Appunti per *un* corso di Teoria dei Sistemi Digitali

Davide Tambuchi

Como, 9 marzo 2004

Avvertenze

Informazioni Legali e Copyright

Marchi Registrati

Altre Avvertenze

Introduzione

Queste dispense vogliono illustrare i principi fondamentali della Teoria degli Automi a Stati Finiti, utilizzando le tecniche note nella Teoria della Commutazione.

Questo volume possiede, per deliberata scelta dell'Autore, un forte carattere *anarchico* rispetto ai testi scolastici tradizionali. Si é scelto di privilegiare fortemente i *contenuti* rispetto agli *schemi*, e pertanto il presente testo non possiede alcuna strutturazione modulare.

Le presenti dispense sono state scritte con LaTeX 2_ε su una macchina `LINUX`; il presente testo é stato compilato nei formati

- *Device Independent* – `DVI`,
- *Postscript* – `PS`,
- *Portable Document Format* – `PDF`.

Per quanto riguarda la distribuzione, si rimanda alle Informazioni Legali.

Chiunque voglia scrivermi, puó farlo all'indirizzo di posta elettronica `davide.tambuchi@tin.it`.

Como, 9 marzo 2004

Davide Tambuchi

Indice

1 Introduzione alla Teoria degli Automi **1**

1.1 Macchine combinatorie e sequenziali 1
1.2 Circuiti sequenziali sincroni ed asincroni 4
1.3 Progettazione dei circuiti sequenziali sincroni 6
1.3.1 Riconoscitore di sequenza 6
1.3.2 Progetto di un automa con flip-flop di tipo T 18
1.3.3 Progetto di un automa con flip-flop di tipo SR 21
1.3.4 Progetto di un automa con flip-flop di tipo JK 24
1.3.5 Full adder seriale 26
1.3.6 Le macchine iterative 28
1.3.7 Equivalenza tra automi sincroni e macchine iterative . 29
1.4 Progetto di contatori sincroni 31
1.4.1 Progetto di un contatore modulo 4 con flip-flop di tipo D 31
1.4.2 Contatore modulo 4 con flip-flop di tipo T 33
1.4.3 Progetto con flip-flop di tipo SR 36
1.4.4 Progetto con flip-flop di tipo JK 37

2 Riduzione del numero degli stati **41**

2.1 Introduzione 41
2.2 Un algoritmo di minimizzazione 43
2.2.1 Rappresentazione canonica di un automa 45
2.3 Macchine incompletamente specificate 46
2.3.1 Completamento delle transizioni 46
2.3.2 La non unicitá della riduzione degli stati 47
2.3.3 Osservazione importante 49
2.3.4 L'introduzione provvisoria di nuovi stati come metodo di riduzione 50
2.4 Un metodo generale 52
2.4.1 Il grafo di fusione 53
2.4.2 L'insieme degli stati compatibili 54
2.4.3 L'insieme chiuso di compatibilitá 56

2.4.4 Il grafo di compatibilitá 57
2.4.5 La tabella di fusione 60

3 Progettazione di circuiti sequenziali asincroni 64

4 Le macchine lineari 65

5 Circuiti sequenziali a relé 66

6 Le reti di Petri 67

A Rappresentazione dei circuiti digitali 68

B Dispositivi di memoria elementari 70
B.1 Il latch SR 70
B.2 Il flip-flop SR 71
B.3 Il flip-flop JK 72
B.3.1 Il JK master-slave 74
B.3.2 Il JK edge-triggered 75
B.3.3 Ingressi di preset e clear 76
B.4 Il flip-flop di tipo D 77
B.5 Il flip-flop di tipo T 78

C Full adder combinatorio 80

Elenco delle figure

1.1 Circuito combinatorio sommatore binario 1
1.2 Circuito riconoscitore di coppie di 1 3
1.3 Andamento ingresso-stato-uscita del riconoscitore di coppie di 1 4
1.4 Esempio di circuito sequenziale asincrono 5
1.5 Relazione ingresso-uscita per il riconoscitore di sequenza . . . 6
1.6 Diagramma degli stati (ancora incompleto) 7
1.7 Diagramma degli stati del riconoscitore di sequenza 9
1.8 Relazione tra ingresso, stato e uscita del riconoscitore di sequenza 9
1.9 Mappa di Karnaugh per Y_1 12
1.10 Mappa di Karnaugh per Y_2 12
1.11 Mappa di Karnaugh per z 12
1.12 Rete combinatoria "bozza" del circuito riconoscitore di sequenza 13
1.13 Riconoscitore della sequenza 0110 14
1.14 Mappe di Karnaugh per il nuovo assegnamento 15
1.15 Circuito del riconoscitore di sequenza con il nuovo assegnamento 16
1.16 Diagramma degli stati per il riconoscimento di coppie di 1 non annidate 17
1.17 mappe di Karnaugh per t_1 e t_2 21
1.18 Circuito del riconoscitore con flip-flop di tipo T 22
1.19 Mappe per il riconoscitore realizzato con flip-flop SR 23
1.20 Riconoscitore realizzato con flip-flop SR 24
1.21 Mappe per il riconoscitore realizzato con flip-flop JK 25
1.22 Riconoscitore realizzato con flip-flop jk 26
1.23 Diagramma degli stati di un full adder seriale 27
1.24 Mappe per il full adder seriale 28
1.25 Realizzazione di un full adder seriale con flip-flop di tipo D . . 29
1.26 Confronto tra full adder sequenziale e combinatorio nella somma di 3 bit 30
1.27 Macchina iterativa riconoscitrice di sequenza 30
1.28 Diagramma degli stati di un contatore modulo 4 32

1.29 Mappe per il contatore modulo 4 con flip-flop di tipo D 33
1.30 Contatore modulo 4 con flip-flop di tipo D 34
1.31 Mappe per il contatore modulo 4 con flip-flop di tipo T 34
1.32 Contatore modulo 4 con flip-flop di tipo T 34
1.33 Contatore modulo 4 con flip-flop di tipo T con assegnamento (1.10) 36
1.34 Mappe per il contatore con flip-flop SR 36
1.35 Contatore modulo 4 con flip-flop SR 37
1.36 Mappe per il contatore con flip-flop JK 38
1.37 Contatore modulo 4 con flip-flop JK 38
1.38 Diagramma degli stati per un contatore modulo 3 38
1.39 Mappe per il contatore modulo 3 39
1.40 Contatore modulo 3 40

2.1 Automa con stati ridondanti 41
2.2 Automa con stati non ridondanti 42
2.3 Grafo di fusione 54
2.4 Coppie di stati compatibili ottenuti dal grafo di fusione 55
2.5 Grafo di fusione 58
2.6 Grafo di fusione dopo la semplificazione 58
2.7 Grafo di compatibilitá 58
2.8 Grafo di compatibilitá e suo sottografo chiuso 59
2.9 Tabella di fusione 61
2.10 Tabella di fusione semplificata 62
2.11 Grafo di compatibilitá ricavato dalla tabella 2.10 63

A.1 Rappresentazione multifilare di un circuito digitale 68
A.2 Corrispondenza tra rappresentazione multifilare ed unifilare . . 69
A.3 Rappresentazione unifilare del circuito 69

B.1 Il latch SR 70
B.2 Il flip-flop SR 72
B.3 Il flip-flop JK 73
B.4 Il flip-flop JK master-slave 74
B.5 Flip-flop JK edge-triggered 76
B.6 JK con ingressi di preset e di clear 76
B.7 JK master-slave con ingressi di preset e di clear 77
B.8 Flip-flop di tipo D 78
B.9 Flip-flop di tipo T 79

Capitolo 1

Introduzione alla Teoria degli Automi

1.1 Differenza tra macchine combinatorie e sequenziali

Un circuito digitale *combinatorio* è un sistema le cui uscite dipendono, istante per istante, dai valori degli ingressi.

Ad esempio, se consideriamo il circuito di 1.1 che effettua la somma di due bit x_1, x_2, possiamo dire che le su uscite s_1 e s_2 dipendono dai *valori istantanei* degli ingressi, e che il circuito non tiene conto in alcun modo dei *valori passati* di x_1 ed x_2.

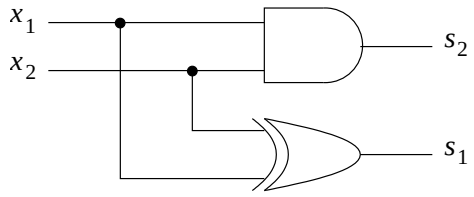

Figura 1.1: Circuito combinatorio sommatore binario

La tabella della verità del sommatore è la 1.1; in essa possiamo vedere come l'uscita sia composta dei due bit s_1, s_2 di cui s_2 è il più significativo.

Consideriamo ora il circuito di figura 1.2; la sua uscita si porta a livello ogniqualvolta applichiamo una stringa costituita da una coppia di bit alti in ingresso. Dato che l'uscita dipende sia dall'ingresso attuale che dalla storia degli ingressi passati, possiamo affermare che questo sistema possiede la capacità di memorizzazione (seppur parziale) della sua storia passata.

Tabella 1.1: Tabella della verità del sommatore di due bit

x_2	x_1	s_2	s_1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Infatti questo circuito deve essere in grado di memorizzare l'ultimo bit ricevuto, per poter decidere, in base al bit attualmente presente in ingresso, se l'uscita deve essere a livello alto oppure a livello basso. Un circuito, come quest'ultimo, in cui l'uscita dipende dalla storia passata degli ingressi ad esso applicati, è detto *circuito* (o *automa*) *sequenziale*.

Esaminiamo in dettaglio il funzionamento del circuito: innanzitutto notiamo che l'ingresso x viene "letto" dal circuito ad istanti di tempo discreti, e precisamente quando il segnale di clock c_k è a livello alto (cioè negli intervalli di tempo $t_1, t_2, \ldots, t_6$).

Supponiamo che all'istante iniziale l'uscita y del flip-flop D sia a livello basso (ciò puó essere ottenuto agendo sul segnale di *clear*, come illustrato nell'appendice nel paragrafo B.3.3). Supponiamo altresí che il flip-flop commuti sul *fronte di discesa* dell'impulso di clock.

Se applichiamo in ingresso il segnale $x = 0$, accade che, durante l'intervallo di tempo t_1, il flip-flop avrà in ingresso il segnale $Y = xc_k\overline{y} = 0$, e pertanto la sua uscita non commuterà sul fronte di discesa del clock, al termine dell'intervallo di tempo considerato. Si avrà ancora $y = 0$.

Nell'intervallo di tempo t_2, in cui $c_k = 1$, troviamo in ingresso un 1, e pertanto l'uscita Y della porta P_1 si troverá a livello alto. Di conseguenza, il flip-flop commuterà al termine di t_2, e la sua uscita y si porterá al valore $y = 1$.

Notiamo che possiamo pensare al flip-flop come ad un *elemento di memoria*; quando la sua uscita y é alta, vuol dire che in ingresso é stato applicato il primo uno della coppia di 1.

Come vedremo fra poco, l'applicazione in ingresso del circuito di uno 0 o di un altro 1 deve riportare l'uscita del flip-flop a 0; infatti se in ingresso abbiamo applicato uno 0 dopo aver applicato un 1, ci troviamo con la successione 10, e pertanto dobbiamo memorizzare il fatto che l'ultima cifra applicata all'ingresso è uno zero.

Invece, se in ingresso si ha il secondo 1, l'uscita si porterá a livello alto, mentre il flip-flop dovrá tener presente che ora tutto é pronto per "rico-

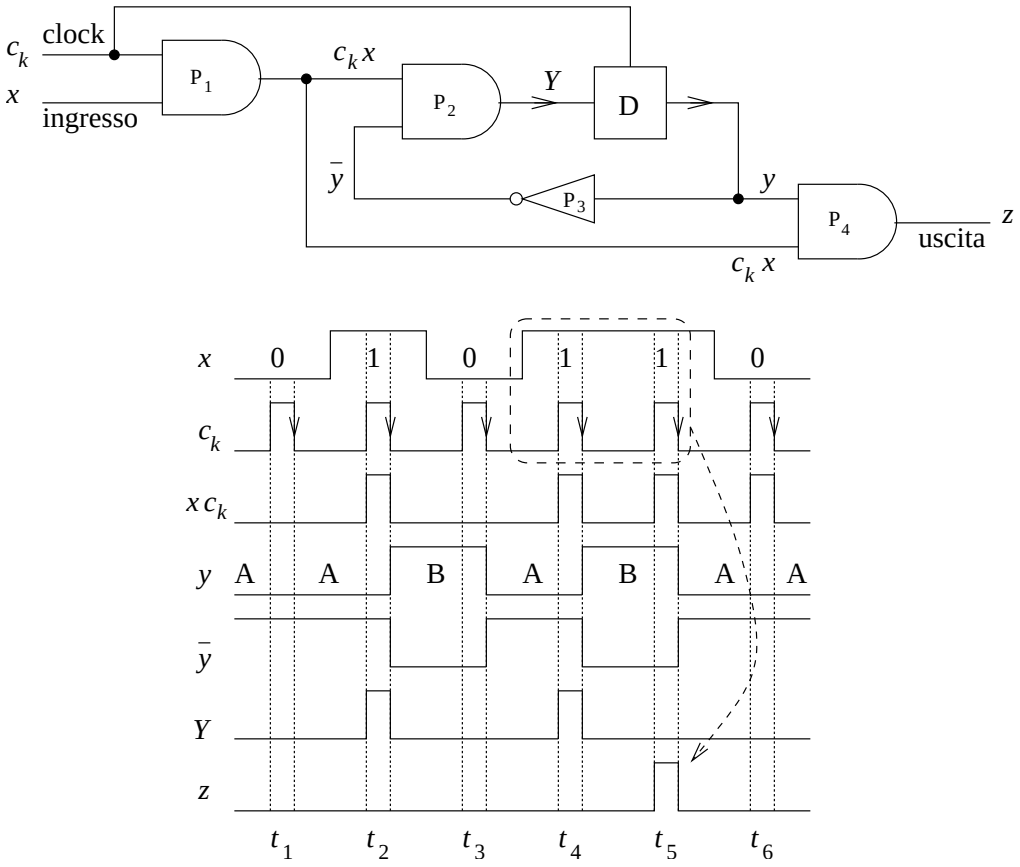

Figura 1.2: Circuito riconoscitore di coppie di 1

minciare daccapo", ovvero per attendere un ingresso la successiva coppia di uno.

In figura 1.2, sono indicati con le lettere A e B gli stati possibili in cui si puó trovare la memoria del flip-flop; la lettera A corrispondente al valore $y = 0$ della sua uscita; la B al valore $y = 1$. La variabile y che individua, istante per istante, lo stato di memorizzazione, é detta *stato attuale*; e dato che il valore che essa assumerá dopo il fronte di discesa del clock é dato dal valore di Y, chiameremo quest'ultima variabile *stato successivo*.

Consideriamo l'intervallo di tempo t_3; in esso si ha $Y = 0$ (avendosi $\overline{y} = 0$ all'ingresso della porta P_1), e pertanto l'uscita del flip-flop si riporta al valore basso.

Sino ad ora, abbiamo applicato in ingresso la successione di cifre 010; proviamo ora ad applicare in ingresso la coppia di 1.

Nell'intervallo di tempo t_4 il cui si ha $x = 1$, l'uscita della porta P_1 é alta,

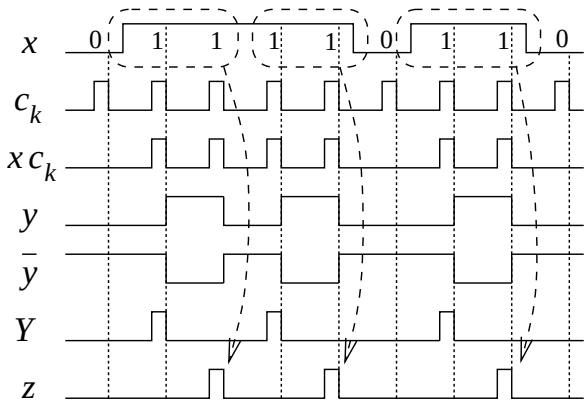

Figura 1.3: Andamento ingresso-stato-uscita del riconoscitore di coppie di 1

e pertanto l'uscita y di D si porterá di nuovo a livello alto.

Applichiamo il secondo 1 della coppia; in questo modo nell'intervallo t_5 l'uscita Y della porta P_1 é bassa, ed il flip-flop riporta la sua uscita a livello basso.

Notiamo ora che, se prendiamo come uscita z il prodotto logico dei segnali xc_k ed y, otteniamo che z si trova a livello alto se e solo se:

- L'uscita y del flip flop è alta (ció corrisponde ad aver giá ricevuto un 1 in ingresso)
- l'ingresso x é a livello alto (ció corrisponde ad applicare un altro 1 in ingresso, consecutivamente al precedente 1).

In questo modo, possiamo dire che il circuito è in grado di riconoscere coppie di 1 applicate all'ingresso. Ad esempio, se in ingresso si ha la successione 011110110 l'uscita sará 001010010, come illustrato in figura 1.3.

1.2 Circuiti sequenziali sincroni ed asincroni

Il circuito riconoscitore di coppie di 1 rappresentato in figura 1.2 é detto *sincrono*, in quanto le operazioni di memorizzazione dello stato (come variabile di uscita del flip-flop) sono scandite dalla presenza di un segnale di clock. Occorre tuttavia notare come possano esistere circuiti sequenziali sprovvisti di clock; tali circuiti sono detti *asincroni*.

Ad esempio, si consideri il circuito di figura 1.4, che rappresenta un circuito in cui le uscite dipendono dalla storia degli ingressi, ma in cui non é presente alcun segnale di clock.

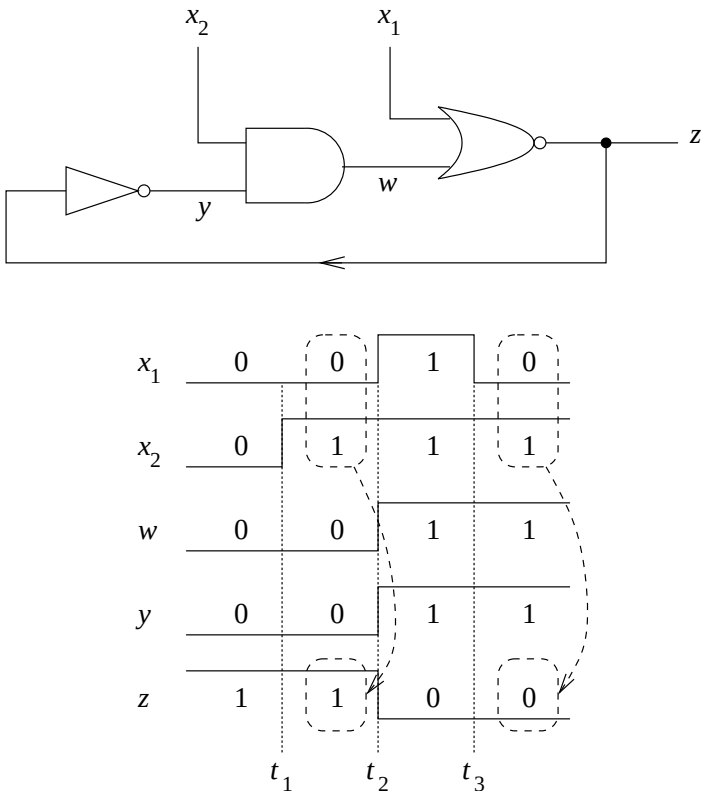

Figura 1.4: Esempio di circuito sequenziale asincrono

Le variabili x_1 ed x_2 rappresentano gli ingressi del circuito, la z la sua uscita. Inizialmente, applichiamo gli ingressi $x_1 = x_2 = 0$. In questo caso, l'uscita w della portau AND sará sicuramente 0 (indipendentemente dal valore dell'altro suo ingresso y), e pertanto entrambi gli ingressi della porta NOR valgono 0. Si avrá pertanto $z = 1$, e dato che $y = \overline{z}$, si avrá $y = 0$. Questa situazione é rappresentata sempre in figura 1.4, per $t < t_1$.

A partire dall'istante di tempo t_1, cambiamo il valore dell'ingresso x_2, portandolo a livello alto, senza variare x_1 (basso). L'uscita w della porta AND, prodotto di $y = 0$ con x_2 rimarrá a livello basso, e pertanto l'uscita del circuito $z = \overline{x_1 + w}$ rimarrá a livello alto. Conseguentemente, y rimarrá ancora a livello basso.

A partire dall'istante t_2, cambiamo il solo valore di x_1, cioé applichiamo agli ingressi la combinazione $x_1 = x_2 = 1$. L'uscita della porta OR (e del circuito) si porterá a livello basso, e conseguentemente si avrá $y = 1$, da cui $w = x_2 y = 1$.

A partire da t_3, riportiamo x_1 a livello basso (con $x_2 = 1$). Gli ingressi

hanno ora gli stessi valori che avevano nell'intervallo di tempo compreso tra t_1 e t_2. Ma come vedremo fra breve, l'uscita z avrá ora un valore differente. Infatti, avendosi $w = 1$, su ha necessariamente $z = 0$ e $y = 1$.

Pertanto notiamo, che pur presentandosi *due situazioni temporali aventе la stessa combinazione degli ingressi* (zone racchiuse in tratteggio nel grafico di figura 1.4), *l'uscita non dipende solo da essi*, ma é dipendente dall'*andamento storico* degli ingressi. Non si tratta dunque di un circuito combinatorio, e pertanto il circuito in esame costituisce un esempio di circuito sequenziale (asincrono, per la mancanza del clock).

1.3 Progettazione dei circuiti sequenziali sincroni

1.3.1 Riconoscitore di sequenza

Nel seguente paragrafo, ci proponiamo di progettare un automa sincrono sequenziale in grado di riconoscere la sequenza di bit $s = 0110$. Precisamente, vogliamo che l'uscita dell'automa sia un 1 se e solo se in ingresso si é presentata la sequenza s. Inoltre, come specifica di progetto, vogliamo che vengano riconosciute anche sequenze *annidate*, cioé tali che la fine di una sequenza possa essere considerata come l'inizio della sequenza successiva. L'andamento temporale ingresso-uscita desiderato è illustrato in figura 1.5.

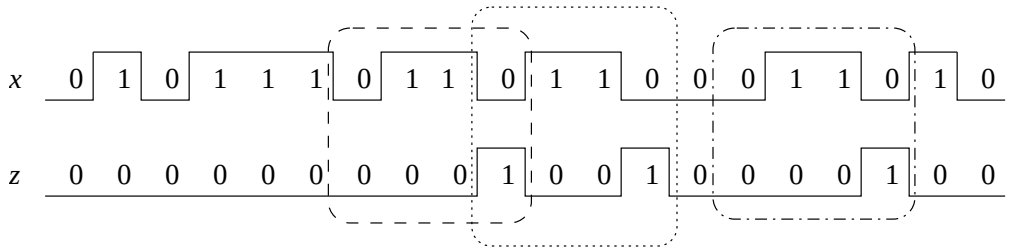

Figura 1.5: Relazione ingresso-uscita per il riconoscitore di sequenza

Nella figura 1.5, l'ingresso é indicato con x, e l'uscita con z. Notiamo come l'ultimo zero della sequenza s possa essere utilizzato anche come primo zero di una nuova sequenza (si vedano le zone evidenziate in tratteggio e in punteggiato), secondo le specifiche di progetto.

Il diagramma degli stati

Per poter continuare nel progetto, occorre tracciare il *diagramma degli stati*, ovvero cercare di rappresentare matematicamente il fatto che l'automa conserva memoria storica degli ingressi ricevuti.

Piú precisamente, dato che la sequenza s é formata da quattro bit, occorre memorizzare quali sono gli ultimi quattro bit ricevuti in ingresso dalla macchina.

Supponiamo che la macchina parta in una condizione (o stato) iniziale in cui non sia mai stato applicato un ingresso; individuiamo questa condizione con lo *stato* A. Se ora in ingresso ricevessimo un 1, possiamo pensare che esso non costituisce l'inizio della sequenza desiderata; la macchina deve pertanto rigettare qualsiasi sequenza che inizia per 1.

In questo caso, la macchina rimane nello stato iniziale A, aspettando in ingresso uno 0, e l'uscita della macchina é ovviamente $z = 0$.

Se in ingresso vi é uno 0, possiamo essere certi che questo bit costituisce il primo simbolo della sequenza da riconoscere; per tale motivo dobbiamo memorizzare questo fatto, e pertanto portiamo la macchina nello stato B. L'uscita si mantiene ovviamente a zero, dato che la sequenza non é stata ancora ricevuta nella sua interezza.

Possiamo indicare questo ragionamento con un grafo, detto *diagramma degli stati*; in esso appaiono gli stati della macchina (A, B, ...) racchiusi in circoletti, e questi stati sono collegati da degli archi orientati. Ad esempio, in figura 1.6 l'arco che va da A a B presenta come etichetta la dicitura $x = 0, z = 0$, che sta a significare che, quando la macchina si trova nello stato iniziale A, e l'ingresso x é uno zero, l'automa si porta nel nuovo stato B, con uscita $z = 0$.

Analogamente, l'arco che inizia e termina in A indica che, con ingresso $x = 1$, lo stato della macchina rimane A, e che l'uscita é $z = 0$.

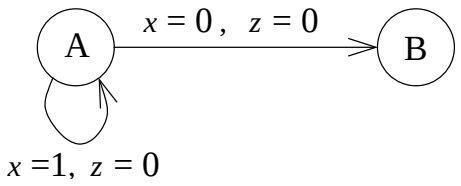

Figura 1.6: Diagramma degli stati (ancora incompleto)

Ovviamente, il diagramma di figura 1.6 é ancora incompleto, rappresentando il comportamento della macchina solo dopo che essa ha ricevuto in ingresso un unico bit.

Da esso possiamo comunque notare che la macchina resta nello stato A se in ingresso si ha una successione di 1, e si porta nello stato B appena riceve il primo 0 in ingresso.

Se ora siamo nello stato B, dobbiamo pensare che l'automa sta aspettando in ingresso il secondo bit della sequenza s, che é un 1. Pertanto, se il secondo bit vale $z = 1$ la macchina riconosce la prima metá della sequenza s (cioè la stringa 01), e per memorizzare questo fatto, si porta in un nuovo stato C. L'uscita si mantiene ovviamente a 0, perché la sequenza é ancora incompleta.

Se invece in ingresso vi é uno zero, la macchina rimane in B (con uscita 0), utilizzando lo 0 ricevuto come primo bit della sequenza, e rimane in B se in ingresso continua a ricevere una successione di zeri.

Consideriamo ora lo stato C; arrivati in esso, abbiamo giá riconosciuto la prima metá della sequenza. Il terzo bit della sequenza é un 1, e pertanto se in ingresso si ha $x = 1$ accettiamo tale input come terzo bit valido, e ci portiamo in un nuovo stato D, ovviamente sempre con uscita $z = 0$. Abbiamo ora riconosciuto la stringa 011.

Se invece, a partire dallo stato C, abbiamo in ingresso uno 0, dobbiamo rigettare tale sequenza (perché in ingresso abbiamo ricevuto la stringa 010 anziché 011). Notiamo come tuttavia non bisogna rigettare completamente la sequenza ricevuta (e tornare di conseguenza nello stato iniziale A), ma possiamo utilizzare l'ultimo 0 ricevuto come inizio di una nuova sequenza (e pertanto andiamo dallo stato C allo stato B, ove attenderemo l'arrivo di un 1 in ingresso).

Se nello stato D (in cui abbiamo ricevuto la stringa 011) riceviamo in ingresso un altro 1, ci ritroveremo con la stringa 0111, che dobbiamo rigettare, e pertanto ci riposizioneremo nello stato iniziale A (con uscita ovviamente 0, dato che abbiamo rigettato la stringa ricevuta in ingresso), ed attenderemo di nuovo la stringa $s = 0110$.

Se invece nello stato D riceviamo in ingresso il simbolo $x = 0$, abbiamo ricevuto la stringa $s = 0110$ desiderata. Per questo motivo, l'uscita si porterá al valore $z = 1$. Nello stesso tempo, l'ultimo 0 ricevuto puó essere considerato, oltre che come ultimo simbolo della sequenza s, come primo simbolo della sequenza successiva. Per tale motivo, ci posizioneremo nello stato B, in attesa dei rimanenti tre bit (110) della stringa da riconoscere.

Ricordiamo che se la coda di una sequenza viene riutilizzata come testa di una nuova sequenza, diremo che l'automa é in grado di riconoscere sequenze *annidate*.

Notiamo invece che il riconoscitore di sequenza 11 di figura 1.2 lavora su sequenze *non annidate* (vedi figura 1.3).

Il diagramma degli stati, cosí completato, é illustrato in figura 1.7. In questo diagramma, la coppia ingresso uscita non é piú rappresentata come

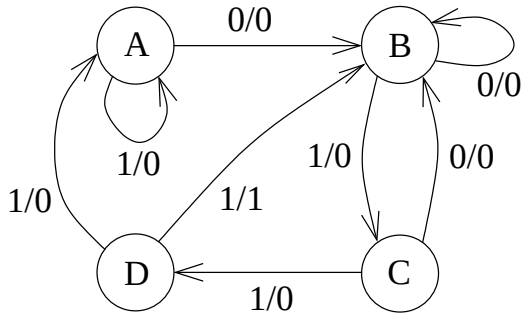

Figura 1.7: Diagramma degli stati del riconoscitore di sequenza

$x = 0$, $z = 0$ (e combinazioni simili), ma semplicemente (per brevitá) come $0/0$, e simili, ove il simbolo prima della barra '/' rappresenta l'ingresso, e quello dopo la barra rappresenta l'uscita.

In figura 1.8 é altresí illustrata la relazione tra l'ingresso x, lo stato y e l'uscita z della macchina.

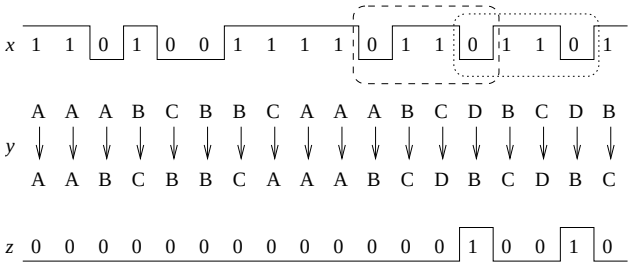

Figura 1.8: Relazione tra ingresso, stato e uscita del riconoscitore di sequenza

La tabella degli stati

Il passo successivo consiste nel ricavare una rappresentazione equivalente, detta *tabella degli stati*. Questa tabella ha tante righe quanti sono gli stati dell'automa (nel notro caso quattro), e tante colonne quante sono le possibili combinazioni degli ingressi (due nel nostro caso, avendosi solo i casi $x = 0$ ed $x = 1$).

All'interno della tabella vengono scritti lo stato successivo in cui si porta la macchina dopo aver applicato l'ingresso, e la corrispondente uscita. Per il riconoscitore di sequenza, abbiamo la tabella 1.2

Ad esempio, la dicitura B, 0 nella riga individuata dallo stato (attuale) A e nella colonna individuata dall'ingresso $x = 0$ significa che, quando l'automa

Tabella 1.2: Tabella degli stati del riconoscitore di sequenza

	$x = 0$	$x = 1$
A	B, 0	A, 0
B	B, 0	C, 0
C	B, 0	D, 0
D	B, 1	A, 0

si trova nello stato A, e in ingresso si ha uno zero, lo stato si porta nel nuovo stato (o stato successivo) B, con uno zero in uscita.

L'assegnamento degli stati

Se identifichiamo ciascuno stato dell'automa con un numero binario, in modo che a stati distinti corrispondano numeri distinti, abbiamo fatto l'operazione di *assegnamento* degli stati. Questo assegnamento non é univoco; ad esempio nel nostro caso possiamo scegliere di assegnare allo stato A il numero 00, allo stato B il numero 10, allo stato C il numero 11, ed a D il numero 01.

Nulla ci vieta di fare un qualsiasi altro assegnamento, ad esempio con le associazioni A$\leftrightarrow$11; B $\leftrightarrow$ 00; C $\leftrightarrow$ 01; D $\leftrightarrow$ 10.

Ad ogni cifra binaria dell'assegnamento, assegnamo un nome (nel nostro caso, avendo due bit, utilizzeremo due nomi di variabile, y_1 ed y_2), come nella tabella 1.3 che rappresenta l'assegnamento scelto.

Tabella 1.3: Assegnamento degli stati

y	$y_1\, y_2$
A	0 0
B	1 0
C	1 1
D	0 1

Sostituiamo ora le coppie di bit $y_1\, y_2$ nella tabella degli stati 1.2; per far ciò indicheremo con lettere *minuscole* lo stato attuale, e con *maiuscole* lo stato successivo. Cioè, all'esterno della tabella gli stati saranno indicati con la coppia $y_1\, y_2$, ed all'interno della tabella con la coppia di variabili $Y_1\, Y_2$. Il risultato della sostituzione é illustrato in tabella 1.4

Tabella 1.4: Tabella degli stati dopo l'assegnamento

	$x = 0$	$x = 1$
$y_1\, y_2$	$Y_1\, Y_2,\, z$	$Y_1\, Y_2,\, z$
00	10, 0	00, 0
10	10, 0	11, 0
11	10, 0	01, 0
01	10, 1	00, 0

La minimizzazione delle funzioni di stato, e di uscita

Ora, si tratta di scrivere le *equazioni logiche* per l'automa. Per far ció, *si procede esattamente come nella minimizzazione dei circuiti combinatori*, ad esempio con l'uso delle mappe di Karnaugh. E precisamente:

- Le variabili di ingresso (x), e quelle che indicano lo stato attuale della macchina (y_1 ed y_2) sono considerate come *variabili di ingresso delle espressioni combinatorie*, e pertanto appaiono all'*esterno* delle mappe di Karnaugh.

- L'uscita (z) e le variabili che indicano lo *stato successivo* (cioé Y_1 ed Y_2) sono considerate come variabili *dipendenti* (cioè di *uscita*) nelle espressioni booleane, e pertanto appariranno all'*interno* delle mappe.

.

Consideriamo ad esempio la variabile Y_1; essa assumerá il valore 1 unicamente nei seguenti casi, come si osserva dalla tabella 1.4:

- $x = 1, Y_1 = 1, Y_2 = 0$ (seconda riga e terza colonna della tabella 1.4).

- $x = 1, Y_1 = 1, Y_2 = 1$ (terza riga e terza colonna della tabella 1.4).

La mappa di Karnaugh, relativa alla variabile Y_1 é rappresentata in figura 1.9

Da essa ricaviamo l'equazione per Y_1, in funzione delle variabili y_1, y_2, x:

$$Y_1 = \overline{x} + y_1\, \overline{y}_2 \tag{1.1}$$

Analogamente, possiamo ricavare la mappa per la variabile Y_2; essa é rappresentata in figura 1.10

L'equazione per Y_2 é la seguente:

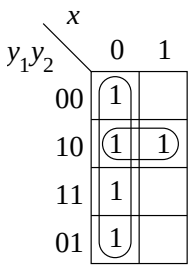

Figura 1.9: Mappa di Karnaugh per Y_1

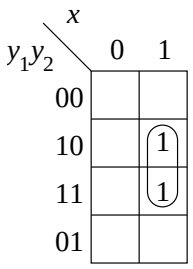

Figura 1.10: Mappa di Karnaugh per Y_2

$$Y_2 = x\, y_1 \tag{1.2}$$

Le equazioni 1.1 e 1.2, che legano lo stato successivo, individuato dalle variabili Y_1, Y_2 allo stato attuale (individuato da y_1 e da y_2) ed all'ingresso x sono dette *equazioni di stato*.

Non ci resta, infine, che ricavare l'espressione dell'uscita z in funzione delle variabili y_1, y_2 ed x; l'equazione che otterremo é detta *equazione di uscita*.

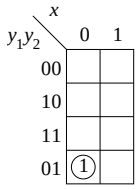

Figura 1.11: Mappa di Karnaugh per z

La mappa per l'uscita z é riportata in figura 1.11, e l'equazione di uscita é la 1.3.

$$z = \overline{y}_1 y_2 \overline{x} \tag{1.3}$$

Notiamo come, avendosi un solo caso in cui $z = 1$, si poteva fare a meno di utilizzare la mappa di Karnaugh per la semplificazione (che invece sará utile in tutti i casi in cui vi é piú di una combinazione dei valori di y_1, y_2, x che dia luogo ad una uscita $z = 1$).

La realizzazione del circuito

La realizzazione del circuito avviene in due fasi: si disegna innanzitutto la rete combinatoria ottenibile dalle equazioni di stato 1.1 e 1.2 e dall'equazione di uscita 1.3, come in figura 1.12

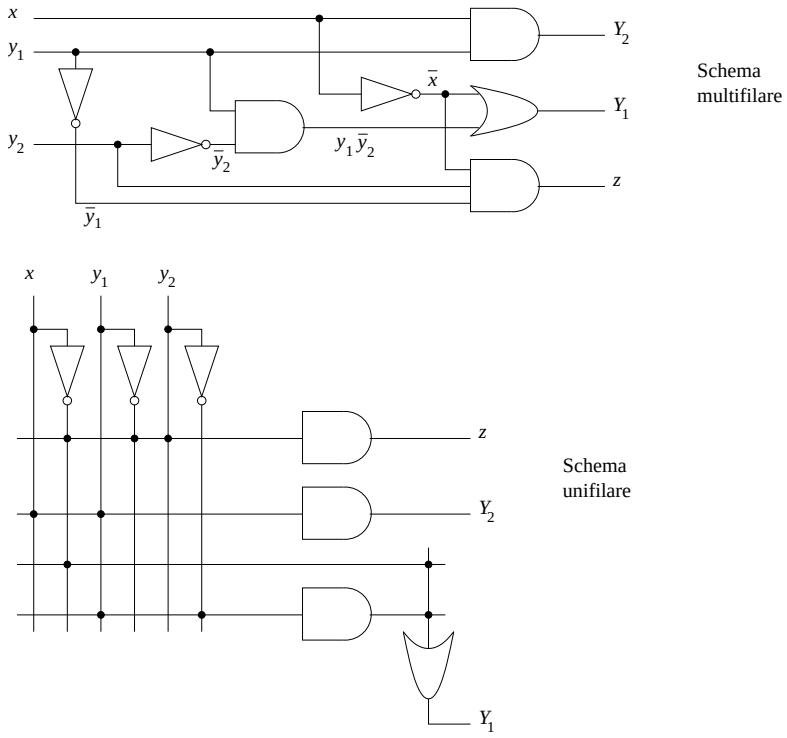

Figura 1.12: Rete combinatoria "bozza" del circuito riconoscitore di sequenza

Notiamo come la rappresentazione unifilare (vedi appendice A) sia piú comoda della consueta rappresentazione multifilare utilizzata nella maggior parte dei libri di elettronica digitale.

Ricordando che ogni stato successivo individuato da Y_1, Y_2 é canditato, dopo un breve instante di tempo dall'applicazione dell'ingresso e del segnale

di clock, a diventare il nuovo stato attuale (individuato da y_1, y_2), possiamo pensare di realizzare ció semplicemente interponento un elemento di ritardo tra Y_1 ed y_1, cosí come tra Y_2 ed y_2.

Il piú semplice elemento di ritardo é costituito da un flip flop di tipo D, con in ingresso lo stato Y_1 (o Y_2) e in uscita lo stato y_1 (o y_2). Dato che ciascuno stato é individuato da due bit, occorreranno pertanto due flip-flop.

Occorre poi inviare il clock c_k sia all'ingresso (con una porta AND avente x come secondo ingresso) che agli stessi flip-flop. Il riconoscitore di sequenza é cosí ottenuto é riportato in figura 1.13. Notiamo come abbiamo disegnato tratteggiato il comando c_k inviato ai flip-flop; nei prossimi circuiti che disegneremo considereremo scontato tale collegamento, che non verrá piú disegnato. La rappresentazione multifilare dell'automa é omessa ed é lasciata al lettore per esercizio.

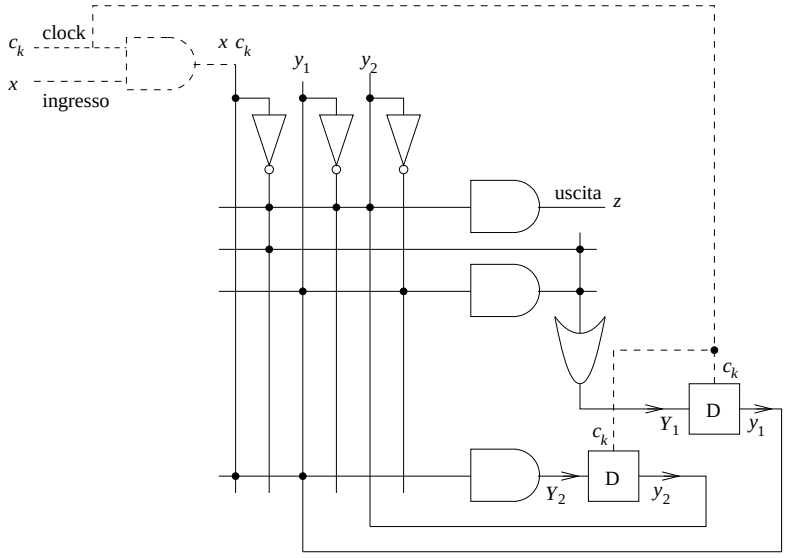

Figura 1.13: Riconoscitore della sequenza 0110

Un diverso assegnamento degli stati

L'assegnamento degli stati é, come abbiamo detto, arbitrario. Possiamo ad esempio realizzare il riconoscitore di sequenza descritto nella tabella degli stati 1.2 mediante un differente assegnamento, ad esempio quello di tabella 1.5.

Otteniamo allora la nuova tabella degli stati (con assegnamento) 1.6. Le corrispondenti mappe, solo riportate in figura 1.14.

Tabella 1.5: Nuovo assegnamento degli stati

y	$y_1\, y_2$
A	1 1
B	0 0
C	1 0
D	0 1

Tabella 1.6: Tabella degli stati dopo il nuovo assegnamento

	$x=0$	$x=1$
$y_1\, y_2$	$Y_1\, Y_2,\, z$	$Y_1\, Y_2,\, z$
11	00, 0	11, 0
00	00, 0	10, 0
10	00, 0	01, 0
01	00, 1	11, 0

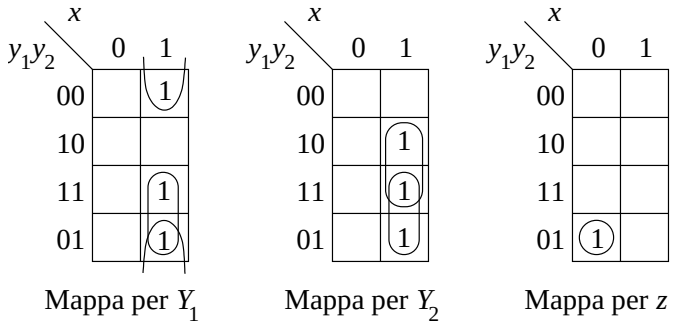

Figura 1.14: Mappe di Karnaugh per il nuovo assegnamento

Da esse possiamo ricavare le nuove equazioni di stato 1.4 e 1.5 e l'equazione di uscita 1.6:

$$Y_1 = y_2 x + \overline{y}_1 x \tag{1.4}$$

$$Y_2 = y_1 x + y_2 x \tag{1.5}$$

$$z = \overline{x}\,\overline{y}_1\, y_2 \tag{1.6}$$

Il circuito corrispondente a queste equazioni é rappresentato in figura 1.15 (ove sono stati omessi i collegamenti del segnale di clock).

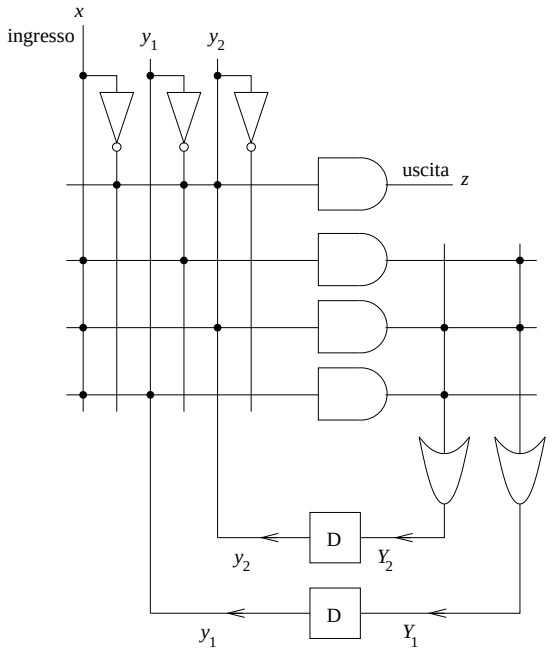

Figura 1.15: Circuito del riconoscitore di sequenza con il nuovo assegnamento

Notiamo come questo circuito sia piú complesso, per numero di porte utilizzate, di quello di figura 1.13, ottenuto con il primo assegnamento. L'assegnamento scelto influenza quindi la complessitá circuitale.

Purtroppo, non esiste un metodo generale per stabilire quale sia il miglior assegnamento (cioé quello che permette di ottenere il circuito con il minor numero di porte), ed occorrono diversi tentativi, con differenti assegnamenti, sino ad ottenere un circuito di complessitá accettabile.

Esercizio

Progettare, a partire dal diagramma degli stati, il riconoscitore di coppie di 1 di figura 1.2.

Soluzione

Il diagramma degli stati é riportato in figura 1.16. In esso notiamo che lo stato A viene abbandonato se e solo se si riceve il primo uno della sequenza in ingresso, per portarci nel nuovo stato B.

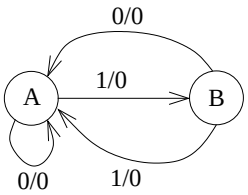

Figura 1.16: Diagramma degli stati per il riconoscimento di coppie di 1 non annidate

Se il bit successivo é uno 0, occorre ricominciare daccapo: torniamo in A, mantenendo l'uscita z a zero. Se invece é un 1, abbiamo completato la sequenza; torniamo ancora in A pronti a ricevere una nuova coppia di 1, ma segnaliamo il corretto riconoscimento della sequenza portando l'uscita z a livello alto.

Tabella 1.7: Tabella degli stati del riconoscitore di coppie di 1

	$x=0$	$x=1$
A	A, 0	B, 0
B	A, 0	A, 1

La tabella 1.7 degli stati é ricavata dal diagramma degli stati. Scegliamo l'assegnamento $A \leftrightarrow y=0$, $B \leftrightarrow y=1$, ottenendo coí la tabella 1.8 (L'altro assegnamento possibile, ovvero $A \leftrightarrow 1$, $B \leftrightarrow 0$ é lasciato al lettore come esercizio).

Le equazioni sono allora:

$$Y = x\overline{y}$$

$$z = xy$$

dalle quali si ricava il circuito di figura 1.2.

Tabella 1.8: Tabella del riconoscitore di coppie di 1 dopo l'assegnamento

	$x = 0$	$x = 1$
y	Y, z	Y, z
0	0, 0	1, 0
1	0, 0	0, 1

1.3.2 Progetto di un automa con flip-flop di tipo T

Come elemento di memoria, non siamo necessariamente obbligati ad utilizzare un flip-flop di tipo D. La scelta fatta nei paragrafi precedenti era dovuta al fatto che, con tale dispositivo, il valore attuale di una variabile di stato y appariva all'uscita del flip-flop, e il valore successivo come suo ingresso Y, in modo tale che sul fronte di discesa del clock, il nuovo stato y assume il valore Y (lo stato successivo diventa il nuovo stato attuale).

Se prescindiamo da questa scelta, possiamo pensare di utilizzare altri tipi di flip-flop, come illustreremo in questo paragrafo e nei successivi due.

Consideriamo un flip-flop di tipo T, la cui evoluzione temporale é riportata in tabella 1.9. Per dettagli sul suo funzionamento, si rimanda all'appendice B.

Tabella 1.9: Tabella del flip-flop T

t	y_a	$\rightarrow$	y_s
0	0	$\rightarrow$	0
0	1	$\rightarrow$	1
1	0	$\rightarrow$	1
1	1	$\rightarrow$	0

Da questa tabella, si vede che con ingresso $t = 0$, l'uscita del flip-flop rimane invariata, mentre applicando un ingresso $t = 1$ (ed il clock c_k), l'uscita cambia il suo valore (da 1 a 0 o viceversa) sul fronte di discesa del clock. La variabile y_a indica il valore dell'uscita prima della commutazione (valore *attuale*), mentre y_s indica il valore *successivo*, cioé dopo il fronte di discesa del clock. Salvo avviso contrario, *tutti i flip-flop utilizzati nel presente libro commuteranno sul fronte di discesa del clock.*

Proviamo ora a realizzare il riconoscitore della sequenza 0110 utilizzando dei flip-flop di tipo T. Riportiamo, per maggiore chiarezza, la sua tabella degli stati (vedi 1.10).

Tabella 1.10: Tabella degli stati del riconoscitore della sequenza 0110

	$x = 0$	$x = 1$
A	B, 0	A, 0
B	B, 0	C, 0
C	B, 0	D, 0
D	B, 1	A, 0

Scegliamo poi l'assegnamento di tabella 1.3, cioé: A $\leftrightarrow y_1 = 0, y_2 = 0$, B $\leftrightarrow y_1 = 1, y_2 = 0$, C $\leftrightarrow y_1 = 1, y_2 = 1$, D $\leftrightarrow y_1 = 0, y_2 = 1$. Riportiamo altresí la tabella degli stati dopo l'assegnamento (vedi 1.11).

Tabella 1.11: Tabella degli stati del riconoscitore dopo l'assegnamento

	$x = 0$	$x = 1$
$y_1\, y_2$	$Y_1\, Y_2$, z	$Y_1\, Y_2$, z
00	10, 0	00, 0
10	10, 0	11, 0
11	10, 0	01, 0
01	10, 1	00, 0

A questo punto, occorre introdurre la cosiddetta *tabella di eccitazione*; per far ció dobbiamo immaginare di avere due flip-flop T_1 e T_2 rispettivamente con uscite y_1 ed y_2 (le variabili di stato dell'automa), e con ingressi t_1, t_2. Notiamo che occorrono tanti flip-flop quante sono le variabili di stato della macchina, esattamente come capitava con i dispositivi di tipo D.

Si tratta ora di indicare quali valori applicare agli ingressi dei due flip-flop al fine di far commutare le loro uscite, in corrispondenza delle transizioni delle variabili di stato y_1 ed y_2 nel passare dallo stato attuale al successivo.

Consideriamo ad esempio la prima riga della tabella 1.11. Se applichiamo in ingresso all'automa il valore $x = 0$, si avrá una commutazione della variabile di stato y_1 dal valore attuale 0 (prima colonna della tabella) al nuovo valore 1 (riportato nella seconda colonna). Se pensiamo ad y_1 come all'uscita di T_1, possiamo dire che, al fine di far commutare y_1, dobbiamo applicare il segnale $t_1 = 1$ all'ingresso del dispositivo di memoria.

Analogamente, sempre con in ingresso $x = 0$, la variabile y_2 rimarrá a livello basso (si veda sempre la prima riga della tabella 1.11). Considerandola come uscita di T_2, occorrerá che si abbia $t_2 = 0$ affinché l'uscita del dispositivo non cambi di valore.

Possiamo pertanto iniziare a riempire la tabella di eccitazione 1.12. Questa tabella é simile alla tabella degli stati, ma al suo interno, anziché contenere i valori degli stati Y_1 ed Y_2 compaioni i valori degli *ingressi dei flip-flop* t_1 e t_2. I valori delle uscite z rimangono gli stessi della tabella degli stati.

Sempre considerando la prima riga della tabella 1.11, osserviamo che con ingresso $x = 1$ le variabili di stato y_1 ed y_2 non cambiano il loro valore; pertanto nella colonna relativa all'ingresso $x = 1$ della tabella di eccitazione si dovrá avere $t_1 = t_2 = 0$.

Nella tabella 1.12, i posti lasciati bianchi devono ancora essere completati.

Tabella 1.12: Tabella di eccitazione (incompleta) del riconoscitore con flip-flop di tipo T

	$x = 0$	$x = 1$
$y_1\, y_2$	$t_1\, t_2,\, z$	$t_1\, t_2,\, z$
00	10, 0	00, 0
10		
11		
01		

Ripetendo il ragionamento per tutte le rimanenti righe della tabella 1.11, si ottiene la tabella di eccitazione completa 1.13.

Tabella 1.13: Tabella di eccitazione del riconoscitore con flip-flop di tipo T

	$x = 0$	$x = 1$
$y_1\, y_2$	$t_1\, t_2,\, z$	$t_1\, t_2,\, z$
00	10, 0	00, 0
10	00, 0	01, 0
11	01, 0	10, 0
01	11, 1	01, 0

A questo punto, si tratta di scrivere le equazioni per t_1 e t_2 (e non piú di Y_1 ed Y_2) in funzione delle variabili y_1, y_2, x. L'equazione per l'uscita z rimane uguale a quella giá trovata con il progetto che utilizzava flip-flop D, ovvero:

$$z = \overline{x}\,\overline{y}_1\, y_2 \tag{1.7}$$

Per quanto riguarda le espressioni per t_1 e t_2, possiamo ricavarle dalle mappe rappresentate in figura 1.17.

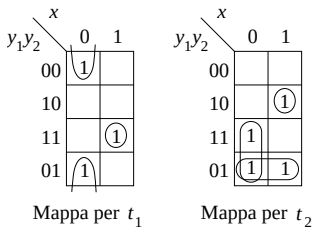

Figura 1.17: mappe di Karnaugh per t_1 e t_2

Le equazioni sono allora:

$$t_1 = \overline{x}\,\overline{y}_1 + y_1\,y_2\,x \tag{1.8}$$

$$t_2 = \overline{x}\,y_2 + \overline{y}_1\,y_2 + x\,y_1\,\overline{y}_2 \tag{1.9}$$

Il circuito é rappresentato in figura 1.18; la porta indicata con una A al suo interno corrisponde al termine utilizzato per scongiurare l'esistenza di alee nei circuiti reali.

1.3.3 Progetto di un automa con flip-flop di tipo SR

Riprogettiamo ora il riconoscitore di sequenza 0110 utilizzando dei flip-flop di tipo SR; per far ció consideriamo la tabella 1.14 di questo dispositivo. Per una maggiore informazione sul flip-flop SR, si rimanda all'appendice B.

Tabella 1.14: Tabella flip-flop SR

s	r	y_a	$\rightarrow$	y_s
0	X	0	$\rightarrow$	0
X	0	1	$\rightarrow$	1
1	0	0	$\rightarrow$	1
0	1	1	$\rightarrow$	0

Ricordiamo che la combinazione $s = r = 1$ non é ammessa, dando luogo ad una *corsa critica*, come illustrato in appendice B.

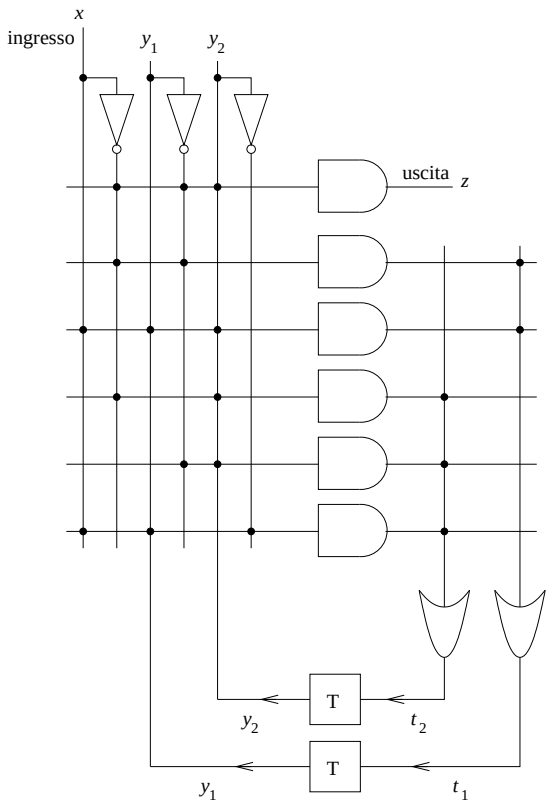

Figura 1.18: Circuito del riconoscitore con flip-flop di tipo T

Occorre utilizzare ancora due flip-flop, le cui uscite sono le variabili di stato y_1 ed y_2, le quali sono rispettivamente funzione delle variabili di ingresso s_1, r_1 ed s_2, r_2.

Si tratta ora di costruire la tabella di eccitazione a partire dalla tabella degli stati; consideriamo ancora la tabella degli stati dopo l'assegnamento 1.15 e da essa costruiamo la tabella di eccitazione in modo simile a quanto fatto con i flip-flop di tipo T.

Notiamo che, applicando in ingresso $x = 0$, lo stato attuale $y_1 = 0$, $y_2 = 0$ diventa lo stato $Y_1 = 1$, $Y_2 = 0$. Per poter cambiare il valore di y_1 da 0 ad 1 occorre applicare al primo flip-flop la combinazione degli ingressi $s_1 = 1$, $r_1 = 0$. Analogamente, per poter mantenere y_2 al valore 0, occorre e basta che si abbia $s_2 = 0$ ed $r_2 =$ X, ovvero è indifferente porre $r_2 = 0$ oppure $r_2 = 1$, come si deduce dalla tabella 1.14,

Vedremo che l'uso delle indifferenze permetterà di avere una maggiore libertà nelle semplificazioni delle equazioni booleane.

La tabella di eccitazione viene completata, riga per riga, ottenendo così

Tabella 1.15: Tabella degli stati del riconoscitore dopo l'assegnamento

	$x = 0$	$x = 1$
$y_1\, y_2$	$Y_1\, Y_2,\, z$	$Y_1\, Y_2,\, z$
00	10, 0	00, 0
10	10, 0	11, 0
11	10, 0	01, 0
01	10, 1	00, 0

la tabella 1.16.

Tabella 1.16: Tabella di eccitazione del riconoscitore con flip-flop di tipo SR

	$x = 0$	$x = 1$
$y_1\, y_2$	$s_1\, r_1\, s_2\, r_2,\, z$	$s_1\, r_1\, s_2\, r_2,\, z$
00	10 0X, 0	0X OX, 0
10	X0 0X, 0	X0 10, 0
11	X0 10, 0	01 X0, 0
01	10 01, 1	0X 01, 0

Si tratta ora di scrivere le equazioni relative ad s_1, r_1, s_2, r_2; per quanto riguarda l'uscita z dell'automa, l'equazione booleana non è cambiata e rimane la stessa già ricavata nel progetto con flip-flop di tipo D.

Riportiamo in figura 1.19 le mappe relative alle variabili s_1, r_1, s_2, r_2.

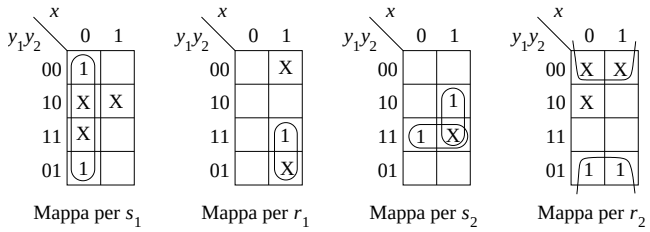

Figura 1.19: Mappe per il riconoscitore realizzato con flip-flop SR

Le equazioni sono:

$$s_1 = \overline{x}$$

$$r_1 = y_2\, x$$

$$s_2 = x\, y_1 + y_1\, y_2$$

$$r_2 = y_1$$

Il circuito è rappresentato in figura 1.20.

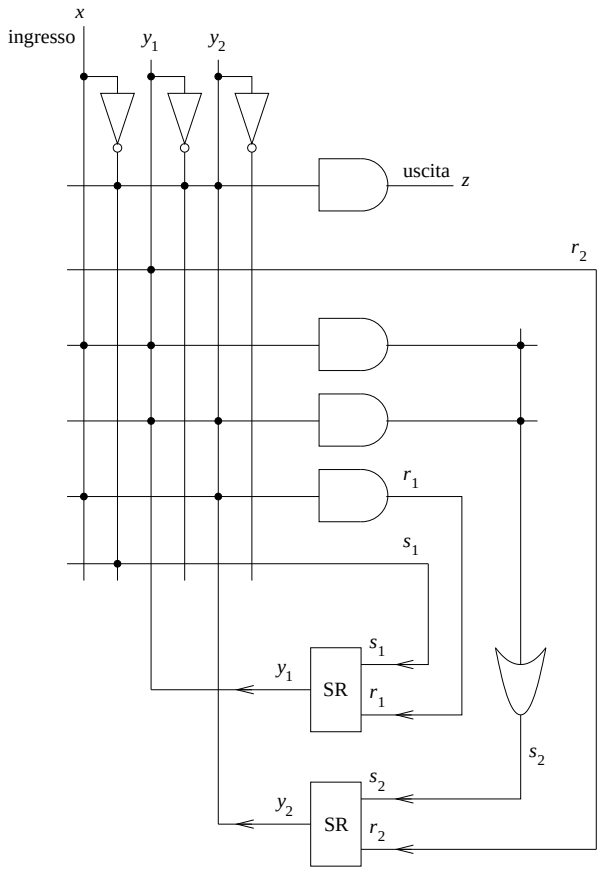

Figura 1.20: Riconoscitore realizzato con flip-flop SR

1.3.4 Progetto di un automa con flip-flop di tipo JK

Con un procedimento del tutto analogo, è possibile costruire la tabella di eccitazione nel caso si desiderino utilizzare dei flip-flop di tipo JK. Ricordiamo che in un flip-flop JK non si ha, a differenza dell'SR, la condizione di corsa critica, in quanto per $j = k = 1$ l'uscita del flip-flop cambia valore in presenza del segnale di clock.

Riportiamo la tabella 1.17 del flip-flop JK, dalla quale, procedendo esattamente come nel paragrafo precedente, è possibile ricavare la tabella di eccitazione dell'automa, riportata in tabella 1.18.

Tabella 1.17: Tabella flip-flop JK

s	r	y_a	$\rightarrow$	y_s
0	X	0	$\rightarrow$	0
X	0	1	$\rightarrow$	1
1	X	0	$\rightarrow$	1
X	1	1	$\rightarrow$	0

Tabella 1.18: Tabella di eccitazione del riconoscitore con flip-flop di tipo JK

	$x = 0$	$x = 1$
$y_1\, y_2$	$s_1\, r_1\, s_2\, r_2,\ z$	$s_1\, r_1\, s_2\, r_2,\ z$
00	1X 0X, 0	0X OX, 0
10	X0 0X, 0	X0 1X, 0
11	X0 1X, 0	X1 X0, 0
01	1X X1, 1	0X X1, 0

Le mappe sono riportate in figura 1.21.

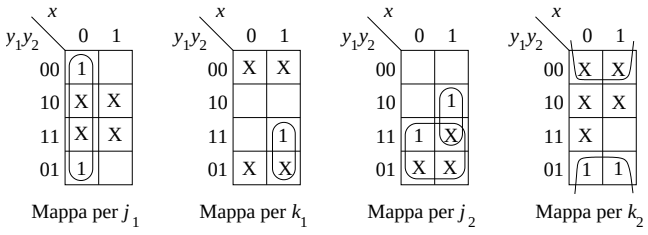

Figura 1.21: Mappe per il riconoscitore realizzato con flip-flop JK

Notiamo come il maggior numero di indifferenze ottenibili utilizzando dei flip-flop JK permetta di scrivere espressioni più semplici rispetto all'uso di SR, consentendo raccoglimenti più ampi all'interno delle mappe. Le equazioni sono le seguenti:

$$j_1 = \overline{x}$$

$$r_1 = y_2\, x$$

$$s_2 = x\, y_1 + y_2$$

$$r_2 = y_1$$

Il circuito è riportato in figura 1.22.

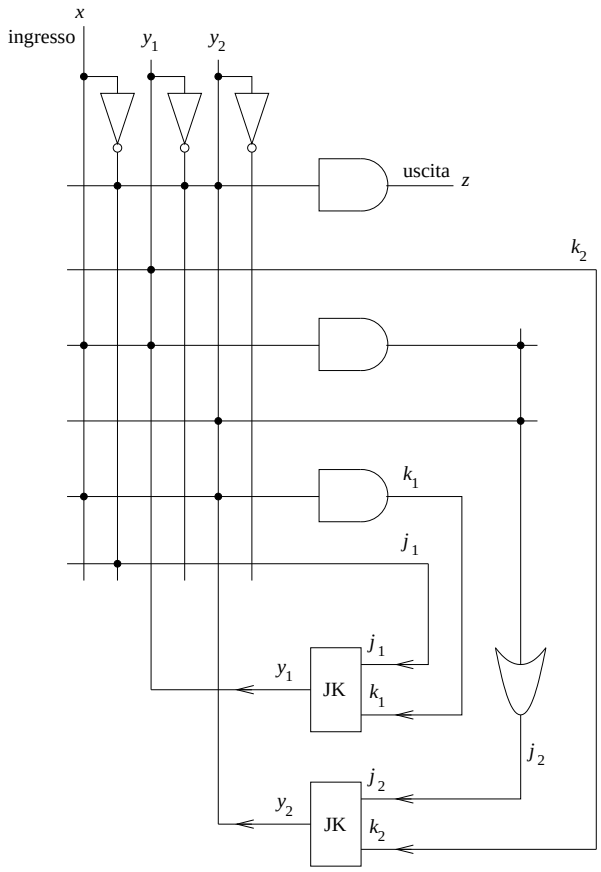

Figura 1.22: Riconoscitore realizzato con flip-flop jk

1.3.5 Full adder seriale

Passiamo ora al progetto di un automa avente due ingressi, e precisamente di un *full adder seriale*. Si tratta di un addizionatore binario in cui in ingresso appaiono le cifre da addizionare a coppie, a partire dai due bit meno significativi, sino agli ultimi due più significativi. In corrispondenza di ciascuna coppia di bit viene inviato alla macchina un impulso di clock.

Rispetto al full adder parallelo (combinatorio), discusso in appendice C, quello seriale tiene in memoria il riporto dell'ultima operazione effettuata con uno stato; se il riporto dell'ultima operazione è zero, si troverà nello stato A, se invece è un uno, nello stato B.

Indichiamo gli ingressi del full adder con x_1 ed x_2, e la sua uscita con z. Osservando che inizialmente, quando si devono sommare i due bit meno significativi il riporto in ingresso (carry-in) è zero, segue che lo stato iniziale è A. In figura 1.23 è rappresentato il diagramma degli stati del dispositivo. Notiamo che, quando siamo nello stato A, e la somma ha riporto (carry-out) uno, ci si sposta nello stato B. Viceversa, se con carry-in pari ad 1 si ha un carry-out nullo ci si riporta di nuovo in A. L'uscita z é il bit ottenuto dalla somma di x_1, x_2 e del carry-in, esattamente come nel full adder parallelo. Notiamo infine che per poter avere in uscita l'ultima cifra (l'ultimo carry out ottenuto dalla somma dei due bit più significativi), occorre applicare in ingresso $x_1 = x_2 = 0$; in questo modo l'ultima uscita disponibile coinciderà con l'ultimo riporto.

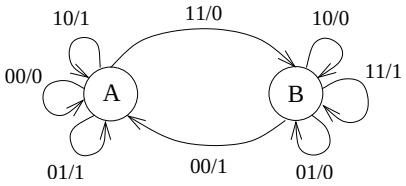

Figura 1.23: Diagramma degli stati di un full adder seriale

Nel diagramma degli stati di figura 1.23 le prime due cifre indicano gli ingressi, e l'ultima l'uscita; ad esempio 10/0 significa $x_1 = 1$, $x_2 = 0$, $z = 0$. Dal diagramma degli stati è possibile ricavare la tabella degli stati.

Tabella 1.19: Tabella degli stati del full adder seriale

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
A	A, 0	A, 1	B, 0	A, 1
B	A, 1	B, 0	B, 1	B, 0

Assegniamo ora allo stato A la variabile $y = 0$, ed allo stato B il valore $y = 1$[1].

Si ha allora la tabella 1.20, dalla quale è possibile ricavare le mappe per Y e per z, in funzione di x_1, x_2, y.

Le equazioni sono le seguenti:

[1]Notiamo come l'assegnamento sia del tutto arbitrario, e non dipende dal fatto che A rappresenta un carry-in pari a 0 e B un carry-in pari ad 1; potevamo tranquillamente assegnare $y = 1$ ad A ed $y = 0$ a B.

Tabella 1.20: Tabella degli stati del full adder dopo l'assegnamento

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
y	Y, z	Y, z	Y, z	Y, z
0	0, 0	0, 1	1, 0	0, 1
1	0, 1	1, 0	1, 1	1, 0

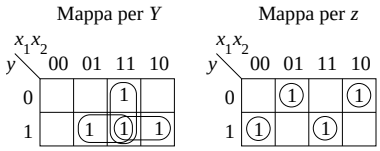

Figura 1.24: Mappe per il full adder seriale

$$Y = x_1\,x_2 + x_1\,y + x_2\,y$$

$$z = \overline{x}_1\,x_2\,\overline{y} + x_1\,\overline{x}_2\,\overline{y} + \overline{x_1}\,\overline{x_2}\,y + x_1\,x_2\,y$$

Il circuito è rappresentato in figura 1.25.

La realizzazione del dispositivo con flip-flop di tipo T, SR o JK è lasciata al lettore come esercizio.

1.3.6 Le macchine iterative

Se consideriamo la parte racchiusa all'interno del tratteggio del circuito di figura 1.25, notiamo che è identica ad una singola cella del full adder combinatorio descritto nell'appendice C. Notiamo altresì che per avere in output le somme di due numeri binari di n cifre ciascuno, occorrono n impulsi di clock per il circuito sequenziale sincrono[2] ed n celle per il circuito combinatorio. Possiamo pertanto pensare al circuito *sequenziale* come una singola cella di un full adder il cui carry-out Y è riportato in ingresso come carry-in y mediante un elemento di ritardo, costituito dal flip-flop D. Se invece il carry-out di una cella viene utilizzato come carry-in della cella successiva, si ha il corrispondente circuito combinatorio (in cui ovviamente si rinuncia al clock). Si veda in proposito la figura 1.26, in cui un full adder combinatorio per la somma di 3 bit, costituito da 3 celle identiche è rappresentato accanto alla sua realizzazione come automa seriale. L'automa seriale ha il vantaggio di essere circuitalmente più semplice (è formato da una sola cella),

[2]Naturalmente occorrono $n + 1$ impulsi se si vuole anche l'ultimo riporto.

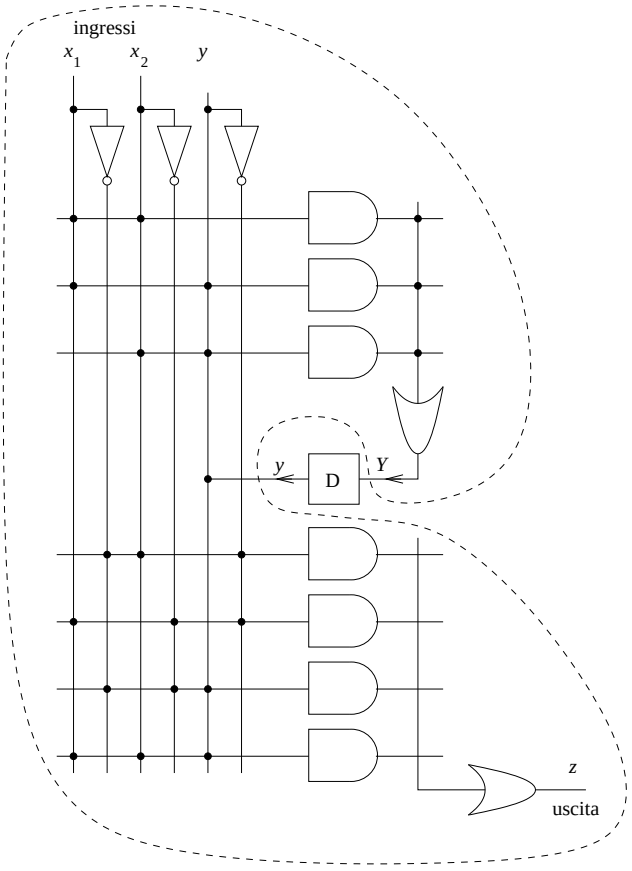

Figura 1.25: Realizzazione di un full adder seriale con flip-flop di tipo D

quello combinatorio è più veloce (l'uscita si ha immediatamente dopo aver applicato i due ingressi).

Una macchina, in cui una cella è identica all'altra (come nel full adder combinatorio) è detta *macchina iterativa*. Oltre al full adder combinatorio di figura 1.26, incontreremo altre macchine iterative nel paragrafo 1.4.

1.3.7 Equivalenza tra automi sincroni e macchine iterative

È importante sottolineare che un qualsiasi automa sincrono seriale realizzato con flip-flop di tipo D può essere trasformato in una macchina iterativa che riceve i suoi input in parallelo. Ad esempio, se consideriamo un qualsiasi ricoscitore di sequenza (ad esempio quello di figura 1.13, in grado di riconoscere la sequenza 0110), possiamo costruire una macchina iterativa parallela

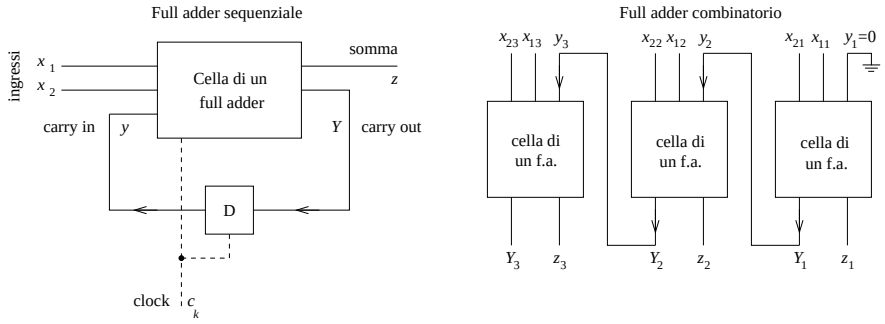

Figura 1.26: Confronto tra full adder sequenziale e combinatorio nella somma di 3 bit

prendendo 4 celle identiche, ciascuna costituita da un riconoscitore identico a quello della figura considerata, e collegando i segnali Y_i all'uscita di una cella agli ingressi y_i della cella successiva. Il tutto è rappresentato in figura 1.27, ove notiamo che l'uscita costituita dai segnali z_1, z_2, z_3, z_4 è una funzione *combinatoria* degli ingressi.

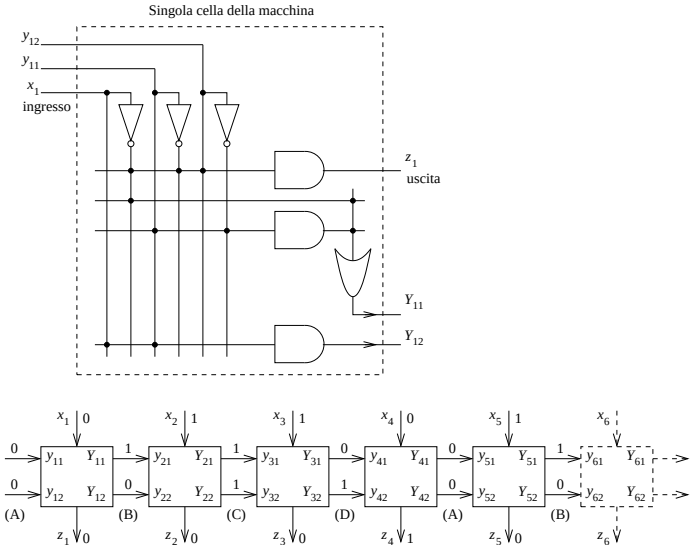

Figura 1.27: Macchina iterativa riconoscitrice di sequenza

In pratica, anzichè collegare i segnali Y_i ed y_i della stessa cella mediante un elemento di ritardo, il flip-flop D, si elimina tale elemento (ed ovviamente anche il clock) e si collegano le celle in cascata. Per quanto riguarda i valori

di y_1 ed y_2 della prima cella, si prendono quelli relativi allo stato iniziale dell'automa, nel nostro caso 00, corrispondente allo stato A.

Se consideriamo l'assegnamento utilizzato per il progetto del full-adder seriale, ovvero: A $\leftrightarrow y_1 = 0, y_2 = 0$, B $\leftrightarrow y_1 = 1, y_2 = 0$, C $\leftrightarrow y_1 = 1, y_2 = 1$, D $\leftrightarrow y_1 = 0, y_2 = 1$,

In figura, si è scelto di indicare gli stati con y_{ni} ed Y_{ni}, ove n indica il numero della cella, ed i l'indice della variabile di stato. Così all'ingresso della prima cella si ha lo stato iniziale $y_{11} = y_{12} = 0$ (lo stato A), ed avendo tale cella ingresso $x_1 = 0$ (il primo bit della sequenza), essa ha in uscita i valori $z_1 = 0$, $Y_{11} = 1$, $Y_{12} = 0$, corrispondenti allo stato B. La seconda cella avrà questi ultimi valori in ingresso (essendo $y_{21} = Y_{11}$, $y_{22} = Y_{22}$) ed avendo al suo ingresso il valore $x_2 = 0$, il suo stato di uscita sarà C, ovvero $Y_{21} = 1$, $Y_{22} = 1$. In questo modo, lo stato successivo di una cella diventa lo stato attuale della successiva, come indicato in figura 1.27.

Come si vede dalla figura, é possibile utilizzare un numero arbitrario di celle, in modo da riconoscere più sequenze all'interno di una stringa di bit. Se si utilizzano k celle, è possibile avere in input (ed ovviamente anche in output) k bit in parallelo.

1.4 Progetto di contatori sincroni

Nel presente paragrafo, ci occuperemo della progettazione di contatori sincroni. Contrariamente a quanto scritto in numerosi libri di elettronica elementare, per ciascun tipo di flip-flop scelto (D, T, SR, JK) il progetto non è univoco, ma dipende anche dall'assegnamento scelto, come vedremo nel seguito della trattazione.

1.4.1 Progetto di un contatore modulo 4 con flip-flop di tipo D

Consideriamo il diagramma degli stati di figura 1.28, esso si riferisce ad un contatore modulo quattro, ovvero ad un dispositivo che presenta un 1 in uscita se e solo se in ingresso sono stati applicati quattro 1, indipendentemente dal numero di 0 applicati. Dopo aver contato i quattro 1, il contatore si riporta nello stato iniziale A per ricominciare il conteggio.

Come si vede, si avanza di uno stato se e solo se in ingresso si ha $z = 1$. Da esso possiamo facilmente ricavare la tabella degli stati 1.21:

Consideriamo ora l'assegnamento:

In questo modo si ottiene la tabella 1.23 degli stati, dalla quale possiamo ricavare le mappe e le equazioni logiche.

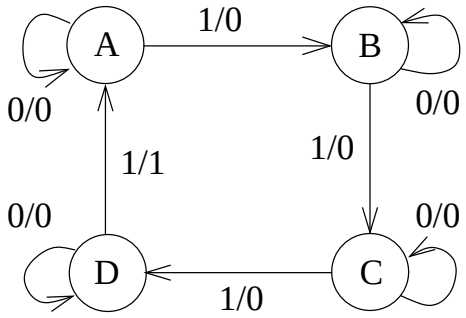

Figura 1.28: Diagramma degli stati di un contatore modulo 4

Tabella 1.21: Tabella degli stati di un contatore modulo 4

	$x = 0$	$x = 1$
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1

Le mappe sono riportate in figura 1.29.
Le equazioni logiche sono allora:

$$Y_1 = \overline{x}\, y_1 + y_1\, \overline{y}_2 + \overline{y}_1\, y_2\, x$$

$$Y_2 = \overline{x}\, y_2 + x\, \overline{y}_2$$

$$z = x\, y_1\, y_2$$

Il circuito è rappresentato in figura 1.30; per un miglior confronto con i risultati dei prossimi paragrafi, si è scelta la rappresentazione multifilare (omettendo comunque il clock).

Tabella 1.22: Primo assegnamento degli stati per il contatore modulo 4

y	$y_1\, y_2$
A	0 0
B	0 1
C	1 0
D	1 1

Tabella 1.23: Tabella degli stati del contatore dopo l'assegnamento

	$x = 0$	$x = 1$
$y_1\, y_2$	$Y_1\, Y_2,\, z$	$Y_1\, Y_2,\, z$
00	00, 0	01, 0
01	01, 0	10, 0
10	10, 0	11, 0
11	11, 0	00, 1

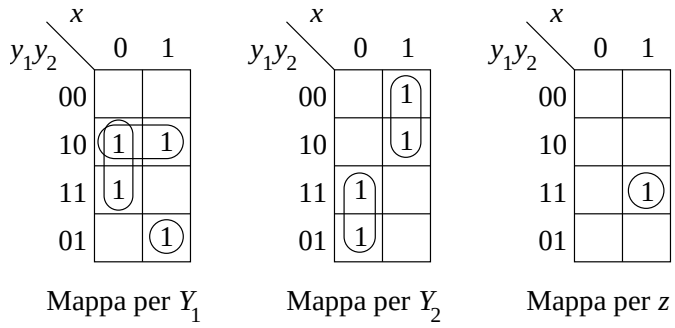

Figura 1.29: Mappe per il contatore modulo 4 con flip-flop di tipo D

1.4.2 Contatore modulo 4 con flip-flop di tipo T

Se vogliamo progettare il contatore con flip-flop di tipo T, dobbiamo costruirne la tabella di eccitazione 1.24 a partire dalla tabella degli stati 1.23, la quale si riferisce all'assegnamento di tabella 1.22.

Tabella 1.24: Tabella di eccitazione per il contatore con flip-flop T

	$x = 0$	$x = 1$
$y_1\, y_2$	$t_1\, t_2,\, z$	$t_1\, t_2,\, z$
00	00, 0	01, 0
01	00, 0	11, 0
10	00, 0	01, 0
11	00, 0	11, 1

Da essa possiamo ricavare le mappe, riportate in figura 1.31. Non si riporta la mappa per l'uscita z, che rimane identica a quella del paragrafo precedente.

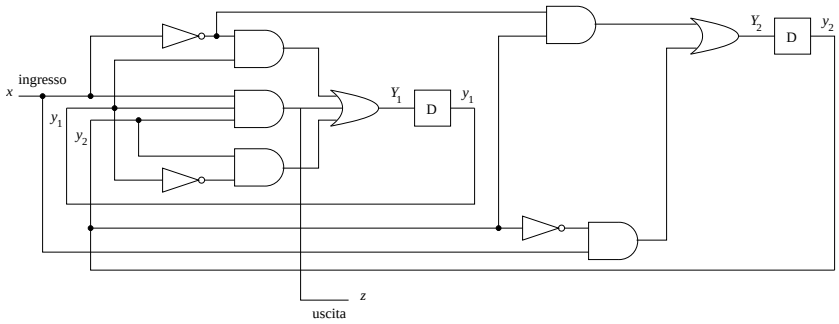

Figura 1.30: Contatore modulo 4 con flip-flop di tipo D

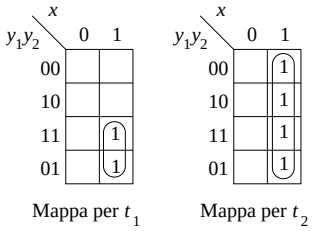

Figura 1.31: Mappe per il contatore modulo 4 con flip-flop di tipo T

Le equazioni sono allora le seguenti:

$$t_2 = x$$

$$t_1 = y_2\, x$$

$$z = y_1\, y_2\, x$$

Notiamo che, con l'uso di flip-flop di tipo T, le equazioni, e quindi il circuito (rappresentato in figura 1.32) risulta notevolmente semplificato.

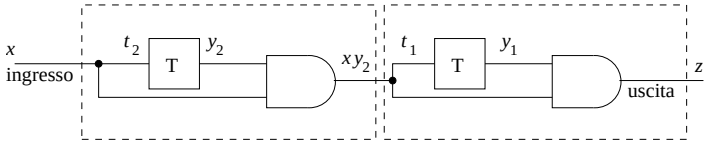

Figura 1.32: Contatore modulo 4 con flip-flop di tipo T

Notiamo altresì che questo circuito costituisce un esempio di *macchina iterativa* costituita da 2 celle identiche. Notiamo tuttavia che permane la

presenza dei flip-flop, esclusi invece nella trasformazione di un automa seriale sincrono in uno parallelo.

L'assegnamento scelto, consistente nel numerare gli stati nell'ordine da essi rappresentato nel conteggio (cioè A come primo stato, 00 in binario, B come secondo, cioè 01 in binario, eccetera) è quello che fornisce il miglior risultato, in termini di semplicità del circuito, con flip-flop di tipo T (ed anche con SR e con JK, come vedremo nei prossimi paragrafi).

Se si progetta con tale criterio un contatore modulo 8, si otterranno 3 celle identiche in cascata, ed allo stesso modo si avranno 4 celle per un contatore modulo 16. In generale, se il modulo m di conteggio è una potenza di due, $m = 2^n$, occorreranno n celle identiche, ciascuna costituita da un flip-flop di tipo T, SR o JK. Il numero di celle resta invariato cambiando l'assegnamento, tuttavia con altri assegnamenti le celle saranno in genere differenti l'una dall'altra.

Il lettore è invitato a provare ad esempio con l'assegnamento seguente:

$$\begin{aligned} \mathrm{A} &\leftrightarrow y_1 = 0\ y_2 = 0 \\ \mathrm{B} &\leftrightarrow y_1 = 0\ y_2 = 1 \\ \mathrm{C} &\leftrightarrow y_1 = 1\ y_2 = 1 \\ \mathrm{D} &\leftrightarrow y_1 = 1\ y_2 = 0 \end{aligned} \tag{1.10}$$

Si otterrà il circuito di figura 1.33, molto più complesso del precedente, le cui equazioni sono:

$$t_1 = \overline{y}_1\, t_2\, x + y_1\, \overline{y}_2\, x$$

$$t_2 = x\, y_1 + x\, \overline{y}_2$$

$$z = x\, y_1\, \overline{y}_2$$

Notiamo come il cambiare assegnamento comporti un cambiamento anche dell'equazione dell'uscita z.

Se il modulo di m di conteggio non è una potenza di 2, le celle saranno differenti una dall'altra. Se $q = 2^n$ è la prima potenza di 2 tale che $m \leq q$ occorreranno esattamente n celle per la realizzazione del contatore, ciascuna delle quali conterrà un flip-flop. Ciascuna cella si comporta come um contatore modulo 2, ovvero fornirà in uscita un 1 solo dopo che sono stati applicati in input due 1.

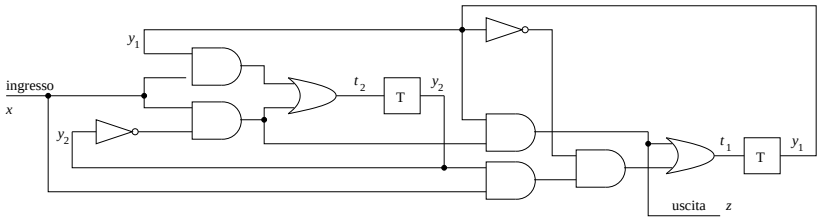

Figura 1.33: Contatore modulo 4 con flip-flop di tipo T con assegnamento (1.10)

Tabella 1.25: Tabella di eccitazione di un contatore modulo 4 con flip-flop SR

	$x = 0$	$x = 1$
$y_1\, y_2$	$s_1\, r_1,\ s_2\, r_2,\ z$	$s_1\, r_1,\ s_2\, r_2,\ z$
00	0X, 0X, 0	0X, 10, 0
01	0X, X0, 0	10, 01, 0
10	X0, 0X, 0	X0, 10, 0
11	X0, X0, 0	01, 01, 1

1.4.3 Progetto con flip-flop di tipo SR

Se consideriamo la tabella degli stati dopo l'assegnamento 1.23, ed utilizzando la tabella del flip-flop SR, è possibile ricavare la tabella di eccitazione 1.25.

Le mappe sono riportate in figura 1.34.

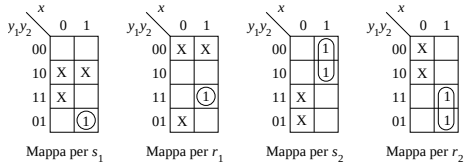

Figura 1.34: Mappe per il contatore con flip-flop SR

Le equazioni sono le seguenti:

$$s_1 = x\, \overline{y}_1\, y_2$$

$$r_1 = x\, y_1\, y_2$$

$$s_2 = x\, \overline{y}_2$$

$$r_2 = x\, y_2$$

$$z = y_1\, y_2\, x$$

Il circuito è rappresentato in figura 1.35. Come per il flip-flop di tipo T, l'assegnamento che conta in ordine gli stati del flip-flop da luogo a due celle identiche. Ed ancora, si dimostra che con n celle identiche si ottiene un contatore modulo 2^n.

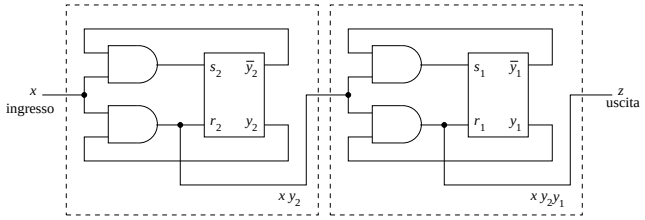

Figura 1.35: Contatore modulo 4 con flip-flop SR

1.4.4 Progetto con flip-flop di tipo JK

Utilizzando dei flip-flop di tipo JK, a partire dalla tabella degli stati 1.23 si ottiene la tabella di eccitazione 1.26

Tabella 1.26: Tabella di eccitazione di un contatore modulo 4 con flip-flop JK

	$x = 0$	$x = 1$
$y_1\, y_2$	$j_1\, k_1,\ j_2\, k_2,\ z$	$j_1\, k_1,\ j_2\, k_2,\ z$
00	0X, 0X, 0	0X, 1X, 0
01	0X, X0, 0	1X, X1, 0
10	X0, 0X, 0	X0, 1X, 0
11	X0, X0, 0	X1, X1, 1

Le mappe sono rappresentate in figura 1.36.
Le equazioni sono le seguenti:

$$j_1 = x\, y_2$$

$$k_1 = x\, y_2$$

$$j_2 = x$$

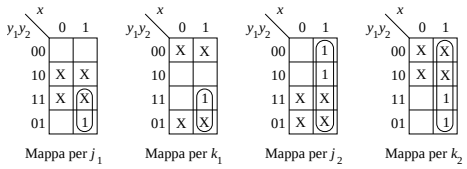

Figura 1.36: Mappe per il contatore con flip-flop JK

$$k_2 = x$$

Avendosi $j_1 = k_1$ e $j_2 = k_2$ il circuito risulta molto semplice, come si può vedere dalla figura 1.37. Valgono le stesse considerazioni del precedente paragrafo, con n celle identiche si ottiene un contatore modulo 2^n.

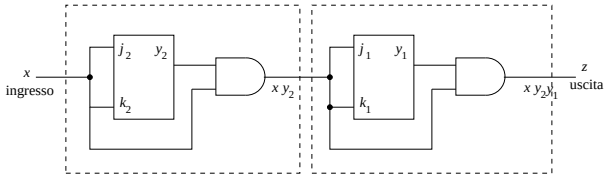

Figura 1.37: Contatore modulo 4 con flip-flop JK

Notiamo come il maggior numero di indifferenze nella tabella del JK permetta di ottenere un circuito più semplice rispetto all'uso di SR. Ciò è dovuto al fatto che per un JK è possibile ottenere la commutazione applicando agli ingressi J e K sia la combinazione 10 che la combinazione 11, mentre quest'ultima è vietata negli SR, dando luogo ad una corsa critica.

Esercizio

Progettare un contatore sincrono modulo 3 con flip-flop di tipo JK.

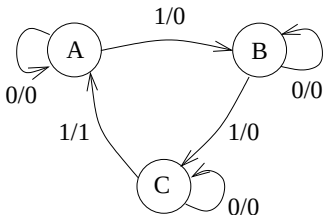

Figura 1.38: Diagramma degli stati per un contatore modulo 3

Soluzione

Si costruisce innanzitutto il diagramma degli stati (vedi figura 1.38), e da questo si ricava la tabella degli stati 1.27.

Tabella 1.27: Tabella degli stati di un contatore modulo 3

	$x=0$	$x=1$
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	A, 1

Scegliamo l'assegnamento seguente (il lettore è invitato a provare altri assegnamenti):

$$A \leftrightarrow y_1 = 0\ y_2 = 0$$
$$B \leftrightarrow y_1 = 0\ y_2 = 0$$
$$C \leftrightarrow y_1 = 1\ y_2 = 0$$

Tabella 1.28: Tabella degli stati del contatore modulo 3 dopo l'assegnamento

	$x=0$	$x=1$
$y_1\,y_2$	$Y_1\,Y_2,\ z$	$Y_1\,Y_2,\ z$
00	00, 0	01, 0
01	01, 0	10, 0
10	10, 0	00, 1

Si ottiene allora la tabella 1.28, dalla quale si ricava la tabella di eccitazione 1.29.

Le mappe sono rappresentate in figura 1.39.

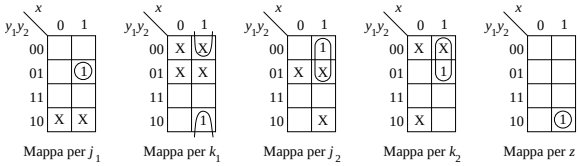

Figura 1.39: Mappe per il contatore modulo 3

Tabella 1.29: Tabella di eccitazione di un contatore modulo 3 con flip-flop JK

	$x = 0$	$x = 1$
$y_1\, y_2$	$j_1\, k_1,\ j_2\, k_2,\ z$	$j_1\, k_1,\ j_2\, k_2,\ z$
00	0X, 0X, 0	0X, 1X, 0
01	0X, X0, 0	1X, X1, 0
10	X0, 0X, 0	X1, X0, 1

Da esse si ricavano le seguenti equazioni:

$$j_1 = x\, \overline{y}_1\, y_2$$
$$k_1 = \overline{y}_2\, x$$
$$j_2 = x\, \overline{y}_1$$
$$k_2 = x\, \overline{y}_1$$
$$z = x\, \overline{y}_1\, y_2$$

Notiamo che si ha $k_2 = j_2$ come nel contatore modulo 4, ma non più $k_1 = j_1$. Il circuito è rappresentato in figura 1.40; notiamo come le due celle non siamo più identiche, essendo il modulo di conteggio diverso da una potenza di due.

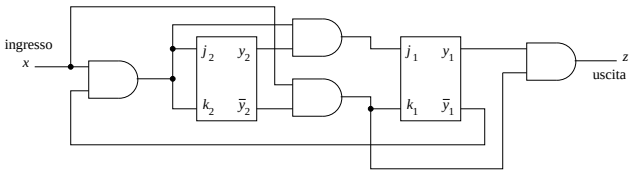

Figura 1.40: Contatore modulo 3

Capitolo 2

Riduzione del numero degli stati

2.1 Introduzione

Consideriamo il diagramma degli stati, rappresentato in figura 2.1.

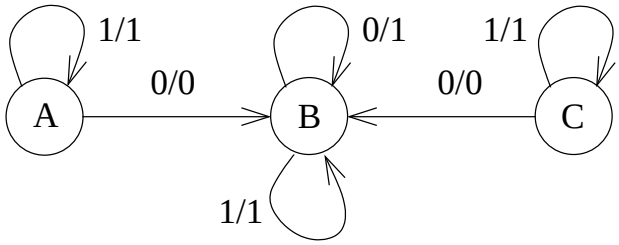

Figura 2.1: Automa con stati ridondanti

Come possiamo osservare, il comportamento degli stati A e C é identico; entrambi hanno come successori (con ingresso 0) lo stato B e (con ingresso 1) se stessi. Inoltre le loro uscite relative ad ingresso zero coincidono, cosí come quelle con ingresso uno. Possiamo pertanto dire che gli stati A e C sono *indistinguibili*.

Piú precisamente, possiamo dare la seguente

Definizione 2.1 *Due stati di un automa sono indistinguibili (o ridondanti), se dall'osservazione dei soli ingressi e delle sole uscite non é possibile capire in quale stato ci si trova.*

Quando un automa contiene stati indistinguibili é detto in *forma non minima*; in caso contrario esso é detto in *forma minima*.

Consideriamo ora l'automa di figura 2.2; da una osservazione superficiale potremmo concludere che gli stati A e D sono indistinguibili, avendo le stesse uscite (zero con ingresso zero, ed uno con ingresso uno).

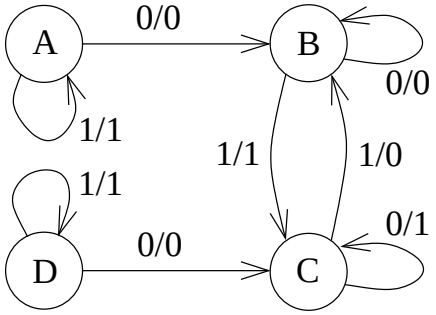

Figura 2.2: Automa con stati non ridondanti

Tuttavia, i successori degli stati A e D sono differenti; essi sono rispettivamente B e C. Ora, il comportamento ingresso/uscita degli stati B e C é differente; pertanto é possibile riconoscere se ci troviamo nello stato A, oppure nello stato D.

Infatti, se ci troviamo nello stato A ed applichiamo in ingresso la seguenza 00, otteniamo in uscita la sequenza 00 (passando per lo stato B e rimanendo in esso), mentre se applichiamo la stessa sequenza di ingresso partendo dallo stato D, abbiamo in uscita la sequenza binaria 01.

Di conseguenza, con una sequenza di due cifre binarie, é possibile distinguere gli stati A e D, osservando il differente comportamento delle uscite.

Pertanto, due stati sono indistinguibili non se il loro comportamento ingresso/uscita é identico, ma anche se i loro successori ottenuti applicando sequenze di ingresso sufficientemente lunghe, sono indistinguibili.

Possiamo pertanto dare la seguente

Definizione 2.2 *Due stati sono k-indistinguibili se l'applicazione di ogni sequenza di ingresso di lunghezza k non consente di distinguerli, osservando solo la sequenza di uscita.*

Ad esempio, gli stati A e D dell'automa di figura 2.2 sono 2-distinguibili (la sequenza di ingresso 00 permette di distinguerli) ma sono 1-indistinguibili (l'applicazione di un solo 1 o di un solo 0 in ingresso non permette di distinguerli).

2.2 Un algoritmo di minimizzazione

Vediamo ora un semplice algoritmo di minimizzazione, che verrá illustrato per mezzo di un esempio.

Consideriamo, innanzitutto, l'automa descritto dalla tabella degli stati 2.1

Tabella 2.1: Tabella dell'automa da rendere in forma minima

	$x = 0$	$x = 1$
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

ove, come al solito, nella prima colonna viene indicato lo stato attuale, e nelle altre sono indicati gli stati successivi, e le relative uscite (colonne distinte corrispondono a differenti valori della variabile di ingresso x).

Per prima cosa, si elencano tutti gli stati dell'automa, e si scrive sotto di essi l'elenco dei loro successori, sia per $x = 0$ che per $x = 1$. Tale operazione é illustrata nella tabella 2.2.

Tabella 2.2: Tabella dei successori degli stati

stati	A	B	C	D	E	F
successori per $x = 0$	E	F	E	F	C	B
successori per $x = 1$	D	D	B	B	F	C

Successivamente, si cercano gli stati 1-indistinguibili; essi si possono riconoscere perché hanno, a pari ingressi, le stesse uscite.

Dall'osservazione della sole uscite nella tabella degli stati 2.1 é allora possbile separare gli stati A, C, E dagli stati B, D, F; tale operazione é riportata nella seconda riga della tabella 2.3 (la prima riga é solo un elenco di tutti gli stati dell'automa).

Osserviamo ora, che per $x = 1$ i successori di B e D appartengono al

Tabella 2.3: Separazione degli stati in blocchi indistinguibili

(A, B, C, D, E, F)
(A, C, E) (B, D, F)
(A, C, E) (B, D) (F)
(A, C) (E) (B, D) (F)
(A, C) (E) (B, D) (F)

blocco di stati (B, D, F), mentre il successore dello stato F é lo stato C, che appartiene al blocco (A, C, E)[1].

Ma dato che gli stati A, C, E hanno uscite differenti dagli stati B, D, F, segue che é possibile distinguere gli stati (B, D) dallo stato F, applicando in ingresso una sequenza opportuna di lunghezza 2.

Possiamo pertanto separare gli stati (B, D) dallo stato F, come appare nella terza riga della tabella 2.3.

Procediamo ancora; osserviamo ora che i successori (per ingresso uno) degli stati A e C appartengono al blocco (B, D), mentre il successore di E é lo stato F, che appartiene ad un altro blocco (come unico elemento).

Ció ci permette di separare la coppia di stati (A, C) dallo stato E, come appare nella quarta riga della tabella 2.3.

Infine, non potendo fare ulteriori separazioni all'interno dei blocchi, segue che gli stati A e C sono tra loro indistinguibili, cosí come lo sono gli stati B e D.

Possiamo pertanto cancellare dalla tabella degli stati 2.1 uno stato a scelta tra A e C (ad esempio C, sostituendolo con lo stato A) ed uno stato a scelta tra B e D (ad esempio D, sostituendolo con lo stato B).

In questo modo otteniamo un automa in forma minima, con il minor numero possibile di stati, che si comporta, dal punto di vista delle proprietá ingresso–uscita, come l'automa di tabella 2.1.

La tabella degli stati dell'automa in forma minima é riportata in tabella 2.4

La riduzione di un automa in forma minima permette di ottenere un circuito piú semplice, in quanto la riduzione degli stati riduce, ad esempio, il numero di cifre binarie necessarie per l'assegnamento (nell'automa in forma non minima, per distinguere i sei stati occorrono tre cifre binarie; in quello in forma minima per distinguere quattro stati bastano due cifre binarie).

[1]In questo esempio gli stati sono indistinguibili per ingresso $x = 0$; in altri casi la loro separazione puó essere fatta anche osservando i successori con ingresso $x = 0$

Tabella 2.4: Automa in forma minima

	$x = 0$	$x = 1$
A	E, 0	B, 1
B	F, 0	B, 0
E	A, 0	F, 1
F	B, 0	A, 0

2.2.1 Rappresentazione canonica di un automa

Ritornando all'automa rappresentato in tabella 2.4, possiamo osservare che la denominazione degli stati é arbitraria; ad esempio potevamo cancellare gli stati A e B anziché gli stati C e B.

Per questo motivo é desiderabile rappresentare l'automa ridotto con una particolare denominazione degli stati; tale rappresentazione é detta *forma standard* o *forma canonica* dell'automa.

Per rappresentare un automa in forma canonica, dopo la sua riduzione, si parte da uno stato (preferibilmente lo stato iniziale), e lo si indica con la prima lettera dell'alfabeto greco, la α.

Gli altri stati sono rinominati in modo che, leggendo in successione le righe della tabella degli stati, ciascuna riga da sinistra a destra, si trovano in ordine gli stati β (seconda lettera dell'alfabeto greco), γ (terza lettera), e cosí via (cioé gli stati appaiono man mano nella tabella in ordine alfabetico).

Ad esempio, per quanto riguarda l'automa ridotto rappresentato in tabella 2.4, si ha la ridenominazione degli stati rappresentata in tabella 2.5:

Tabella 2.5: Ridenominazione degli stati per portare l'automa in forma normale

	$x = 0$	$x = 1$
$A = \alpha$	$E = \beta$, 0	$B = \gamma$, 1
$B = \gamma$	$F = \delta$, 0	$B = \gamma$, 0
$E = \beta$	$A = \alpha$, 0	$F = \delta$, 1
$F = \delta$	$B = \gamma$, 0	$A = \alpha$, 0

La tabella degli stati dell'automa in forma standard é pertanto rappresentata in tabella 2.6

Tabella 2.6: Ridenominazione degli stati per portare l'automa in forma normale

	$x = 0$	$x = 1$
α	β, 0	γ, 1
β	α, 0	δ, 1
γ	δ, 0	γ, 0
δ	γ, 0	α, 0

2.3 Macchine incompletamente specificate

Puó capitare, che in alcuni automi, non sia sempre necessario specificare, nella tabella degli stati, tutti i successori degli stati e/o tutte le uscite. In questo caso, si dice che l'automa *non é completamente specificato*.

2.3.1 Completamento delle transizioni

Consideriamo l'automa descritto dalla tabella degli stati 2.7; in essa possiamo osservare come l'uscita e lo stato successivo non siano specificati nella prima riga, nel caso in cui $x = 1$, mentre quando ci si trova nello stato B, non é specificato lo stato successivo, nel caso in cui si abbia in ingresso $x = 0$.

Tabella 2.7: Esempio di automa incompletamente specificato

	$x = 0$	$x = 1$
A	B, 1	–, –
B	–, 0	C, 0
C	A, 1	B, 0

In questo caso, possiamo specificare tutte le transizioni verso gli stati successivi introducendo un nuovo stato T (che rappresenterá lo stato terminale dell'automa) al posto di tutti i trattini che sono messi ove lo stato successivo non é specificato; invece possiamo sfruttare la non specificazione dell'uscita come una condizione di indifferenza; essa potrá essere utile nella semplificazione dell'equazione booleana della variabile di uscita.

La nuova tabella degli stati é rappresentata in tabella 2.8

Si noti che le uscite non sono specificate quando ci si trova nello stato terminale T; pertanto verranno indicate con il simbolo di indifferenza (la X).

Tabella 2.8: Introduzione dello stato terminale T

	$x = 0$	$x = 1$
A	B, 1	T, X
B	T, 0	C, 0
C	A, 1	B, 0
T	T, X	T, X

2.3.2 La non unicitá della riduzione degli stati

Vediamo ora, come partendo da un automa non completamente specificato, sia possibile effettuare due riduzioni, differenti tra loro per quanto riguarda il numero degli stati.

Consideriamo, a tal scopo, l'automa rappresentato dalla tabella 2.9, in cui, in due casi, non sono specificate le uscite.

Tabella 2.9: Automa incompletamente specificato per le uscite

	$x = 0$	$x = 1$
A	C, 1	E, –
B	C, –	E, 1
C	B, 0	A, 1
D	D, 0	E, 1
E	D, 1	A, 0

Proviamo a sostituire, al posto di entrambi le indifferenze, un uno. Otteniamo allora la tabella 2.10:

Tabella 2.10: Primo assegnamento delle uscite indeterminate

	$x = 0$	$x = 1$
A	C, 1	E, 1
B	C, 1	E, 1
C	B, 0	A, 1
D	D, 0	E, 1
E	D, 1	A, 0

Scriviamo la tabella dei successori degli stati (tabella 2.11) e mediante

essa eseguiamo il processo di divisione degli stati in blocchi indistinguibili, come giá esposto nel paragrafo 2.2.

Tabella 2.11: Successori degli stati

stati	A	B	C	D	E
successori per $x = 0$	C	C	B	D	D
successori per $x = 1$	E	E	A	E	A

Otteniamo allora la suddivisione rappresentata in tabella 2.12, da cui vediamo che gli stati A e B sono tra loro equivalenti.

Tabella 2.12: Separazione degli stati in blocchi

(A, B, C, D, E)
(A, B) (C, D) (E)
(A, B) (C) (D) (E)
(A, B) (C) (D) (E)

É allora possibile eliminare lo stato B, ottenendo l'automa descritto (in forma non standard) in tabella 2.13.

Tabella 2.13: Automa ridotto - prima possibilitá

	$x = 0$	$x = 1$
A	C, 1	E, 1
C	A, 0	A, 1
D	D, 0	E, 1
E	D, 1	A, 0

Proviamo ora a sostituire le due indifferenze presenti nella tabella 2.9 con due zeri; si ottiene allora l'automa della tabella 2.14:

La suddivisione in blocchi di stati indistinguibili é ora rappresentata in tabella 2.15; da essa notiamo che possiamo considerare indistinguibili gli stati A, E e gli stati B, C, D.

Cancellando lo stato E (sostituito dallo stato A) e cancellando gli stati C e D (sostituiti dallo stato B abbiamo pertanto ridotto l'automa ad una macchina con solo due stati, come da tabella 2.16.

Tabella 2.14: Secondo assegnamento delle uscite indeterminate

	$x = 0$	$x = 1$
A	C, 1	E, 0
B	C, 0	E, 1
C	B, 0	A, 1
D	D, 0	E, 1
E	D, 1	A, 0

Tabella 2.15: Separazione degli stati in blocchi per il secondo assegnamento

(A, B, C, D, E)
(A, E) (B, C, D)
(A, E) (B, C, D)

Concludiamo il paragrafo con due osservazioni:

- La riduzione non é univoca; a differenti specificazioni delle uscite corrispondono differenti automi in forma minima.
- Tra le possibili semplificazioni, alcune portano ad una considerevole riduzione del numero di stati, altre a semplificazioni meno riuscite, ed altre a nessuna semplificazione.

2.3.3 Osservazione importante

Entrambi le riduzioni ottenute sono perfettamente equivalenti, nel senso che *ove le uscite dell'automa incompletamente specificato* sono note, gli automi ridotti si comportano nella stessa maniera per quanto riguarda il comportamento ingresso/uscita dell'automa originario; quando invece si trova (nell'automa incompletamente specificato) una condizione di indifferenza nell'uscita,

Tabella 2.16: Automa ridotto in forma minima - seconda possibilitá

	$x = 0$	$x = 1$
A	B, 1	A, 0
B	B, 0	A, 1

allora le uscite degli automi in forma minima possono anche differire tra loro, ma ció non ha alcuna importanza, dato che nell'automa originario non era richiesta alcuna specifica sulle uscite.

2.3.4 L'introduzione provvisoria di nuovi stati come metodo di riduzione

Vediamo ora un'altra tecnica, che permetterá di ottenere una riduzione del numero degli stati, dopo averne introdotto provvisoriamente degli altri, sdoppiando uno o piú stati esistenti.

Vedremo che questa tecnica porterá a differenti possibilitá di semplificazione, e che anche a paritá di numero di stati due automi in forma ridotta possono differire nella tabella delle transizioni.

Ancora una volta, le soluzioni ottenute sono tutte valide, nel senso indicato dall'osservazione del paragrafo 2.3.3.

Consideriamo l'automa descritto nella tabella 2.17.

Tabella 2.17: Tabella degli stati di un automa incompletamente specificato

	$x = 0$	$x = 1$
A	A, 0	C, 0
B	B, 0	B, –
C	B, 0	A, 1

Se al posto dell'indifferenza nella seconda riga, inseriamo un 1, possiamo tentare di raggruppare gli stati B e C in un unico blocco di stati equivalenti, ovvero di effettuare la prima suddivisione come (A) (B, C). Tuttavia, con $x = 1$, i successori di B e C appartengono a gruppi diversi, e quindi la suddivisione finale risulta essere (A) (B) (C). Non é allora possibile semplificare l'automa.

Una analoga considerazione si ha sostituendo il valore 0 al posto dell'indifferenza presente nella seconda riga. La prima suddivisione in blocchi risulta essere (A, B) (C), ma osservando il successori di A e di B per $x = 1$ si giunge alla suddivisione definitiva (A) (B) (C), che vanifica la possibilitá di riduzione del numero degli stati.

Tuttavia, esiste un metodo alternativo, consistente nello sdoppiare lo stato B in due stati, B_1 e B_2.

In questo modo, un abbozzo della nuova tabella degli stati é rappresentato in tabella 2.18; qui al posto del simbolo B_X possiamo mettere, indifferentemente B_1 oppure B_2.

Tabella 2.18: Sdoppiamento dello stato B

	$x=0$	$x=1$
A	A, 0	C, 0
B_1	B_X, 0	B_X, $--$
B_2	B_X, 0	B_X, $--$
C	B_X, 0	A, 1

Se pensiamo infatti agli stati B_1 e B_2 come stati gemelli, risulta del tutto indifferente avere come stato successore di un arbitrario stato uno o l'altro dei gemelli.

Analogamente, al posto delle indifferenze nelle uscite, possiamo utilizzare arbitrariamente il valore 0 o il valore 1.

Tabella 2.19: Prima scelta nelle transizioni verso gli stati gemelli

	$x=0$	$x=1$
A	A, 0	C, 0
B_1	B_1, 0	$B_2, 0$
B_2	B_1, 0	$B_1, 1$
C	B_1, 0	A, 1

Ad esempio, possiamo costruire l'automa descritto dalla tabella 2.19; una riduzione degli stati porta in questo caso ai blocchi (A, B_1) e (B_2, C). Cancellando gli stati B_1 e B_2 si ottiene l'automa in forma minima della tabella 2.20

Tabella 2.20: Automa ottenuto con la prima scelta

	$x=0$	$x=1$
A	A, 0	C, 0
C	A, 0	A, 1

Se invece scegliamo la possibilitá offerta dalla tabella 2.21, otteniamo ancora i blocchi di stati equivalenti (A, B_1) e (B_2, C).

Cancellando gli stati B_1 e B_2 si ottiene l'automa in forma minima della tabella 2.22, che differisce dal precedente per quanto riguarda la transizione

Tabella 2.21: Seconda scelta nelle transizioni verso gli stati gemelli

	$x = 0$	$x = 1$
A	A, 0	C, 0
B_1	B_1, 0	B_2, 0
B_2	B_2, 0	B_1, 1
C	B_2, 0	A, 1

(per $x = 0$) dallo stato C allo stato successivo (che ora é C stesso, anziché A).

Tabella 2.22: Automa ottenuto con la seconda scelta

	$x = 0$	$x = 1$
A	A, 0	C, 0
C	C, 0	A, 1

Pertanto, abbiamo ottenuto due automi in forma minima, entrambi con due soli stati, ma con transizioni differenti dallo stato C. Entrambi gli automi sono comunque, per quanto detto nel paragrafo 2.3.3, delle corrette minimizzazioni dell'automa originario.

2.4 Un metodo generale

Analizziamo ora una procedura di uso generale per la minimizzazione del numero di stati degli automi incompletamente specificati; osserviamo a tal proposito che le tecniche illustrate nei precedenti paragrafi sono difficilmente applicabili per automi con numerosi stati; infatti tali tecniche si basano essenzialmente su uno o piú tentativi di sostituzione delle indifferenze e delle transizioni mancanti, o sullo sdoppiamento di alcuni stati. Tali tecniche sono inapplicabili per un automa contenente un elevato numero di stati o di uscite non specificate, richiedendo di effettuare un numero elevato di tentativi.

Vi é pertanto la necessitá di utilizzare un metodo generale di minimizzazione; alla base di tale metodo vi sono i concetti di *grafo di fusione* e di *grafo di compatibilitá*, che verranno illustrati nei paragrafi seguenti.

2.4.1 Il grafo di fusione

Consideriamo ora la tebella degli stati 2.23, relativa ad un automa incompletamente specificato. Dato il gran numero di dati mancanti nella tabella (stati successori e uscite), risulta evidente l'impossibilitá di adottare i metodi illustrati nei paragrafi precedenti al fine di minimizzarne il numero di stati.

Tabella 2.23: Automa incompletamente specificato con gran numero di dati mancanti

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 10$	$x_1x_2 = 11$
A	–,–	C, 0	E, 0	B, 0
B	E, 1	–,–	–,–	–,–
C	F, 1	F, 0	–,–	–.–
D	–,–	–,–	B, 0	–,–
E	–,–	F, 1	A, 1	D, 0
F	C, 1	–,–	B, 1	C, 0

A partire dalla tabella possiamo costruire il *grafo di fusione*, rappresentato in figura 2.3, e cos costruito:

- Si prendono tanti vertici (A, B, C, ...) quanti sono gli stati.
- Si esamina una coppia di stati per volta: prima la coppia AB, poi AC, ..., poi AF, poi BC, sino ad EF.
- Se per quella data coppia di stati, gli stati successori e le uscite, quando specificati sono identiche, si collegano i sue stati con una linea. Nel nostro esempio, ció accade per gli stati A e B, ma non per B e C (i loro successori nella prima colonna della tabella sono differenti tra loro, sono infatti gli stati E (per B) ed F (per C)).
- Se le uscite, quando specificate, sono identiche, ma gli stati successori sono differenti, la coppia di stati viene collegata da una linea a fianco della quale sono indicate le coppie di successori che differiscono tra loro (sempre nel nostro esempio, a fianco della linea che collega B e C si scrive la coppia (E, F)). Non é necessario indicare la coppia dei successori differenti se essi coincidono con i vertici della linea stessa.
- Se le uscite sono differenti (vedi ad esempio stati C ed E, che hanno uscite differenti nella seconda colonna della tabella), gli stati non devono essere collegati tra loro.

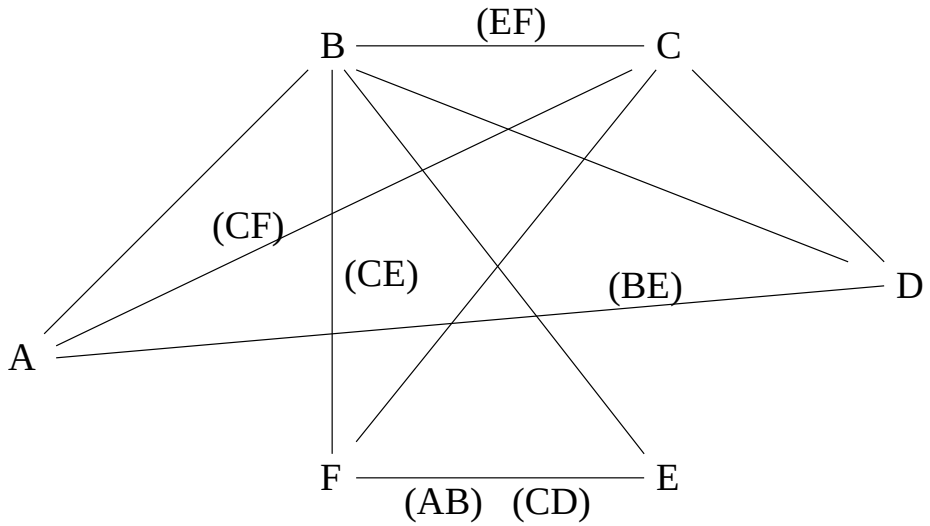

Figura 2.3: Grafo di fusione

Consideriamo ora tutte le linee del grafo etichettate con coppie di stati incompatibili; ad esempio tra gli stati B e F vi é una linea a fianco dei quali é indicata una coppia (C, E). Occorre adesso guardare se questi ultimi due stati sono collegati da una linea; nel caso che non lo siano, gli stati C, E successori rispettivamente di B, F hanno uscite differenti e pertanto sono distinguibili tra loro. In questo caso sono distinguibili anche B ed F, ed occorre pertanto eliminare la linea che li collega. Invece, non si elimina la linea che collega tra loro gli stati A e C perché pur avendo successori distinti (C ed F), questi ultimi sono collegati da una linea. Per lo stesso motivo, non si elimina la linea che collega gli stati B e C (I loro successori distinti, E ed F sono collegati tra di loro) Si ottiene pertanto il grafo di figura 2.4, ove al solo scopo didattico é stata evidenziata (con una crocetta, e ridisegnandola tratteggiata) la linea eliminata.

2.4.2 L'insieme degli stati compatibili

Tutte le coppie di stati collegati da una linea nel grafo di fusione di figura 2.4 sono dette *coppie compatibili*; nel nostro caso le coppie di stati compatibili sono le seguenti: U= {(A,B), (A,C), (A,D), (B,C), (B,D), (B,E), (C,D), (C,F), (E,F)}. Vale inoltre la propriet transitiva: cos avendosi A compatibile con B, A compatibile con C e B compatibile con C si pu dire che la terna (A, B, C) rappresenta tre stati compatibili tra loro. Osserviamo che gli stati A,

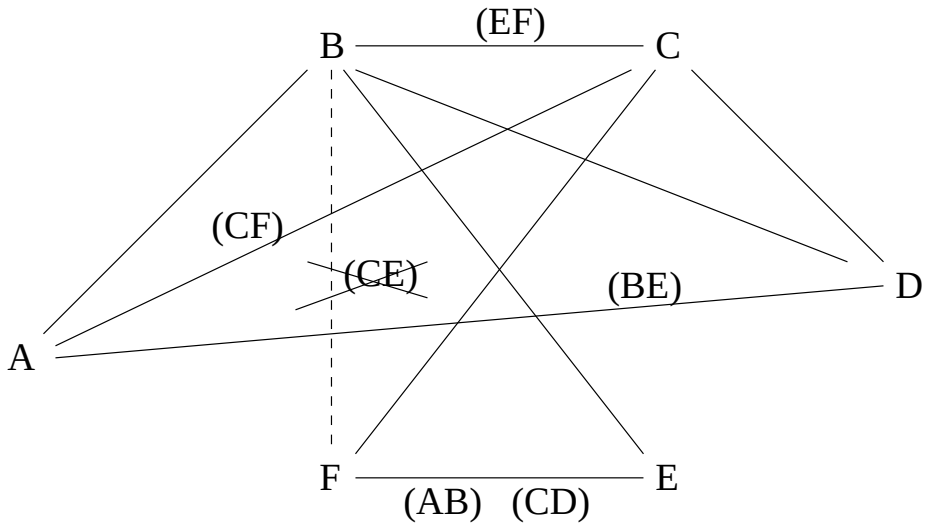

Figura 2.4: Coppie di stati compatibili ottenuti dal grafo di fusione

B, C sono collegati da un poligono completo[2]. Per cercare di raggruppare il piú possibile stati tra loro compatibili, basta cercare tutti i poligoni completi nel grafo di fusione ottenuto dopo le operazioni di cancellazione esposti in precedenza. Ad esempio, dalla figura 2.4 si nota che (A, B, C, D) sono vertici di un poligono completo, e che inoltre (B, E), (C, F), (E,F) sono coppie di stati compatibili.

La collezione di insiemi di stati compatibili cosí ottenuta é detta *insieme massimo di compatibilitá*.

Nel nostro caso, tale insieme é: S = {(A, B, C, D), (B, E), (C, F), (E, F)}.

Se prendiamo il sottoinsieme T di S seguente: T = { (A, B, C, D), (E, F)} notiamo che esso comprende tutti gli stati dell'automa; l'insieme T é inoltre composto di due soli elementi. Ogni altro sottoinsieme di S che contenga tutti gli stati dell'automa possiede piú di due elementi. Pertanto T é detto *insieme minimo di compatibilitá*.

Questo insieme é importante, perché ci dice quale é il limite inferiore del numero di stati ottenibili durante il proseguimento di minimizzazione dell'automa. Nel nostro caso, dopo una corretta minimizzazione, dobbiamo ottenere un automa con numero di stati maggiore o uguale a 2 (la dimensione di T)

[2]Un poligono é detto completo se sono rappresentati sia i suoi lati che tutte le sue diagonali

2.4.3 L'insieme chiuso di compatibilitá

Cerchiamo innanzitutto di ottenere un automa con 2 stati; vogliamo vedere se con due soli stati: α = (A, B, C, D) e β = (E, F) possiamo realizzare un automa minimo equivalente a quello originario; per far ció prendiamo l'insieme T = {(A, B, C, D), (E, F)}, e vediamo quali sono i distinti successori della quaterna (A, B, C, D): nella tabella degli stati 2.23 o nel grafo di fusione vediamo che gli stati B, C hanno come successori gli stati E, F (la cui coppia é un elemento di T. Invece, i successori distinti di A, C sono C, F, e la coppia (C, F) non appartiene a T. Analogamente, i successori distinti di A e D sono E e B, e la coppia (B, E) non appartiene a T. Se troviamo, come in questo esempio, coppie di successori che non appartengono a T, non possiamo prendere i raggruppamenti di T come stati dell'automa minimo.

Ci serve, per proseguire, la seguente

Definizione 2.3 *Un insieme V di stati compatibili é detto chiuso, se per ogni coppia di elementi appartenente a ciascun raggruppamento di V, le corrispondenti coppie di successori distinti appartengono a V.*

Ad esempio, l'insieme W={ (A, D), (B, E), (C, D) } é chiuso. Infatti i successori distinti di (A, D), che sono (B, E), appartengono a V. Inoltre (B, E) e (C, D) non hanno successori distinti, come si puó vedere dal grafo di fusione o dalla tabella degli stati.

Tuttavia, l'insieme W non é un candidato per la costruzione dell'automa minimo, perché non contiene tutti gli stati dell'automa (manca lo stato F).

Ci serve, ora, un'altra

Definizione 2.4 *Un insieme chiuso di stati compatibili V, contenente tutti gli stati dell'automa, é detto un rivestimento (o ricoprimento) chiuso dell'automa.*

Ad esempio, l'insieme Q = { (A, B), (C, D), (E, F) } é un ricoprimento chiuso.

Trovato un ricoprimento chiuso, é possibile costruire l'automa in forma minima; nel nostro caso siano a=(A, B), b=(C, D), c=(E, F) i nuovi stati. Si ottiene cosí un automa a tre stati, rappresentato dalla tabella 2.24, (in forma non standard).

Osserviamo che, in generale, un automa puó avere piú di un ricoprimento chiuso, con lo stesso numero di elementi; e ció conferma che la riduzione in forma minima di un automa non é unica, come avevamo notato giá in precedenza.

Tabella 2.24: Automa incompletamente specificato con gran numero di dati mancanti

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 10$	$x_1x_2 = 11$
a	c, 1	b, 0	c, 0	a, 0
b	c, 1	c, 0	a, 0	–,–
c	b, 1	c, 1	a, 1	b, 0

2.4.4 Il grafo di compatibilitá

Il grafo di fusione é comodo, e semplice da utilizzare; tuttavia la fase della determinazione del ricoprimento chiuso é ancora svolta per via empirica, a tentativi. Introduciamo ora un nuovo procedimento, che fa uso del cosiddetto textitgrafo di compatibilitá, in modo da semplificare la ricerca del ricoprimento chiuso dell'automa.

Consideriamo, a tal scopo, la tabella degli stati 2.25 di un altro automa incompletamente specificato:

Tabella 2.25: Automa incompletamente specificato con gran numero di dati mancanti

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
A	–,–	–,–	E, 0	–,–
B	C, 1	A, 0	B, 1	–,–
C	C, 1	D, 0	–,–	A, 1
D	–,–	E, 0	B, –	–,–
E	B, 1	–, –	C, –	B, 1

Costruiamo il suo grafo di fusione (figura 2.5).

E semplifichiamolo, come in figura 2.6.

Il *grafo di compatibilitá* si ottiene prendendo come vertici tutte le coppie collegate da una linea nel grafo di fusione; nel nostro caso queste coppie sono U = { (A,C), (A,D), (B,C), (B,E), (C,D),(D,E) }. Per ciascuna coppia di elementi di U, si guarda il grafo di fusione; se sulla linea congiungente questa coppia sono indicate una o piú coppie di elementi successori, occorre congiungere con una freccia la coppia di elementi di U (punto partenza) con la coppia (o le coppie) di successori distinti (punto di arrivo).

Il grafo di compatibilitá relativo all'automa in esame é riportato in figura 2.7; in esso osserviamo ad esempio che la coppia (A,C) non é collegata a

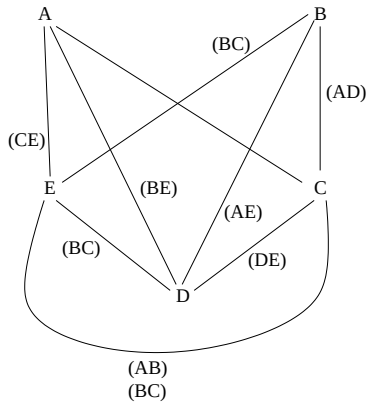

Figura 2.5: Grafo di fusione

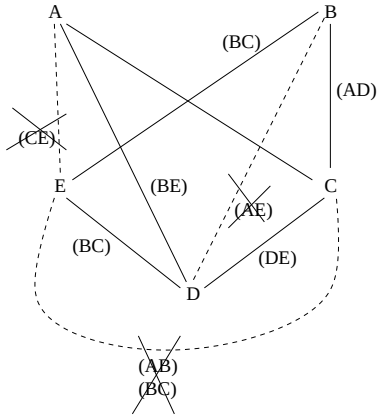

Figura 2.6: Grafo di fusione dopo la semplificazione

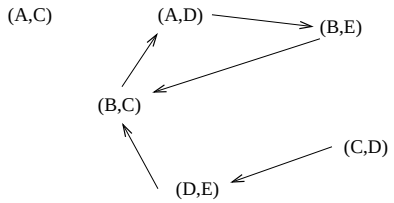

Figura 2.7: Grafo di compatibilitá

nessun'altra, in quanto nel grafo di fusione di figura 2.6 i vertici A, C sono collegati da una linea priva di indicazioni di coppie di successori. Invece la coppia (A,D) é collegata alla coppia (B,E) perché nel grafo di fusione, accanto alla linea che collega i vertici A e D é segnata la coppia di successori distinti (B,E).

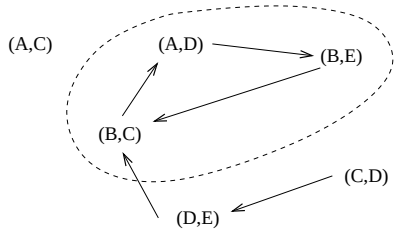

Figura 2.8: Grafo di compatibilitá e suo sottografo chiuso

Si tratta ora di cercare, all'interno del grafo di compatibilitá un sottografo chiuso; esso é una parte del grafo di compatibilitá dal quale non fuoriescono linee. Ad esempio, la parte tratteggiata di figura 2.8 rappresenta un sottografo chiuso (non ci sono linee che escono dall'area tratteggiata.

Il sottografo chiuso cosí trovato contiene i vertici W= { (A,D), (B,C), (B,E) }. Il lettore puó verificare che esistono altri sottografi chiusi, ad esempio come l'intero grafo stesso, come U={ (A,C) }, come Z={ (C,D), (D,E), (B,C), (A,D), (B,E) }. Ma, ad esempio, l'insieme V={ (A,D), (B,E) } non é un sottografo chiuso, perché esiste una linea che esce da (B,E) (verso (B,C)).

Una osservazione: se nel grafo di compatibilitá non sono elencati tutti gli stati dell'automa, occorre aggiungerli come stati a se stanti. Se per esempio non fosse stato elencato lo stato E, occorreva aggiungerlo come una coppia (E,E) non collegata a nessun altra coppia.

La cosa importante é che il grafo di compatibilitá forma un ricoprimento chiuso. Tuttavia é spesso desiderabile trovare dei ricoprimenti chiusi con un numero minimo di elementi. E per questo motivo, ogni sottografo chiuso contenente tutti gli stati della macchina é un ricoprimento chiuso. Se poi un sottografo chiuso contiene piú coppie di stati, essi possono essere raggruppati assieme secondo la proprietá transitiva: ad esempio, se esso contiene le coppie (A,B), (A,C), (B,C) é possibile raggruppare tali coppie in un unico stato rappresentante la terna (A, B, C). (Se tuttavia contiene, ad esempio, solo le coppie (A,B) e (B,C) ma non la coppia (A,C) ció non é possibile).

Nel nostro caso, il sottografo chiuso W= { (A,D), (B,C), (B,E) } contiene tutti gli stati dell'automa, e pertanto costituisce un ricoprimento chiuso.

Possiamo allora semplificare l'automa, riducendolo ad un automa in forma

minima di tre soli stati: a=(A,D), b=(B,C) e c=(B,E). L'automa in forma minima, non standard, é rappresentato in tabella 2.26.

Tabella 2.26: Riduzione dell'automa incompletamente specificato con gran numero di dati mancanti

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
a	–,–	c, 0	c, 0	–,–
b	b, 1	a, 0	b/c, 1	a, 0
c	c, 1	a, 0	b, 0	b/c, 0

Osserviamo che in alcune posizioni della tabella 2.26 compare la specifica b/c; infatti in quelle caselle é possibile utilizzare, indifferentemente lo stato b (corrispondente alla coppia di stati indistinguibili (B,C)) oppure lo stato c (corrispondente alla coppia (B,E)). Infatti lo stato B puó essere indifferentemente sostituito dal nuovo stato b oppure dal nuovo stato c.

2.4.5 La tabella di fusione

Se il numero di stati dell'automa é elevato, é possibile sostituire al grafo di fusione, la cosiddetta *tabella di fusione*, la cui semplificazione risulta piú rapida per la maggior visibilitá delle coppie di stati incompatibili tra loro.

Si consideri ad esempio l'automa rappresentato dalla tabella degli stati di tabella 2.27; e per esso si costruisca una tabella di forma triangolare in cui sulle colonne sono indicati gli stati escludendo il primo (lo stato A), e sulle cui righe sono indicati ancora gli stati dell'automa, con esclusione dell'ultimo (lo stato G). Questa tabella, detta *tabella di fusione*, é rappresentata in figura 2.9

La tabella di fusione può essere completata osservando le seguenti regole:

- Quando una coppia stati hanno uscite compatibili e successori compatibili, la casella che li individua viene riempita con una V (ció corrisponde ad una linea che li collega se avessimo utilizzato il grafo di fusione).

- Quando una coppia di stati ha uscite compatibili ma successori distinti, tutte le coppie di successori tra loro incompatibili vengono inseriti nella casella della tabella di fusione nella casella che individua tale coppia (ció corrisponde ad una linea con etichettate le coppie di successori nel grafo di fusione).

Tabella 2.27: Automa utilizzato nell'esempio di costruzione della tabella di fusione

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
A	–,–	G, 1	G, 0	–,–
B	A, 0	–,–	–,–	–,–
C	F, –	D, 0	–,–	A, 1
D	–, 1	E, 0	B, –	–,–
E	–, 0	E, 0	C, –	B, 1
F	–,–	G, –	A, 0	–,–
G	–,–	–, 1	–, –	–, –

- Quando una coppia di stati ha uscite tra loro incompatibili, la corrispondente casella viene barrata con una croce.

Ad esempio, la coppia (A,B) della tabella 2.27 ha uscite e stati compatibili; pertanto nella tabella di fusione, nella casella individuata dalla colonna A e dalla riga B viene inserita una V. Gli stati (B,C) hanno invece uscite compatibili, ma successori distinti (A ed F) per $x_1 = 0$, $x_2 = 0$. Pertanto nella casella individuata dalla colonna C e dalla riga B della tabella scriveremo la coppia (A,F). Se consideriamo la coppia di stati D ed E, notiamo che hanno uscite distinte per $x_1 = 0$, $x_2 = 0$. La casella della colonna D e riga E sará allora riempita con una croce. Procedendo in questo modo, per ciascuna coppia di stati, si ottiene la tabella di figura 2.9.

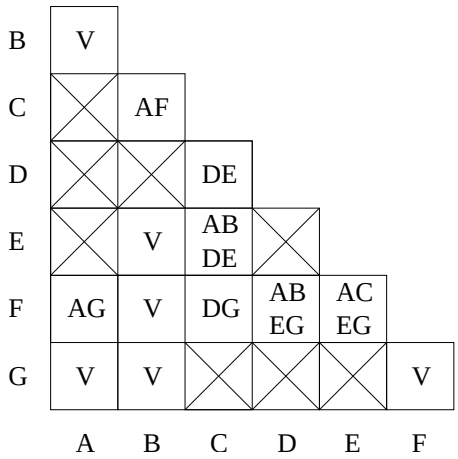

Figura 2.9: Tabella di fusione

Passiamo ora alla semplificazione della tabella di fusione; essa procede in modo analogo a quanto visto per il grafo di fusione. Ad esempio, osserviamo che la coppia di stati (A,F) é compatibile se e solo se lo é la coppia (A,G). E siccome quest'ultima é compatibile, non cancelliamo nulla nella colonna A, riga F della tabella. Per lo stesso motivo, non cancelliamo nulla nella casella che individua la coppia di stati (B,C). Consideriamo la coppia (C,D); essa é compatibile se e solo se lo sono (D,E). Ma questi ultimi non lo sono (la loro casella é contrassegnata con una croce). Barriamo allora con una croce la casella che individua la coppia (C,D). Ció corrisponderebbe, nel diagramma di fusione, nella cancellazione della linea che collega tale coppia di stati. Per lo stesso motivo, barriamo le caselle corrispondenti alle coppie (C,E), (C,F), (D,F), (E,F). A questo punto, una rapida occhiata alla tabella ci permette di dire che non ci sono altre coppie di stati da cancellare. La tabella cosí semplificata è rappresentata in figura 2.10.

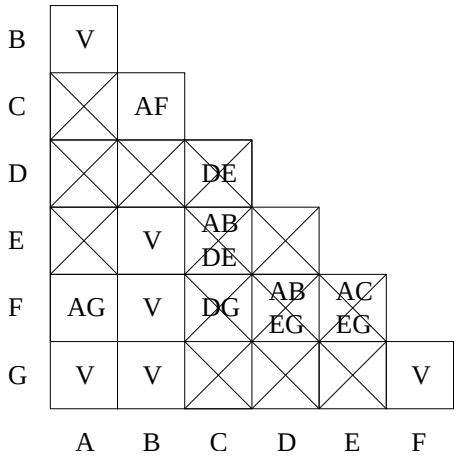

Figura 2.10: Tabella di fusione semplificata

Le coppie di stati compatibili sono allora: (A,B), (A,F), (A,G), (B,C), (B,E), (B,F), (B,G), (F,G). Osservando che (A, F) é compatibile se e solo se lo é (A,G), e che (B,C) lo é se e solo se lo é (A,F), possiamo tracciare il corrispondente grafo di compatibilitá, rappresentato in figura 2.11. Osserviamo che in tale grafo abbiamo dovuto aggiungere la coppia (D,D), dato che lo stato D non é compatibile con nessun altro stato.

Lo stato E appare solo nella coppia (B,E). Lo stato D da solo. Pertanto dobbiamo prendere come stati dell'automa in forma minima i nuovi stati a=(B,E) e b=D. Inoltre lo stato C compare solo nella coppia (B,C). Ma siccome c'é una freccia uscente da (B,C) verso (A,F), ed un'altra da (A,F) verso (A,G), dobbiamo prendere anche queste coppie di stati. Notiamo poi

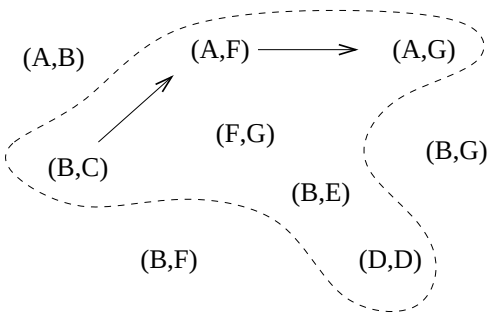

Figura 2.11: Grafo di compatibilitá ricavato dalla tabella 2.10

che, aggiungendo la coppia (F,G), possiamo creare un unico stato (A,F,G). Pertanto gli altri due stati sono c=(B,C) e d=(A,F,G). L'automa in forma minima ha allora quattro stati, come illustrato nella sua tabella 2.28.

Tabella 2.28: Automa in forma minima ricavato dalla tabella di fusione

	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
a	d,0	a, 0	c, –	a/c, 1
b	–, 1	a, 1	a/c ,–	–,–
c	d, –	b, 0	–,–	d, 1
d	–, –	d, 1	d, 0	–,–

Capitolo 3

Progettazione di circuiti sequenziali asincroni

In preparazione.

Capitolo 4

Le macchine lineari

In preparazione.

Capitolo 5

Circuiti sequenziali a relé

In preparazione.

Capitolo 6

Le reti di Petri

In preparazione.

Appendice A

Rappresentazione dei circuiti digitali

La rappresentazione grafica dei circuiti digitali diventa sempre piú complessa all'aumentare del numero di porte che compongono il circuito stesso. Solo quando il numero di porte é molto piccolo, la rappresentazione non crea problemi e la lettura dello schema elettrico risulta agevole. A titolo di esempio, si osservi la figura A.1 che rappresenta un circuito, di piccole dimensioni utilizzando la classica rappresentazione multifilare, in cui vengono messi in evidenza tutti gli ingressi delle porte. In questa figura il circuito realizza la funzione combinatoria $u = \overline{x}yz + x\overline{y}z + \overline{y}z\overline{w}$

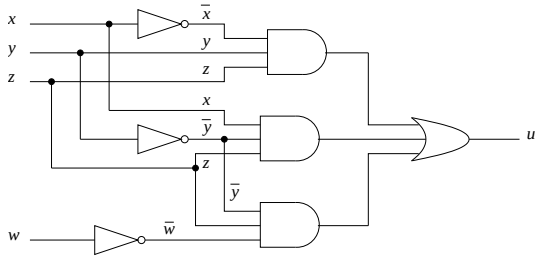

Figura A.1: Rappresentazione multifilare di un circuito digitale

Per semplificare il disegno dello schema elettrico, possiamo utilizzare la cosiddetta *rappresentazione unifilare*, in cui gli ingressi di una porta vengono rappresentati con una sola linea L, collegata agli ingressi, rappresentati da linee ortogonali ad L, che intersecano la linea L stessa (i punti di intersezione sono individuati con una crocetta, o equivalentemente, seppur piú raramente, con un circoletto nero). Ad esempio, in figura A.2 troviamo l'equivalenza tra alcune rappresentazioni multifilari ed unifilari.

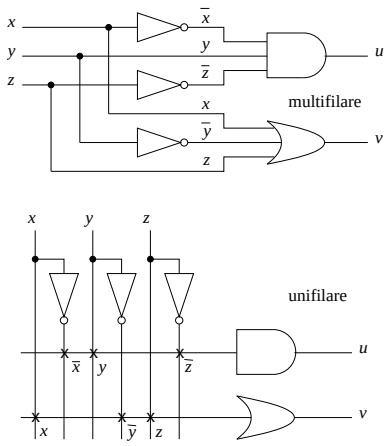

Figura A.2: Corrispondenza tra rappresentazione multifilare ed unifilare

Infine, in figura A.3 troviamo la rappresentazione unifilare del circuito di figura A.1. Nelle presenti dispense si utilizza la convenzione di utilizzare il circoletto nero, anziché la crocetta, per indicare i collegamenti delle porte ai rispettivi ingressi.

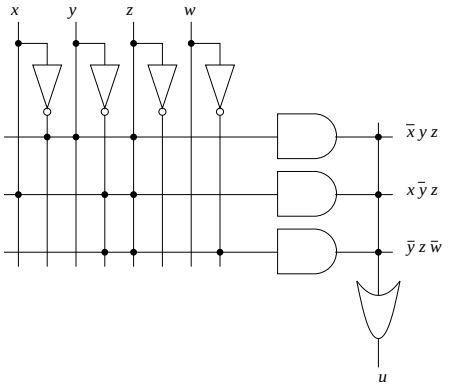

Figura A.3: Rappresentazione unifilare del circuito

Appendice B

Dispositivi di memoria elementari

Nella presente appendice verranno brevemente illustrati i principali dispositivi di memoria, capaci di memorizzare un bit, distinguendo tra dispositivi sincroni ed asincroni.

B.1 Il latch SR

Il più semplice dispositivo capace di memorizzare un bit è il latch SR; si tratta di un circuito *asincrono*, con due ingressi s ed r (detti rispettivamente *set* e *reset*, e con due uscite y_1 ed y_2. Il circuito può essere realizzato sia a porte NOR, che a porte NAND, come illustrato in figura B.1. Analizzeremo il funzionamento del circuito riferendoci alla versione con porte NOR; analoghe considerazioni possono essere fatte per l'altra architettura.

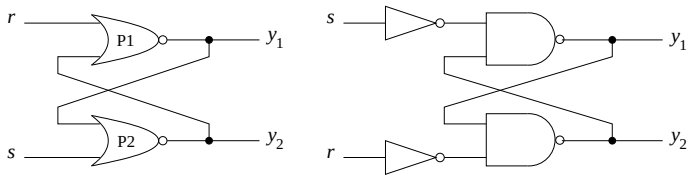

Figura B.1: Il latch SR

Supponiamo di applicare agli ingressi i valori $s = 1$, $r = 0$. In questo modo l'uscita y_2 della porta P2 (che coincide con un ingresso della porta P1) è a livello basso. La porta P1 ha cosí entrambi gli ingressi a livello basso, da cui la sua uscita y_1 è alta. Portiamo ora s a livello basso, lasciando $r = 0$.

Dato che si ha $y_1 = 1$, l'uscita y_2 di P2 rimane a livello basso, e di conseguenza la porta P1 ha ancora tutte e due gli ingressi a zero, da cui $y_1 = 0$. Le uscite sono rimaste invariate. Se ora poniamo $r = 1$ ed $s = 0$, si ha la situazione speculare rispetto al caso $s = 1$, $r = 0$, e per simmetria otteniamo $y_1 = 0$, $y_2 = 1$. Se ora lasciamo invariato s a livello basso e riportiamo r a zero, troviamo la porta P2 con entrambi gli ingressi a zero, da cui $y_2 = 1$, e di conseguenza, P1 ha in ingresso $y_2 = 1$ da cui $y_1 = 0$. Abbiamo pertanto visto che con $s = r = 0$ le uscite non cambiano valore rispetto a quello ottenuto rispettivamente con $s = 1$, $r = 0$ e con $s = 0$, $r = 1$.

Di conseguenza, si può memorizzare un bit (y_1, oppure $y_2 = \overline{y}_1$) applicando $s = 1$, $r = 0$ o viceversa (nel primo caso memorizzeremo $y_1 = 1$, nel secondo $y_1 = 0$) e poi mantenerlo in memoria per un tempo indefinito riportando entrambi gli ingressi a zero.

Ma cosa succede, se applichiamo in ingresso $s = r = 1$? Entrambi le porte hanno un ingresso a livello alto, e di conseguenza $y_1 = y_2 = 0$. Questa situazione è però delicata, dando luogo ad un fenomeno noto come *corsa critica*. Riportiamo infatti *entrambi* gli ingressi a zero; otteniamo che entrambi le porte hanno tutti gli ingressi nulli, si ottiene $y_1 = y_2 = 1$. Ma y_1 ed y_2 sono anche gli ingressi delle porte, che allora dovrebbero avere le loro uscite a livello basso. Di conseguenza, *non è possibile determinare il valore delle uscite*. In pratica, il passaggio da $s = r = 1$ ad $s = r = 0$ non è mai simultaneo per entrambi gli ingressi. Ad esempio, il passaggio di s da 1 a 0 potrebbe avvenire un istante prima di quello di r (ed in questo caso, avendosi $s = 0$, $r = 1$ si avrebbe $y_1 = 0$, $y_2 = 1$), oppure potrebbe andare a zero prima r di s, ottenendo così $y_1 = 1$, $y_2 = 0$. L'uscita non è quindi determinabile, e di conseguenza il passaggio da $s = r = 1$ ad $s = r = 0$, detto *corsa critica* è da evitare. Per questo motivo, al latch SR è opportuno evitare l'applicazione degli ingressi $s = r = 1$.

Osserviamo infine che, evitando la combinazione $s = r = 1$, possiamo dire che in tutti gli altri casi l'uscita y_2 é la negata di y_1.

Possiamo infine rappresentare in una tabella (B.1) il funzionamento del latch SR.

B.2 Il flip-flop SR

Nel caso si desideri sincronizzare la commutazione delle uscite con un impulso di clock, si può ricorrere al circuito di figura B.2, in cui entrambi gli ingressi s ed r vengono moltiplicati per il segnale di clock c_k. In questo caso, all'ingresso delle porte P1 e P2 abbiamo i segnali $s \cdot c_k$ ed $r \cdot c_k$, e quando $c_k = 0$ tali segnali sono entrambi nulli e il flip-flip mantiene invariate le sue uscite. Per

Tabella B.1: Tabella input-output del latch SR

s	r	y_1
1	0	si porta a 1 (o ci rimane se lo è già)
0	1	si porta a 0 (o ci rimane se lo è già)
0	0	rimane invariata
1	1	da evitare (corsa critica)

poter cambiare i valori delle uscite y_1 ed y_2 non basta più applicare $s = 1$, $r = 0$ oppure $s = 0$ ed $r = 1$, ma anche il segnale di clock deve trovarsi a livello alto. Pertanto il circuito è simile al precedente, ma la commutazione avviene sul livello alto di c_k. Pertanto la commutazione si ottiene tenendo costanti $s = 1$, $r = 0$ (o $s = 0$, $r = 1$) per un breve istante in cui si applica un impulso di clock. Le considerazioni fatte per la realizzazione con porte NOR si applicano, per analogia, a quella con porte NAND.

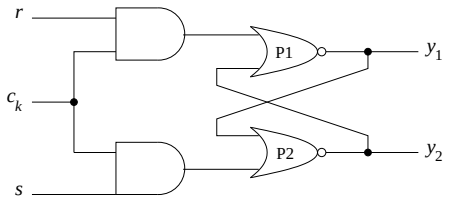

Figura B.2: Il flip-flop SR

B.3 Il flip-flop JK

Il flip-flop JK è simile all'SR, ma presenta il vantaggio di non avere condizioni di corsa critica. Infatti, l'ingresso j corrisponde all'ingresso s dell'SR, e l'ingresso k al reset r, e quando si ha in ingresso una coppia di uno (cioè $j = k = 1$) l'uscita commuta. Lo schema elettrico di un JK è rappresentato in figura B.3, e come si vede è ottenuto da un latch SR con l'aggiunta di un paio di porte AND a tre ingressi. Il flip-flop JK è attivo solo in presenza di un segnale di clock a livello alto, come per il flip-flop SR.

Notiamo subito che con $j = k = 0$, le uscite s ed r delle porte P1 e P2 sono a livello basso, e pertanto il latch SR tratteggiato non commuta. Di conseguenza, applicare $j = k = 0$ non fa commutare il flip-flop. Supponiamo ora di avere le uscite $y_1 = 0$, $y_2 = 1$. Se poniamo $j = 1$ e $k = 0$, in presenza

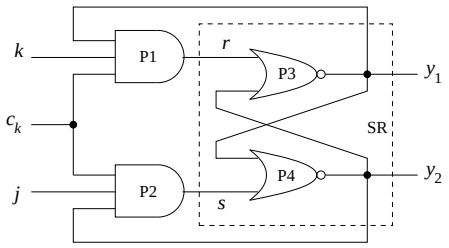

Figura B.3: Il flip-flop JK

di un impulso di clock otteniamo $s = jy_2c_k = 1$, $r = ky_1c_k = 0$, e ció fa commutare le uscite dell'SR (e quindi anche del JK), ottenendo cosíi nuovi valori $y_1 = 1$, $y_2 = 0$. Notiamo che se applichiamo gli ingressi $j = 1$ e $k = 0$ quando le uscite sono $y_1 = 1$, $y_2 = 0$, otteniamo $s = jy_2c_k = 0$, $r = ky_1c_k = 0$, e le uscite non cambiano. Possiamo pertanto dire, che con ingressi $j = 1$, $k = 0$ le uscite si portanto al valore $y_1 = 1$, $y_2 = 0$, oppure se hanno gia' questo valore, rimangono invariate. Per ragioni di simmetria, possiamo dire che se applichiamo gli ingressi $j = 0$, $k = 1$ le uscite si portano al valore $y_1 = 0$, $y_2 = 1$ oppure se hanno gia' questo valore, non variano. Infine, supponiamo di avere $y_1 = 1$, $y_2 = 0$, e di applicare l'ingresso $j = k = 1$. Si ha allora $s = jy_2c_k = 0$, $r = ky_1c_k = 1$ e ciò porta le uscite al nuovo valore $y_1 = 0$, $y_2 = 1$. Per simmetria, se si ha $y_1 = 0$, $y_2 = 1$, e si applica in corrispondenza dell'impulso di clock la combinazione di ingresso $j = k = 1$, si ottiene $s = 1$, $r = 0$ e le uscite commutano, assumendo il nuovo valore $y_1 = 1$, $y_2 = 0$. Notiamo che l'uscita y_2 é *sempre* la negata di y_1. Il funzionamento del flip-flop JK è riassunto in tabella B.2

Tabella B.2: Tabella input-output del latch SR

j	k	y_1
1	0	si porta a 1 (o ci rimane se lo è già)
0	1	si porta a 0 (o ci rimane se lo è già)
0	0	rimane invariata
1	1	passa da 0 ad 1, oppure da 1 a 0

Occorre tuttavia notare che se gli ingressi permangono ancora al valore $j = k = 1$, e l'impulso di clock non è sufficientemente breve, si ha dopo pochi istanti una nuova commutazione delle uscite. Per questo motivo, l'utilizzo di questo flip-flop è limitato ad impulsi di clock assai brevi; il segnale di clock non deve permanere a livello alto dopo la commutazione. Per questo motivo

questo tipo di flip-flop non è quasi mai utilizzato. Questo inconveniente è superato utilizzato l'architettura master-slave o quella edge-triggered, descritte nei prossimi paragrafi.

B.3.1 Il JK master-slave

Questo flip-flop è ottenuto collegando in cascata due flip-flop JK come quelli di figura B.3, di cui il primo è attivo quando il clock è a livello alto, ed il secondo quando il clock è a livello basso (infatti nel secondo flip-flop è applicato in ingresso alle porte P5 e P6 il segnale $\overline{c}_k$).

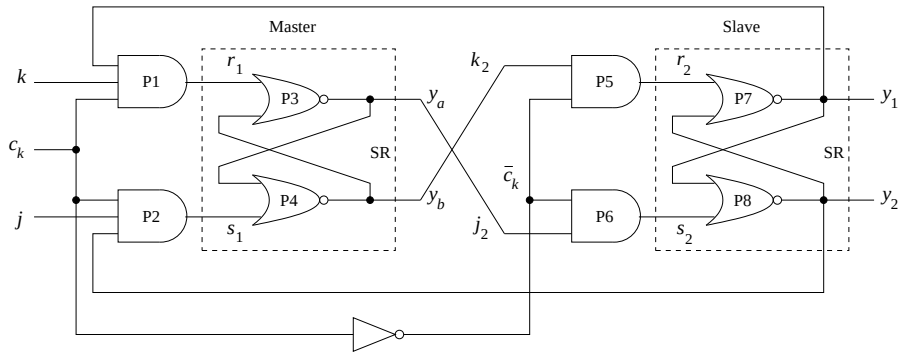

Figura B.4: Il flip-flop JK master-slave

Per capire il funzionamento del flip-flop, osserviamo che quando il clock si porta dal livello basso a quello alto, lo slave è disabilitato, in quanto le porte P5 e P6 hanno uscita nulla, e pertanto le uscite dell'SR slave non cambiano. Mentre il clock é a livello alto, avviene la commutazione delle uscite y_a ed y_b dell'SR master, che coincidono con gli ingressi j_2 e k_2 delle porte P5 e P6. Quando il clock si riporta a livello basso, risulta abilitata la cella slave, le cui uscite commutano in base ai valori di j_2 e k_2, mentre risulta disabilitatta la cella master, che non commuta piú. E ciò elimina il fenomeno del rischio di commutazione multipla in presenza degli ingressi $j = k = 1$ analizzato nel precedente paragrafo.

Per analizzarne meglio il funzionamento, siano inizialmente $y_1 = 1$, $y_2 = 0$. Applichiamo agli ingressi la combinazione $j = k = 1$. Portiamo il clock a livello alto. In questo modo la cella master è abilitata, quella slave no. L'uscita di P_1 è $r_1 = y_1 k c_k = 1$, mentre quella di P2 è $s_1 = y_2 j c_k = 0$. La cella SR master commuta, e le sue uscite (coincidenti con gli ingressi di P5 e P6) saranno $j_2 = y_a = 0$, $k_2 = y_b = 1$. Ricordando che il clock è a livello

alto, e quindi $\overline{c}_k = 0$, le uscite di P5 e P6 sono a livello basso, da cui si vede che lo slave non commuta e rimane per il momento disabilitato.

Se ora il clock si riporta a livello basso, viene disabilitata la cella master, mentre le uscite di P5 e P6 saranno: $s_2 = \overline{c}_k j_2 = 0$, $r_2 = \overline{c}_k k_2 = 1$. La cella slave commuterà, e le sue uscite varranno $y_1 = 0$, $y_2 = 1$, commutando come desiderato. La tabella del flip-flop master-slave è sempre la B.2, che vale per tutti i flip-flop JK, con la sola differenza che mentre il JK semplice visto nel precedente paragrafo commuta quando il clock è a livello alto, i JK master-slave ed edge-triggered commutano sul fronte di discesa del clock.

B.3.2 Il JK edge-triggered

Il JK master-slave, analizzato nel precedente paragrafo, presenta il difetto di richiedere che gli ingressi non cambino di valore durante il permanere a livello alto del clock. Se infatti, per un qualsiasi motivo, ad esempio la presenza di un disturbo, gli ingressi cambiano valore prima che il clock sia ritornato a livello basso, si possono avere dei malfunzionamenti. Più precisamente, supponiamo che, con c_k alto, e con lo slave nello stato di reset ($y_1 = 0$, $y_2 = 1$), e con l'ingresso j a livello basso, si verifichi un disturbo su J, portandolo per breve tempo a livello alto. Avendosi $s_1 = c_k y_2 j$, il disturbo arriva all'ingresso della porta P1, portanto per un instante s_1 a livello alto, e di conseguenza facendo commutare la cella master in modo che le sue uscite siano $y_a = 1$, $y_b = 0$. E siccome si ha $y_1 = 0$, l'altro ingresso $r_1 = k c_k y_1$ dello slave rimane a livello basso, *impedendo il corretto reset della cella master al termine del disturbo.* Di conseguenza, l'uscita $y_a = 1$, $y_b = 0$ viene riportata all'ingresso dello slave mentre il clock torna a livello basso, causando una commutazione indesiderata di quest'ultima. Le uscite del flip-flop pertanto commutano (in modo indesiderato) in seguito al disturbo. Questo fenomeno, è noto come *ones catching*. Per rendere poco probabile la presenza di questo fenomeno, la durata dell'impulso di clock va tenuta più breve possibile, in modo da ridurre la probabilità di cambio dei valori degli ingressi.

Una possibile solutione è data dall'utilizzo del flip-flop JK edge-triggered, riportato in figura B.5.

Quando il clock viene portato a livello alto, i segnali presenti agli ingressi j e k risultano disponibili alle uscite delle porte P1 e P2. Tuttavia, dato che alle porte P3 e P4 è applicato il segnale di clock negato, quando $c_k = 1$ le uscite di tali porte rimangono a livello basso. Se ora il clock si riporta a livello basso, vengono disabilitate le uscite di P1 e P2. Ma se si suppone che la transizione da 1 a 0 del clock è sufficientemente rapida, otteniamo agli ingressi s ed r del latch due brevi inpulsi, di durata pari ai tempi di ritardo delle porte NAND, di valore uguale a quello di J e di K. In questo modo

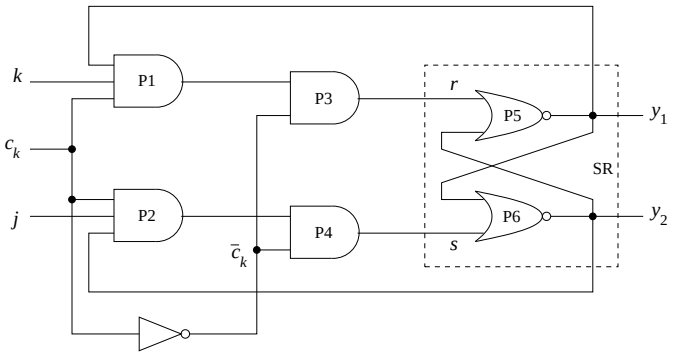

Figura B.5: Flip-flop JK edge-triggered

il latch commuta durante il fronte di discesa dell'impulso di clock. Questo tempo è generalmente inferiore a 20 nanosecondi.

I flip-flop JK più utilizzati sono di tipo master-slave o edge-triggered. Useremo indifferentemente uno o l'altro nella progettazione degli automi a stati finiti.

B.3.3 Ingressi di preset e clear

I flip-flop sono generalmente corredati di due ingressi, detti preset (p_r) e clear c_r), che servono per inizializzarne le uscite. Nel caso di un JK, questi segnali sono applicati all'ingresso della cella del latch SR (vedi figura B.6). In questa figura le celle dei latch sono realizzate a porte NAND, metendo in evidenza una possibile architettura alternativa dei flip-flop, che possono tutti essere realizzati sia con porte NAND che con porte NOR.

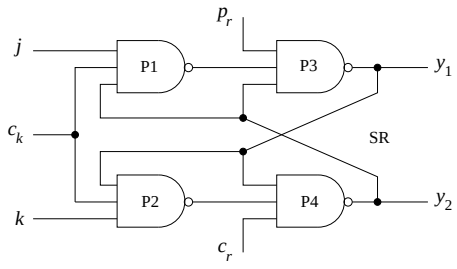

Figura B.6: JK con ingressi di preset e di clear

Supponiamo di volere inizializzare il dispositivo in modo che inizialmente le sue uscite siano $y_1 = 1$, $y_2 = 0$. Per far ciò, sia $c_r = 0$, $p_r = 1$, e sia

$c_k = 0$. In questo caso l'uscita di P1 è alta, e la porta P3 ha tutti gli ingressi a livello alto. Di conseguenza, l'uscita di P3 è zero, e dato che essa è collegata all'ingresso di P4, quest'ultima avrà uscita a livello alto. Abbiamo così inizializzato il dispositivo con uscite $y_1 = 1$, $y_2 = 0$. Per simmetria, se volessimo inizializzare il dispositivo con $y_1 = 0$, $y_2 = 1$ occorre e basta che sia $c_l = 1$, $p_r = 0$ quando il clock è a livello basso.

Una volta stabilito lo stato iniziale delle uscite, è possibile far funzionare normalmente il flip-flop ponendo $c_r = p_r = 1$. Infine, in figura B.7 è riportato un JK master-slave dotato di ingressi di preset e di clear; l'inizializzazione avviene sulla cella master, e lo stato iniziale di questà è poi riportato nella cella slave.

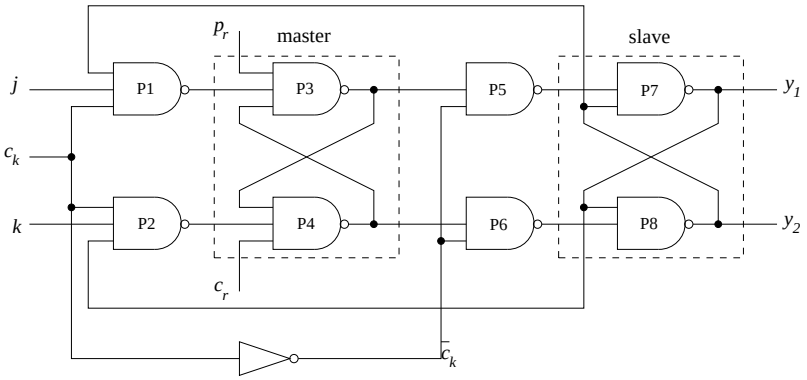

Figura B.7: JK master-slave con ingressi di preset e di clear

B.4 Il flip-flop di tipo D

Se, utilizzando una porta NOT, facciamo in modo che gli ingressi j e k di un flip-flop JK siano sempre uno negato dell'altro, otteniamo un flip-flop di tipo D, riportato in figura B.8. Useremo indifferentemente JK di tipo edge-triggered o master-slave.

Nel caso in cui sia $d = 1$, si ha $j = 1$, $k = 0$, ed applicando un impulso di clock le uscite saranno $y_1 = 1$, $y_2 = 0$. Viceversa, se l'ingresso d è zero, si avrà $j = 0$, $k = 1$ e le uscite saranno $y_1 = 0$, $y_2 = 1$.

In pratica, l'ingresso d viene memorizzato, dopo l'applicazione del segnale di clock, nell'uscita y_1 (mentre y_2 è sempre la negata di y_1). Anche questo flip-flop, è generalmente dotato di ingressi di preset e di clear. Il funzionamento è riassunto in tabella B.3, ove la transizione di y_1 avviene ovviamente solo in presenza di un impulso di clock.

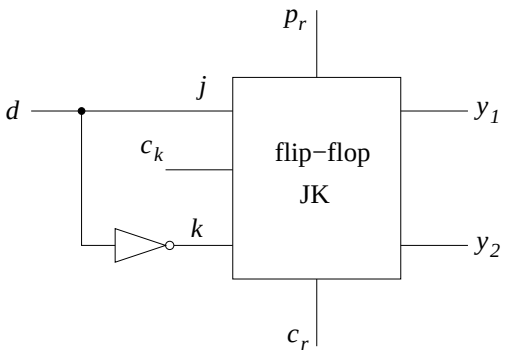

Figura B.8: Flip-flop di tipo D

Tabella B.3: Tabella input-output del flip-flop D

d	y_1
1	si porta a 1 (o ci rimane se lo è già)
0	si porta a 0 (o ci rimane se lo è già)

Notiamo come il flip-flop di tipo D possa essere utilizzato come semplice elemento di memoria per memorizzare il valore della variabile d. Questo dispositivo costituisce dunque una memoria avente la capacità di un bit. Dato che l'uscita assume il valore dell'ingresso dopo l'applicazione del clock, e quindi con un certo ritardo, questo elemento è chiamato di tipo D, dall'inglese *delay* (ritardo).

B.5 Il flip-flop di tipo T

Se gli ingressi del JK sono collegati entrambi insieme, e se si indica con t il loro ingresso comune, si possono avere, in presenza del clock, due casi:

- Nel primo caso, si ha $t = 0$, da cui $j = k = 0$ e le uscite del dispositivo rimangono invariate.
- Se invece si ha $t = 1$, si ottiene $j = k = 1$ e le uscite cambiano il loro valore

Il dispositivo é riportato in figura B.9. Anche qui sono presenti solitamente gli ingressi di preset e di clear. Nella dispensa, tali ingressi si considereranno sempre presenti, anche se spesso non sono indicati per semplicità.

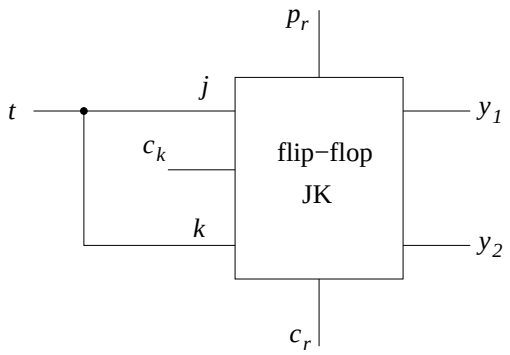

Figura B.9: Flip-flop di tipo T

Il nome T deriva dall'inglese *toggle*, in quando le uscite di questo dispositivo commutano sono applicando un segnale alto all'ingresso, in presenza del clock, come illustrato in tabella B.4. Ovviamente, y_2 è la negata di y_1.

Tabella B.4: Tabella input-output del flip-flop T

t	y_1
1	Passa da 1 a 0, oppure da 0 ad 1
0	Rimane invariata

Appendice C

Full adder combinatorio

In preparazione.

Bibliografia

[1] S.J. Cahill. *Digital and Microprocessor Engineering, second edition*, Ellis Horwood, New York, (1993).

[2] D.A. Hodges, H.G. Jackson. *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill, Singapore, (1985).

[3] Z. Kohavi. *Switching and Finite Automata Theory*, McGraw-Hill, New York, (1978).

[4] J. Millman. *Microelectronics: Digital and Analog Circuits and Systems*, McGraw-Hill, Singapore, (1986).

[5] M. Sami, R. Negrini. *Calcolatori Elettronici, Appunti del Corso*, CLUP, Milano, (1986).

[6] D. Tambuchi. *I Malfunzionamenti nei Circuiti Combinatori*, dispensa, (2003).

[7] S.H. Unger. *Asynchronous Sequential Switching Circuits*, R.E. Krieger Publishing Company, Malabar, (1983).