Reti Logiche

November 11, 2025

1 Macchine digitali

1.1 Segnali

Un segnale può essere:

Analogico continuo e con valori reali

Digitale valori discreti ad intervalli

1.2 Macchina digitale

Una macchina digitale interpreta dei segnali digitali.

1.3 Interruttore

Gli interruttori sono alla base del funzionamento delle macchine digitali. Possono essere:

Manuali interruttore della luce, tasti di una tastiera, ...

Elettronici transistor, relays, ...

1.4 Microprocessore

Macchina digitale alla cui base troviamo il componente transistor. L'architettura dei microprocessori è molto complicata. Possiamo dividere l'architettura in livelli di astrazione, individuando componenti primitivi, che permettono di realizzare componenti di sempre più alto livello fino ad avere il microprocessore desiderato. Dei componenti primitivi interessa soltanto il "comportamento", ovvero la relazione tra input ed output da un punto di vista esterno.

2 Informazione

2.1 Codifica binaria

In una macchina digitale, le informazioni vengono rappresentate sotto forma di bit (0 e 1). Questo sistema è detto sistema binario. Ci sono diversi modi per rappresentare le informazioni. Prendendo come esempio il numero 28:

Sistema binario	Sistema decimale	Sistema esadecimale
11100	28	1C

Viene detto numero in base n il numero che viene rappresentando usando l'n-esimo sistema numerico.

2.2 Cambio di base

Per convertire un numero da un sistema all'altro, esistono diversi metodi, dette conversioni. Alcune di queste sono:

2.2.1 Conversione iterativa

In base n: si divide un numero in parte intera e parte decimale.

Parte intera si divide per n il numero iniziale, ottenendo un quoziente intero e un resto (*0 o 1*); si segna il resto e si ripete l'operazione dividendo il quoziente intero precedente, finché esso risulterà = 0; per finire, riscrivere tutti i resti "al contrario" (*partendo dall'ultimo calcolato e arrivando al primo*).

Parte decimale si prende il numero iniziale e lo si moltiplica per n, segnando separatamente la parte intera e decimale del risultato. Poi bisogna ripetere lo stesso procedimento usando la parte decimale segnata precedentemente come fattore, iterando finché la parte decimale risulterà = 0 oppure in base alla precisione decisa.

2.3 Numeri binari

Per rappresentare un numero senza segno N su una macchina digitale, usiamo il sistema binario, base 2. La quantità di numeri che possiamo rappresentare è determinata dalla quantità di bit che vengono assegnati. Per un numero n finito di bit, la quantità di numeri senza segno N in base 2 è:

$$2^n - 1$$

2.4 Problemi con la conversione A/D

Nel mondo reale, l'allineamento meccanico perfetto è impossibile, ovvero due segnali non possono variare contemporaneamente. Per evitare letture scorrette nei punti di disallineamento, è opportuno utilizzare configurazioni relative a posizioni consecutive che differiscono di un solo valore.

2.5 Codice di Gray

A differenza del codice binario, il codice di Gray è una configurazione di bit che codifica informazioni adiacenti tali che differiscono soltanto di 1 bit. Viene usato generalmente per ridurre errori nella codifica da analogico a digitale.

2.6 Transcodifica

Possiamo dividere i codici in esterno e interno.

Esterno ridondante e standardizzato

Interno non ridondante

2.7 Codici standard e proprietari

Standard Un codice scelto da norme internazionali o da un costruttore molto diffuso, e permette a tutti di adottare quel codice per rendere le proprie informazioni facilmente interpretabili

Proprietario Un codice scelto dal costruttore con l'unico scopo di interconnettere i propri prodotti.

3 Reti logiche

Una rete logica è un'astrazione che rappresenta una combinazione di "interruttori" che elaborano segnali binari.

3.1 Gate

Definiamo "gate" tutti i componenti elementari di cui non conosciamo il come sono fatti, ma il loro comportamento. Il numero di funzioni diverse di n ingressi binari con un'uscita binaria è:

$$2^{2^n}$$

I componenti elementari, o funzioni possibili, limitandosi ai componenti con un unico segnale binario di ingresso x sono 4, $(n = 1)$.

esempio:

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

Ogni gate è descritto da:

Tabella della verità con ogni riga che riporta un possibile ingresso e la corrispondente uscita

Simbolo circuitale per rappresentarlo graficamente e distinguerlo

Espressione un modo di rappresentare la relazione tra ingressi ed uscite

3.1.1 Gate NOT

Tabella della verità:

x	y
0	1
1	0

Espressione: $y = \overline{x}$ oppure $y = x'$

3.1.2 Gate AND

Tabella della verità:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Espressione: $z = x * y$ oppure $z = xy$

3.1.3 Gate OR

Tabella della verità:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

Espressione: $z = x + y$

3.1.4 Gate EXOR/XOR

Tabella della verità:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Espressione: $z = x \oplus y$

lo XOR viene anche detto somma modulo 2, in quando il suo output può essere interpretato come il risultato della somma di due bit, escludendo il riporto.

3.1.5 Gate NAND

Tabella della verità:

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

Espressione: $z = x \uparrow y$ oppure $z = \overline{xy}$

3.1.6 Gate NOR

Tabella della verità:

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

Espressione: $z = x \downarrow y$ oppure $z = \overline{x + y}$

3.1.7 Gate EXNOR

Tabella della verità:

x	y	z
0	0	1
0	1	0
1	0	0
1	1	1

Espressione: $z = x \equiv y$ oppure $z = \overline{x \oplus y}$

3.1.8 AND e OR con n ingressi

esempio, AND con $n = 3$:

x	y	w	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3.1.9 Tabelle per gate con 2 ingressi

x	y	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

3.1.10 Bus di segnali

Un gruppo di segnali viene detto bus. Per indicare un bus di n segnali che codificano un'informazione, si usa la notazione con parentesi quadre:

$$\text{TEST}[n-1 \dots 0]$$

esempio:

- bus a 3 bit per il colore: COLORE$[2 \dots 0]$

Per riferirci ad uno dei segnali del bus, si usa la notazione:

$$\text{TEST}0, \dots, \text{TEST}n-1$$

esempio:

- secondo segnale del bus dei colori: COLORE1

3.1.11 Ritardo di propagazione

Pur lavorando con componenti astratti, bisogna tenere in considerazione il fatto che possono essere componenti reali. La differenza principale tra i due, da prendere in considerazione, è il "ritardo di propagazione", indicato con τ_p, ed esprime il tempo che un segnale impiega per completare la transizione tra stati. Un impulso di durata inferiore a τ_p su uno degli ingressi non appare in uscita.

4 Reti combinatorie

4.1 Definizione

Una rete logica la dove l'uscita dipende unicamente dagli ingressi correnti.

4.2 Comportamento

La tabella della verità, con tutte le combinazioni di ingressi possibili e i rispettivi segnali d'uscita.

4.3 Struttura

I componenti che realizzano la rete logica indicata. Può essere dichiarata sotto forma di:

Espressione esempio: $z = (x \equiv y)(x \oplus w)$

Schema logico

4.4 Dalle tabelle di verità a espressioni

Uno dei metodi per passare da una tabella di comportamento ad una espressione, è mediante l'uso delle così dette espressioni canoniche.

4.5 Espressioni canoniche

Espressione canonica SP Somma di prodotti, prima forma canonica

Espressione canonica PS Prodotto di somme, seconda forma canonica

4.5.1 Full adder

Una rete logica con 3 ingressi (a, b, r) e due uscite (S, R). Questo rappresenta:

$S = 1$ quando il numero di uno dei suoi ingressi è dispari

$R = 1$ quando in ingresso ci sono due o più 1

Questa rete è combinatoria perchè l'uscita dipende solo dagli ingressi attuali. Sarà una rete fondamentale per realizzare operazioni aritmetiche tra numeri binari.

a	b	r	S	R
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Per trovare un'espressione, prendiamo come esempio l'output S:

$$S = 1 \implies Riga_1, Riga_2, Riga_4, Riga_7$$

Perciò $S = 1$ quando:

- $C_1 = 1 \implies a = 0, b = 0, r = 0$
- $C_2 = 1 \implies a = 0, b = 1, r = 0$

- $C_4 = 1 \implies a = 1,\ b = 0,\ r = 0$
- $C_7 = 1 \implies a = 1,\ b = 1,\ r = 1$

Che possiamo trasformare in:

- $C_1 = 1 \implies a' = 1,\ b' = 1,\ r' = 1$
- $C_2 = 1 \implies a' = 1,\ b = 1,\ r' = 1$
- $C_4 = 1 \implies a = 1,\ b' = 1,\ r' = 1$
- $C_7 = 1 \implies a = 1,\ b = 1,\ r = 1$

Che ci permette di ottenere:

$$S = C_1 + C_2 + C_4 + C_7 = a'b'r' + a'br' + ab'r' + abr$$

Usando lo stesso ragionamento otteniamo che:

$$R = a'br + ab'r + abr' + abr$$

4.6 Notazione simbolica

Mintermine il vettore dei bit, rappresentati mediante il loro indice, che assumono il valore 1

Maxtermine il vettore dei bit, rappresentati mediante il loro indice, che assumono il valore 0

Mintermine e maxtermine sono complementari tra loro. Una configurazione è o un mintermine o un maxtermine. Il pedice degli operatori Σ e Π corrisponde al numero di ingressi che danno origine al mintermine o maxtermine.

esempio, Full adder:

i	a	b	r	S	R
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

- $S(a, b, r) = \Sigma_3 m(1, 2, 4, 7)$
- $S(a, b, r) = \Pi_3 M(0, 3, 5, 6)$
- $R(a, b, r) = \Sigma_3 m(3, 5, 6, 7)$
- $R(a, b, r) = \Pi_3 M(0, 1, 2, 4)$

4.7 Equivalenza tra espressioni

L'equivalenza tra due espressioni si indica con:

$$E_1 = E_2$$

Si può definire tale se e solo se le espressioni esprimono la stessa funzione (o TdV).

4.8 Funzioni incomplete

Anche delle espressioni che forniscono una valutazione uguale, limitate al dominio di una funzione incompleta, sono dette equivalenti. Un esempio di funzione incompleta è un encoder da 1 a N bit. Prendendo come esempio più specifico, un encoder con 3 ingressi, a seconda del valore che assumono i bit di uscita delle configurazioni non rilevanti (fuori dal dominio della funzione), l'espressione risultante cambia.

4.9 Equivalenze notevoli

Proprietà della somma e del prodotto logico:

Commutativa $x + y = y + x$ e $x * y = y * x$

Associativa $(x + y) + z = x + y + z$ e $(x * y) * z = x * y * z$ (utile per ridurre il fan-in)

Distributiva $(x * y) + (x * z) = x * (y + z)$ e $(x + y) * (x + z) = x + (y * z)$ (valida soltanto in algebra binaria)

Idempotenza $x + x = x$ e $x * x = x$

Identità $x + 0 = x$ e $x * 1 = x$

Limite $x + 1 = 1$ e $x * 0 = 0$

Involuzione $(x')' = x$ (viene usata per amplificare il segnale)

Limitazione $x + x' = 1$ e $x * x' = 0$

Combinazione $xy + xy' = x$ e $(x + y) * (x + y') = x$

Prima legge di De Morgan $(x + y)' = x' * y'$

Seconda legge di De Morgan $(x * y)' = x' + y'$

Consenso $xy + x'z + yz = xy + x'z$ e $(x + y) * (x' + z) * (y + z) = (x + y) * (x' + z)$

4.10 Manipolazione algebrica di espressioni

Data l'espressione per il ritorno di un full adder:

$$R = a'br + ab'r + abr' + abr$$

Essa può essere semplificata usando i seguenti passaggi:

Distribuzione $a'br + ab'r + ab * (r' + r)$

Limitazione $a'br + ab'r + ab1$

Identità $a'br + ab'r + ab$

Oppure una versione ancora più semplificata è:

Idempotenza $a'br + ab'r + abr' + abr + abr + abr$

Distribuzione $br * (a' + a) + ar * (b' + b) + ab * (r' + r)$

Limitazione $br1 + ar1 + ab1$

Identità $br + ar + ab$

Rispetto all'espressione originale $R = a'br + ab'r + abr' + abr$, da una rete formata da:

- 1 OR da 4 ingressi
- 4 AND da 3 ingressi
- 3 NOT

Siamo passati, usando l'espressione $R = br + ar + ab$, ad una rete composta da:

- 1 OR a 3 ingressi
- 3 AND da 2 ingressi

4.11 Il problema della sintesi

La sintesi è il processo per individuare l'espressione "migliore" per la realizzazione della funzione assegnata. "Migliore" può essere definito con criteri anche opposti tra loro, come:

- Rapidità di progetto
- Massima flessibilità
- Massima velocità
- Minima complessità

Abbiamo due possibili obbiettivi:

Reti di costo minimo Massima velocità e Minima complessità

Reti programmabili Rapidità di progetto e Massima flessibilità

5 Reti di costo minimo

5.1 Ritardi e velocità

Quando cambia un ingresso di un gate, l'uscita non cambia istantaneamente, ma dopo un tempo τ_p che dipende dalla tecnologia utilizzata. Questo ritardo varia da gate a gate e anche se il passaggio è da H a L o viceversa. Nel caso peggiore, il ritardo totale della rete è dato dalla somma dei ritardi dei gate sul percorso più lungo tra ingressi e uscite. Si assegna il ritardo peggiore alla rete complessiva.

5.2 Complessità e velocità

Per confrontare complessità e velocità di risposta di reti combinatorie equivalenti, si usano i seguenti indicatori:

N_{gate} il numero di gate nella rete (maggiore è l'N_{gate} , maggiore è la complessità)

N_{conn} il numero di connessioni in una rete (maggiore è l'N_{conn} , maggiore è la complessità)

N_{casc} il numero massimo di gate disposti in cascata, ovvero in serie tra ingressi e uscite (minore è l'N_{gate} , maggiore è la velocità)

5.3 Rete di "costo minimo"

Ipotesi:

- Ingressi disponibili in forma vera e negata
- fan-in dei gate quando serve

Una rete combinatoria, per essere considerata "di costo minimo", è una rete con:

- Non più di 2 gate in cascata tra ingressi e uscita
- Minimo numero di gate
- Minimo numero di ingressi per gate

Il numero di gate e/o connessioni della rete di costo minimo di tipo SP è in generale diverso da quello della rete di costo minimo di tipo PS.

E' possibile che più espressioni dello stesso tipo (SP o PS) siano minime (abbiano cioè valori uguali di N_{gate}, N_{conn} e $N_{casc} \leq 2$).

5.4 Implicanti e implicanti primi

Viene detto implicante, un termine di n ingressi che assume il valore 1 solo la dove la funzione vale 1 o per indifferenza. Un implicante che cessa di essere tale quando si rimuove un suo letterale viene detto implicante primo. Un implicante primo essenziale è l'unico ad assumere valore 1 per alcune configurazioni degli ingressi in cui la funzione assume valore 1 (non per indifferenza). L'espressione minima SP è la somma di implicanti primi essenziali.

esempio:

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	-
1	0	0	-
1	0	1	0
1	1	0	0
1	1	1	1

Implicanti rispetto alla TdV:

- 3 termini (mintermini): $a'b'c', a'b'c, a'bc', a'bc, ab'c', abc$
- 2 termini: $a'b', a'b, a'c, a'c', b'c', bc$
- 1 termine: a'

a	b	c	z	implicanti primi attivi
0	0	0	1	$a', b'c'$
0	0	1	1	a'
0	1	0	1	a'
0	1	1	-	a', bc
1	0	0	-	$b'c'$
1	0	1	0	
1	1	0	0	
1	1	1	1	bc

Implicanti primi: a', bc

$$F(a,b,c) = a' + bc$$

5.5 Implicati e implicati primi

Viene detto implicato, un termine di n ingressi che assume il valore 0 solo la dove la funzione vale 0 o per indifferenza. Un implicato che cessa di essere tale quando si rimuove un suo letterale viene detto implicato primo. Un implicato primo essenziale è l'unico ad assumere valore 0 per alcune configurazioni degli ingressi in cui la funzione assume valore 0 (non per indifferenza). L'espressione minima PS è il prodotto degli implicati primi essenziali.

esempio:

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	-
1	0	0	-
1	0	1	0
1	1	0	0
1	1	1	1

Implicati $a + b' + c', a' + b + c, a' + b + c', a' + b' + c, a' + c, a' + b$

Implicati primi $a' + c, a' + b, a + b' + c'$

Implicati primi essenziali $a' + c, a' + b$

$$F(a,b,c) = (a' + c)(a' + b)$$

6 Mappe di Karnaugh

La mappa di Karnaugh è una rappresentazione bidimensionale della tabella della verità di una funzione di 2,3,4 variabili, i cui valori sono elencati sui bordi in maniera tale che due configurazioni consecutive differiscano per il valore di un solo bit (codice di Gray).

esempio, Uscita R del Full Adder:

a	b	r	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

La mappa di Karnaugh associata è:

$a\backslash br$	00	01	11	10
0	0	0	1	0
1	1	1	1	1

Lo scopo della mappa è quello di identificare graficamente, configurazioni adiacenti, ovvero che differiscono per un solo bit, aventi il medesimo valore di uscita (utile per trovare a sua volta implicanti e implicati primi essenziali).

6.1 Adiacenza tra celle

Una coppia di celle adiacenti su una mappa di Karnaugh è detta tale quando differiscono per un solo bit. Il numero di celle adiacenti corrisponde al numero di ingressi.

esempio, O è la cella scelta, X sono le celle considerate adiacenti ad O:

- 2 ingressi:

$a\backslash b$	0	1
0	O	X
1	X	

- 3 ingressi:

$a\backslash bc$	00	01	11	10
0			X	
1		X	O	X

- 4 ingressi:

$ab\backslash cd$	00	01	11	10
00	X		X	O
01				X
11				
10				X

- 5 ingressi:
  $a = 0$

$bc\backslash de$	00	01	11	10
00				
01		X		
11	X	O	X	
10		X		

$a = 1$

$ab\backslash cd$	00	01	11	10
00				
01				
11		X		
10				

6.2 Manipolazione algebrica per via grafica

Due termini di una espressione canonica (SP o PS) corrispondenti a configurazioni diverse che individuano celle adiacenti, equivalgono ad un unico termine con un letterale in meno (quello che cambia valore).

esempio:

$ab\backslash cd$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	1
10	X	X	X	1

L'espressione canonica SP $\cdots + abcd' + ab'cd' + \ldots$ diventa: $\cdots + acd' + \ldots.$

$ab\backslash cd$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	0	0	X
10	X	X	X	X

L'espressione canonica PS $\cdots * (a'b'cd') * (a'b'c'd') * \ldots$ diventa: $\cdots + a'b'd' + \ldots.$

$ab\backslash cd$	00	01	11	10
00	X	X	X	X
01	X	1	1	X
11	X	1	1	X
10	X	X	X	X

L'espressione canonica SP $\cdots + a'bc'd + a'bcd + abc'd + abcd + \ldots$ diventa: $\cdots + a'bd + abd + \ldots,$ che a sua volta diventa: $\cdots + bd + \ldots.$

6.3 Raggruppamenti rettangolari

Un raggruppamento rettangolare (RR) di ordine p è un insieme di 2^p celle appartenenti ad una mappa, all'interno del quale ogni cella ha esattamente p celle adiacenti. Un RR di ordine p costituito da celle contenenti valore 1 o una condizione di indifferenza, individua un implicante della funzione. Nel prodotto compaiono solo le $(n-p)$ variabili che rimangono costanti nelle coordinate del RR, in forma vera se valgono 1, in forma complementata se valgono 0. Lo stesso vale se il RR è costituito da celle con valore 0. Esso costituisce un implicato della funzione e nel prodotto compariranno $(n-p)$ variabili in forma vera se valgono 0 e in forma complementata se valgono 1. Se un RR non è interamente incluso in un'altro RR di ordine superiore, allora esso individua un implicante/implicato primo.

esempio:

$ab\backslash cd$	00	01	11	10
00	-	1	1	-
01	-	1	1	-
11	-	1	1	-
10	0	1	1	-

- il RR dove bd assumono valore 1 non è un implicante primo
- il RR dove d assume valore 1 è un implicante primo

$ab\backslash cd$	00	01	11	10
00	0	x	x	0
01	0	x	x	0
11	0	x	x	0
10	0	x	1	0

- il RR $c + d$ non è un implicato primo
- il RR $c' + d$ non è un implicato primo
- il RR d è un implicato primo

6.4 Copertura minima

La copertura di una funzione su una mappa è l'insieme di RR che coprono tutte le celle di valore 1 o 0. Una copertura può individuare una possibile struttura per un'espressione (SP per gli 1, PS per gli 0). Una copertura minima è una copertura costituita dal numero minore possibile di RR di dimensione massima. Questa corrisponde alla espressione minima.

esempio:

$ab\backslash cd$	00	01	11	10
00	1	1	0	0
01	1	-	0	-
11	1	1	0	1
10	1	1	0	1

La copertura $c' + acd'$ è valida ma non è una copertura minima. Infatti se prendiamo in considerazione la copertura $c' + ad'$, essa è una copertura valida con lo stesso numero di RR ma entrambi di dimensione massima.

6.5 Analisi di un circuito con le mappe

1. Si scrive l'espressione associata allo schema dato e, se necessario, la si manipola fino ad ottenere una espressione SP o PS.
2. Si predispone una mappa di dimensioni adeguate e si tracciano sulla mappa i RR che corrispondono ai termini dell'espressione.
3. Nelle celle coperte dagli RR, si indica valore 1 se l'espressione analizzata è SP, 0 in caso sia una espressione PS. Nelle celle non coperte si mette il valore opposto (0 con le SP, 1 con le PS).

7 Algebre binarie

Si definisce algebra binaria, un sistema matematico formato da un insieme di operatori definiti assiomaticamente ed atti a descrivere con una espressione ogni possibile funzione di variabili binarie. Oggi è possibile rappresentare ogni funzione binaria con soltanto operatori NAND ($\uparrow$) o NOR ($\downarrow$).

7.1 NAND

NOT $x' = x \uparrow x$

AND $xy = (x \uparrow y)'$

OR $x + y = x' \uparrow y'$

7.2 NOR

NOT $x' = x \downarrow x$

AND $x + y = (x \downarrow y)'$

OR $xy = x' \downarrow y'$

7.3 Sintesi con NAND

La sintesi "a NAND" può essere effettuata trasformando un'espressione SP (SP, SPS, SPSP, ...) che descrive la funzione assegnata, in una nuova espressione contenente esclusivamente operatori "↑".

1. Si parte dall'espressione SP, SPS. SPSP, ...
2. Si sostituisce il simbolo "↑" ad ogni simbolo "∗"
3. Si sostituisce il simbolo "↑" ad ogni simbolo "+" e si completano le variabili singole affiancate a tale simbolo senza aggiungere o togliere parentesi
4. (Opzionale) Se compaiono segnali di ingresso in forma negata e non sono disponibili, si sostituiscono con il NAND del segnale in forma vera

7.4 Sintesi con NOR

1. Si parte da un'espressione PS, PSP, PSPS, ...
2. Si sostituisce il simbolo "↓" ad ogni simbolo "+"
3. Si sostituisce il simbolo "↓" ad ogni simbolo "∗" e si completano le variabili singole affiancate a tale simbolo senza aggiungere o togliere parentesi
4. (Opzionale) Se compaiono segnali di ingresso in forma negata e non sono disponibili, li sostituisco con il NOR del segnale in forma vera

8 Decoder

esempio:

Una rete combinatoria che converte un numero binario a 3 bit a codice 1 su 8 (2^3). La convenzione per indicare gli ingressi è usare le lettere maiuscole a partire dalla A, la quale rappresenta l'ingresso con la cifra meno significativa. Ogni uscita ha solo una configurazione per cui essa vale "1", ovvero quella che codifica il numero dell'uscita stessa. Data la proposizione precedente, è chiaro che la sintesi canonica minima SP prevede quell'unico mintermine nell'espressione.

8.1 Decoder (DEC) generico $n : 2^n$

Una rete che transcodifica un numero binario a n bit a codice 1 su 2^n. Gli n ingressi vengono anche indicati come indirizzi (A addresses), con A_0 indirizzo di minor peso. L'indice i dell'uscita U_i attivata è pari al numero rappresentato dalla configurazione binaria degli stessi ingressi.

$$i = A_{n-1} * 2^{n-1} + \cdots + A_1 * 2^1 + A_0 * 2^0$$

A_{n-1}	...	A_2	A_1	A_0
0	...	0	0	0
0	...	0	0	1
0	...	0	1	0
0	...	0	1	1
...	...	...	...	...
1	...	1	1	1

U_{2^n-1}	...	U_2	U_1	U_0
0	...	0	0	1
0	...	0	1	0
0	...	1	0	0
0	...	0	0	0
...	...	...	...	...
1	...	0	0	0

8.2 Effetto di carico: fan-out

Il fan-out è il numero massimo di gate che possono collegarsi all'ingresso di un singolo gate. La tecnologia di oggi ci permette di avere numero di fan-out ≥ 10, ma rimane comunque un fattore importante da considerare.

8.3 MSI e LSI

MSI Medium Scale Integration

LSI Large Scale Integration

Esistono DEC in forma integrata con 2, 3, o 4 bit di indirizzo. Questi ricadono tutti sotto l'ala della MSI.

9 Multiplexer

esempio:

Un multiplexer o selettore a 2 vie. Un selettore è l'equivalente, in hardware, di un "if". Permette di decidere tramite un segnale A, quale tra le due vie d'ingresso, I_0 o I_1, sarà replicata dall'uscita. La tabella della verità e l'equazione SP risultante sono le seguenti:

$A \backslash I_1, I_0$	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$$Z = A'I_0 + AI_1$$

9.1 Multiplexer (MUX) generico

Il concetto di selettore può essere generalizzato a n bit di indirizzo. Gli n bit di indirizzo selezionano uno tra i 2^n ingressi detti "vie" o anche "bit di programmazione", a seconda dell'uso che si fa del MUX. Al crescere d n cresce esponenzialmente il numero delle vie. L'ingresso replicato sull'uscita si determina nel seguente modo:

$$i = A_{n-1}2^{n-1} + \cdots + A_1 2^1 + A_0 2^0$$

9.2 Teorema dell'espansione di Shannon

Data una funzione F di n variabili binarie, vale la relazione:

$$F(x_1, \ldots, x_i, \ldots, x_n) = x_i F(x_1, \ldots, 1, \ldots, x_n) + \overline{x_i} F(x_1, \ldots, 0, \ldots, x_n)$$

La relazione duale è ugualmente valida:

$$F(x_1, \ldots, x_i, \ldots, x_n) = (x_i + F(x_1, \ldots, 0, \ldots, x_n))(\overline{x_i} + F(x_1, \ldots, 1, \ldots, x_n))$$

9.3 Espressioni generali

In generale, applicando ripetutamente il teorema di espansione di Shannon, è possibile dedurre le seguenti espressioni generali:

Caso SP ogni funzione di n variabili è descritta da una espressione in cui compaiono, in somma logica, tutti i mintermini di n variabili, ciascuno in prodotto logico con il valore della funzione quando in ingresso compare la configurazione riconosciuta dal mintermine

$$F(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{2^n-1} m(i) * F(i)$$

Caso PS ogni funzione di n variabili è descritta da una espressione in cui compaiono, in prodotto logico, tutti i maxtermini di n variabili, ciascuno in somma logica con il valore della funzione quando in ingresso compare la configurazione riconosciuta dal maxtermine

$$F(x_1, x_2, \ldots, x_n) = \prod_{i=0}^{2^n-1} (M(i) + F(i))$$

9.4 Il MUX come rete programmabile

Il MUX si adatta bene a realizzare l'espressione generale nel caso SP. Con n variabili occorre un MUX a n bit di indirizzo. Il MUX viene utilizzato, in questo caso, come generatore di funzioni.

9.5 MUX in forma integrata

Esistono MUX a 2,4,8 o 16 bit di programmazione. Il circuito integrato ha un numero di pin limitati, ma gli ingressi di un MUX crescono esponenzialmente, perciò saranno disponibili in forma integrata, al massimo MUX a 16 bit di programmazione.

10 Reti programmabili

10.1 LSI e VLSI

Rispettivamente Large Scale Integration e Very Large Scale Integration. Appartengono a queste categorie i chip composti da milioni di gate. Sono molto costosi da produrre e per ammortizzare le spese necessitano di essere prodotti e, soprattutto, venduti in massa.

10.2 Reti combinatorie programmabili

Una rete combinatoria in grado di presentare diverse relazioni ingresso/uscita selezionabili mediante l'attribuzione di una determinata configurazione di valori ad un gruppo di segnali interni, detti bit di programmazione.

10.3 Tabella della verità come memoria non modificabile

Le reti programmabili possono essere usate per realizzare reti combinatorie a partire da memorie immutabili. Una memoria è un circuito che realizza fisicamente un array, ovvero una lista di dati indicizzabile con un intero (indirizzo). Una tabella della verità può essere interpretata come una memoria non modificabile, ovvero una lista di 2^n dati immutabili, tra cui gli n ingressi mi permettono di scegliere.

10.3.1 Memoria di sola lettura, ROM

Una Read Only Memory è un circuito che realizza una memoria a sola lettura, ovvero che contiene ad ogni indirizzo un dato fissato. Esse sono dunque delle reti programmabili in grado di realizzare qualsiasi funzione di n variabili. Le ROM sono programmabili una sola volta (OTP, one time programmable) e sono quindi memorie di sola lettura non volatili.

10.3.2 Memorie non volatili, a sola lettura e programmabili

ROM programmabile una volta, soltanto dal costruttore, fuori dal circuito

PROM programmabile una volta, dall'utente, fuori dal circuito

EPROM programmabile più volte, dall'utente, fuori dal circuito

EEPROM / FLASH programmabile più volte, dall'utente, dentro il circuito

10.3.3 CPLD e FPGA

Le EEPROM sono un esempio di CLB, Configurable Logic Block. Cambiando i bit di programmazione possiamo sintetizzare funzioni diverse. La loro evoluzione ha portato a rendere disponibili delle versioni più avanzate, come i Complex PLD e gli FPGA, Field Programmable Gate Array. In un FPGA, anche le connessioni tra CLB sono programmabili.

10.4 Hardware description language, HDL

I circuiti programmabili e le connessioni in CPLD e FPGA sono configurati a partire da file scritti in linguaggi di descrizione hardware. Sono dei linguaggi di programmazione a tutti gli effetti, specializzati per gestire la concorrenza intrinseca a livello hardware e la nozione di tempo. I progetti sono poi passati al sintetizzatore, che realizza una sintesi mappabile sui gate, le altre risorse e le connessioni disponibili in uno specifico FPGA. I due principali HDL sono:

- VHDL
- Verilog

11 Aritmetica binaria

11.1 Numeri e aritmetica binaria senza segno

Come abbiamo visto, un numero senza segno $\mathbb{N}$ in una macchina digitale è rappresentato in notazione posizionale in base 2. Altra differenza fondamentale con la rappresentazione a cui siamo abituati su carta, è che il numero di cifre disponibili è fisso e finito: l'hardware che uso stabilisce quanti segnali e quindi bit ho a disposizione per rappresentarlo. Dati n bit, i numeri senza segno rappresentabili sono compresi tra 0 e $2^n - 1$.

La somma di due numeri a n bit può non essere rappresentabile con il numero di bit disponibile (overflow). La condizione di overflow è segnalata dall ultimo riporto (carry out).

11.2 Adder

Per definire una rete combinatoria che realizza la somma di due numeri A e B a n bit, avrei teoricamente bisogno di $2n$ ingressi. Se si osserva il problema, si scopre che in realtà dobbiamo ripetere la stessa operazione per ogni singolo bit. Usando un approccio modulare al problema, possiamo definire una rete combinatoria che dati in ingresso 3 bit (a_i, b_i, r_i) restituisce 2 bit in output, la somma s_i e il riporto r_{i+1}.

La rete che realizza la somma di 2 bit con il bit di riporto in ingresso viene detto Full adder. Quando l'adder ha soltanto 2 bit di ingresso, si dice Half adder.

Half adder è realizzabile con un EX-OR per la somma s_i e un AND per il riporto r_{i+1}.

Full adder è realizzabile con due Half adder e un OR per il riporto.

Per realizzare un Adder a n bit, è sufficiente disporre in serie un Half adder e $n-1$ Full adder. Il riporto dell'ultimo Full adder rappresenta il bit di carry out, che indica la validità del risultato.

11.3 Complemento a 1

Il complemento a $\beta - 1$ di un numero in base β a n cifre è il risultato della seguente operazione:

$$(\beta^n - 1) - A$$

esempio:

- complemento base 9 di 001234_{10}: $(10^6 - 1) - 001234 = 999999 - 001234 = 998765$
- complemento base 1 di 001110_2: $(2^6 - 1) - 001110 = 111111 - 001110 = 110001$

Nel caso del sistema binario, il complemento a 1 equivale all'operazione NOT applicata ad ogni cifra del numero.

11.4 Complemento a 2

Il complemento a β di un numero in base β a n cifre è il risultato della seguente operazione:

$$\beta^n - A$$

Nel caso del sistema binario, il complemento a 2 può essere calcolato anche nel seguente modo:

$$NOT(A) + 1$$

La rappresentazione di un numero in complemento a 2 serve a:

- Implementare la sottrazione con somma
- Rappresentare i numeri con segno quando sono negativi

11.5 Sottrazione tra numeri senza segno

Dati due numeri binari senza segno A e B a n bit, con A maggiore di B, si può calcolare $A - B$ sommando A al complemento a 2 di B e ignorando il carry out uguale ad 1.

$$A - B \Rightarrow A + (2^n - B) = (A - B) + 2^n$$

Se $(A - B) < 0$, il risultato non è rappresentabile come numero senza segno. Nella sottrazione, se il bit di carry out vale 1, l'operazione è valida. Il carry out = 0 indica un overflow.

11.6 Numeri con segno

Esistono molteplici rappresentazioni per i numeri con segno. La più semplice, ma non utilizzata nelle reti logiche, è la rappresentazione in segno e valore assoluto:

- Dato un numero $A_2 = a_{n-1}a_{n-2}\dots a_0$, la cifra a_{n-1} rappresenta il segno $(0:+,1:-)$ e il resto delle cifre rappresentano il valore assoluto del numero.
- I numeri rappresentabili equivalgono a: $-(2^{n-1}-1) \leq A \leq 2^{n-1}-1$
- Due modi per rappresentare lo 0

La rappresentazione usata è quella in complemento a 2:

- Dato un numero $A_2 = a_{n-1}a_{n-2}\dots a_0$, si ottiene:
  - se A è positivo, come per segno e valore assoluto
  - se A è negativo, come per il complemento a 2 del numero positivo $-A$, bit di segno incluso
- I numeri rappresentabili equivalgono a: $-2^{n-1} \leq A \leq 2^{n-1}-1$
- Solo un modo per rappresentare lo 0
- Per calcolare il valore di un numero A a complemento a 2 con n cifre, devo applicare la seguente formula: $A_{2^{n-1}} * -2^{n-1} + A_{2^{n-2}} * 2^{n-2} + A_{2^{n-3}} * 2^{n-3} + \dots + A_0 * 2^0$

11.7 Aritmetica con segno

Anche per i numeri con segno, il risultato può non essere rappresentabile con il numero di bit disponibile (overflow). La condizione di overflow, con i numeri a complemento a 2, non viene più segnalata dal bit di carry out. Per verificare la condizione di overflow nelle operazioni tra numeri con segno, bisogna verificare se il bit di carry out è diverso dal bit di riporto r^{n-1}.

11.8 Ritardi di un Adder

Il limite principale di un Adder realizzato come disposizione in serie di Full adder, è la lentezza dovuta alla catena dei riporti. Questa realizzazione viene detta ripple-carry adder. Ogni Full adder impiega $2\tau_p$ per portare a regime il riporto in uscita da quando è a regime il riporto in ingresso. Se eseguo operazioni tra dati di 64 bit, nel caso peggiore ottengo il risultato aggiornato dopo $128\tau_p$.

11.9 Carry lookahead

Una rete più veloce, ma più costosa dal punto di vista del numero di gate impiegati, si ottiene sintetizzando un circuito di carry lookahead, che elabori gli output del primo Half adder. Definiamo per ogni Full adder, i due segnali, corrispondenti alle uscite del primo Half adder:

Carry propagate $p_i(a_i \oplus b_i)$

Carry generate $g_i(a_i b_i)$

Questi segnali sono a regime τ_p per tutti i Full adder. Le uscite del Full adder possono essere espresse in funzione di questi due segnali come:

- $s_i = r_i \oplus p_i$
- $r_{i+1} = r_i p_i + g_i$

In generale, il riporto $i+1$ è dato da:

$$r_{i+1} = \sum_{j=0}^{i} [g_j \prod_{k=j+1}^{i} p_k] + r_0 \prod_{j=0}^{i} p_j$$

ovvero è calcolabile a partire da segnali a regime τ_p da quando sono disponibili gli ingressi dell'adder (uscite dell'Half adder), con un circuito SP. Ogni riporto è disponibile dopo $3\tau_p$, indipendentemente dal numero di bit dell'adder. Serve poi un XOR tra il carry propagate e il riporto r_i per calcolare i bit di uscita s_i, che quindi sono pronti dopo $4\tau_p$.

11.10 Arithmetic Logic Unit (ALU)

La codifica dei numeri a complemento a 2 è alla base della costruzione delle ALU. Le ALU sono delle reti combinatorie in grado di eseguire diverse operazioni (somma, sottrazione, complemento a 2, scorrimento a destra e a sinistra, AND, OR, XOR, ...) su due operandi in base al codice operativo (op-code) in ingresso. Oltre al risultato stesso, diversi bit detti flags indicano il verificarsi di diversi casi speciali durante il calcolo del risultato.

12 Reti sequenziali asincrone

12.1 Stato

Lo stato è un informazione che la rete memorizza al suo interno. Gli stati sono tutti i "riassunti" utili della storia passata degli ingressi. Utile indica la proprietà degli stati che gli permette di cambiare il comportamento futuro della rete. Oltre a decidere il valore di uscita, la rete sequenziale deve decidere se "vale la pena ricordare" qualsiasi evento che influenzerà il comportamento di essa. La rete, quando memorizza un evento, cambia stato. La rete calcola perciò lo stato futuro, ovvero il nuovo stato che riassume la storia tenendo in considerazione anche l'ultimo ingresso visto.

12.2 Reti sequenziali asincrone (RSA)

Esistono due tipi di reti sequenziali:

- Asincrone
- Sincrone

In una rete asincrona, lo stato futuro sovrascrive lo stato presente il momento in cui viene calcolato. Segue una sequenza di lavoro tipica per una rete sequenziale asincrona:

1. Si parte da una situazione stabile, dove lo stato presente è uguale allo stato futuro
2. Cambia un ingresso
3. Dopo un tempo pari al ritardo della rete, sono disponibili un nuovo stato futuro e una nuova uscita.
4. Lo stato futuro diventa lo stato presente, sovrascrivendolo nella memoria della rete.
5. Nuova situazione stabile, con uno stato presente uguale allo stato futuro, ed un ingresso costante.

La rete riinizierà il ciclo ad ogni cambio di ingressi.

12.3 Codifica degli stati

Gli stati vengono rappresentati attraverso segnali binari organizzati in un codice. In una rete sequenziale con K stati, ci saranno almeno $k = [\log_2 K]bit$ per rappresentare lo stato. Gli ingressi di stato presente e le uscite di stato futuro sono ognuno rappresentati da k segnali binari.

Una notazione comune per questi segnali è $y_{k-1}, \ldots, y_1, y_0$ per lo stato presente, e $Y_{k-1}, \ldots, Y_1, Y_0$ per lo stato futuro. In una rete sequenziale asincrona si hanno:

m segnali di uscita

k segnali di stato futuro

n segnali di ingresso

k segnali di stato presente

Una volta che lo stato presente diventa disponibile, le uscite dipendono solo dagli ingressi attuali e dallo stato presente, perciò le uscite sono funzioni combinatorie di questi segnali. Una RSA è perciò composta da $m + k$ reti combinatorie, ognuna con $n + k$ ingressi. Una convenzione comune è indicare con F le funzioni che calcolano i bit di uscita e con G le funzioni che calcolano i bit di stato futuro.

12.4 Retroazione

Data una rete combinatoria, provare a collegare l'uscita di essa in retroazione su uno dei propri ingressi, analizzando il tutto soltanto con usando le pure relazioni matematiche tra i gate, risulterebbe impossibile. Però i gate non sono istantanei, ogni gate reale ha un ritardo di propagazione, indicato dal tempo τ_p. Il ritardo complessivo di una rete combinatoria nel caso peggiore è definito come:

$$T_p = N_{\text{casc}} * \tau_p$$

Quando analizziamo una rete, possiamo immaginare un segnale ideale, che cambia istantaneamente al variare degli ingressi, insieme al segnale reale, che cambia dopo il ritardo complessivo della rete. Grazie ai ritardi delle reti, è possibile connettere reti combinatorie in retroazione diretta, garantendo che la rete calcoli correttamente la nuova uscita.

12.5 Struttura generale di una RSA

Ogni rete combinatoria con anelli di retroazione diretta è in realtà una RSA. La rete ha k bit di stato se ci sono k segnali di retroazione. I ritardi agiscono della rete combinatoria G agiscono come memoria. Lo stato futuro in se non è osservabile. Lo è soltanto quando esso diventa stato presente.

12.6 Finite State Machine (FSM)

Una RSA è un caso particolare di Automa a stati finiti M, anche rappresentabile come sistema matematico.

$$FSM = \{I, U, S, F, G\}$$

Formato da:

I alfabeto di ingresso

U alfabeto di uscita

S insieme degli stati

$F : S \times I \to U$ funzione di uscita,

$G : S \times I \to S$ funzione di aggiornamento dello stato interno

Due tipologie:

- Automa di Mealy
- Automa di Moore (un caso particolare dell'automa di Mealy, dove l'uscita dipende soltanto dallo stato presente)

12.7 Comportamento e struttura

Per analizzare una rete combinatoria, si usavano:

- Tabella della verità (TdV) → Comportamento della rete
- Espressione o schema logico → Struttura della rete

Per analizzare una rete sequenziali asincrona invece si usano:

- Grafo degli stati e Tabella di flusso → Comportamento della rete
- Espressione o schema logico → Struttura della rete (per ognuna delle $m + k$ reti combinatorie che la compongono)

12.8 Grafo degli stati

Un grafo ad archi orientati dove:

Nodo rappresenta uno stato presente

Arco rappresenta una trasformazione, da stato presente a stato futuro, in corrispondenza di una o più configurazioni di ingressi

Da ogni nodo devono uscire tanti rami quante configurazioni possibili di ingressi. Uno stato stabile è un nodo con un arco che ritorna sul nodo da cui parte.

12.8.1 Grafo per un'automa di Mealy

Nel caso dell'automa di Mealy, si indica il valore dell'uscita su ciascuna transazione. Per rendere il grafo più leggibile ogni nodo verrà indicato con una lettera diversa (A,B,C,D,...). Quando il valore d'uscita cambia tra due stati, bisogna contrassegnare l'uscita nella transazione con il simbolo di indifferenza. Per progettare una RSA che opera con continuità, bisogna assicurarsi che:

- esista sempre almeno un percorso per passare da un nodo arbitrariamente scelto ad un altro (grafo strettamente connesso)
- non esistano stati irrangiungibili o assorbenti (dotati di solo frecce entranti)

12.8.2 Grafo per un'automa di Moore

Nel caso dell'automa di Mealy, si indica il valore dell'uscita sugli stati invece che sulle transazioni.

12.9 Stato iniziale e reset

In una rete sequenziale asincrona, l'uscita dipende dagli ingressi e dallo stato presente. Per assicurarsi un comportamento deterministico, la certezza che la rete parta sempre dallo stesso stato, si introduce un segnale di reset, che assume valore 1 per pochi istanti all'accensione, e rimane 0 per il resto del tempo.

12.10 Tabella di flusso

La tabella di flusso è un modo alternativo per descrivere il comportamento di una RSA.

12.10.1 Tabella per un'automa di Mealy

Nel caso di Mealy, in ogni casella vengono riportati i valori delle funzioni F e G per il rispettivo ingresso dello stato presente. Ogni riga deve contenere almeno una condizione di stabilità. Le condizioni di instabilitità devono indicare uno stato futuro stabile.

	00	01	11	10
A	A,0	A,0	B,0	A,0
B	A,0	C,0	B,0	A,0
C	A,0	C,0	D,-	A,0
D	A,-	A,0	D,1	A,-

12.10.2 Tabella per un'automa di Moore

Poichè l'uscita dipende solo dallo stato presente, è costante su ogni riga. Esiste perciò una colonna aggiuntiva che riporta il valore dell'uscita per ogni stato.

	00	01	11	10	z
A	A,0	A,0	B,0	A,0	0
B	A,0	C,0	B,0	A,0	0
C	A,0	C,0	D,-	A,0	0
D	A,-	A,0	D,1	A,-	1

esempio:

	00	01	11	10
A	A,0	A,0	B,0	A,0
B	A,0	C,0	B,0	A,0
C	A,0	C,0	D,-	A,0
D	A,-	A,0	D,1	C,-

1. Partiamo dallo stato presente stabile B con gli ingressi $y_1y_0 = 11$
2. Appena cambiamo ingressi $y_1y_0 = 10$, lo stato immediatamente futuro è D
3. Però lo stato D con ingressi $y_1y_0 = 10$ non è stabile e il seguente stato futuro è C
4. Anche lo stato C con ingressi $y_1y_0 = 10$ non è stabile e il seguente stato futuro è A
5. Lo stato A con ingressi $y_1y_0 = 10$ è stabile, perciò il nostro spostamento è terminato

13 Alee

13.1 Comportamento a regime e in transitorio

Per capire i vincoli di corretto impiego di una RSA, dobbiamo approfondire come risponde ai cambiamenti degli ingressi una rete combinatoria. Esistono due tipi di comportamenti:

In transitorio che descrive la fase successiva al cambiamento dei segnali di ingresso di una rete combinatoria, caratterizzato dal fatto che l'uscita di essa non ha ancora presentato il valore previsto per la nuova configurazione

A regime ovvero il comportamento al termine della fase transitoria

13.2 Ritardo puro

Il caso più semplice è di comportamento transitorio, la dove al variare di un ingresso, che provoca la variazione di un'uscita di una rete combinatoria, la rete mantiene il precedente valore dell'uscita per il tempo di propagazione T_p della rete combinatoria, prima di sostituirla col nuovo valore. Questo transitorio è ineliminabile, ma non dannoso.

13.3 Alea dinamica

L'uscita, in regime di transitorio, varia più volte prima di assestarsi sul nuovo valore. Questo malfunzionamento è causato dai diversi ritardi di propagazione dei percorsi che agiscono sull'uscita Z. Una rete combinatoria descritta da espressioni SP o PS non presenta mai alee dinamiche.

13.4 Alea statica

L'uscita dovrebbe rimanere costante, ma subisce, in regime di transitorio, una temporanea variazione.

esempio:

I due ingressi di un gate OR cambiano valore contemporaneamente. Nel caso ideale, l'uscita rimane costante, ma per motivazioni legate al contesto (la disposizione del circuito, le caratteristiche fisiche dei gate, ...), si verifica una minuscola variazione indesiderata dell'uscita. Per evitare alee statiche in una rete combinatoria, è necessario, ma non sufficiente, variare un solo ingresso alla volta. Un esempio di non sufficienza nell'evitare le alee statiche è il caso del multiplexer. Usando l'espressione minima SP per un multiplexer, il cambio di A, per colpa del disallineamento dei componenti, può provocare alea statica (nello specifico, a causa del ritardo dei NOT). Per evitare a priori l'insorgere di alee statiche, quando si effettua la sintesi con le mappe di Karnaugh, si deve scegliere un'espressione ridondante (quindi non minima) che racchiuda in uno stesso raggruppamento rettangolare ogni coppia di 1 o 0 adiacenti.

$A\backslash I_1I_0$	00	01	11	10
0	0	1	1	0
1	0	0	1	1

- espressione SP minima: $A'I_0 + AI_1$
- espressione SP priva di alee: $A'I_0 + AI_1 + I_1I_0$

$c\backslash ab$	00	01	11	10
0	0	1	0	0
1	0	1	1	1

- espressione SP minima: $(a+b)(a'+c)$
- espressione SP priva di alee: $(a+b)(a'+c)(b+c)$

$cd\backslash ab$	00	01	11	10
00	0	1	0	0
01	0	1	1	1
11	0	1	0	0
10	0	1	1	1

- espressione SP minima: $a'd + ad' + ab + c'd$
- espressione SP priva di alee: $a'd + ad' + ab + c'd + bd + ac'$

14 Funzionamento corretto di RSA

14.1 Regole di corretto impiego

Ci sono 4 regole per ottenere una RSA a funzionamento corretto.

1. Durata degli ingressi
2. Cambiamento dei bit di ingresso
3. Alee statiche
4. Codifica degli stati

14.1.1 Durata degli ingressi

Esiste, anche per le RSA, un limite superiore alla velocità di funzionamento della rete. Per calcolare il tempo minimo di durata di una configurazione binaria in ingresso, devo considerare il ritardo massimo tra $T_{pG_0}, \ldots, T_{pG_{k-1}}$ che viene indicato con T_{pG}. In generale d'ingresso deve permanere per almeno $2T_{pG}$: la prima volta per calcolare lo stato futuro a partire dal vecchio stato presente, la seconda per renderlo stabile. Se sono presenti transizioni multiple e la più lunga prevede t transizioni, il tempo sarà $(1+t)T_{pG}$, ovvero T_{pG} moltiplicato per il numero necessario di transizioni per rendere lo stato futuro stabile.

14.1.2 Cambiamento dei bit di ingresso

Il progetto e l'uso di una RSA devono prevedere che cambi solo un bit di ingresso alla volta.

esempio:

	00	01	11	10	z
A	A	C	B	D	0
B	A	D	B	C	0
C	A	C	C	C	1
D	A	D	D	D	0

Se si parte dallo stato A e si riceve la sequenza $00-11-10$ (sapendo che non possiamo cambiare più di 1 bit alla volta):

caso $00-01-11-10$ la rete raggiunge lo stato C ma troppo velocemente

caso $00-10-11-10$ la rete finisce nello stato D invece che lo stato C

Cambiare più di bit di ingresso alla volta non è soltanto fisicamente impossibile, ma potrebbe causare un aumento nell'imprevedibilità della rete. Per evidenziare questa regola, si deve modificare la tabella di flusso, eliminando le configurazioni non adiacenti a quelle per cui lo stato è stabile.

esempio:

	00	01	11	10
A	A,0	A,0	B,0	A,0
B	A,0	C,0	B,0	A,0
C	A,0	C,0	D,-	A,0
D	A,-	A,-	D,1	A,-

$\Downarrow$

	00	01	11	10
A	A,0	A,0	B,0	A,0
B	-,-	C,0	B,0	A,0
C	A,0	C,0	D,-	-,-
D	-,-	A,-	D,1	A,-

14.1.3 Alee statiche

L'alea statica può non essere un problema per reti di calcolo non all interno di anelli combinatori in retroazioni, però può introdurre stati spuri all'interno della sequenza di una RSA. Per eliminare a priori le alee statiche, è necessario usare la tecnica di copertura ridondante, per ottenere le espressioni SP e PS durante la sintesi.

14.1.4 Codifica degli stati

Le configurazioni binarie associate ad ogni coppia (stato presente e stato futuro) devono essere adiacenti. Se applicato a tutte le coppie, questo principio richiederebbe un codice molto ridondante, che utilizzerebbe più bit necessario. Fortunatamente si tratta di un vincolo spesso eccessivo: si può garantire il corretto funzionamento della rete anche in presenza di variazione contemporanea di più di un bit alla volta, per alcune configurazioni di ingresso. Si parla di corse critiche e corse non critiche. Per passare dalla tabella di flusso alla tabella delle transizioni, è necessario scegliere una codifica degli stati. Non tutte le codifiche sono valide.

esempio:

	00	01	11	10
A	A	B	A	A
B	A	B	D	B
C	A	C	C	C
D	A	C	D	D

$\Downarrow$

	00	01	11	10
A=00	00	01	00	00
B=01	00	01	10	01
C=11	00	11	11	11
D=10	00	11	10	10

Questa configurazione non è valida poiché le celle $B, 11$ e $C, 00$ non rispettano il vincolo di adiacenza tra stato presente e stato futuro.

14.2 Determinare se una configurazione è valida o meno

I segnali in retroazione per cui è stata prevista una modifica contemporanea di valore si trovano in una situazione di corsa. Una corsa può essere:

Critica se si possono raggiungere stabilità diverse

Non critica se passa da uno o più stati intermedi prima di raggiungere quello stabile

Le transizioni multiple non sono un problema affinché:

- L'uscita non presenti andamenti diversi dal comportamento voluto durante la transizione
- Gli ingressi rimangono stabili fino al raggiungimento dello stato stabile

esempio:

Usando l'esempio di prima, se dallo stato $C, 01$ cambia l'ingresso in 00 lo stato dovrà passare per due possibili stati intermedi:

- $11 \to 01 \to 00$
- $11 \to 10 \to 00$

In entrambe i casi il comportamento non varia l'ingresso rimane stabile, perciò questa situazione di corsa non è critica. Invece se dallo stato $B, 01$ cambia l'ingresso in 11, lo stato ha questi due possibili stati intermedi:

- $01 \to 00 \to \times$
- $01 \to 11 \to \times$

Il comportamento varia e in entrambe i casi porta a risultati diversi e non desiderati. Questa situazione di corsa è critica.

14.3 Prevenzione a priori delle corse critiche

La presenza di corse critiche si ha nei casi in cui una colonna della tabella di flusso presenta più di uno stato stabile. In caso di colonne con una sola stabilità, se tutti gli stati non stabili riducono allo stato stabile, si avranno corse non critiche. Si possono eliminare le situazioni di corse critiche a priori seguendo le seguenti regole:

1. Nelle colonne con una sola stabilità, si inserisce il simbolo dello stato stabile al posto di eventuali condizioni di indifferenza
2. Per le colonne con più stabilità, si traccia il grafo delle adiacenze:
   - Un nodo associato per ogni stato
   - Un ramo orientato per ogni coppia stato presente-futuro
3. Si sovrappone il grafo ad una mappa con codici di Gray su righe e colonne (come per le mappe di Karnaugh) e si verifica se è possibile assegnare configurazioni adiacenti ad ogni coppia di stati coinvolta in una transazione
4. Se è impossibile soddisfare i vincoli di adiacenza, si cerca di ridurli ricorrendo a transazioni multiple
5. Se non ci si riesce, si incrementa il numero delle variabili di stato e si ritorna al punto 3

15 Sintesi RSA

5 passaggi:

1. Individuazione del Grafo degli stati
2. Definizione della Tabella di flusso
3. Codifica degli stati e definizione della Tabella delle transizioni
4. Sintesi della rete combinatoria di uscita e stato futuro
5. Schema logico che includa anche ingresso di reset, se presente

15.1 Tabella di Flusso

2 controlli:

- In ogni riga ci deve essere almeno una condizione di stabilità
- Le situazioni di instabilità devono indicare uno stato futuro stabile nella colonna

esempio:

	0	1
A	A,0	B,0
B	C,-	B,0
C	C,1	D,1
D	A,-	D,1

15.2 Tabella delle transizioni

In ogni rete sequenziale, lo stato è rappresentato da una configurazione binaria dei bit di stato. Bisogna quindi scegliere un modo per codificare gli stati. Data la codifica, posso tradurre la tabella di flusso in tabella delle transizioni, sostituendo ad ogni stato la codifica binaria. Non tutte le codifiche producono un funzionamento corretto della rete.

esempio:

	0	1
A	A,0	B,0
B	C,-	B,0
C	C,1	D,1
D	A,-	D,1

$\Downarrow$

	0	1
0	A	B
1	D	C

$\Downarrow$

	0	1
A=00	00,0	01,0
B=01	11,-	01,0
C=11	11,1	10,1
D=10	00,-	10,1

15.3 Espressioni combinatorie

La tabella delle transizioni è in realtà una composizione di TdV combinatorie, ovvero di mappe di Karnaugh, con cui è possibile eseguire la sintesi combinatoria delle funzioni che implementano il comportamento richiesto con le metodologie già studiate.

esempio:

	0	1
A=00	00,0	01,0
B=01	11,-	01,0
C=11	11,1	10,1
D=10	00,-	10,1

$\Downarrow$

	00	01	11	10
0	0	-	1	-
1	0	0	1	1

$$Z = y1$$

	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$$Y_1 = xy_1 + x'y_0 + y_1y_0$$

	00	01	11	10
0	0	1	1	0
1	1	1	0	0

$$Y_0 = xy_1' + x'y_0 + y_1'y_0$$

16 Memorie binarie

Uno degli usi più importanti delle RSA è la realizzazione di memorie binarie, ovvero circuiti che hanno lo scopo di memorizzare il valore di un bit. Sono categorizzati come componenti primitivi, inoltre sono i principali blocchi di costruzione per le reti sincrone. Ci sono 3 tipi principali di memorie:

- Latch SR
- Latch CD
- Flip-flop D

Si differenziano per quale sequenza di ingressi porta alla scrittura di un bit, quindi quando sono sensibili a comandi di modifica dell'informazione memorizzata.

16.1 Latch SR

La più semplice memoria binaria. Questo componente ha:

2 Ingressi set S e reset R

2 Uscite Q e il suo complemento Q'

Il comportamento della rete è il seguente:

Comando	Ingresso S	Ingresso R	Uscita Q
Memorizza	0	0	Ultimo valore scritto
Scrivi 1	1	0	1
Scrivi 0	0	1	0

La coppia di ingressi 11 non può presentarsi per un corretto funzionamento della rete.

16.1.1 Sintesi del latch SR

	00	01	11	10	Q
A	A	A	-	B	0
B	B	A	-	B	1

Tabella di flusso

	00	01	11	10	Q
A=0	0	0	-	1	0
B=1	1	0	-	1	1

Tabella delle transizioni

Espressione SP $Y = S + yR'$

Espressione PS $Y = R' * (y + S)$

Sintesi combinatoria

16.1.2 Stato iniziale di un latch SR

Come ogni RSA, il bit di stato di un latch SR avrà un valore casuale all'avvio della macchina. Ci sono casi in cui, però, serve che il latch SR abbia uno stato iniziale predeterminato all'accensione, perciò esistono delle varianti del latch SR dotate di 2 ingressi aggiuntivi, normalmente attivi bassi, prioritari rispetto a S e R:

Preset PRE' quando $PRE' = 0$, il latch SR memorizza 1 indipendentemente dai valori di S e R

Clear CLR' quando $CLR' = 0$, il latch SR memorizza 0 indipendentemente dai valori di S e R

Anche per gli ingressi di preset e clear vale la regola che non possono mai essere contemporaneamente attivi. Poiché gli ingressi di inizializzazione PRE' e CLR' non devono essere usati durante la normale operatività del latch ma solo all'avvio, sono rappresentati tipicamente in alto e in basso, rispetto agli altri ingressi, situati sul lato sinistro della rete.

16.2 Metastabilità

Uno stato stabile è uno stato in cui $Q = q$ (stato futuro uguale allo stato presente). Idealmente, il bit di stato può assumere soltanto i valori H o L. In realtà i segnali che si trovano sull'anello di retroazione, come tutti i segnali di una rete reale, sono segnali analogici. Questo significa che i segnali possono assumere un valore anche leggermente superiore o inferiore ad H e L.

Per garantire un corretto funzionamento di un circuito, esso deve rimanere stabile anche a fronte di disturbi sui segnali analogici sottostanti. Se il segnale analogico da cui è derivato q si sposta di poco all'interno della banda in cui viene interpretato, la risposta Q della rete combinatoria deve riavvicinarsi al valore predefinito. Se si allontanasse, amplificando il disturbo, la rete non riuscirebbe a garantire la stabilità in una situazione reale. Questo si ottiene a livello elettronico, facendo si che la relazione ingresso-uscita analogica del circuito combinatorio f_{rc}, che poniamo in retroazione, abbia tratti a pendenza minore di 1 intorno ad H e L.

Per unire due tratti di pendenza ¡ 1, è inevitabile creare un tratto di pendenza ¿ 1 che interseca di nuovo la retta $Q = q$. Esiste quindi un'altro punto M oltre ai valori H e L in cui la rete può fermarsi. Questo punto viene detto stato meta-stabile e se il una rete combinatoria si trova in tale punto, si dice che è in metastabilità. Il problema è che non si può sapere in quale dei due stati si troverà la rete a seguito del disturbo, poiché dipende soltanto dal disturbo stesso, che è casuale.

Un latch, per esempio, può trovarsi in uno stato meta-stabile se la durata degli ingressi non rispetta il vincolo t_w.

16.3 Latch CD

Il latch CD è un'altra memoria binaria. La differenza principale da un latch SR è il modo in cui viene comandato. Il latch CD memorizza il valore del segnale D soltanto quando C ha valore 1. Chi lo comanda non dice esplicitamente cosa scrivere come nel caso del latch SR, ma quando scrivere D, pilotando opportunamente C. Quindi diventa possibile, pilotando correttamente il segnale C, condiviso da n latch, memorizzare contemporaneamente n dati binari.

Comando	Ingresso C	Ingresso D	Uscita Q
Memorizza	0	-	Ultimo valore scritto
Scrivi 1	1	1	1
Scrivi 0	1	0	0

Il campionamento del segnale D avviene soltanto quando C è alto (1). Significa che il latch CD è una memoria level-triggered. Il latch CD è utile perché la forma d'onda nell'ingresso D, pilotando opportunamente C, viene ripulita da andamenti indesiderati in transitorio o valori che non doveva essere memorizzati, quando la si va a leggere dall'uscita Q. Per questo motivo è il componente di memoria usato all'interno del registro buffer e della memoria RAM.

16.3.1 Sintesi formale

$$Y = CD + C'y + Dy$$
$$Q = y$$

16.3.2 Sintesi diretta

Si può realizzare una sintesi di un latch CD utilizzando i componenti precedentemente introdotti. Avendo il latch SR, una RSA che può memorizzare un bit, basterebbe ideare una rete che transcodifichi il codice CD in codice SR. La rete è costituita da 2 AND, corrispondenti ai due mintermini della tabella seguente:

Comando	C	D	Q	S	R
Memorizza	0	-	Ultimo valore scritto	0	0
Scrivi 1	1	1	1	1	0
Scrivi 0	1	0	0	0	1

- $S = CD$
- $R = CD'$

16.3.3 Tempi di set-up, di hold e di risposta

Avendo più gate in cascata, il transitorio del latch CD dura più di quello del latch SR. Come nel latch SR, anche il comando di campionamento ($C = 1$) deve avere una durata minima di t_w. Non rispettare questo vincolo può causare metastabilità, come nel latch SR. Esistono altri due tempi da considerare nel latch CD, rispettivamente prima e dopo il fronte di discesa di C.

Tempo di set-up il tempo di propagazione attraverso i gate

Tempo di hold il tempo necessario per innescare la retroazione

L'ingresso D deve essere costante durante $t_{su} + t_h$ per garantire il corretto funzionamento del latch. Questi vincoli evitano che la rete veda cambiamenti "simultanei" degli ingressi, proibiti nelle RSA. L'uscita in forma vera e complementata è disponibile a partire dal fronte di salita di C, dopo un tempo di risposta t_r, tipicamente diverso se Q passa da L a H o da H a L. Se quando il segnale $C = 1$, D modifica il suo valore e lo mantiene per almeno il tempo di setup, allora la stessa modifica si riscontra anche su Q, con un ritardo pari al tempo di risposta (uscita trasparente).

16.4 Flip-flop D

Una RSA con due ingressi CLK e D. L'ingresso CLK (clock) è tipicamente rappresentato da un triangolo. CLK svolge il ruolo di segnale di campionamento, D il segnale campionato. Il campionamento ha luogo quando CLK transita dal valore logico 0 al valore logico 1 (campionamento su fronti di salita, positive edge-triggered). Q, dopo un piccolo tempo dal fronte di salita, riflette l'ultimo valore campionato.

Questo ritardo è fondamentale, in quanto garantisce che quando Q cambia, la rete non è più in trasparenza. Con il flip-flop D è possibile qualunque montaggio in retroazione, senza alcun rischio di instabilità. Esistono due tipi principali di flip-flop D:

Master-slave realizzato con due latch CD in cascata con C in forma vera e negata

Edge-triggered 6 NAND invece che 8 e altri miglioramenti che vengono da scelte di progetto non intuitive, ottenute utilizzando strumenti di CAD (Computer-Aided Design)

16.4.1 Tempi di set-up, di hold e di risposta

I tempi caratteristici di un flip-flop D hanno gli stessi nomi e significati simili a quelli del latch CD, con la differenza che fanno riferimento al fronte di salita del clock.

Tempo di set-up t_{su} tempo minimo in cui D deve essere costante prima del fronte

Tempo di hold t_h tempo minimo in cui D deve essere costante dopo il fronte

Tempo di risposta t_r tempo massimo di durata del transitorio sulle uscite Q e Q' dopo il fronte

Come nel latch CD, D deve rimanere costante sia durante t_{su}, sia durante t_h. Transitorio più lungo di un latch CD, perciò viene usato solo quando serve la sua robustezza, per esempio nelle retroazioni di reti sequenziali sincrone.

16.4.2 FF-D come elemento di ritardo

Se all'ingresso CLK viene inviato un segnale periodico, il FF-D ritarda il segnale di uscita Q, rispetto al segnale D, di un tempo pari al periodo di clock T. Il clock divide il tempo in intervalli discreti tra cui vale la relazione:

$$Q^{n+1} = D^n$$

Questa è una caratteristica che lo rende fondamentale nella realizzazione di reti sequenziali sincrone.

17 Reti sequenziali sincrone

17.1 Vantaggi e limiti di una RSA

Il vantaggio principale è che la rete risponde non appena cambiano gli ingressi, grazie ai segnali in retroazione diretta. Avere i segnali in retroazione diretta, per una rete, porta ad una serie di malfunzionamenti dovuti principalmente ai ritardi con cui si aggiornano i segnali in retroazione. La difficoltà di realizzare un progetto cresce molto al crescere della complessità della funzione da realizzare. In pratica le RSA vengono usate soltanto per realizzare memorie, latch e flip-flop.

17.2 Reti sequenziali sincrone

Mentre in una rete sequenziale asincrona, l'aggiornamento dello stato avviene immediatamente, anche durante il transitorio, nelle reti sequenziali sincrone, l'aggiornamento dello stato avviene quando tutti i segnali di stato sono a regime. Questo permette alle RSS di sfruttare i seguenti vantaggi:

- Espressioni minime
- Codice d'ingresso arbitrario
- Codice di stato arbitrario
- Progettazione più semplice rispetto alle RSA

17.3 Campionamento periodico

Per leggere lo stato futuro ed aggiornare lo stato presente a regime, bisogna utilizzare la tecnica del campionamento periodico. Campionare, o memorizzare, è l'azione che ci permette di leggere lo stato futuro ed aggiornare lo stato presente, ed è una azione che va eseguita periodicamente, perciò ad intervalli regolari, sufficientemente lunghi. Grazie al campionamento, lo stato presente si aggiorna ogni T_0 unità di tempo, con il valore che lo stato futuro ha assunto alla fine del periodo, filtrando tutte le oscillazioni e le corse. Per campionare con periodo T_0, si possono utilizzare k flip-flop D (tanti quanti i bit di stato da memorizzare), il cui ingresso CLK è collegato ad un segnale periodico di clock, con periodo T_0, che posso generare da un circuito oscillatore, con frequenza $\frac{1}{T_0}$.

17.4 Struttura generale di una RSS

Ogni rete combinatoria con anelli in retroazione su cui sono inseriti FF-D comandati da un generatore di clock in comune è quindi una RSS. La rete ha k bit di stato se ci sono k segnali in retroazione. I FF-D agiscono da memoria dello stato presente e ritardo costante nel suo aggiornamento. Stato futuro e stato presente sono entrambi osservabili. Sequenza tipica di lavoro di una rete sequenziale sincrona:

1. Al fronte di salita del segnale di clock, vengono campionati i bit di stato futuro dai FF-D
2. Dopo un tempo pari al tempo di risposta dei FF-D, lo stato futuro diventa stato presente

3. L'eventuale cambiamento dello stato presente e l'eventuale cambiamento degli ingressi porta la rete a calcolare una nuova uscita e un nuovo stato futuro (gli stati spuri intermedi, se esistenti, vengono ignorati dalla rete)
4. I nuovi bit di stato futuro sono a regime con un anticipo sulla fine del periodo T_0 pari almeno al tempo di set-up dei FF-D, pronti per essere campionati alla fine del periodo e iniziare un nuovo ciclo

17.5 Temporizzazione di una RSS

Le RSS non funzionano ad inseguimento degli ingressi come le reti combinatorie o le RSA. Ogni stato presente è stabile per almeno un periodo di clock. Lo stato futuro e l'uscita possono modificarsi anche se non cambiano gli ingressi. Ciò che fa evolvere stato ed uscita di una RSS è l'arrivo di un fronte di salita del clock e nient'altro.

L'unico vincolo in una RSS è che il periodo di clock T_0 ha un valore minimo. I nuovi ingressi e lo stato presente devono avere tempo di propagarsi attraverso la funzione G. Oltre al ritardo della rete G, anche gli FF-D impongono dei vincoli su T_0:

- I bit di stato futuro $Y_{0\ldots k-1}$ devono rimanere costanti sia durante t_{su}, sia durante t_h
- I bit di stato presente $y_{0\ldots k-1}$ saranno ingressi stabili per G dopo t_r dal fronte del clock

Per garantire il corretto funzionamento della rete, il periodo T_0 deve essere maggiore della somma di:

- tempo necessario per avere ingressi a regime
- ritardo della funzione G
- tempo di set-up dei FF-D

17.6 Finite State Machine

Le RSS sono un caso particolare di automa a stati finiti M, ovvero un sistema matematico:

$$FSM = \{I, U, S, F, G\}$$

I alfabeto di ingresso

U alfabeto di uscita

S insieme degli stati

$F : S \times I \to U$ funzione di uscita

$G : S \times I \to S$ funzione di aggiornamento dello stato interno

In questo caso la memoria che mantiene il vecchio stato s fino a quando non è necessario sostituirlo con il nuovo stato s^* è realizzata dai FF-D. È possibile realizzare RSS in accordo al modello di Mealy o al modello di Moore e la scelta ha impatti più significativi di quanti non ne avesse per le RSA.

17.7 Sincronizzazione di ingressi asincroni

Il modello delle RSS assume ingressi sincroni, ovvero che variano una sola volta in ogni ciclo di clock al fronte del clock stesso. Non è possibile rendere sincrono un segnale che varia con una frequenza maggiore del clock stesso, perciò in questo caso bisogna aumentare la frequenza di clock. Se invece la frequenza con cui varia l'ingresso è minore del di quella del clock, allora la sincronizzazione è possibile e il risultato è il segnale X_{sync}, le cui variazioni rispecchiano quelle di X (il segnale asincrono), ma avvengono in corrispondenza dei fronti di salita del clock.

Una rete che realizza un segnale X_{sync} partendo da un ingresso asincrono X viene detta sincronizzatore. Il più semplice tra i sincronizzatori è un FF-D. Per diminuire la probabilità che la rete veda un segnale in metastabilità, il sincronizzatore viene spesso realizzato con 2 o più FF-D in cascata.

17.8 Segnali $A\_$

Un segnale di ingresso asincrono viene indicato dal suo nome preceduto da $A\_$.

esempio:

- A_RESET e A_RES $\implies$ Segnali di reset asincroni
- $RESET$ e RES $\implies$ Segnali di reset sincroni

17.9 Sintesi di una RSS

17.9.1 Sintesi formale

Il procedimento di sintesi formale di una rete sequenziale sincrona è formato da 5 passi e consente di dedurre lo schema logico dalle specifiche di comportamento:

1. Comprensione delle specifiche e individuazione del grafo degli stati
2. Definizione della tabella di flusso a partire dal grafo degli stati
3. Codifica degli stati e definizione della tabella delle transizioni
4. Sintesi della parte combinatoria
5. Disegno dello schema logico

17.9.2 Sintesi diretta

La progettazione diretta di una RSS si esegue combinando RSS notevoli più semplici (registri, shift register, contatori, ...) tramite opportuna logica combinatoria.

18 Registro

Un registro a k bit è una rete logica sincrona in grado di memorizzare un dato formato da k bit. La rete ad ogni fronte di salita del clock, memorizza e rende disponibile sulle uscite $OUT[k-1 \dots 0]$ il dato $IN[k-1 \dots 0]$ in ingresso se l'ingresso $WE = 1$ (write enable) mentre mantiene il valore precedentemente memorizzato se $WE = 0$. Inoltre la rete è dotata di un ingresso asincrono A_RESET che, se 1, pone a livello logico 0 tutti i bit del registro, indipendentemente dal valore dei segnali WE, IN e del clock. WE è un comando sincrono, agisce al fronte del clock, quindi al termine degli intervalli di tempo in cui vale 1.

18.1 Varianti

Molto spesso esistono più varianti delle RSS notevoli, che si differenziano per:

- comandi disponibili (registri con o senza A_RESET)
- se i comandi disponibili sono sincroni o asincroni
- per l'ordine di priorità dei comandi (i comandi asincroni sono sempre prioritari rispetto a quelli sincroni)

Bisogna sapere quale ingresso sincrono è prioritario se vi sono più ingressi sincroni.

18.2 Flip-flop T

Usando un FF-D, il flip-flop T (toggle) è realizzato prendendo l'uscita (Q) e il suo complemento (Q') dal FF-D e collegarle ad un selettore che a sua volta è collegato in retroazione diretta all'ingresso D del FF-D.

18.3 Shift register

Uno shift register (o registro a scorrimento) a k bit è una rete in grado di memorizzare gli ultimi k bit ricevuti su un segnale di ingresso seriale IN, rendendoli disponibili attraverso le uscite $OUT[k-1 \dots 0]$. Tutti i bit memorizzato possono essere portati a 0 asserendo l'ingresso asincrono A_RESET. Questo componente è utile per:

- Ritardare da 1 a k intervalli di tempo il segnale IN.
- Riconoscere il verificarsi di stringe d'ingresso
- Convertitore Seriale/Parallelo e Parallelo/Seriale
- Conteggio
- Rotazione verso destra o sinistra
- Moltiplicazione o divisione per una potenza di 2

18.3.1 Comandi

EN **o enable** l'ingresso IN viene memorizzato solo se $EN = 1$ e le uscite non cambiano quando $EN = 0$

LD **o load** se $LD = 1$ il valore memorizzato diventa IN

R/L' **o destra/sinistra'** consente variare la direzione dello shift

18.3.2 Universal shift register

Quando tutti i comandi elencati precedentemente sono disponibili, si parla di universal shift register. 4 comandi possibili:

Hold corrispondente a $LD = 0$, $EN = 0$, $R/L' = -$

Shift right corrispondente a $LD = 0$, $EN = 1$, $R/L' = 1$

Shift left corrispondente a $LD = 0$, $EN = 1$, $R/L' = 0$

Load corrispondente a $LD = 1$, $EN = -$, $R/L' = -$

Sono codificabili da 2 bit di comando S_1 e S_0. Posso avere un solo ingresso IN che entra a sinistra o a destra a seconda del valore dei bit di comando, o due ingressi diversi IN_R e IN_L.

19 Monoimpulsore

Un monoimpulsore (o edge detector) è una rete che asserisce la sua uscita esattamente per un ciclo di clock quando l'ingresso, anche asincrono, ha un fronte di salita. Questo circuito deve anche sincronizzare, e quindi funziona sotto l'ipotesi che il segnale in ingresso abbia una frequenza minore del clock. Se il fronte di salita capita nell'intervallo $[t_{su}, t_h]$ intorno al fronte del clock, è accettabile segnalare il fronte del segnale con più di un clock di ritardo, come indicato in figura.

19.1 Versione base

Il monoimpulsore base è realizzato con due FF-D in serie. L'output della rete OUT è il risultato dell'output Q, detto IN_sync del primo FF-D e l'output Q' del secondo ($IN\_sync * Q'$). In questa versione, in caso di fronte dell'ingresso che causa metastabilità, il rispetto delle specifiche dipende da:

- come viene interpretato dal gate AND lo stato di metastabilità
- lo stato casuale in cui si trova il FF-1 all'uscita dalla metastabilità

Se lo stato metastabile viene interpretato come 1 e FF-1 esce in stato 1 oppure se la metastabilità viene vista come 0 e similmente FF-1 esce in stato 0, allora ho $OUT = 1$ per un clock. Se però l'interpretazione dello stato metastabile e lo stato di uscita non sono identici ho:

$OUT = 1$ per meno di un clock, in caso di uscita dalla metastabilità in stato 0

$OUT = 0$ per l'inizio di un periodo di clock e per tutto il periodo successivo, in caso di uscita in stato 1

Se il segnale OUT è usato in una rete di Moore, potrebbe non essere un problema, purché la rete abbia il tempo di calcolare lo stato futuro nel rispetto del tempo di setup dei suoi FF-D.

19.2 Versione avanzata

Consiste nella stessa rete della versione base, con l'aggiunta di un FF-D prima del FF-1 della versione base, per eliminare i casi limite di metastabilità. L'uscita del nuovo FF-1 si chiama IN_meta. L'unico lato negativo di questa versione è l'aumento di un periodo, dal momento in cui la rete rileva il fronte sul segnale IN all'uscita OUT anche per fronti del segnale asincrono che non violano il tempo di set-up e hold.

20 Contatore

Un contatore è una rete sincrona senza ingressi (nel caso più semplice) che effettua continuamente un ciclo attraverso tutti gli stati interni e rende disponibile sulle uscite lo stato interno. Un contatore binario modulo n è un contatore in cui gli stati sono codificati con i primi n numeri binari.

20.1 Contatore binario x4

$(Q_1)^n$	$(Q_0)^n$	$(Q_1)^{n+1}$	$(Q_0)^{n+1}$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Dalla sequenza di stati si può dedurre:

- Il bit di peso minore commuta ad ogni clock, ovvero è il NOT del suo valore nel ciclo di clock precedente
- Il secondo bit è l'EXOR dello stato precedente

Perciò:

- $(Q_0)^{n+1} = (Q_0')^n$
- $(Q_1)^{n+1} = (Q_0 \oplus Q_1)^n$

20.2 Comandi sincroni

20.2.1 ENABLE

Il comando di ENABLE (EN) abilita/disabilita il conteggio. Il contatore con ENABLE conta i cicli di clock in cui $EN = 1$. Essendo un comando sincrono, $EN = 0$ blocca il conteggio al clock successivo.

esempio:

La rete si può realizzare con dei selettori che intercettano il segnale prima di entrare nell'ingresso D di ogni FF-D. Sull'ingresso $I0$ del selettore abbiamo l'uscita del FF-D Q, mentre su $I1$ abbiamo il complemento dell'uscita (Q') del FF-D. Una possibile ottimizzazione sarebbe sostituire il selettore per Q_0 con un EXOR, poiché il comportamento è lo stesso.

EN	Q_0	Q_0'	z	$EN \oplus Q_0$
0	0	1	0	0
1	0	1	1	1
0	1	0	1	1
1	1	0	0	0

Oltre a Q_0, è possibile ottimizzare il calcolo del valore Q_1. Questo infatti equivale a un AND e un EXOR in cascata.

EN	Q_1	Q_0	$Q_1 \oplus Q_0$	z	ENQ_0	$Q_1 \oplus (ENQ_0)$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	1	1	1	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

20.2.2 RESET

Tipicamente prioritario rispetto al comando di ENABLE, il comando di RESET (RES) serve a riportare allo stato 0 il contatore, in modo sincrono. Normalmente A_RESET è usato all'inizializzazione del sistema, mentre RES per resettare il conteggio durante la normale operatività.

20.2.3 LOAD

Il comando di LOAD (LD), tipicamente prioritario rispetto ad ENABLE e RESET, imposta il valore di conteggio ad un valore fornito dall'esterno sugli ingressi $I[k-1 \dots 0]$.

20.2.4 UP/DOWN'

Il comando di UP/DOWN' (U/D') stabilisce se il conteggio va effettuato in avanti ($U/D' = 1$) o all'indietro ($U/D' = 0$). Ovviamente il comando ENABLE è sempre prioritario rispetto a U/D'.

20.2.5 Contatore binario x8

Per realizzare un contatore binario x8, il bit Q_2 deve commutare ogni qualvolta $Q_1Q_0 = 1$. Un concetto equivalente al fatto che nel sistema decimale, la cifra delle centinaia incrementa soltanto quando quelle che la precedono formano il numero 99. La commutazione può essere realizzata come quella per Q_1 vista nello schema con ENABLE.

20.2.6 Incremento della base di conteggio

La regola che il bit Q_i deve commutare quando tutti i bit da Q_{i-1} a Q_0 valgono 1 ha valenza generale. Similmente, nel conteggio all'indietro, il bit Q_i deve commutare quando tutti i bit da Q_{i-1} a Q_0 valgono 0. Per estendere la base di conteggio è quindi utile dotare il contatore di un'ulteriore uscita di *carry out* (CO), che segnala il raggiungimento dell'ultimo stato. $CO = 1$ in caso di conteggio in avanti, quando tutti i bit valgono 1, in caso di conteggio all'indietro, quando tutti i bit valgono 0. Collegando questa uscita all'ingresso ENABLE di un'altro contatore, è possibile abilitarlo al conteggio solo quando deve effettivamente contare. Così si ottiene un contatore privo di comandi sincroni con base di conteggio pari al prodotto delle due basi. Se si vuole avere un contatore composto con il comando ENABLE, bisogna:

- Portarlo all'ingresso del contatore meno significativo
- Condizionare con EN anche l'ingresso ENABLE del contatore successivo

20.2.7 Decremento della base di conteggio

Dato un contatore per n, per realizzare un contatore per m tale che $m < n$, devo:

- Trovare il mintermine corrispondente a $m - 1$
- Aggiungere un AND con il segnale EN per evitare che il contatore venga reimpostato anche quando $EN = 0$
- Collegare il segnale precedentemente descritto al comando RES del contatore in questione

20.2.8 Divisore di frequenza

Ogni uscita Q_i del contatore mentre $EN = 1$ evolve come un'onda quadra con frequenza $\frac{f}{2^{i+1}}$, se f è la frequenza del clock. Per esempio, Q_0 ha la frequenza di $\frac{f}{2}$, Q_1 ha la frequenza di $\frac{f}{4}$ e così via.

21 Riconoscitore di sequenze

Progettare una rete sincrona che controlla se gli ultimi n byte che si sono presentati sull'ingresso $IN[7 \dots 0]$ mentre il segnale EN era a livello logico 1 erano uguali ad una sequenza predefinita. Nel caso sia rilevata la sequenza desiderata, nel periodo di clock successivo a quello in cui si è ricevuto l'ultimo valore, l'uscita OUT deve essere portata al valore logico 1 e rimanere tale finché non viene asserito il segnale asincrono di reset A_RESET. In seguito ad un reset, la rete deve riprendere immediatamente il controllo della sequenza di ingresso come se non fosse stato ricevuto alcun carattere.

21.1 Versione base

Un modo, inefficiente, per riconoscere sequenze secondo Moore negli ultimi n dati di ingresso è quello di memorizzare in uno shift register a n stadi i dati stessi e poi verificare ad ogni clock se gli n dati rappresentano la sequenza predefinita. Se i dati hanno parallelismo k, saranno necessari $k * N$ FF-D.

21.2 Versione ottimizzata

La soluzione precedente non ottimizza l'uso delle risorse. Non è necessario memorizzare l'intera lunghezza n del dato, servirebbe semplicemente sapere se il primo numero è apparso n clock prima, il secondo $n - 1$ clock prima, ... Un modo migliore per riconoscere una sequenza secondo Moore negli ultimi n dati di ingresso è quindi quello di memorizzare in vari shift register a n, $n-1$, ..., 1 bit, se si è visto il simbolo richiesto n, $n-1$, ..., 1 clock prima, infine verificare ad ogni clock se gli n bit sono tutti a 1. Questa soluzione richiede il seguente numero di FF-D:

$$n + (n-1) + \cdots + 1 = n!$$

21.3 Versione con i contatori

Si può ridurre ulteriormente l'utilizzo di FF-D rispetto alle versioni precedente, usando un contatore con EN e LD. Non è necessario ricordare se n clock prima abbiamo visto il dato k. In ogni momento la rete aspetta un simbolo che la rete attende e che la fa avanzare al passo successivo, ovvero fa incrementare di 1 il numero di simboli corretti visti in sequenza. Il modo più efficiente per riconoscere sequenze secondo Moore negli ultimi n dati di ingresso è quindi quello di memorizzare in un contatore x$(n+1)$ se si è visto 0, 1, 2, ..., o tutti e n i simboli della sequenza, abilitando il conteggio quando la rete riceve un simbolo corretto e resettando a 0 quando compare un simbolo fuori sequenza. Così facendo sono necessari soltanto $[log_2(n+1)]$ FF-D.

22 Clock gating e clock skew

22.1 Clock gating

Quando un segnale viene usato per "fermare" il clock, questo viene definito come clock gating. Un rischio del clock gating è che, se in presenza di alee, introdotte dalla rete responsabile per il segnale di attivazione del clock, possono verificarsi fronti di salita spuri del segnale uscente dal clock gating (CK_G). Per evitare questo, bisogna assicurarsi che sull'ingresso clock di un FF-D vengano mandati soltanto segnali sincroni (ovvero che commutano una sola volta all'inizio del periodo di clock).

22.2 Clock skew

Il clock gating, oltre a generare potenziali glitch, può portare anche al fenomeno del clock skew. Il clock skew è un ritardo che si forma a causa del ritardo introdotto dalla rete combinatoria che viene utilizzata per il clock gating. Questo è potenzialmente dannoso poiché la RSS non sincronizzata con il clock potrebbe aggiornare il suo stato con nuovi valori prodotti dalla RSS sincronizzata (che potrebbe essere anche in regime di transitorio).