ2：リストの第一グループに含まれるキューブと第二グループに含まれるキューブを比較し 1 か所だけ異なる場合には，それらのキューブにチェックを付ける．それらを併合して得られるキューブを次の段階のリストの第一グループに入れる．
- 同様に第二グループと第三グループに対して比較し，併合結果は次の段階のリストの第二グループに入れる．この隣どうしのグループに対する比較・併合を最後のグループまで行う．
- ステップ3：次の段階のリストにグループが二つ以上あれば，ステップ2 に戻る．
- ステップ4：チェックのついていないキューブが主項である．

第一段階のリスト

	x_3	x_2	x_1	x_0	最小項	
第一グループ	0	0	0	0	(0)	v
第二グループ	0	0	1	0	(2)	v
	1	0	0	0	(8)	v
	0	0	1	1	(3)	v
第三グループ	0	1	0	1	(5)	v
	0	1	1	0	(6)	v
	1	0	0	1	(9)	v
第四グループ	0	1	1	1	(7)	v

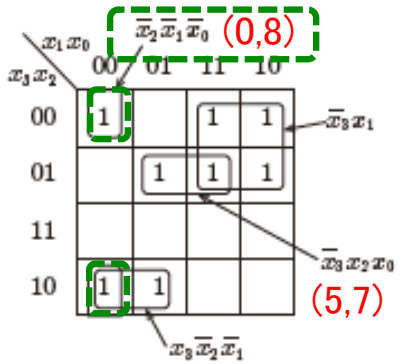
第二段階のリスト

x_3	x_2	x_1	x_0	最小項	
0	0	−	0	(0, 2)	
−	0	0	0	(0, 8)	
0	0	1	−	(2, 3)	v
0	−	1	0	(2, 6)	v
1	0	0	−	(8, 9)	
0	−	1	1	(3, 7)	v
0	1	−	1	(5, 7)	
0	1	1	−	(6, 7)	v

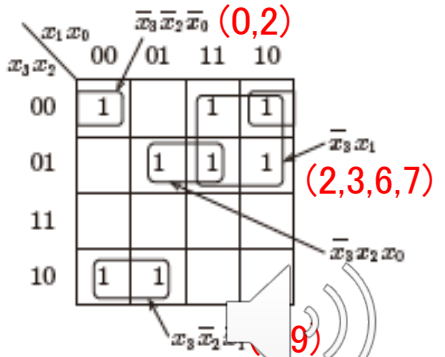
第三段階のリスト

x_3	x_2	x_1	x_0	最小項
0	−	1	−	(2, 3, 6, 7)

図 6・6　例題 6.3 の主項の生成



（a）　f_a のカルノー図



（b）　f_a のカルノー図

クワイン・マクラスキー法（例題6・3）

すべての主項を求める手続き

- ステップ1：すべての最小項に対してキューブ表現を求める．キューブに含まれる 1 の個数によって昇順にグループ分けし，第一段階のリストを生成する．
- ステップ2：リストの第一グループに含まれるキューブと第二グループに含まれるキューブを比較し 1 か所だけ異なる場合には，それらのキューブにチェックを付ける．それらを併合して得られるキューブを次の段階のリストの第一グループに入れる．
- 同様に第二グループと第三グループに対して比較し，併合結果は次の段階のリストの第二グループに入れる．この隣どうしのグループに対する比較・併合を最後のグループまで行う．
- ステップ3：次の段階のリストにグループが二つ以上あれば，ステップ2 に戻る．
- ステップ4：チェックのついていないキューブが主項である．

第一段階のリスト

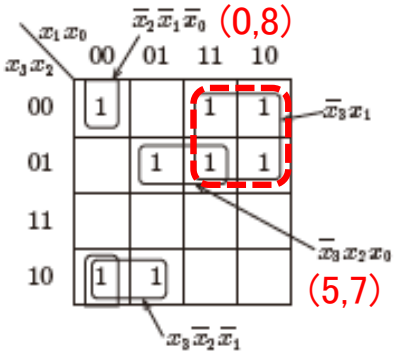
	x_3	x_2	x_1	x_0	最小項	
第一グループ	0	0	0	0	(0)	v
第二グループ	0	0	1	0	(2)	v
	1	0	0	0	(8)	v
第三グループ	0	0	1	1	(3)	v
	0	1	0	1	(5)	v
	0	1	1	0	(6)	v
	1	0	0	1	(9)	v
第四グループ	0	1	1	1	(7)	v

第二段階のリスト

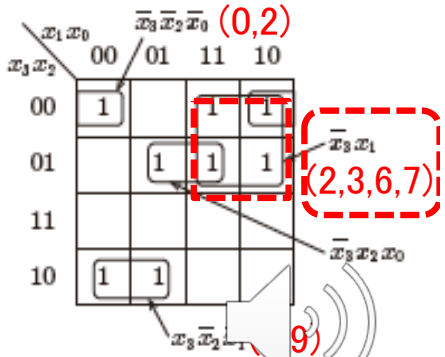
x_3	x_2	x_1	x_0	最小項	
0	0	−	0	(0, 2)	
−	0	0	0	(0, 8)	
0	0	1	−	(2, 3)	v
0	−	1	0	(2, 6)	v
1	0	0	−	(8, 9)	
0	−	1	1	(3, 7)	v
0	1	−	1	(5, 7)	
0	1	1	−	(6, 7)	v

第三段階のリスト

x_3	x_2	x_1	x_0	最小項
0	−	1	−	(2, 3, 6, 7)



（a） f_a のカルノー図



（b） f_a のカルノー図

図 6・6　例題 6.3 の主項の生成

クワイン・マクラスキー法（例題6・3）

すべての主項を求める手続き

- ステップ1：すべての最小項に対してキューブ表現を求める．キューブに含まれる 1 の個数によって昇順にグループ分けし，第一段階のリストを生成する．
- ステップ2：リストの第一グループに含まれるキューブと第二グループに含まれるキューブを比較し 1 か所だけ異なる場合には，それらのキューブにチェックを付ける．それらを併合して得られるキューブを次の段階のリストの第一グループに入れる．
- 同様に第二グループと第三グループに対して比較し，併合結果は次の段階のリストの第二グループに入れる．この隣どうしのグループに対する比較・併合を最後のグループまで行う．
- ステップ3：次の段階のリストにグループが二つ以上あれば，ステップ2 に戻る．
- ステップ4：チェックのついていないキューブが主項である．

第一段階のリスト

	x_3	x_2	x_1	x_0	最小項	
第一グループ	0	0	0	0	(0)	v
第二グループ	0	0	1	0	(2)	v
	1	0	0	0	(8)	v
第三グループ	0	0	1	1	(3)	v
	0	1	0	1	(5)	v
	0	1	1	0	(6)	v
	1	0	0	1	(9)	v
第四グループ	0	1	1	1	(7)	v

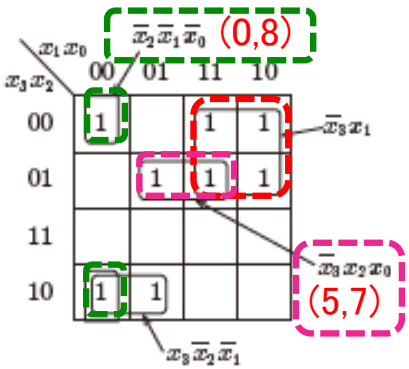
第二段階のリスト

x_3	x_2	x_1	x_0	最小項	
0	0	−	0	(0,2)	
−	0	0	0	(0,8)	
0	0	1	−	(2,3)	v
0	−	1	0	(2,6)	v
1	0	0	−	(8,9)	
0	−	1	1	(3,7)	v
0	1	−	1	(5,7)	
0	1	1	−	(6,7)	v

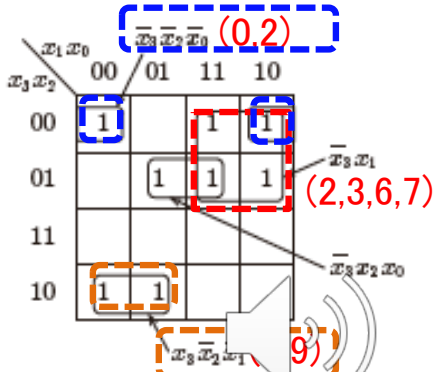
第三段階のリスト

x_3	x_2	x_1	x_0	最小項
0	−	1	−	(2,3,6,7)

図 6・6　例題 6.3 の主項の生成



（a）f_a のカルノー図



（b）f_a のカルノー図

最小被覆

主項表

- 次に主項表を用いて関数の最小項の最小被覆を求める．主項表では主項を縦軸に，論理関数の最小項を横軸に並べる．主項はリテラル数が少ないものを上に記載する

例題6・4

- 例題6・3 で求めた f_a の主項表を示せ
- 特異最小項は 3, 5, 6, 9
- それらを被覆する $\bar{x}_3x_1$（3, 6 を被覆）$x_3\bar{x}_2\bar{x}_1$（9 を被覆），$\bar{x}_3x_2x_0$（5 を被覆）が必須主項（必須項）

主項	最小項 0	2	3	5	6	7	8	9
$\bar{x}_3x_1$		1	1		1	1		
$\bar{x}_3\bar{x}_2\bar{x}_0$	1	1						
$\bar{x}_2\bar{x}_1\bar{x}_0$	1						1	
$x_3\bar{x}_2\bar{x}_1$							1	1
$\bar{x}_3x_2x_0$				1		1		

図 6・7 例題 6.4 の主項表

最小被覆

最小被覆の求め方

- ステップ1：必須行（必須項の行）をすべて探し出し，最小被覆に含めたのち，表から除去する．必須行に被覆されている列も除去する（列 2, 3, 5, 6, 7, 8, 9 が除去され列 0 のみが残る）．
- ステップ2：列 r が 1 をもつすべての行に列 s が 1 を持つとき，列 s は列 r を支配するという．他の行に支配されている行をすべて探し出して除去する．
- ステップ3：他の列を支配している列をすべて探し出して除去し，ステップ1 に戻る．

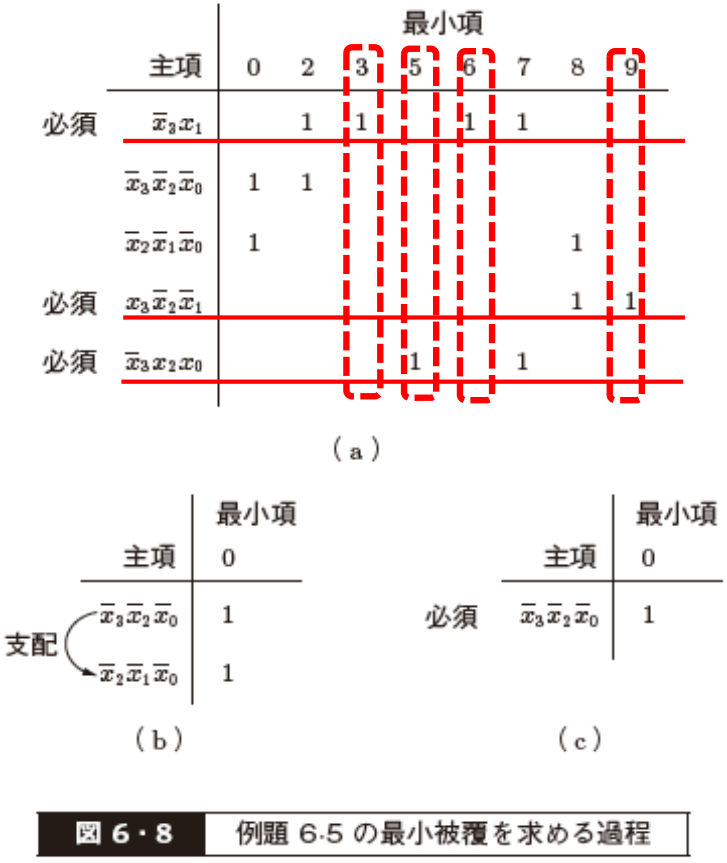
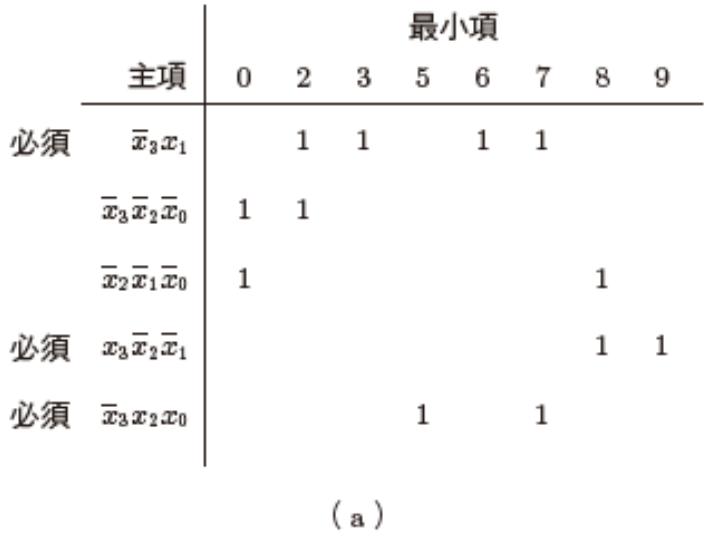


図 6・8　例題 6.5 の最小被覆を求める過程

最小被覆

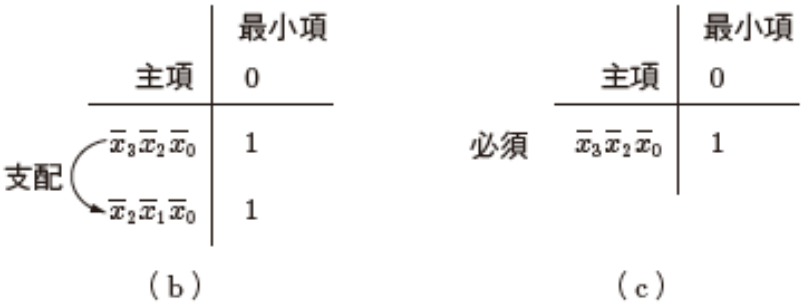
例題6・4

- 例題6・3 で求めた f_a の主項表を示せ



	主項	最小項 0	2	3	5	6	7	8	9
必須	$\bar{x}_3x_1$		1	1		1	1		
	$\bar{x}_3\bar{x}_2\bar{x}_0$	1	1						
	$\bar{x}_2\bar{x}_1\bar{x}_0$	1						1	
必須	$x_3\bar{x}_2\bar{x}_1$							1	1
必須	$\bar{x}_3x_2x_0$				1		1		

(a)



	主項	最小項 0
支配	$\bar{x}_3\bar{x}_2\bar{x}_0$	1
	$\bar{x}_2\bar{x}_1\bar{x}_0$	1

(b)

	主項	最小項 0
必須	$\bar{x}_3\bar{x}_2\bar{x}_0$	1

(c)

図 6・8　例題 6.5 の最小被覆を求める過程

- 特異最小項は 3, 5, 6, 9
- それらを被覆する $\bar{x}_3x_1$（3, 6 を被覆），$x_3\bar{x}_2\bar{x}_1$（9 を被覆），$\bar{x}_3x_2x_0$（5 を被覆）が必須主項（必須項）
- 必須行（必須項の行）と必須行に被覆された列を除去して，図6·8（b）を得る（ステップ1）
- 図6·8(b) では，$\bar{x}_2\bar{x}_1\bar{x}_0$ が $\bar{x}_3\bar{x}_2\bar{x}_0$ に支配されているため，$\bar{x}_2\bar{x}_1\bar{x}_0$ の行を除去し，図6·8(c) を得る（ステップ2）なお，本例では支配関係を逆にすることも可能であり，別の最簡積和形も求まる.
- $f_a = \bar{x}_3x_1 \vee x_3\bar{x}_2\bar{x}_1 \vee \bar{x}_3x_2x_0 \vee \bar{x}_3\bar{x}_2\bar{x}_0$

($f_a = \bar{x}_3x_1 \vee x_3\bar{x}_2\bar{x}_1 \vee \bar{x}_3x_2x_0 \vee \bar{x}_2\bar{x}_1\bar{x}_0$ も最簡積和形)

多出力論理関数の簡単化

多出力論理回路の簡単化

例題７・３

- 論理関数が複数あり，それぞれを積和形で表して，組合せ回路として実現することを考える．これらの積和形の論理式に共通の積項が存在すると，組合せ回路として実現するときにANDゲートが共有でき，必要なゲート数が削減できるので，全体の積項数の最小化が重要となる．
- 一方，個々の論理関数の簡単化では，複数の論理関数を実現するために必要な積項数が最小になるとは限らない．
- 複数の論理関数（多出力論理関数）の簡単化においては，各関数の主項のみならず，関数のすべての組合せについてそれらの積の主項も求め，これらの主項による各々の関数の最小被覆を求める．具体的な方法を次の例題（図７・９）を用いて説明する．
- f_a, f_b, f_cの最簡積和形は
  - $f_a = \bar{x}_1 x_0 \vee x_2 x_0 \vee \bar{x}_2 x_1 \bar{x}_0$
  - $f_b = x_1 \bar{x}_0 \vee x_2 \bar{x}_0 \vee \bar{x}_2 \bar{x}_1 x_0$
  - $f_c = x_1 \bar{x}_0 \vee x_2 \bar{x}_0 \vee x_2 \bar{x}_1$
  - 論理積７個，論理和３個

    （ $x_1 \bar{x}_0$，$x_2 \bar{x}_0$ は f_b, f_c で共用）

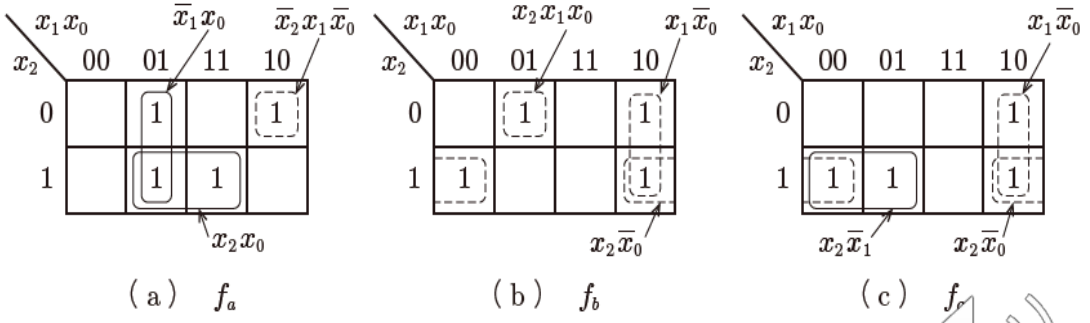


図 7・9　例題 7.3 の f_a, f_b, f_c のカルノー図

多出力論理回路の簡単化

例題７・３

- f_a, f_b, f_cの最簡積和形は
  - $f_a = \bar{x}_1 x_0 \vee x_2 x_0 \vee \bar{x}_2 x_1 \bar{x}_0$
  - $f_b = x_1 \bar{x}_0 \vee x_2 \bar{x}_0 \vee \bar{x}_2 \bar{x}_1 x_0$
  - $f_c = x_1 \bar{x}_0 \vee x_2 \bar{x}_0 \vee x_2 \bar{x}_1$
  - 論理積７個，論理和３個（ $x_1\bar{x}_0$，$x_2\bar{x}_0$ は f_b, f_c で共用）
- $x_1\bar{x}_0$，$x_2\bar{x}_0$のように f_a, f_b, f_cで共用できる主項が見つかれば積項数を削減できる
- $x_1\bar{x}_0$，$x_2\bar{x}_0$は $f_b \cdot f_c$ の主項になっている（図７・10参照）
- f_b の最簡積和形の候補： f_b，$f_a \cdot f_b$，$f_b \cdot f_c$，$f_a \cdot f_b \cdot f_c$ の主項

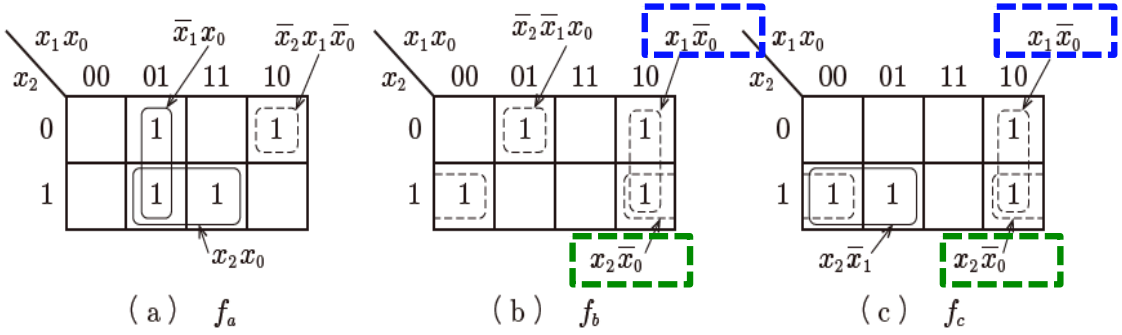


図 7・9　例題 7.3 の f_a, f_b, f_c のカルノー図

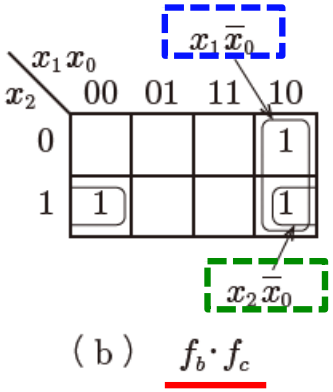


図 7・10

多出力論理回路の簡単化

例題７・３

- f_a, f_b, f_cの最簡積和形は
  - $f_a = \bar{x}_1x_0 \vee x_2x_0 \vee \bar{x}_2x_1\bar{x}_0$
  - $f_b = x_1\bar{x}_0 \vee x_2\bar{x}_0 \vee \bar{x}_2\bar{x}_1x_0$
  - $f_c = x_1\bar{x}_0 \vee x_2\bar{x}_0 \vee x_2\bar{x}_1$
  - 論理積７個，論理和３個

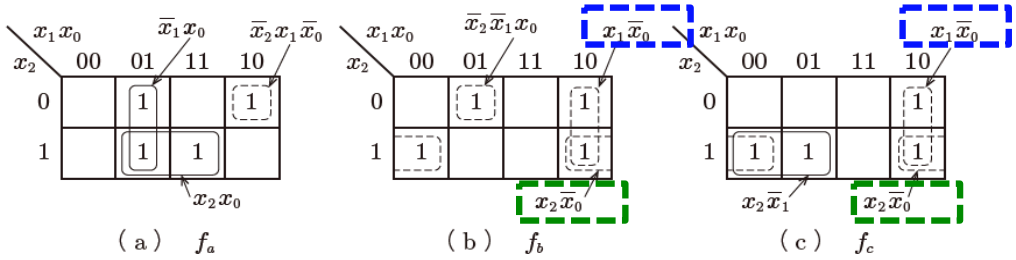


図 7・9　例題 7.3 の f_a, f_b, f_c のカルノー図

- f_a の最小項は，f_a, $f_a \cdot f_b$, $f_c \cdot f_a$, $f_a \cdot f_b \cdot f_c$ に含まれる主項を用いて被覆する．ここで $f_a \cdot f_b \cdot f_c$ に含まれる主項 $\bar{x}_2x_1\bar{x}_0$ は f_a, $f_a \cdot f_b$, $f_c \cdot f_a$ にも主項として含まれている．このとき，$f_a \cdot f_b \cdot f_c$ に含まれる主項 $\bar{x}_2x_1\bar{x}_0$は，f_a だけでなく f_b, f_cの被覆にも利用できるため，f_a, $f_a \cdot f_b$, $f_c \cdot f_a$に含まれる $\bar{x}_2x_1\bar{x}_0$ よりも優先して被覆に利用すべきである．
- このような優先関係により被覆に用いられない主項を図7·9，図7·10 では点線で示している．

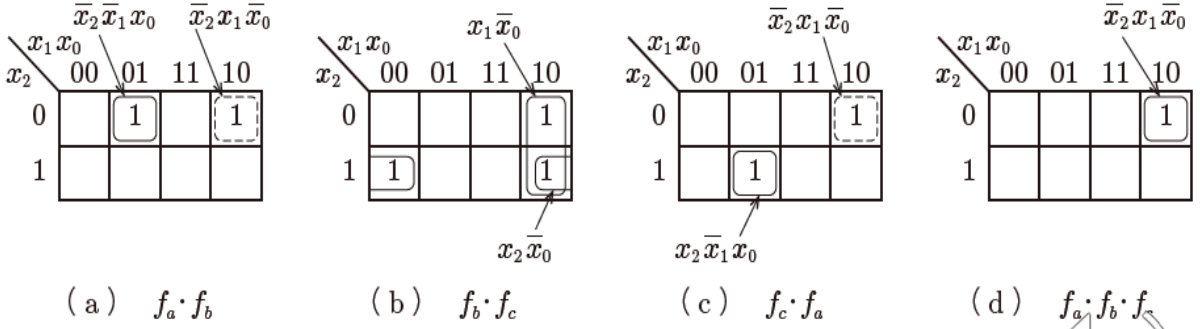


図 7・10　例題 7.3 の $f_a \cdot f_b, f_b \cdot f_c, f_c \cdot f_a, f_a \cdot f_b \cdot f_c$ のカルノー図

多出力論理回路の簡単化

例題７・３

- クワイン・マクラスキー法で用いた主項表による最小被覆の導出を適用する（図7·11）.
- f_a, f_b, f_c それぞれについて必須行を探し出して，最小被覆に加える.
- ここでは 図7·11(a) の左に必須と書いた行が該当する（縦の列に１が一つしか書かれていない場合，その最小項を覆う主項が必須項）.

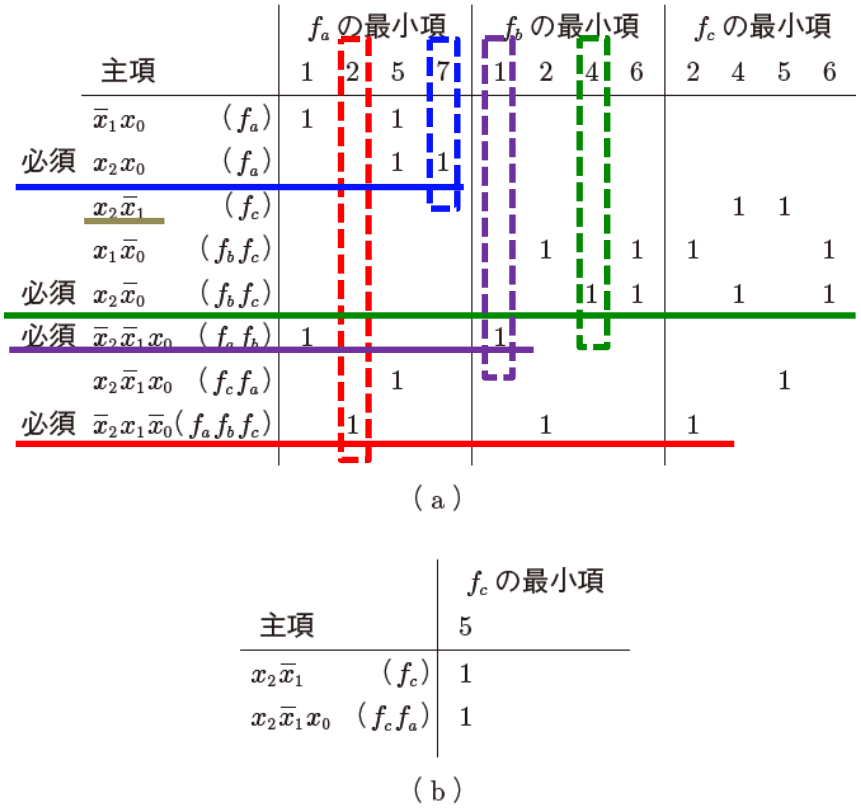


図 7・11　例題 7·3 の主項表を用いた最小被覆の導出

多出力論理回路の簡単化

例題７・３

- クワイン・マクラスキー法で用いた主項表による最小被覆の導出を適用する（図7·11）.
- f_a, f_b, f_c それぞれについて必須行を探し出して，最小被覆に加える.
- ここでは 図7·11(a) の左に必須と書いた行が該当する．これらの必須項に被覆された最小項を除くと，残された被覆すべき最小項は f_c の 5 のみである.
- この最小項を被覆可能な主項のうち，リテラル数の小さい $x_2\bar{x}_1$ を選択する.
- この結果 f_a, f_b, f_c の最簡多出力論理関数は以下のようになる（論理積５個，論理和３個に減少する）.
  - $f_a = x_2x_0 \vee \bar{x}_2\bar{x}_1x_0 \vee \bar{x}_2x_1\bar{x}_0$
  - $f_b = x_2\bar{x}_0 \vee \bar{x}_2\bar{x}_1x_0 \vee \bar{x}_2x_1\bar{x}_0$
  - $f_c = x_2\bar{x}_0 \vee x_2\bar{x}_1 \vee \bar{x}_2x_1\bar{x}_0$

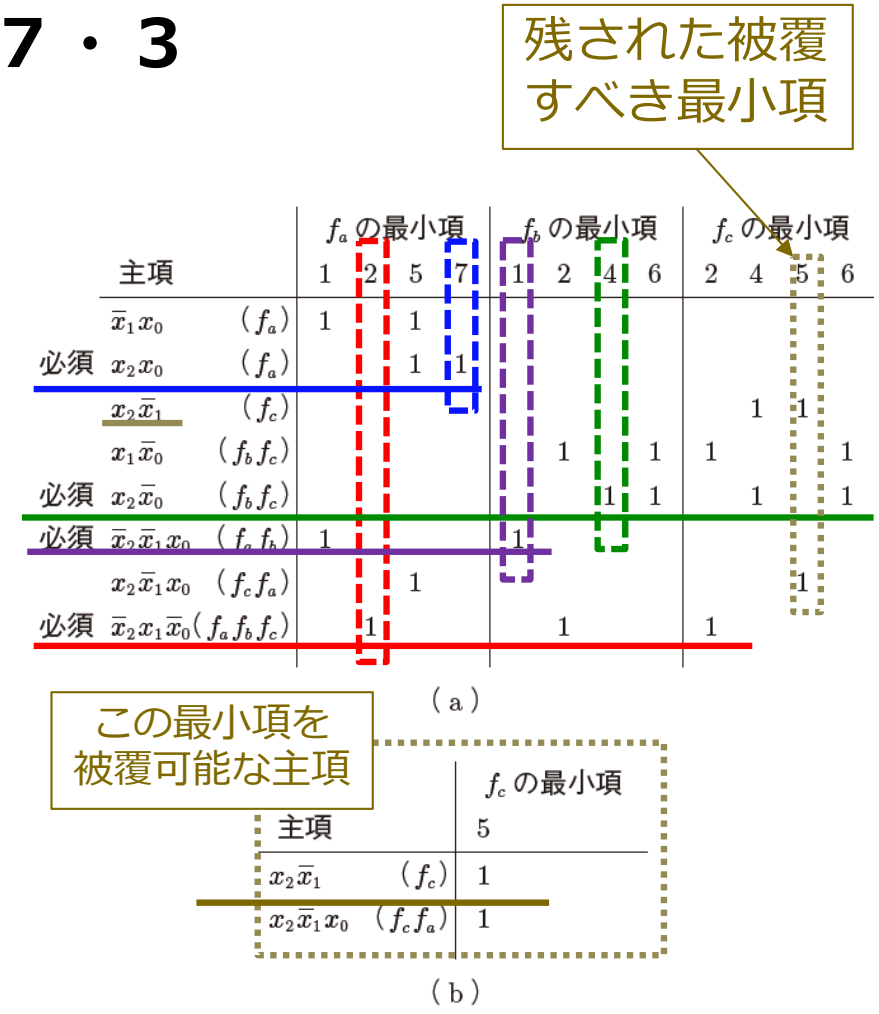


図 7・11　例題 7.3 の主項表を用いた最小被覆の導出

状態遷移図の作り方とDフリップ・フロップを用いた同期式順序回路の生成

順序回路の設計の流れ

- 順序回路の設計は，一般に次の順序で行う．
  1. 状態遷移図，状態遷移出力表の作成
  2. 状態割当ての決定
  3. 状態遷移関数，出力関数の実現
  4. 論理ゲートを使った順序回路の実現
- 文字列検出回路を用いて説明する．文字列検出回路とは，ある特定のパターンが入力されたときに，そのパターンの入力検出を行う回路である．例えば，クロックに同期して入力 x が入ってくるときに，パターン “110” を見つけて，出力を 1とする回路である．初期値としては 0 が入力されているとする．
- 時刻，入力パターン x およびそれに対する出力 z の例を以下に示す．x, z の初期値は 0 とする．
  - 時刻 t: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 …
  - 入力 x: 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0…
  - 出力 z: 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 …
- 本順序回路では，パターン “110” が見つかった 7 クロック後および 16 クロック後に出力 z は 1 となっている．パターン “110” の最後の文字 0 が入力されると同時に 1 を出力しているため，Mealy 型順序回路で実現できる．

1. 状態遷移図の作成
例題11・1

- パターン“110” を見つけるための文字列検出回路を Mealy型順序回路として実現せよ.
- パターン “110” の初めの 2 文字目までのパターン “11” を見つけ，その後 ‘0’ を見つける方針で検出回路を設計する.
- この動作を実現するために，‘1’ が 1 回現れた状態である S_1，‘1’ が 2 回以上連続して現れた状態である S_2，その他の状態 S_0 の 3 状態を定義する.
- 本状態遷移は Mealy型順序機械として実現すると，図11·5 の状態遷移図を得る．また，このときの状態遷移出力表を表11·3 に示す.
- 図11·5 では，11 の後に 0 が入力されるとすぐに 1 を出力する順序回路が得られる.

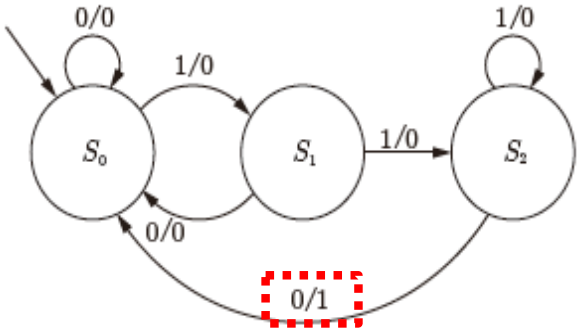


図 11・5 “110” 文字列検出回路（Mealy 型）

表 11・3 “110” 文字列検出回路の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
	$x=0$	$x=1$	$x=0$	$x=1$
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_0	S_2	1	0

2. 状態割当ての決定

- 次に状態をフリップフロップを用いて実現するため，状態を表す変数を割当てる.
- 今回のMealy 型順序機械の場合，3 状態を使用するので，3 状態を表現するために最小でも 2 ビットを割り当てる必要がある．ここでは，表11·4 のように状態を割り当てるものとする.
- なお，状態を表す変数の割当て方法は一通りではなく複数の方法があり，回路の大きさなども変化する.

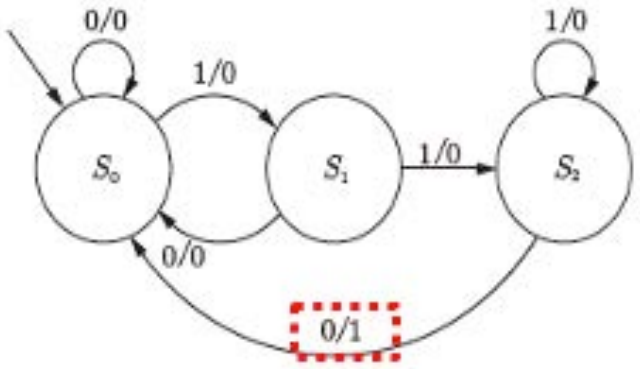


図 11・5 "110" 文字列検出回路（Mealy 型）

表 11・4 状態割当て（Mealy 型）

状態	状態割当て
S_0	00
S_1	01
S_2	11
使用しない	10

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.

表 11・3 "110" 文字列検出回路の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
	$x=0$	$x=1$	$x=0$	$x=1$
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_0	S_2	1	0

表 11・5 状態割当て決定後の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
$q_1 q_0$	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	00	11	0	0
11	00	11	1	

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.

表 11・5　状態割当て決定後の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
$q_1 q_0$	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	00	11	0	0
11	00	11	1	0

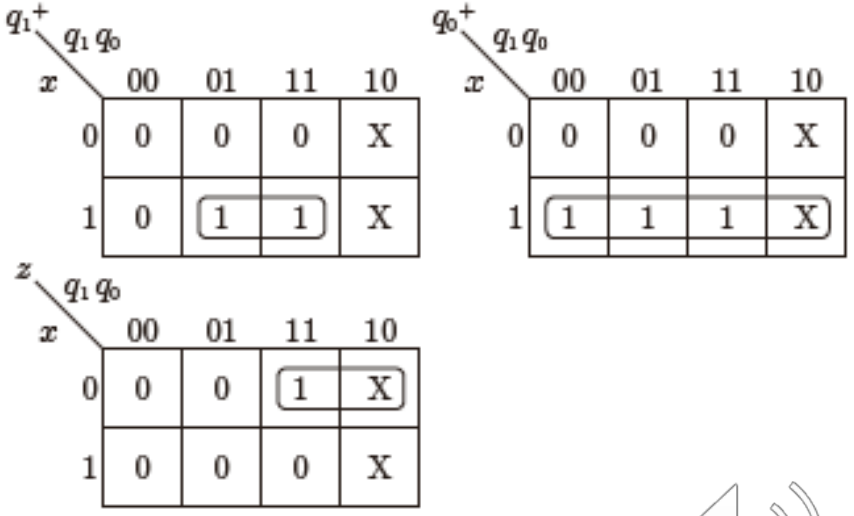


図 11・6　カルノー図（Mealy 型）

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.

表 11・5　状態割当て決定後の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
$q_1 q_0$	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	00	11	0	0
11	00	11	1	0

q_1^+　q_1^+

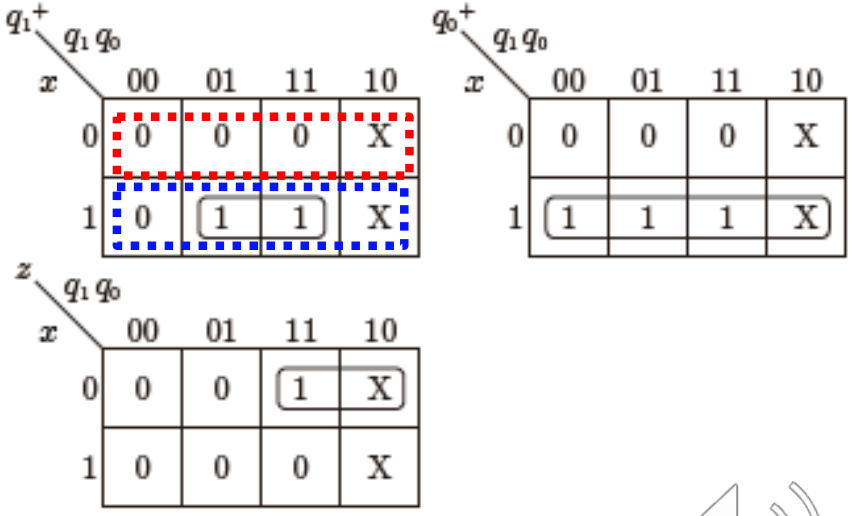


図 11・6　カルノー図（Mealy 型）

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.

表 11・5　状態割当て決定後の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
$q_1 q_0$	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	00	11	0	0
11	00	11	1	0

q_0^+　　q_0^+

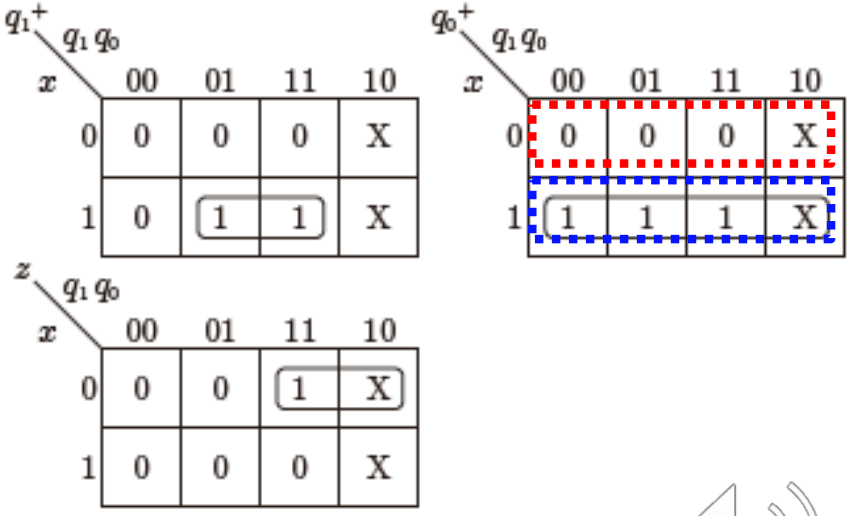


図 11・6　カルノー図（Mealy 型）

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.

表 11・5　状態割当て決定後の状態遷移出力表（Mealy 型）

現在の状態	次の状態		出力	
q_1q_0	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	00	11	0	0
11	00	11	1	0

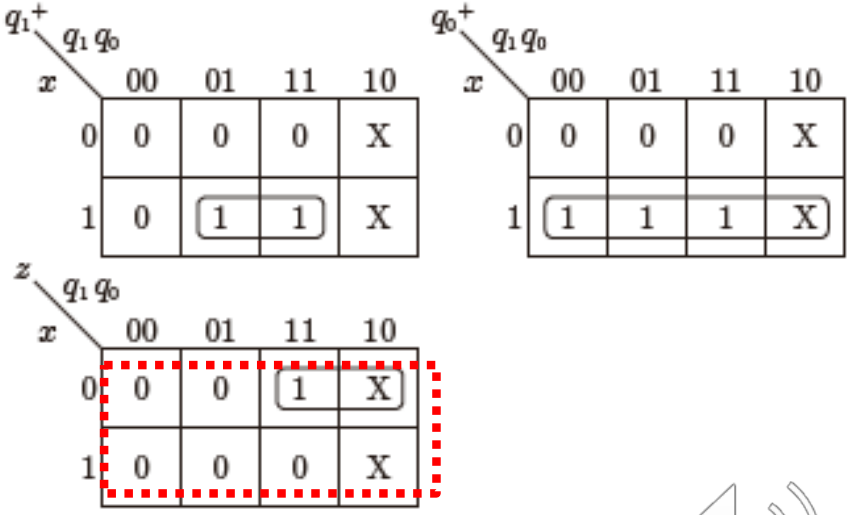


図 11・6　カルノー図（Mealy 型）

3.状態遷移関数，出力関数の実現

- Mealy型順序機械の状態割当て決定後の状態遷移出力表を表11･5 に示す.
- ここで，状態を表すための変数 q_1, q_0 を導入した．次の時刻の状態を表現する状態変数を q_1^+, q_0^+ と表すと，状態変数q_1^+, q_0^+ および出力 z は，現在時刻の状態変数 q_1, q_0 および入力 x を使った論理関数として表現できる．状態遷移出力表から， q_1^+, q_0^+, z に関するカルノー図を作成すると図11･6 のカルノー図が得られる．10 は状態割当てされていないため， q_1, q_0 の論理式を決める際にはドントケアの組合せとなり，図11･6 中では，‘X’ と表されている.
- 図11･6 のカルノー図より，状態遷移関数，出力関数を求めると

$$q_1^+ = x \cdot q_0$$
$$q_0^+ = x$$
$$z = \bar{x} \cdot q_1$$

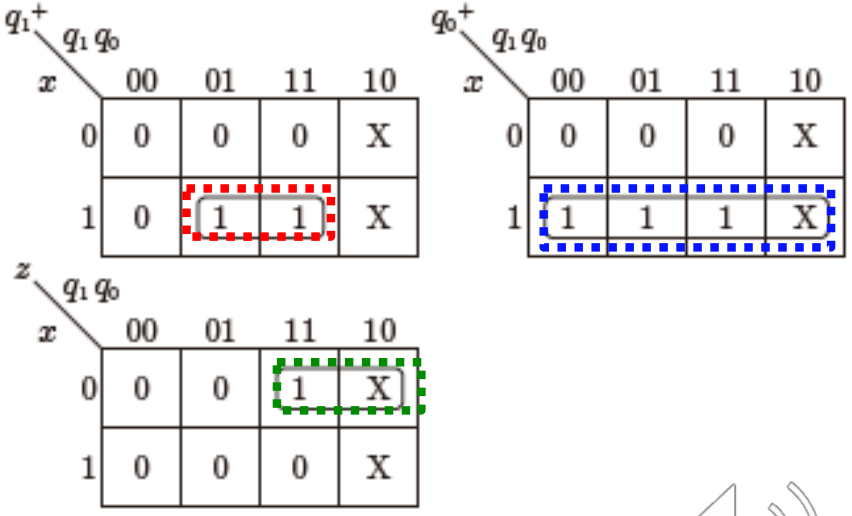


図 11・6 カルノー図（Mealy 型）

論理ゲートを使った順序回路の実現

- 状態遷移関数，出力関数が求まっているので，それらを実現する論理回路を構成する．記憶要素としては，状態変数の個数分の Dフリップフロップを用いる．
- 本文字列検出回路では，状態保持するために 2ビットの変数を使用するので，Dフリップフロップを 2 個使用する．
- 初期状態は動作開始前に Dフリップフロップのリセット端子 RST に ‘1’ の信号を与え，Dフリップフロップを初期化し，状態を “00” としている．
- Mealy 型順序回路で実現した回路構成を図11·7に示す．

$$q_1^+ = x \cdot q_0$$
$$q_0^+ = x$$
$$z = \bar{x} \cdot q_1$$

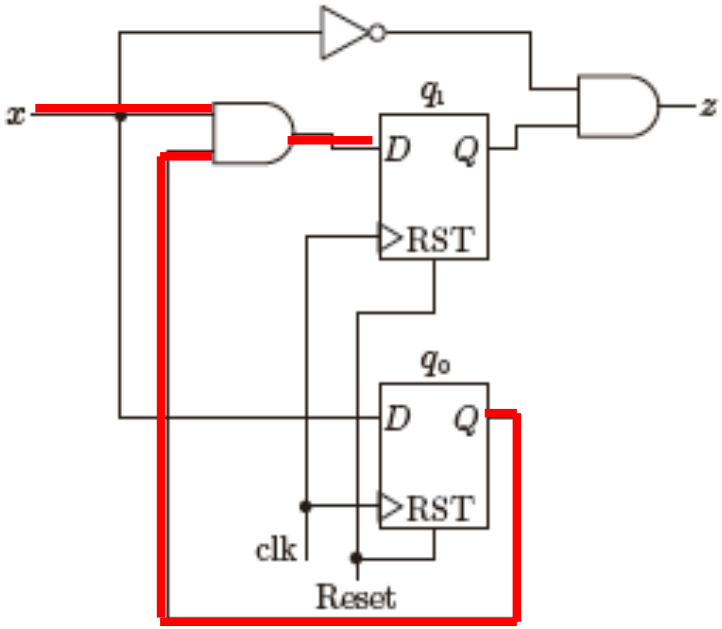


図 11・7　順序回路（Mealy 型）

論理ゲートを使った順序回路の実現

- 状態遷移関数，出力関数が求まっているので，それらを実現する論理回路を構成する．記憶要素としては，状態変数の個数分の Dフリップフロップを用いる．
- 本文字列検出回路では，状態保持するために 2ビットの変数を使用するので，Dフリップフロップを 2 個使用する．
- 初期状態は動作開始前に Dフリップフロップのリセット端子 RST に ‘1’ の信号を与え，Dフリップフロップを初期化し，状態を “00” としている．
- Mealy 型順序回路で実現した回路構成を図11·7に示す．

$$q_1^+ = x \cdot q_0$$
$$q_0^+ = x$$
$$z = \bar{x} \cdot q_1$$

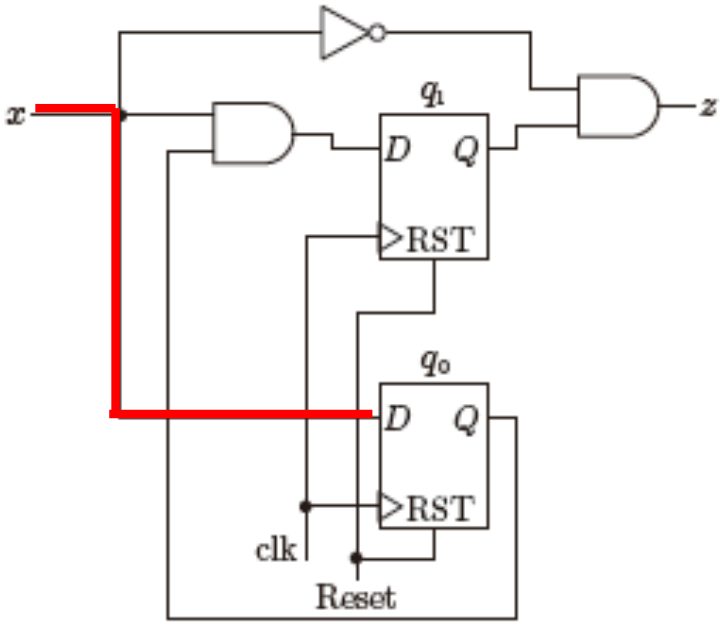


図 11・7　順序回路（Mealy 型）

論理ゲートを使った順序回路の実現

- 状態遷移関数，出力関数が求まっているので，それらを実現する論理回路を構成する．記憶要素としては，状態変数の個数分の Dフリップフロップを用いる．
- 本文字列検出回路では，状態保持するために 2ビットの変数を使用するので，Dフリップフロップを 2 個使用する．
- 初期状態は動作開始前に Dフリップフロップのリセット端子 RST に ‘1’ の信号を与え，Dフリップフロップを初期化し，状態を “00” としている．
- Mealy 型順序回路で実現した回路構成を図11·7に示す．

$$q_1^+ = x \cdot q_0$$
$$q_0^+ = x$$
$$z = \bar{x} \cdot q_1$$

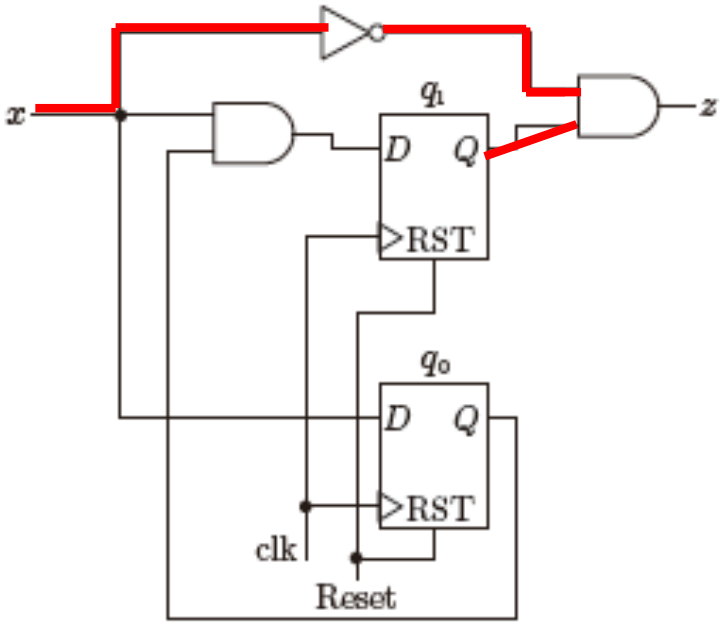


図 11・7　順序回路（Mealy 型）

OSAKA UNIVERSITY

中間試験対策

中間試験対策

- 試験範囲
  - 論理回路の教科書の１章～１１章（基本的に中間試験の対象は，論理設計の１回目から６回目の授業で説明した６章から１１章の内容）
- 中間試験対策
  - カルノー図（ドント・ケアを含むカルノー図）の簡単化
  - クワイン・マクラスキー法を用いた主項や最簡積和形の生成
  - 多出力論理関数の簡単化
  - 状態遷移図の作り方と Dフリップ・フロップ を用いた同期式順序回路の生成
    - 同期式順序回路の動きを状態遷移図（Mealy型とMoore型の両方）で表現
    - Dフリップ・フロップを用いて与えられた状態遷移図を実現する同期式順序回路を生成

OSAKA UNIVERSITY

７回目の授業終了

授業終了

皆さん
今日はレポート課題はありません