Reti Combinatorie

Giovanni Stea

Reti Logiche

Corso di Laurea in Ingegneria Informatica, Universita' di Pisa

a.a. 2020/21

Materiale di riferimento

- Paolo Corsini, "Dalle porte AND, OR, NOT al Sistema calcolatore", edizioni ETS

- Appunti sulle reti combinatorie - versione 25/09/2020 (con esercizi svolti)

Rete logica come modello astratto

- Sistema fisico

- Modello astratto

0 1 binary digit

bit

circuiti elettronici

Rete logica come modello astratto

- Una rete logica è un **modello astratto** di un **sistema fisico**, quest'ultimo costituito da **dispositivi tra loro interconnessi**.
- Tali dispositivi si scambiano **informazioni codificate**.
- Le informazioni sono codificate tramite **fenomeni fisici** che si presentano ad un osservatore in **due aspetti distinti**.
- Ad esempio:
  - corrente forte, corrente debole
  - tensione alta, tensione bassa
  - perforazione di una zona di un foglio di carta, assenza di perforazione
  - magnetizzazione positiva/negativa di un'areola di materiale ferromagnetico
  - …

Caratterizzare una rete logica

- Una rete logica è caratterizzata da:

1. un insieme di **N variabili logiche di ingresso**. Il loro valore all'istante *t* si chiama **stato di ingresso all'istante *t***. L'insieme di tutti i 2^N stati di ingresso verrà indicato con $\boldsymbol{X}$. $\boldsymbol{X} = \{(x_{N-1}x_{N-2}\ldots x_1x_0)\}$.

2. un insieme di **M variabili logiche di uscita**. Il loro valore all'istante *t* si chiama **stato di uscita all'istante *t***. L'insieme di tutti i 2^M stati di uscita verrà indicato con $\boldsymbol{Z}$.$\boldsymbol{Z} = \{(z_{M-1}z_{M-2}\ldots z_1z_0)\}$.

3. una **legge di evoluzione nel tempo**, che dice come le uscite si evolvono in funzione degli ingressi

Classificazione delle reti logiche

$z = F(x)$

clock

- in base a **due criteri** riguardanti il tipo di legge di evoluzione nel tempo

a) **presenza/assenza di memoria:**

presente

- reti **combinatorie**: lo stato di uscita dipende **soltanto** dallo stato di ingresso. Ad un certo stato di ingresso corrisponde uno ed un solo stato di uscita. Analogia con il concetto di **funzione matematica**.
- reti **sequenziali**: lo stato di uscita dipende dalla **storia** degli stati di ingresso precedenti. Queste ultime sono **reti con memoria**, in quanto per decidere quale sia l'uscita hanno bisogno di **ricordare** il passato.

b) **temporizzazione della legge di evoluzione**

$\forall t$

- reti **asincrone**, in cui l'aggiornamento delle uscite avviene continuamente nel tempo
- reti **sincronizzate**, in cui l'aggiornamento delle uscite avviene ad istanti (di sincronizzazione) separati nel tempo

Classificazione delle reti logiche

	asincrono	**sincronizzato**
combinatorio	1) Reti combinatorie	(non hanno un nome, sotto-caso del caso 3)
sequenziale	2) Reti sequenziali asincrone	3) Reti sequenziali sincronizzate

Interfaccia, intepretazione e fisica

- Una rete logica comunica con l'esterno tramite **variabili logiche** (0 e 1)
- La descrizione che ne faremo non dipende da:
  - L'interpretazione di queste variabili logiche (e.g., come numero intero, colore di un pixel, etc.) – che sta nella testa di chi le produce/consuma
  - Il fenomeno fisico con cui si codificano

- Useremo **in genere** questo modello per descrivere **circuiti elettronici** all'interno del calcolatore
  - Le variabili logiche sono codificate come **tensioni** (bassa: 0, alta: 1)
  - Scala di tensione dipende dall'implementazione (5V, 3V…)

Limiti del modello

• massa
→ dimensioni / struttura interna

- Un modello è un modo **semplificato** di descrivere alcuni aspetti di una realtà complessa.
  - Fisica: **punto materiale**
  - Elettrotecnica: **un circuito a parametri concentrati**

- Tutte le volte che si modella un sistema, si **trascurano** alcuni suoi aspetti
- Necessario conoscere i **limiti** del modello

Transizione dei segnali

- all'istante t_0 la variabile logica w **si setta**, oppure transisce ad 1, etc.
- all'istante t_1 la variabile logica w **si resetta**, oppure transisce a 0, etc.

valore di regime

Non contemporaneita'

- Il ragionamento di prima non implica che si possano assumere variazioni **contemporanee** di piu' variabili

$x_1 x_0$

$$\boldsymbol{X} = \{(00), (10), (01), (11)\}, \boldsymbol{Z} = \{0,1\}$$

- Supponiamo che **prima dell'istante t_1** lo stato di ingresso presente sia, ad esempio, (1,0), e che ***all'istante* t_1** lo stato di ingresso diventi (0,1).

Non contemporaneita'

Ciò che *secondo me* accade in ingresso

Ciò che la rete vede (esempio)

Non contemporaneita'

- **Evitare di supporre che due variabili di ingresso varino contemporaneamente**
  - Nessun **sistema** obbedira' mai a questa ipotesi
  - Per un certo intervallo di tempo vedra' in ingresso uno stato intermedio
- Rete **sincronizzata**, -> non e' un problema, in genere
- Rete **asincrona**
  - Potrebbe presentare in uscita stati **spuri**
  - Se **sequenziale**, si evolve in modo **imprevedibile**

Pibtaggio

differiscono per 1 bit

- **Stati di ingresso consecutivi devono essere adiacenti**

Reti Combinatorie

Caratterizzare una rete combinatoria

- Una rete combinatoria è caratterizzata da:

1. Un insieme di **N variabili logiche di ingresso**.
2. Un insieme di **M variabili logiche di uscita**.
3. Una **descrizione funzionale** $F: \boldsymbol{X} \rightarrow \boldsymbol{Z}$, che mappa stati di ingresso in stati di uscita
4. Una **legge di evoluzione nel tempo**, che dice: continuamente, se $\boldsymbol{X}$ e' lo stato di ingresso presente, adegua lo stato di uscita ad $F(\boldsymbol{X})$

Tempo di attraversamento

invertitore

$N = 1, M = 1, \boldsymbol{X} = \{0,1\}, \boldsymbol{Z} = \{0,1\}$

$F(0)=1, F(1)=0$

- Tempo di **accesso**, o di **risposta**
- Il tempo che ci mette la rete ad adeguare lo stato di uscita al nuovo stato di ingresso
- **Non nullo**

1) bit di ingresso si cambiano 1 alla volta

2)

- Necessario attendere che l'uscita sia andata **a regime** prima di variare ancora lo stato di ingresso

Pilotaggio in modo fondamentale

ns

Descrizione di reti combinatorie

- Una rete combinatoria si descrive dicendo:
  - quanti sono gli **ingressi**;
  - quante sono le **uscite**;
  - qual è la **funzione *F***, cioè la descrizione funzionale

- **a parole**
- **notazioni testuali** (vedremo il linguaggio **Verilog**)
- **tabelle di verità**

Tabella di verita'

non specificate
don't care

X Z

2^3

x_2	x_1	x_0	z_1	z_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

x_2	x_1	x_0	z_1	z_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	-	0
1	0	0	1	-
1	0	1	1	1
1	1	0	0	-
1	1	1	-	-

z_1 "riconosce"

Descrizione e sintesi

- La **descrizione** di una rete è un modo **formale** per dire **che cosa fa** quella rete, qual è cioè il suo comportamento osservabile.
  - Una tabella di verità.
- La **sintesi** di una rete è il **progetto della realizzazione** di una rete
  - la decisione di quali "scatolette" vanno messe, e connesse come, in modo da far sì che la rete si comporti come indicato dalla descrizione

realizzazione implementazione

- Di una rete si da' **prima** la descrizione, e **poi** la sintesi

tutte $\forall N$ $\forall M$ $\forall F(\ )$

Una proprieta' delle reti combinatorie

- **Una rete combinatoria ad N ingressi ed M uscite può essere realizzata interconnettendo M reti combinatorie ad N ingressi ed una uscita**

- Possiamo limitarci a trattare **reti con una sola uscita**

Reti combinatorie elementari: zero ingressi

- Generatori di costante (caso degenere)

Complessità nulla.

Reti combinatorie ad un ingresso

- Invertitore

x	z
0	1
1	0

- Elemento neutro buffer

x	z
0	0
1	1

- genera ritardo (utile per le temporizzazioni)
- **rigenera** i segnali elettrici degradati.

AND a 2^N ingressi

- Disporre le porte ad **albero binario bilanciato**
- Numero di LL attraversati e' minimo
- Stesso ragionamento vale per le OR (fare per casa)

NOR NAND

XOR

Reti comb.

"complesse"

⇑

reti comb. "semplici"

Reti combinatorie a due ingressi

- Quante sono in tutto?
- Tante quante le possibili **tabelle di verita'** diverse che posso costruire con due ingressi

- Una TdV per rete con N ingressi ha 2^N righe
- Le possibili diverse tabelle di verita' sono 2^{2^N}
- Quindi per N=2, abbiamo 16 diverse reti

N

X_{N-1}	X_{N-2}	...	X_1	X_0	$Z^{(1)}$	$Z^{(2)}$	...	$Z^{(?)}$	$Z^{(?)}$
0	0	...	0	0	0	0	...	1	1
0	0	...	0	1	0	0	...	1	1
1	1	...	1	0	0	0	...	1	1
1	1	...	1	1	0	1	...	0	1

2^N

$2^{(2^N)}$

Reti combinatorie a due ingressi

porte

- Ad alcune corrispondono nomi speciali

x_1	x_0	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$	$z^{(4)}$	$z^{(5)}$	$z^{(6)}$	$z^{(7)}$	$z^{(8)}$	$z^{(9)}$	$z^{(10)}$	$z^{(11)}$	$z^{(12)}$	$z^{(13)}$	$z^{(14)}$	$z^{(15)}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		(*)	(A)		(+)		(+)	(B)	(C)	(D)	(E)	(-)		(-)		(F)	(*)

AND

x_1 x_0 NOT NOT

A) porta **AND**: $z = 1 \Leftrightarrow x_0 = x_1 = 1$

B) porta **XOR**: $z = 1 \Leftrightarrow x_0 \neq x_1$

C) porta **OR**: $z = 0 \Leftrightarrow x_0 = x_1 = 0$

F) porta **NAND**: $z = 0 \Leftrightarrow x_0 = x_1 = 1$

E) porta **XNOR**: $z = 1 \Leftrightarrow x_0 = x_1$

D) porta **NOR**: $z = 1 \Leftrightarrow x_0 = x_1 = 0$

AND e OR a piu' ingressi

- Posso pensare di **estendere** la funzione logica realizzata dalle porte AND e OR al caso di *N* ingressi.
  - AND ad *N* ingressi: l'uscita vale 1 se e solo se **tutti gli ingressi valgono 1**
  - OR ad *N* ingressi: l'uscita vale 1 se e solo se **almeno un ingresso vale 1**

x_2	x_1	x_0	y	z
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

AND e OR a piu' ingressi

- Modo poco furbo

8

AND 2 ingressi

- Modo piu' furbo

- Tre livelli di logica

- Due livelli di logica
  - Tempo di attraversamento minore

AND a 2^N ingressi

- Disporre le porte ad **albero binario bilanciato**
- Numero di LL attraversati e' minimo
- Stesso ragionamento vale per le OR (fare per casa)

NOR NAND

XOR

Reti comb.

"complesse"

⇑

reti comb. "semplici"

AND e OR con molti ingressi

- Rappresentazione grafica

Algebra di Boole

- Formalismo per rappresentare le leggi di corrispondenza delle reti combinatorie
- Sistema algebrico basato su:
  - **variabili logiche**, capaci di assumere due valori (0 e 1)
  - **operatori logici**: si applicano alle variabili logiche per costruire espressioni algebriche

Operatori logici

- **Complemento:** Operatore unario. Si indica con: $\overline{x}$, (anche: $!\,x$, $/x$). È definito come: $\overline{0} = 1$, $\overline{1} = 0$.

- **Prodotto logico:** $x \cdot y$, definito come

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

- **Somma logica:** $x + y$, definita come

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

Proprieta' degli operatori booleani

- Alcune sono ovvie, altre molto meno

Proprietà degli operatori booleani
Involutiva (del complemento): $\bar{\bar{x}} = x$
commutativa (della somma e del prodotto): $x + y = y + x,\ x \cdot y = y \cdot x$
associativa (della somma e del prodotto): $x + y + z = (x + y) + z = x + (y + z)$ $x \cdot y \cdot z = (x \cdot y) \cdot z = x \cdot (y \cdot z)$
distributiva della somma rispetto al prodotto **e viceversa**: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ $x + (y \cdot z) = (x + y) \cdot (x + z)$ (disegnare tabella di verità)

x	y	z	$y \cdot z$	$x+(y \cdot z)$	$(x+y)$	$(x+z)$	$(x+y) \cdot (x+z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Esercizio

$\bar{a} \cdot \bar{c} \cdot \bar{d} \cdot (\cancel{b + b})$

$\alpha \cdot 1$

$\rightarrow x + x = x$

- Utilizzando le proprietà dell'algebra, semplificare al massimo la seguente espressione

$$\bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \cdot \bar{d} + a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot d + b \cdot c \cdot d + a \cdot b \cdot d$$

$+ \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$

$= \bar{a} \cdot \bar{c} \cdot \bar{d} + \bar{b} \cdot \bar{c} \cdot \bar{d} + b \cdot d + b \cdot c \cdot d$

$b \cdot d \cdot \cancel{(1 + c)}$

$b + c + d$

$= \bar{a} \cdot \bar{c} \cdot \bar{d} + \bar{b} \cdot \bar{c} \cdot \bar{d} + b \cdot d$

sintesi di costo minimo

Teoremi di De Morgan per N variabili logche

- I Teoremi di De Morgan valgono per un numero qualunque di variabili logiche.

1. $\overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}}$

2. $\overline{x_0 + x_1 + \ldots + x_{N-1}} = \overline{x_0} \cdot \overline{x_1} \cdot \ldots \cdot \overline{x_{N-1}}$

- Si dimostrano **per induzione** sul n. di variabili logiche
  - 1) dimostrare che la proprietà vale per un certo numero n_0 (passo base);
  - 2) dimostrare che, se vale per un generico $n \geq n_0$, allora vale anche per $n + 1$ (passo induttivo)

Dimostrazione della prima tesi

- Tesi: $\overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}}$
- Passo base: $n_0 = 2$: gia' fatto (tabella di verita')
- Passo induttivo:
  - Hp: $\overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}}$
  - Th: $\overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1} \cdot x_N} = \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}} + \overline{x_N}$
- Pongo $x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1} = \alpha$
- $\overline{\alpha} = \overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}}$ (ipotesi induttiva)
- $\overline{x_0 \cdot x_1 \cdot \ldots \cdot x_{N-1}} \cdot x_N = \overline{\alpha \cdot x_N} = \overline{\alpha} + \overline{x_N}$ (passo base)
- $= \overline{x_0} + \overline{x_1} + \ldots + \overline{x_{N-1}} + \overline{x_N}$

Algebra di Boole e reti combinatorie

- **data una rete combinatoria** (comunque complessa), è sempre possibile trovare un'espressione booleana che mette in relazione ogni sua uscita con gli ingressi (un'espressione per ogni uscita, in realtà);
- **data un'espressione booleana**, è sempre possibile sintetizzare una rete combinatoria (ad un'uscita) in cui la relazione tra ingresso ed uscita è data dall'espressione.

$$z = (x_1 + \overline{x_0}) \cdot (\overline{x_1} + (x_3 \cdot x_2))$$

Algebra di Boole e reti combinatorie (cont.)

- Espressioni logiche equivalenti ⇔ reti logiche che svolgono lo stesso compito

$x + (y \cdot z)$

=

$(z + x) \cdot (y + x)$

- Non necessariamente **con lo stesso costo**
  - Le porte logiche **dissipano energia, costano, si rompono**, ritardo
  - Meno ne mettiamo, meglio e'
- Dobbiamo trovare il modo per **minimizzare il costo** della **sintesi** di una data legge di corrispondenza

Esercizio

$\bar{a} \cdot \bar{c} \cdot \bar{d} \cdot (\cancel{b + b})$

$\alpha \cdot 1$

$\rightarrow x + x = x$

- Utilizzando le proprietà dell'algebra, semplificare al massimo la seguente espressione

$$\bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \cdot \bar{d} + a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot d + b \cdot c \cdot d + a \cdot b \cdot d$$

$+ \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$

$= \bar{a} \cdot \bar{c} \cdot \bar{d} + \bar{b} \cdot \bar{c} \cdot \bar{d} + b \cdot d + b \cdot c \cdot d$

$b \cdot d \cdot \cancel{(1 + c)}$

$b + c + d$

$= \bar{a} \cdot \bar{c} \cdot \bar{d} + \bar{b} \cdot \bar{c} \cdot \bar{d} + b \cdot d$

$\rightarrow$ sintesi di costo minimo

Decoder

- È una rete con N ingressi e p uscite, con $p = 2^N$.

2^N

- legge di corrispondenza (a parole): "ogni uscita riconosce uno ed un solo stato di ingresso, in particolare **l'uscita *j*-sima riconosce lo stato di ingresso i cui bit sono la codifica di *j* in base 2**, cioè se $(x_{N-1}x_{N-2}\ldots x_1x_0)_{b2} \equiv j$"

Esempio: Decoder 2 to 4

x_1	x_0	z_0	z_1	z_2	z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

desc

- Come si sintetizza una rete descritta da questa tabella di verita'?
- Prendo in esame l'uscita $\mathbf{z_3}$

$$z_3 = \begin{cases} 1 & x_1 x_0 = 11 \\ 0 & altrimenti \end{cases}$$

$$z_3 = x_1 \cdot x_0$$

Esempio: Decoder 2 to 4

$x_1 x_0 = 01 \iff \overline{x_1} x_0 = 11$

x_1	x_0	z_0	z_1	z_2	z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$z_1 = \overline{x_1} \cdot x_0$

$z_0 = \overline{x_1} \cdot \overline{x_0}$

$x_1 x_0 = 00$

$\overline{x_1}\,\overline{x_0} = 11$

- Prendo in esame l'uscita $\mathbf{z_2}$

$z_2 = x_1 \cdot \overline{x_0}$

$$z_2 = \begin{cases} 1 & x_1\overline{x_0} = 10 \\ 0 & altrimenti \end{cases} \Rightarrow z_2 = \begin{cases} 1 & x_1\overline{x_0} = 11 \\ 0 & altrimenti \end{cases} \Rightarrow$$

Lo stesso ragionamento vale per le altre uscite: $z_1 = \overline{x_1} \cdot x_0$, $z_0 = \overline{x_1} \cdot \overline{x_0}$.

Sintesi del decoder 2 to 4

- $z_3 = x_1 \cdot x_0$
- $z_2 = x_1 \cdot \overline{x_0}$
- $z_1 = \overline{x_1} \cdot x_0$
- $z_0 = \overline{x_1} \cdot \overline{x_0}$

Decoder N to 2^N

$2^N = p$

$$z_0 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot \overline{x_0}$$

$$z_1 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot x_0$$

$$\ldots$$

$$z_{p-2} = x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot \overline{x_0}$$

$$z_{p-1} = x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot x_0$$

Esempio: Decoder 1 to 2

- Caso particolare, non richiede logica

x	z_0	z_1
0	1	0
1	0	1

Decoder con enabler (espandibile)

- Il decoder che abbiamo appena descritto **non è espandibile**.
- Non posso costruire decoder **grandi** combinando decoder **più piccoli**.
- Per questo motivo, esistono decoder dotati di un ingresso aggiuntivo, detto **di abilitazione** (**enabler**)

- Ha $N + 1$ ingressi e 2^N uscite
- **se l'ingresso di enabler è 1, la rete si comporta come un decoder N to 2^N. Altrimenti, tutte le uscite sono a 0**
- l'ingresso di abilitazione **"accende"** il decoder

Sintesi del decoder con enabler

- Prendiamo un decoder **senza enabler**, e chiamiamo $y_0 \dots y_{p-1}$ le sue uscite
- Ciascuna uscita z_i vale 0 se $e = 0$, altrimenti è uguale ad y_i.

$$z_i = \begin{cases} y_i & e = 1 \\ 0 & e = 0 \end{cases}$$

$z_i = y_i \cdot e$

- Dato che: $x \cdot 0 = 0$, $x \cdot 1 = x$

$$z_i = y_i \cdot e$$

Sintesi del decoder con enabler

- Due porte AND in sequenza sono uno spreco
- Meglio aggiungere un ingresso (l'AND e' associativo)

$$z_0 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot \overline{x_0}$$

$$z_1 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot x_0$$

$$\ldots$$

$$z_{p-2} = e \cdot x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot \overline{x_0}$$

$$z_{p-1} = e \cdot x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot x_0$$

Costruzione di decoder 4to16 da decoder 2to4

x_3	x_2	x_1	x_0	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}	
0	0	0	0	1	0	0	0													
0	0	0	1	0	1	0	0		0				0				0			$e_2=0$
0	0	1	0	0	0	1	0													
0	0	1	1	0	0	0	1													
0	1	0	0					1	0	0	0									
0	1	0	1		0			0	1	0	0		0				0			$e_2=0$
0	1	1	0					0	0	1	0									
0	1	1	1					0	0	0	1									
1	0	0	0									1	0	0	0					
1	0	0	1		0				0			0	1	0	0		0			$e_2=1$
1	0	1	0									0	0	1	0					
1	0	1	1									0	0	0	1					
1	1	0	0													1	0	0	0	
1	1	0	1		0				0				0			0	1	0	0	$e_2=\emptyset$
1	1	1	0													0	0	1	0	
1	1	1	1													0	0	0	1	

$e=1$

$e_1=\emptyset$

$e_1=1$

$e=\emptyset$

$e_1=\emptyset$

Costruzione di decoder 4to16 da decoder 2to4

Costruzione di decoder 4to16 da decoder 2to4

Demultiplexer

- Rete con $N + 1$ ingressi e $p = 2^N$ uscite
  - x si chiama **variabile da commutare**,
  - gli altri si chiamano **variabili di comando**
- la *j*-sima uscita insegue la variabile da commutare se e solo se

$$(\boldsymbol{b_{N-1} b_{N-2} \ldots b_1 b_0})_2 \equiv \boldsymbol{j}$$

- altrimenti vale 0

$\overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \ldots \cdot \overline{b_1} \cdot \overline{b_0} = 1$

Demultiplexer (cont.)

- Descrizione in termini algebrici

$\overline{b_{N-1}}, \ldots \overline{b_1}, \overline{b_0} = 11 \ldots 11$

- $z_0 = \begin{cases} x & (b_{N-1} \ldots b_1 b_0) = (0 \ldots 00) \\ 0 & altrimenti \end{cases}$

$x \cdot \alpha$

α

$(\quad)$

Demultiplexer (cont.)

- Descrizione in termini algebrici

$$z_0 = x \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \ldots \cdot \overline{b_1} \cdot \overline{b_0}$$

$$z_1 = x \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \ldots \cdot \overline{b_1} \cdot b_0$$

$$\ldots$$

$$z_{p-2} = x \cdot b_{N-1} \cdot b_{N-2} \cdot \ldots \cdot b_1 \cdot \overline{b_0}$$

$$z_{p-1} = x \cdot b_{N-1} \cdot b_{N-2} \cdot \ldots \cdot b_1 \cdot b_0$$

e

- **identica a quella di un decoder con enabler**

Multiplexer

- rete con $N + 2^N$ ingressi ed 1 uscita
- ingressi b_i si chiamano **variabili di comando**
- duale del demultiplexer
- Le var di comando selezionano l'ingresso connesso all'uscita

$$z = x_i \Leftrightarrow (b_{N-1} \ldots b_1 b_0)_2 \equiv i$$

- Mux/Demux usati (molti anni fa) nelle centrali telefoniche

descrizione sintesi

Multiplexer (cont.)

- Partiamo dalla **sintesi** (piu' semplice)
  - **decoder** $N\ to\ 2^N$, come ingressi ha le var di comando
  - metto in AND a ciascuna uscita una delle var di ingresso

- 2^N variabili logiche in uscita:
  - tutte meno una sono sicuramente nulle.
  - quell'unica che puo' non essere nulla (sia la j-sima) vale x_j

Multiplexer (cont.)

- Come faccio ad ottenere l'uscita z con queste variabili?
- Basta ricordare che $0 + \alpha = \alpha$. Infatti, di queste p variabili, $p - 1$ sono sicuramente nulle
- Metto una OR in fondo

$p = 2^N$

Multiplexer (cont.)

2LL

~~inverti ingressi~~

$\overline{b_{p-1}}$

$\overline{b_{N-1}}$

decoder

- Eliminiamo AND in cascata

Muliplexer (cont.)

14 - - 11

111 - - 10

- Dal disegno si ricava facilmente una descrizione algebrica

$$z = x_0 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \ldots \cdot \overline{b_1} \cdot \overline{b_0} +$$

$$x_1 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \ldots \cdot \overline{b_1} \cdot b_0 +$$

...

$$x_{p-2} \cdot b_{N-1} \cdot b_{N-2} \cdot \ldots \cdot b_1 \cdot \overline{b_0} +$$

$$x_{p-1} \cdot b_{N-1} \cdot b_{N-2} \cdot \ldots \cdot b_1 \cdot b_0$$

- Rete a **due livelli di logica**
- La **strada più lunga** tra ingresso e uscita passa attraverso **due** porte (AND + OR)
- le NOT sugli ingressi non si contano
  - Il motivo sara' chiaro piu' avanti (registri)

Multiplexer come rete combinatoria universale

- un multiplexer con N variabili di comando è in grado di realizzare **qualunque legge combinatoria** ad N ingressi ed un'uscita
  - basta connettere ai 2^N ingressi dei generatori di costante

j	b_2	b_1	b_0	z
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Multiplexer come rete combinatoria universale

1. Un multiplexer si realizza con porte AND, OR, NOT, ed è una rete a **due livelli di logica.**
2. Un multiplexer realizza una **qualunque** rete combinatoria ad un'uscita.
3. una rete combinatoria a più uscite può essere scomposta in reti ad un'uscita messe "in parallelo".

- Quindi:

<u>Ogni</u> rete combinatoria può essere costruita combinando AND, OR, NOT in al più <u>due</u> livelli di logica

MUX N

Esercizio

- Data una qualunque tabella di verità per rete ad N ingressi, è sempre possibile realizzare la rete che la implementa tramite:
  - un multiplexer ad $\boldsymbol{N-1}$ variabili di comando
  - al più porte NOT

x_2	x_1	x_0	z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Modello strutturale universale per reti comb.

- Vediamo adesso **un modo per sintetizzare** una rete logica ad *N* ingressi ed *M* uscite a partire da una tabella di verità.

	x_2	x_1	x_0	z_1	z_0
v_0	0	0	0	0	0
v_1	0	0	1	0	1
v_2	0	1	0	1	0
v_3	0	1	1	1	0
v_4	1	0	0	1	1
v_5	1	0	1	1	1
v_6	1	1	0	0	0
v_7	1	1	1	0	0

$$\begin{cases} z_1 = v_2 + v_3 + v_4 + v_5 \\ z_0 = v_1 + v_4 + v_5 \end{cases}$$

Ridurre il costo della sintesi

- Partiamo dalla sintesi del MSU e vediamo cosa possiamo togliere

Ridurre il costo della sintesi (cont.)

- Cerchiamo aiuto nelle proprieta' dell'Algebra di Boole

Forma CANONICA SP

$$\begin{cases} z_1 = \overline{x_2} \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \\ z_0 = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \end{cases}$$

$\overline{x_2} \cdot x_1 \cdot (\cancel{\overline{x_0} + x_0})$ $\qquad$ $x_2 \cdot \overline{x_1}$

$$z_1 = \overline{x_2} \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0$$

$$= \overline{x_2} \cdot x_1 \cdot (\cancel{\overline{x_0} + x_0}) + x_2 \cdot \overline{x_1} \cdot (\cancel{\overline{x_0} + x_0})$$

$$\boxed{z_1 = \overline{x_2} \cdot x_1 + x_2 \cdot \overline{x_1}}$$

Ridurre il costo della sintesi (cont.)

- Cerchiamo aiuto nelle proprieta' dell'Algebra di Boole

$$\begin{cases} z_1 = \overline{x_2} \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \\ z_0 = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \end{cases}$$

$\overline{x_1} \cdot x_0$ $\quad$ $x_2 \cdot \overline{x_1}$

$$z_0 = \underline{\overline{x_2} \cdot \overline{x_1} \cdot x_0} + \underline{x_2 \cdot \overline{x_1} \cdot \overline{x_0}} + \underline{x_2 \cdot \overline{x_1} \cdot x_0} + \underline{x_2 \cdot \overline{x_1} \cdot x_0}$$

$$= \overline{x_1} \cdot x_0 \cdot (\overline{x_2} + x_2) + x_2 \cdot \overline{x_1} \cdot (\overline{x_0} + x_0)$$

$$\boxed{z_0 = \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1}}$$

Ridurre il costo della sintesi (cont.)

- Esistono **diverse sintesi** per una stessa legge combinatoria
- Hanno **costo diverso** in termini di numero di porte
- Riduzione del costo: basata su proprieta' **algebriche**

- Abbiamo bisogno di un metodo **formale** (algoritmico) per minimizzare il costo di una sintesi

Esercizio

- La rete ha 2 ingressi (x, y), un'uscita (z), e 4 *variabili di comando* a, b, c, d.
- implementa una legge $f(x, y)$ diversa a seconda del valore delle variabili di comando.

a = ...
b = ...
c = ...

x	y	z
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

a) Scrivere l'espressione algebrica che lega z agli ingressi e alle variabili di comando

b) Calcolare a, b, c, d in modo da implementare una generica funzione $f(x, y)$ nota (assumendo, cioè, di conoscere $f(0,0), f(1,0), f(0,1), f(1,1)$).

c) calcolare a, b, c, d per i *casi particolari* a) $f(x, y) = \overline{xy}$, b) $f(x, y) = x\overline{y}$.

Esercizio (cont.)

x	y	z
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

Sintesi di reti in forma SP a costo minimo

- Esistono **due criteri di costo**
  - **a porte:** ogni porta conta per un'unità di costo.
  - **a diodi**: ogni **ingresso** conta per un'unità di costo.

- Costo(AND a 10 ingressi)=Costo(AND a 2 ingressi) con il primo criterio
- Costo(AND a 10 ingressi)=**5** Costo(AND a 2 ingressi) con il secondo

- Metodo che si applica a **reti con un'uscita**
- Produce **reti in forma SP a 2 livelli di logica**
  - Tra tutte queste, trova quella di costo minimo

Espansione di Shannon

- è sempre possibile scrivere **qualunque** legge combinatoria F come **somma di prodotti degli ingressi (diretti o negati)**
- $z = f(x_{N-1}, \ldots, x_0)$

$$
\begin{aligned}
z = {} & f(0,\ldots,0,0) \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot \overline{x_0} \\
& + f(0,\ldots,0,1) \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \ldots \cdot \overline{x_1} \cdot x_0 \\
& \ldots \\
& + f(1,\ldots,1,0) \cdot x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot \overline{x_0} \\
& + f(1,\ldots,1,1) \cdot x_{N-1} \cdot x_{N-2} \cdot \ldots \cdot x_1 \cdot x_0
\end{aligned}
$$

- $f(x_{N-1}, \ldots, x_0) = 0$,
  - $0 \cdot \alpha = 0$, la riga vale 0.
  - $0 + \alpha = \alpha$, posso togliere la riga
- $f(x_{N-1}, \ldots, x_0) = 1$,
  - $1 \cdot \alpha = \alpha$, posso togliere $f(\ldots)$

Forma canonica SP

- **SP:** *z* è somma di prodotti
- **Canonica:** ogni prodotto ha come fattori **tutti gli ingressi** diretti o negati
- ciascuno dei termini della somma si chiama **mintermine**,
- corrisponde ad uno stato di ingresso riconosciuto dalla rete.

x_3	x_2	x_1	x_0	z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
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

$$1 \cdot \alpha = \alpha \qquad 0 + \beta = \beta$$
$$0 \cdot \alpha = 0$$

Espansione di Shannon

$$z = \cancel{0 \cdot \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}}$$
$$\cancel{+1} \cdot \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$$
$$\ldots$$
$$\cancel{+0 \cdot x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0}}$$
$$\cancel{+0 \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0}$$

Forma canonica SP

(lista dei mintermini)

$$z = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}$$
$$+\overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0}$$
$$+\overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0$$
$$+\overline{x_3} \cdot x_2 \cdot x_1 \cdot \overline{x_0}$$
$$+x_3 \cdot x_2 \cdot x_1 \cdot x_0$$
$$+x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}$$
$$+x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0$$
$$+x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}$$

Metodo di Quine-McCluskey

- Modo piu' efficiente di cercare fusioni tra mintermini/implicanti
- Basta ordinarli per **numero crescente di 1** (var. dirette)
  - Fusione possibile solo tra partizioni adiacenti
  - Numero di confronti ridotto

- Efficienza dell'algoritmo dipende dalla facilita' di trovare **stati di ingresso adiacenti** riconosciuti dalla legge

x_j $\overline{x_j}$ grafico Mappe di Karnaugh

Semplificare la forma canonica SP (cont.)

$\overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$

$\overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0}$

$\overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot x_0$

$\overline{x_3} \cdot x_2 \cdot x_1 \cdot \overline{x_0}$

$\overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0$

$x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}$

$x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$

$x_3 \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0}$

$\overline{x_3} \cdot \overline{x_2} \cdot x_0$

$\overline{x_2} \cdot \overline{x_1} \cdot x_0$

$\overline{x_3} \cdot \overline{x_2} \cdot x_1$

$\overline{x_3} \cdot x_1 \cdot \overline{x_0}$

$\overline{x_2} \cdot x_1 \cdot \overline{x_0}$

$\overline{x_3} \cdot x_1 \cdot x_0$

$\overline{x_3} \cdot x_2 \cdot x_1$

$x_3 \cdot \overline{x_2} \cdot \overline{x_1}$

$x_3 \cdot \overline{x_2} \cdot \overline{x_0}$

$\overline{x_3} \cdot x_1$

$\cancel{\overline{x_3} \cdot x_1}$

Semplificare la forma canonica SP (cont.)

- Applicare esaustivamente la legge

$\alpha(\cancel{x+1})$

$$\boldsymbol{\alpha x + \alpha = \alpha}$$

- **Togliere <u>tutti</u> gli implicanti che hanno prodotto qualcosa per fusione.**
- Si ottiene la **lista degli implicanti principali**

$$z = \overline{x_3} \cdot x_1 + \overline{x_3} \cdot \overline{x_2} \cdot x_0 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_3 \cdot \overline{x_2} \cdot \overline{x_1} + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

4 2

Riepilogo

Tabella di verita'

Espansione di Shannon

$$\begin{cases} 1 \cdot \alpha = \alpha \\ 0 \cdot \alpha = 0 \end{cases} \qquad 0 + \beta = \beta$$

Forma canonica SP

fusione

$$\begin{cases} \alpha x + \alpha \overline{x} = \alpha x + \alpha \overline{x} + \alpha \\ \alpha + \alpha = \alpha \end{cases}$$

semplificazione

$$\alpha x + \alpha = \alpha$$

Lista degli implicanti principali

- La lista degli implicanti principali **costa meno** della forma canonica SP.
  - Contiene meno termini
  - Ciascuno con meno ingressi.

- E' sintesi di costo minimo?
  - **Non è detto**. Potrebbe ancora essere **ridondante**.

Liste di copertura (non) ridondanti

- **Lista di copertura**: lista di implicanti, la cui somma è una forma SP per la funzione *f*.
  - lista dei mintermini
  - lista degli implicanti (cioè quella ottenuta prima della semplificazione)
  - lista degli implicanti principali
- Lista di copertura **non ridondante**: se tolgo un elemento smette di essere una lista di copertura.
- La lista dei **mintermini** è una lista **non ridondante**.
  - Infatti, se ne tolgo uno, uno stato di ingresso riconosciuto non sarà coperto (quindi non avrò più una lista di copertura)
- La lista degli **implicanti principali** può essere **ridondante.** Questa lo è.

$$z = \overline{x_3} \cdot x_1 + \overline{x_3} \cdot \overline{x_2} \cdot x_0 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_3 \cdot \overline{x_2} \cdot \overline{x_1} + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

Metodo di Quine-McCluskey

- Modo piu' efficiente di cercare fusioni tra mintermini/implicanti
- Basta ordinarli per **numero crescente di 1** (var. dirette)
  - Fusione possibile solo tra partizioni adiacenti
  - Numero di confronti ridotto

- Efficienza dell'algoritmo dipende dalla facilita' di trovare **stati di ingresso adiacenti** riconosciuti dalla legge

Mappe di Karnaugh

- Per una rete ad N ingressi è una matrice di 2^N celle

coordinate

x_0	0	1
	1	0

Z

$N = 1$

x_0 \ x_1	0	1
0	0	0
1	0	1

AND

$N = 2$

Mappe di Karnaugh (cont.)

- Celle **contigue** sulla mappa hanno coordinate **adiacenti** (e viceversa)

$x_2x_1x_0$

010 110

x_0 \ x_2x_1	00	01	11	10
0	0	1	1	0
1	1	1	1	1

001 101

3

$N = 3$

x_2x_3

x_1x_0 \ x_3x_2	00	01	11	10
00	X 0000			• 1000
01				
11				
10	* 0010			

$N = 4$

Mappe di Karnaugh (cont.)

- Oltre le 4 coordinate serve la 3a dimensione

$N = 5$

Sottocubi di ordine 1

$x_2x_1x_0$

A 011

B 111

- **Sottocubo di ordine 1**
  - casella che contiene un 1, corrispondente quindi ad uno stato di ingresso riconosciuto dalla rete
- **Coordinate** di un sottocubo di ordine 1
  - stato di ingresso corrispondente al sottocubo
- **Adiacenza tra sottocubi di ordine 1**
  - due SO1 sono adiacenti se differiscono tra loro per **una sola** coordinata

x_0 \ x_2x_1	00	01	11	10
0	0	1	1	0
1	1	1	1	1
		A	B	

Esempio

$P = \cancel{4}\ 2$

x_1x_0 \ x_3x_2	00	01	11	10
00	1	1		1
01			1	1
11			1	1
10	1			1

A B C D

Sottocubi principali

$Z = x_3 \cdot x_0 + x_3 \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_0} + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}$

	x_3	x_2	x_1	x_0
A	1	–	–	1
B	1	0	–	–
C	–	0	–	0
D	0	–	0	0

$x_3 \cdot x_0$

Sottocubi di ordine 2

$\overline{x_2} \cdot x_1$

- **Sottocubo di ordine 2**
  - costituito da **sottocubi adiacenti di ordine 1**.
  - C è un SO2.
  - Le sue coordinate sono: $x_2 = - \; x_1 = 1 \; x_0 = 1$
- Il SO2 C **copre** i SO1 A e B
- **Adiacenza tra sottocubi di ordine 2**
  - due SO2 sono adiacenti se differiscono tra loro per una sola coordinata

x_0 \ x_2x_1	00	01	11	10
0	0	E 1	1	0
1	F 1	1	1	1

	x_2	x_1	x_0
E	0	1	-
F	-	0	1

Sottocubi di ordine *n* e liste di copertura

- Sottocubi di ordine 4, 8, …

	x_2	x_1	x_0
C	-	1	1
D	-	1	0
F	-	0	1

SP

{G, H} {C, D, F}

$z = x_1 + x_0$

	x_2	x_1	x_0
G	-	1	-
H	-	-	1

Sottocubi e implicanti

- Sono piu' o meno la stessa cosa

$\bar{x}_2 \cdot x_1 \cdot x_6$

sottocubo (principale) **di ordine p**	$\Leftrightarrow$	**implicante (principale)** **di $N - (\log_2 p)$ variabili**

cosette 4

N-2

- Cercare i sottocubi principali è facile
  - si fa ad occhio

Algoritmo per la ricerca dei S. principali

- Parto dai sottocubi **di ordine più grande** che trovo sulla mappa

1. Considero i sottocubi di ordine p *non interamente contenuti in sottocubi di ordine piu' grande*
   - Li segno **tutti** (attenzione!)
   - Non essendo contenuti in sottocubi di ordine maggiore sono senz'altro principali

2. **Domanda**: l'insieme dei sottocubi che ho trovato finora basta a coprire tutta la mappa?
   - Se sì, ho finito.
   - Se no, $p \leftarrow p/2$ e ripeto il ciclo

- L'algoritmo termina di sicuro (al più quando $p = 1$)

Esempio

$P = \cancel{4}\ 2$

x_1x_0 \ x_3x_2	00	01	11	10
00	1	1		1
01			1	1
11			1	1
10	1			1

Sottocubi principali

$Z = x_3 \cdot x_0 + x_3 \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_0} + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}$

	x_3	x_2	x_1	x_0
A	1	-	-	1
B	1	0	-	-
C	-	0	-	0
D	0	-	0	0

$x_3 \cdot x_0$

Esercizio

$P = \cancel{4}\ 2$

$$Z = \overline{x_3} \cdot x_1 + x_3 \cdot \overline{x_2} \cdot \overline{x_1} + \overline{x_3} \cdot \overline{x_2} \cdot x_0 + x_3 \cdot \overline{x_2} \cdot \overline{x_0} + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0}$$

x_3	x_2	x_1	x_0	z_0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
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

x_1x_0 \ x_3x_2	00	01	11	10
00	0	0	0	1
01	1	0	0	1
11	1	1	0	0
10	1	1	0	1

A B C D E F

Sottocubi principali

	x_3	x_2	x_1	x_0
A	0	–	1	–
B	1	0	0	–
C	0	0	–	1
D	1	0	–	0
E	–	0	0	1
F	–	0	1	0

Ricerca delle liste di copertura non ridondanti

- I sottocubi principali vanno **classificati**

B?

- Alcuni sono **gli unici a coprire un dato sottocubo di ordine 1**.
  - non possono essere tolti, se vogliamo coprire tutti gli stati riconosciuti dalla rete.
  - Sono detti **essenziali,** e costituiscono il **cuore della mappa**.
- Come trovarli?
  - Basta guardare se ci sono degli 1 cerchiati una volta sola.

$z = x_3 \cdot x_0 + \overline{x_2} \cdot \overline{x_0} + \overline{x_3} \cdot x_1 \cdot x_0$

Sottocubi essenziali e assolutamente eliminabili

$\{A, B, C, D\}$ A, C, D ess. B ass. elim.

- Si spuntano tutti i sottocubi di ordine 1 contenuti in sottocubi principali **essenziali**
- Se si copre completamente **anche** qualche altro sottocubo
  - quel sottocubo è **ridondante**.
  - **assolutamente eliminabile**, **non deve comparire** in una lista di copertura irridondante.

- Un sottocubo principale è **assolutamente eliminabile** se riconosce **soltanto** stati di ingresso già riconosciuti da sottocubi principali **essenziali.**

$\{A, C, D\}$ l.c. costo minimo

P=4 2

Classificazione dei sottocubi

A, D • ipotesi alternative

- Ricerca dei **sottocubi principali:** ce n'è uno di ordine 4 (A), e poi devo passare a quelli di ordine 2 (B,C,D)

	x_3	x_2	x_1	x_0	
A	0	1	-	-	ess
B	-	1	0	1	
C	1	-	0	1	
D	1	0	-	1	ess

Semplicemente eliminabili

Lista di copertura di costo minimo

- Generare **tutte le possibili liste di copertura non ridondanti** che
  - includono **tutti** i sottocubi principali essenziali
  - non includono **nessun** sottocubo principale assolutamente eliminabile
  - includono **un sottoinsieme** di sottocubi principali semplicemente eliminabili

- Tra tutte queste, **valuto quella di costo minimo** applicando il criterio di costo richiesto.

Lista di copertura di costo minimo (cont.)

- Considero un **qualunque** sottocubo S.E.
  - formulo **due ipotesi alternative**

a) "Come se fosse" essenziale
  - lo aggiungo alla lista di copertura
  - qualche **altro sottocubo** diventa assolutamente eliminabile

b) " Come se fosse" assolutamente eliminabile
  - lo tolgo da qualunque lista di copertura
  - qualche **altro sottocubo** diventa essenziale

$\{A,D\}$ — B → $\{A,D,B\}$; $\cancel{B}$ → $\{A,D,C\}$

2 + 3 + 3 + 3 = 11

criterio

- **Albero binario di decisioni**
  - Foglie sono L.C. non ridondanti

$\{A,D\} \rightarrow \{A,D,B\}$, $\{A,D,C\}$ $\Leftrightarrow$ $z = \overline{x_3} \cdot x_2 + x_3 \cdot \overline{x_2} \cdot x_0$ $\rightarrow$ $+x_2 \cdot \overline{x_1} \cdot x_0$, $+x_3 \cdot \overline{x_1} \cdot x_0$

Esempio

- Classificare gli implicanti e trovare tutte le LC non ridondanti

- S. essenziali: **A**

Esempio (cont.)

- S. essenziali: **A**
- S. assolutamente eliminabili: **nessuno**
- S. semplicemente eliminabili: **B,C,D,E,F**

- Decisione su **D**
  - Come se ESS -> B,F diventano AE
  - Come se AE -> B,F diventano ESS

Esempio (cont.)

$$Z = \underbrace{\overline{x_3} \cdot x_1}_{A} + \underbrace{\overline{x_2} \cdot \overline{x_1} \cdot x_0}_{E} + \underbrace{x_3 \cdot \overline{x_2} \cdot \overline{x_0}}_{D}$$

- Ramo inferiore {A,C,B}

- Decisione su **D**
  - Come se ESS -> F diventa AE
  - Come se AE -> F diventa ESS

Esempio (fine)

- ho **tutte le liste di copertura non ridondanti**
  - Uso il criterio di costo
  - Trovo la LC di costo minimo

$$z = \overline{x_3} \cdot x_1 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

Esercizio - sintesi di leggi non completamente specificate

Binary - Coded Decimal

- **Decodificatore BCD a 7 segmenti**
  - 4 variabili di ingresso,
  - interpretate come la **codifica in base 2** di una cifra decimale,
  - **7 uscite**, che vanno ad accendere i segmenti di un display LCD

0 - - 9

Esercizio (cont.)

- Sintesi a costo minimo dell'uscita z_E

J	x_3	x_2	x_1	x_0	z_E
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
?	1	0	1	0	-
?	1	0	1	1	-
?	1	1	0	0	-
?	1	1	0	1	-
?	1	1	1	0	-
?	1	1	1	1	-

x_1x_0 \ x_3x_2	00	01	11	10
00	1	0	-	1
01	0	0	-	0
11	0	0	-	-
10	1	1	-	-

1. **Ricerca** degli IP
   - Considero "-" ⇔ 1
   - Trovo IP piu' grandi
2. **Classificazione** degli IP
   - Considero "-" ⇔ 0
   - Non devo coprire "-"

	x_3	x_2	x_1	x_0

Sintesi in forma PS

- Esiste anche – **sempre –** la possibilità di sintetizzare una rete in formato **PS,** cioè prodotto di somme.
- Teoria **duale**: espansione di Shannon, forma canonica PS, maxtermini, implicati, etc.
  - Che non faremo
  - Esiste un metodo piu' veloce
  - Che sfrutta conoscenze gia' acquisite

$$z = S_1 \cdot S_2 \cdot \cdots S_k$$

$S_i = \sum x_j$

Sintesi in forma PS (cont.)

F

	z	$\overline{z}$
	0	1
	⋮	
	1	0
		$\overline{F}$

- **<u>Punto 1</u>**: data F, ricavo $\overline{F}$
  - In pratica, riscrivo la tabella di verità con 1 al posto dello 0 e viceversa.
- **<u>Punto 2</u>**: realizzo una **sintesi SP** della legge $\overline{F}$
- **<u>Punto 3</u>**: ottengo una sintesi di F inserendo un **invertitore** in uscita alla rete SP che sintetizza $\overline{F}$.
- **<u>Punto 4</u>**: applico i **Teoremi di De Morgan**
  - da destra verso sinistra

Sintesi in forma PS (cont.)

$\overline{y_1 + \dots + y_k} = \overline{y_1} \cdot \dots \cdot \overline{y_k}$

- **Punto 1,2**
  - Sintesi di $\overline{F}$ in forma SP
- **Punto 3**
  - Applico un invertitore
  - Ottengo z
- **Punto 4**
  - Applico DM: $\overline{a + b} = \overline{a} \cdot \overline{b}$
  - Applico DM: $\overline{a \cdot b} = \overline{a} + \overline{b}$

$\overline{\alpha_1 \cdot \alpha_2 \cdot \dots \cdot \alpha_e} = \overline{\alpha_1} + \overline{\alpha_2} \dots + \overline{\alpha_e}$

Sintesi in forma PS (cont.)

- Dal punto di vista algebrico:
  - scrivo $\overline{F}$ in forma SP: $\overline{z} = P_1 + P_2 + \ldots + P_k$
  - P_i sono prodotti di variabili di ingresso

- Applico De Morgan, ottenendo:

$$z = \overline{\overline{z}} = \overline{P_1 + P_2 + \ldots + P_k} = \overline{P_1} \cdot \overline{P_2} \cdot \ldots \cdot \overline{P_k}$$

- Applico ancora De Morgan per ottenere

$$\overline{P_i} = \overline{\prod x_j} = \sum \overline{x_j}$$

Sintesi in forma PS (cont.)

$z = ($ tutti $) \cdot ($ $)$

- Con questo procedimento
- Se $\overline{F}$ è in forma **canonica SP**, allora F è in **forma canonica PS**
- Se la sintesi SP di $\overline{F}$ **costa *C***, allora la sintesi PS di F **costa C**
- Se la sintesi SP di $\overline{F}$ è di costo minimo (tra le possibili sintesi SP), lo è anche la sintesi PS di F (tra le possibili sintesi PS)
  - Se esistesse una sintesi PS di F a costo minore, applicando DM[2] troverei una sintesi SP di $\overline{F}$ a costo **minore del minimo**, contro l'ipotesi

$SP(\overline{F}) \equiv PS(\overline{\overline{F}})$ Do Dm2

$SP(\overline{F}) \equiv PS(F)$

Riepilogando

$$\boxed{SP\ F = PS\ \overline{F}}$$

SP F PS F

- Data una legge F, sappiamo effettuarne la sintesi a **costo minimo in forma SP**
- Sappiamo anche effettuarne la sintesi a **costo minimo in forma PS**
  - Sintetizziamo $\overline{F}$ a costo minimo in forma SP
  - Aggiungiamo un invertitore ed applichiamo DM due volte
- Quale delle due sintesi a costo minimo (di F) costa meno?
  - Non e' possibile dirlo
  - Dipende dalla tabella di verita'

SP
PS

Esercizio

$z = \overline{\overline{z}} = \overline{x_2 \cdot \overline{x_1} + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}}$ · SP $\overline{z}$

$= \overline{(x_2 \cdot \overline{x_1})} \cdot \overline{(x_3 \cdot x_2)} \cdot \overline{(x_3 \cdot x_1 \cdot x_0)} \cdot \overline{(\overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0})}$

- Sintesi PS a costo minimo di legge F gia' vista

· PS z

$= (\overline{x_2} + x_1) \cdot (\overline{x_3} + \overline{x_2}) \cdot (\overline{x_3} + \overline{x_1} + \overline{x_0}) \cdot (x_3 + x_1 + x_0)$

x_1x_0 \ x_3x_2	00	01	11	10
00				1
01	1			1
11	1	1		
10	1	1		1

SP 4

Mappa di Karnaugh per *F*

x_1x_0 \ x_3x_2	00	01	11	10
00	1	1	1	
01		1	1	
11			1	1
10			1	

D A B C

$\overline{z}$

Mappa di Karnaugh per $\overline{F}$

		x_3	x_2	x_1	x_0
ess	A	-	1	0	-
ess	B	1	1	-	-
ess	C	1	-	1	1
ess	D	0	-	0	0

5 costo porte

$$\overline{z} = \overline{x_1} \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}$$

Esercizio (cont.)

x_1x_0 \ x_3x_2	00	01	11	10
00	1 D	1 A	1	
01		1	1	
11			1	C 1
10			B 1	

$$\overline{z} = \overline{x_1} \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}$$

$$z = \overline{\overline{z}} = \overline{\overline{\overline{x_1} \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}}}$$

$$= \left(\overline{\overline{x_1} \cdot x_2}\right) \cdot \left(\overline{x_3 \cdot x_2}\right) \cdot \left(\overline{x_3 \cdot x_1 \cdot x_0}\right) \cdot \left(\overline{\overline{\overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}}}\right)$$

$$= \left(x_1 + \overline{x_2}\right) \cdot \left(\overline{x_3} + \overline{x_2}\right) \cdot \left(\overline{x_3} + \overline{x_1} + \overline{x_0}\right) \cdot \left(x_3 + x_1 + x_0\right)$$

Porte logiche universali

- NAND e NOR sono porte logiche **universali**
  - Posso realizzare AND, OR, NOT con **sole** porte NAND o **sole** porte NOR

	Rel. algebrica	Porta	Realizzazione a NAND	Realizzazione a NOR
NOT	$x = x \cdot x \Rightarrow \overline{x} = \overline{x \cdot x}$ $x = x + x \Rightarrow \overline{x} = \overline{x + x}$			
AND	$x \cdot y = \overline{\overline{(x \cdot y)}}$ $x \cdot y = \overline{\overline{x} + \overline{y}}$			
OR	$x + y = \overline{\overline{x} \cdot \overline{y}}$ $x + y = \overline{\overline{(x + y)}}$			

Porte logiche universali

- Visto che **qualunque legge combinatoria** puo' essere sintetizzata con porte AND, OR, NOT
- Qualunque legge combinatoria puo' essere sintetizzata
  - **Soltanto** con porte NAND
  - **Soltanto** con porte NOR

Sintesi a porte NAND

F Z

- Si parte da un circuito **in forma SP**
- Sostituisco la porta OR con il suo equivalente a NAND
- Sostituisco ciascuna AND con il suo equivalente a NAND
- I NOT sugli ingressi li posso lasciar stare, tanto **non fanno parte della sintesi** (sono gratis)

Sintesi a porte NAND (cont.)

- Elimino le coppie di NOT in cascata
- Ottengo una rete a sole porte NAND
- A due livelli di logica

2LL

SP

Sintesi a porte NAND (cont.)

K SP ⟺ NAND K ?

- Dal punto di vista algebrico

SP

$$z = P_1 + P_2 + \cdots + P_K$$

2c

$$= \overline{\overline{P_1 + P_2 + \cdots + P_K}}$$

DM

$$= \overline{\overline{P_1} \cdot \overline{P_2} \cdot \ldots \cdot \overline{P_K}}$$

6

Sintesi a porte NOR

- Si parte da un circuito **in forma PS**
- Sostituisco la porta AND con il suo equivalente a NOR
- Sostituisco la porta OR con il suo equivalente a NOR
- Non sto a toccare i NOT sugli ingressi

Sintesi a porte NOR (cont.)

PS NOR

- Elimino le coppie di NOT in cascata
- Ottengo una rete a sole porte NOR
- A due livelli di logica

SP

Sintesi a porte NOR (cont.)

- Dal punto di vista algebrico

PS

$$z = S_1 \cdot S_2 \cdot \ldots \cdot S_K$$

$$= \overline{\overline{S_1 \cdot S_2 \cdot \ldots \cdot S_K}}$$

DM

$$= \overline{\overline{S_1} + \overline{S_2} + \ldots + \overline{S_K}}$$

PS

NOR

Esercizio

→ Gate minime

- Sintetizzare la rete a porte NAND ed a porte NOR

① SP

PS ②

x_1x_0 \ x_3x_2	00	01	11	10
00	1	0	1	-
01	0	0	1	1
11	1	-	0	-
10	-	0	0	1

SP CM

Sintesi a porte NAND

AC — D: ACD — D: ACE

4

$$Z = \overline{x_3 \cdot \overline{x_1} + \overline{x_2} \cdot \overline{x_0} + \overline{x_2} \cdot x_1}$$

$$= \overline{\left(\overline{x_3 \cdot \overline{x_1}}\right) \cdot \left(\overline{\overline{x_2} \cdot \overline{x_0}}\right) \cdot \left(\overline{\overline{x_2} \cdot x_1}\right)}$$

x_1x_0 \ x_3x_2	00	01	11	10
00	1	0	1	-
01	0	0	1	1
11	1	-	0	-
10	-	0	0	1

	x_3	x_2	x_1	x_0	
A	1	-	0	-	CSS
B	1	0	-	-	ass. elim.
C	-	0	-	0	ESS
D	-	0	1	-	SG
E	0	-	1	1	SE

Sintesi a porte NOR

x_1x_0 \ x_3x_2	00	01	11	10
00	1	0	1	-
01	0	0	1	1
11	1	-	0	-
10	-	0	0	1

x_1x_0 \ x_3x_2	00	01	11	10
00				
01				
11				
10				

	x_3	x_2	x_1	x_0

Porte tri-state

- fa comodo poter connettere insieme le **uscite** delle reti
  - Bus condivisi
  - Linee di ingresso-uscita

Modello elettrico dell'uscita di una rete

Problemi elettrici e logici

Problemi elettrici e logici

- Scorre corrente
  - Anche molta
  - Circuiti si bruciano
- Quale e' il valore logico sulla linea?

Porte tri-state

- servono porte **capaci di disconnettere fisicamente** un'uscita da una linea condivisa.
- Tali porte prendono il nome di **porte tri-state**

b	x	z
1	0	0
1	1	1
0	-	HiZ

N-1 HiZ

- **b=1**: la porta si comporta da elemento neutro
- **b=0**: la porta e' in **alta impedenza**
  - **L'uscita e' disconnessa**

Esempio: multiplexer decodificato

- Se tutte le variabili di comando valgono 0, l'uscita è in **alta impedenza**
- Altrimenti, **una** sola variabile di comando deve essere ad 1
  - l'uscita insegue l'ingresso corrispondente

00 - - 00 → z Hi-Z

01 0 - 00 → z

Esempio: linea di ingresso/uscita

- **Forchetta**
  - b=0: linee di ingresso
  - b=1: line di uscita

L’alta impedenza non e’ un valore logico

- Quanto vale z nelle due reti qui sotto?

- Vale 0 o vale 1
  - Dipende dalla tecnologia sottostante
  - Le uscite degli stadi finali sono connesse alla tensione Vcc o a massa.

Circuiti di ritardo e formatori di impulsi

20 5

- I circuiti di ritardo si fanno con **gli elementi neutri**
  - O, piu' spesso, con un numero pari di porte NOT

- Hanno ritardo **simmetrico**
  - Identico sulle transizioni 0-1 e 1-0

- Vorrei poter realizzare circuiti con ritardo **asimmetrico**
  - **Grande** sulle transizioni 0-1, **piccolo** sulle transizioni 1-0
  - O viceversa

Circuito di ritardo sul fronte di salita

- Si indica con Δ^+
  - Si realizza con una AND
  - Transizione 1-0, il **primo** ingresso che va a 0 (*x*) porta a zero l'uscita. Ritardo **piccolo** della porta AND
  - Transizione 0-1, il **secondo** ingresso che va a 1 (*y*) porta a 1 l'uscita. Ritardo (**grande**) del buffer, più quello (piccolo) della porta AND

Circuito di ritardo sul fronte di discesa

- Si indica con Δ^{-}
  - Si realizza con una OR
  - Transizione 0-1, il **primo** ingresso che va a 1 (*y*) porta a 1 l'uscita. Ritardo (**piccolo**) della porta OR
  - Transizione 1-0, il **secondo** ingresso che va a 0 (*x*) porta a zero l'uscita. Ritardo **grande** del buffer, piu' quello (piccolo) della porta AND

Formatore di impulsi sul fronte di salita

- Rete combinatoria che genera in uscita **un impulso di durata nota** quando l'ingresso ha un fronte di salita
- Si indica con P^+

Formatore di impulsi sul fronte di discesa

- Rete combinatoria che genera in uscita **un impulso di durata nota** quando l'ingresso ha un fronte di discesa
- Si indica con P^-