Università degli Studi di Ferrara
Corso di Laurea in Ingegneria Elettronica

Simulatore Circuitale a Livello Switch

Tesina di Calcolatori Elettronici
di
Tarin Gamberini

Corso di Calcolatori Elettronici (ante riforma 3+2)
Anno Accademico 2000/2001
Docente *M. Favalli*

taringamberini [at] taringamberini [dot] com
www.taringamberini.com

Indice

Elenco delle tabelle

Elenco delle figure

Introduzione

Il simulatore che ho sviluppato è in grado di determinare il valore logico di ogni nodo di un circuito ed in particolare il valore assunto dal nodo d'uscita. Per far questo occorre fornire al simulatore sia una descrizione *topologica* del circuito sia una descrizione dello *stato iniziale* degli switch. Tali descrizioni sono fornite attraverso appropriati file di testo così come mostrato nel capitolo 6.

Il simulatore circuitale a livello switch viene utilizzato in fase di progetto dal punto di vista *logico* di reti circuitali. Simulando una circuito logico è possibile, prima di entrare in fase di produzione, scoprire se implementa correttamente la relativa funzione logica, evitando errori.

Lo studio di circuiti ad un livello più approfondito, in cui è determinante l'evoluzione nel tempo delle grandezze di interesse, è lasciato ad altri tipi di simulatori come per esempio SPICE.

Capitolo 1

Introduzione teorica

La logica binaria può essere rappresentata dal transistor MOS, o meglio da un suo modello matematico semplificato (con perdita di molti altri dettagli elettrici) detto modello switch.

Uno switch può funzionare secondo due possibili logiche, riportate in tabella 1.1, pertanto può assumere uno dei due simboli {0, 1}.

Logica	**Tecnologia**	**Valore logico**	**Stato fisico**
positiva	NMOS	0 oppure 1	1=chiuso, 0=aperto
negativa	PMOS	0 oppure 1	0=chiuso, 1=aperto

Tabella 1.1: Logiche di funzionamento di uno switch

Il simulatore lavora invece con quattro simboli fornendo in uscita {0, 1, X, Z} i cui significati sono spiegati in tabella 1.2.

La descrizione circuitale tramite transistor MOS di reti logiche a gate, che

Simbolo	**Significato**	**Stato dell'uscita**
0	Esiste un percorso connesso verso massa.	0
1	Esiste un percorso connesso verso l'alimentazione.	1
X	Esiste un percorso connesso sia verso massa sia verso l'alimentazione.	Cortocircuito
Z	Non esiste alcun percorso connesso verso massa o verso l'alimentazione.	Alta impedenza

Tabella 1.2: Logica del simulatore

rappresentano funzioni booleane, è possibile garantendo le risposte corrette {0, 1} ed evitando quelle errate {X, Z}, ricorrendo alla topologia *complementary mos*, o più brevemente *CMOS*.

Tale topologia si realizza collegando in serie due circuiti: il primo realizza la funzione booleana diretta tramite NMOS, ed è collegato all'alimentazione; il secondo realizza la funzione booleana duale tramite PMOS, ed è collegato a massa. L'uscita viene comunque prelevata dal nodo comune alle due reti.

Per esempio volendo realizzare la funzione logica ($X\ AND\ Y$), la cui tabella di verità è riportata in tabella 1.3, progetteremo il circuito in logica

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Tabella 1.3: Tabella di verità della funzione logica ($X\ AND\ Y$)

complementare, come in figura 1.1.

La teoria dei grafi permette di descrivere un circuito tramite la matrice M

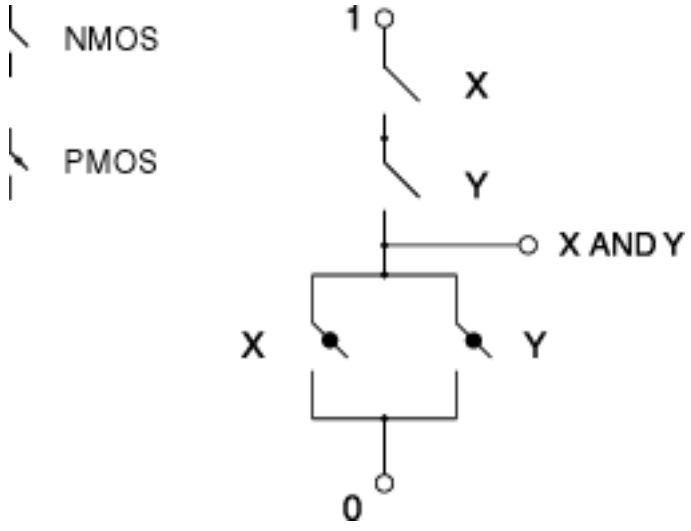

Figura 1.1: Funzione ($X\ AND\ Y$) implementata in logica complementare

delle connessioni che gode delle seguenti proprietà:

1. M è quadrata di dimensione NxN, dove N è il numero di nodi della rete.

2. L'elemento $m_{ij} = 1$ se il nodo i è connesso al nodo j.

3. M è simmetrica: se i è connesso a j anche j è connesso ad i, quindi anche $m_{ji} = 1$.

Un teorema dimostra che il prodotto logico della matrice M delle connessioni per se stessa, ripetuto iterativamente finché $M(k+1) = M(k)$, permette di stabilire se esiste un cammino connesso fra i nodi a e z. È sufficiente infatti che al termine delle k iterazioni sia $M_{az} = 1$.

Capitolo 2

Metodo risolutivo

Nel descrivere la rete si adotta la convenzione di numerare i nodi del circuito con numeri simbolici: 0 per la massa, 1 per l'alimentazione, 2, 3, ecc... per gli altri nodi.

In questo modo al matrice M delle connessioni avrà indici di riga e colonna che varieranno fra 0 ed $N-1$. In particolare la colonna (riga) 0 è la colonna (riga) di massa e permette di verificare se un qualsiasi nodo è collegato a massa, la colonna (riga) 1 è la colonna (riga) di alimentazione e permette di verificare se un qualsiasi nodo è collegato all'alimentazione.

Il simulatore inizializza M a matrice unitaria, il che è formalmente corretto in quanto prima di collegare qualsiasi componente fra i vari nodi ognuno di essi risulta collegato solo con se stesso ($m_{ii} = 1$).

Successivamente viene fornita una descrizione del circuito nella quale sono specificati a quali nodi sono connessi ognuno degli switch.

A fronte di un ingresso occorre settare lo stato dei singoli switch e verificare quali coppie di nodi risultano direttamente collegate da uno switch chiuso, ponendo in tal caso il relativo elemento di M ad 1.

Poi si ripete iterativamente il prodotto logico matriciale di M per se stessa terminando quando $M(k+1) = M(k)$.

Infine occorre confrontare il nodo di uscita i, tramite la riga (colonna) i della matrice delle connessione, con le colonne (righe) di massa e di alimentazione, fornendo l'uscita opportuna {0,1,X,Z}.

Capitolo 3

Descrizione delle strutture dati

*vetSwitch

Per memorizzare la descrizione topologica del circuito e lo stato relativo ad ogni switch ho utilizzato il vettore *vetSwitch*:

```
/* Vettore degli switch */
tipoSwitch *vetSwitch;
```

che ho dichiarato come puntatore in quanto verrà allocato dinamicamente in fase di esecuzione del programma in funzione del numero totale di switch presenti nel circuito. Questi ultimi sono specificati nel relativo file di descrizione del circuito. Ogni elemento del vettore degli switch è così strutturato:

```
typedef struct tipoSwitch
{
  /* Nome simbolico dello switch */
  char nome[5];

  /* Logica dello switch: 1 Positiva NMOS, 0 Negativa PMOS */
  char logica;

  /* Indicano a quale nodo sono collegati i pin dello switch */
  unsigned int pinA,pinB;

  /* Chiuso o aperto a seconda della logica */
  char stato;
};
```

Il campo *logica* memorizza la logica di funzionamento dello switch ed assume solo i valori {0,1} con significato {logica negativa, logica positiva}. Il campo *stato* memorizza la stato dello switch ed assume solo i valori {0,1} con significato {chiuso, aperto}. I campi *pinA* e *PinB* memorizzano il numero del nodo cui sono collegati i morsetti dello switch. Poiché sono consentiti fino a 65536 nodi è possibile simulare circuiti anche molto complessi!

*vetIngressi

Per memorizzare le informazioni relative ad ogni ingresso ho utilizzato il vettore *vetIngressi*:

```
/* Vettore degli ingressi */
tipoIO *vetIngressi;
```

che ho dichiarato come puntatore in quanto verrà allocato dinamicamente in fase di esecuzione del programma in funzione del numero di ingressi. Questi ultimi sono specificati nel relativo file degli ingressi. Ogni elemento del vettore degli ingressi è così strutturato:

```
typedef struct tipoIO
{
  char nome[5]; /* Nome simbolico */
  int valore;   /* Valore */
};
```

Il campo *valore* memorizza il valore dell'ingresso ed assume solo i valori {0,1}.

*vetUscite

Per memorizzare le informazioni relative ad ogni uscita ho utilizzato il vettore *vetUscite*:

```
/* Vettore delle uscite */
tipoIO *vetUscite;
```

che ho dichiarato come puntatore in quanto verrà allocato dinamicamente in fase di esecuzione del programma in funzione del numero di uscite. Queste ultime sono specificate nel relativo file delle uscite. Ogni elemento del vettore delle uscite è così strutturato:

```
typedef struct tipoIO
{
  char nome[5]; /* Nome simbolico */
  int  valore;  /* Valore */
};
```

Questa volta il campo .valore memorizza il numero del nodo che si desidera considerare come uscita, pertanto assumerà valori nell'intervallo (0..65535).

fileDescrittoreCircuito

È il file descriptor associato al file di testo che contiene la descrizione topologica del circuito. Tale file va passato a linea di comando come primo parametro, (*simcirc circuito.txt*):

```
FILE *fileDescrittoreCircuito;
```

Il file deve essere scritto dall'utente e deve essere strutturato come specificato di seguito:

- La prima riga deve contenere il numero di nodi del circuito;
- La seconda riga deve contenere il numero totale di switch impiegati nel circuito;
- Ogni altra riga successiva contiene informazioni su ogni switch tramite quattro campi che devono essere separati dal carattere di tabulazione: nome simbolico dello switch, logica dello switch, numero del nodo cui è collegato il morsetto A, numero del nodo cui è collegato il morsetto B.
- Si deve numerare con 0 il nodo di massa (livello logico basso), con 1 il nodo di alimentazione (livello logico alto), con 2,3,4, ecc... gli altri eventuali nodi della rete.

fileIngressi

È il file descriptor associato al file di testo che contiene la descrizione del vettore degli ingressi da applicare al circuito. Tale file va passato a linea di comando come secondo parametro, (*simcirc circuito.txt ingressi.txt*):

```
FILE *fileIngressi;
```

Il file deve essere scritto dall'utente e deve essere strutturato come specificato di seguito:

- La prima riga deve contenere il numero di ingressi da applicare al circuito;
- Ogni altra riga successiva contiene informazioni su ogni ingresso tramite due campi che devono essere separati dal carattere di tabulazione: nome simbolico dell'ingresso, valore dell'ingresso.

fileUscite

È il file descriptor associato al file di testo che contiene la descrizione delle uscite da prelevare dal circuito. Tale file va passato a linea di comando come terzo parametro, (*simcirc circuito.txt ingressi.txt uscite.txt*):

```
FILE *fileUscite;
```

Il file deve essere scritto dall'utente e deve essere strutturato come specificato di seguito:

- La prima riga deve contenere il numero di uscite da prelevare dal circuito;
- Ogni altra riga successiva contiene informazioni su ogni uscita tramite due campi che devono essere separati dal carattere di tabulazione: nome simbolico dell'uscita, numero del nodo da cui prelevare l'uscita.

**matConnessioni

Per memorizzare la matrice delle connessioni utilizzato la matrice *matConnessioni*:

```
/* Matrice delle connessioni */
char **matConnessioni;
```

che ho dichiarato come puntatore di puntatori in quanto verrà allocata dinamicamente in fase di esecuzione del programma in funzione del numero di nodi presenti nel circuito. Questi ultimi sono specificati nel relativo file di descrizione del circuito. Ogni suo elemento può assumere i valori {0,1}.

numNodiCircuito

Memorizza il numero di nodi del circuito; è acquisito dal file di descrizione del circuito:

```
/* Dimensione matrice connessioni */
unsigned int numNodiCircuito;
```

numSwitch

Memorizza il numero totale di switch utilizzati nel circuito; è acquisito dal file di descrizione del circuito:

```
/* Dimensione vettore degli switch */
unsigned int numSwitch;
```

numIngressi

Memorizza il numero totale di ingressi applicati al circuito; è acquisito dal file degli ingressi:

```
/* Numero di ingressi */
unsigned int numIngressi;
```

numUscite

Memorizza il numero di uscite da prelevare dal circuito; è acquisito dal file delle uscite:

```
/* Dimensione vettore delle uscite */
unsigned int numUscite;
```

Capitolo 4

Descrizione del programma

main(int argc, char **argv)

Il simulatore è avviato da linea di comando con sintassi (per es.): *simcirc circuito.txt ingressi.txt uscite.txt*. Pertanto riceverà in ingresso *argc=4*, *argv[1]=circuito.txt*, *argv[2]=ingressi.txt*, *argv[3]=uscite.txt*.

Vengono quindi chiamate nell'ordine le procedure:

```
Inizializzazione(argc,argv)
LeggiFileDescrittoreCircuito(argv[1]);
LeggiFileIngressi(argv[2]);
LeggiFileUscite(argv[3]);
SettaStatoInizialeSwitch();
SettaStatoInizialeCircuito();
AnalizzaCollegamentiCircuito();
CalcolaValoreDelleUsite();
Dealloca();
```

descritte nelle pagine seguenti.

Inizializzazione(argc,argv)

La procedura controlla la sintassi di *simcirc* a linea di comando, nonché la presenza su disco dei tre file ad esso passati come parametri, interrompendo l'esecuzione in caso di problemi e restituendo un opportuno codice d'errore.

LeggiFileDescrittoreCircuito(argv[1])

La procedura apre il file di descrizione circuitale e legge il numero di nodi del circuito, permettendo così l'allocazione dinamica della matrice delle connessioni che viene immediatamente settata a matrice unitaria.

Si legge poi il numero totale di switch impiegati nel circuito, permettendo così l'allocazione dinamica del vettore degli switch nel quale viene immediatamente caricata la descrizione del circuito.

Infine il file di descrizione circuitale viene chiuso.

LeggiFileIngressi(argv[2])

La procedura apre il file degli ingressi e legge il numero di ingressi, permettendo così l'allocazione dinamica del vettore degli ingressi nel quale vengono immediatamente caricati nome simbolico e valore.

Infine il file degli ingressi viene chiuso.

LeggiFileUscite(argv[3])

La procedura apre il file delle uscite e legge il numero di uscite, permettendo così l'allocazione dinamica del vettore delle uscite nel quale vengono immediatamente caricati i nomi delle uscite ed il nodo da cui prelevarle.

Infine il file delle uscite viene chiuso.

SettaStatoInizialeSwitch()

La procedura setta lo stato degli switch in base alla loro logica di funzionamento ed in base al valore degli ingressi.

Un primo ciclo for permette di scorrere il vettore degli ingressi. Un secondo ciclo for nidificato nel primo permette, per ogni ingresso, di esaminare ogni elemento del vettore degli switch. Se la logica di funzionamento dello switch è positiva allora il suo stato è quello imposto dall'ingresso, se la logica è negativa il suo stato è il *duale* di quello imposto dall'ingresso.

Infine viene chiamata la procedura *StampaVetSwitch()* che stampa a video il vettore degli switch.

SettaStatoInizialeCircuito()

La procedura pone, se è il caso, ad 1 gli elementi m_{ij} e m_{ji} della matrice delle connessioni per indicare che lo switch collegato fra i nodi i e j del circuito è chiuso.

Per far questo un ciclo for scandisce il vettore degli switch e per ogni elemento si considerano i nodi cui lo switch è collegato, così come è stato specificato nella descrizione topologica del circuito. Pertanto si pone lo stato dello switch (1 se chiuso, 0 se aperto) alle coordinate *[vetSwitch.pinA] [vetSwitch.pinB]* della matrice delle connessioni. Quest'ultima operazione viene effettuata solo nel caso a tali coordinate ci sia uno 0: se c'è un 1 vuol dire che i nodi *vetSwitch.pinA* e *vetSwitch.pinB* sono già collegati da qualche altro switch, è pertanto inutile ogni altra operazione.

AnalizzaCollegamentiCircuito()

Questa procedura è un po' il cuore del programma: si preoccupa di verificare se esiste un cammino connesso fra i nodi i e j della rete anche se tali nodi non sono direttamente collegati da un unico switch. Come spiegato nei cenni teorici ciò è possibile effettuando iterativamente l'operazione $M(k+1) = M(k) * M(k)$ dove $*$ è il prodotto logico matriciale.

Si utilizzano le seguenti variabili locali:

```
/* Conta il numero delle iterazioni M(k+1)=M(k)^2 */
unsigned int k;

/* Memorizza M(k+1) */
char **prodottoMatConnessioni;

/* Sfruttata per scambiare le matrici */
char **tmpMat;
```

La matrice $M(k+1)$ viene memorizzata nella *prodottoMatConnessioni* che viene immediatamente allocata dinamicamente e settata ad unitaria.

Il puntatore *tmpMat* viene sfruttato momentaneamente solo come variabile d'appoggio per effettuare l'assegnazione $M(k) = M(k+1)$ senza dover copiare elemento per elemento la matrice *prodottoMatConnessioni* nella *matConnessioni*.

Si setta il contatore k delle iterazioni a 0 e si calcola il primo prodotto logico matriciale invocando la procedura *ProdottoLogicoMatriciale*, analizzata di seguito. Il risultato viene stampato a video tramite la procedura *stampaMatrici*.

Viene ora eseguito un ciclo che termina solo quando $M(k+1) = M(k)$, confrontate invocando *MatriciDiverse*, analizzata di seguito, e che ad ogni iterazione assegna $M(k) = M(k+1)$ e calcola $M(k+1) = M(k) * M(k)$ stampando a video il risultato.

La procedura termina deallocando la matrice *prodottoMatConnessioni*.

ProdottoLogicoMatriciale (prodottoMatConnessioni, matConnessioni)

La procedura calcola $M(k+1) = M(k) * M(k)$ dove $*$ è il prodotto logico matriciale.

Si utilizza la variabile locale:

```
/* Flag di uscita dal ciclo più interno */
char mintermineVero;
```

sfruttata come flag di uscita dal ciclo più interno dedicato al calcolo dell'espressione SP (Somme di Prodotti).

Un primo ciclo for scorre le righe della matrice delle connessioni. Un secondo for, nidificato nel primo, scandisce le colonne solo nel triangolo superiore[1] della matrice. Infine un terzo ciclo permette di effettuare il prodotto logico riga per colonna.

MatriciDiverse(char **matA,char **matB)

La funzione confronta le matrici e ritorna 1 se $matA \neq matB$ o 0 se $matA = matB$.

Si utilizza la variabile locale:

```
/* Flag di uscita dai cicli */
char diverse;
```

sfruttata come flag di uscita dai cicli. La funzione tramite due cicli nidificati scandisce elemento per elemento le due matrici e termina appena trova due elementi differenti, ritornando il valore opportuno.

CalcolaValoreDelleUsite()

Calcola il valore delle uscite.

Un ciclo for scandisce il vettore delle uscite verificando se esiste un collegamento fra il nodo specificato in *vetUscite[i].valore* (riga della matrice delle connessioni) e il nodo dell'alimentazione (colonna 1 della matrice delle connessioni) o il nodo di massa (colonna 0 della matrice delle connessioni). Ciò è realizzato mediante una serie di if nidificati.

Dealloca()

La procedura rilascia lo spazio in memoria richiesto in fase di allocazione dinamica del vettore degli switch, del vettore degli ingressi, del vettore delle uscite e della matrice delle connessioni.

[1] Vedere il capitolo 5 *Breve analisi del costo computazionale*

Capitolo 5

Breve analisi del costo computazionale

La parte del programma che necessita il maggior tempo di esecuzione è senza dubbio la procedura *AnalizzaCollegamentiCircuito*, che verifica se esiste un cammino connesso fra i nodi i e j della rete. In particolare il costo computazionale maggiore è dovuto al ciclo:

```
while (MatriciDiverse(prodottoMatConnessioni,matConnessioni))
{
  /* M(k)=M(k+1) */
  tmpMat=matConnessioni;
  matConnessioni=prodottoMatConnessioni;
  prodottoMatConnessioni=tmpMat;

  /* M(k+1)=M(k)*M(k) */
  ProdottoLogicoMatriciale(prodottoMatConnessioni,matConnessioni);

  StampaMatrici(prodottoMatConnessioni,++k);
}
```

nel quale sono concentrate una serie di operazioni che dipendono dalla dimensione della matrice M delle connessioni, ossia dal numero di nodi della rete.

Ad un primo ragionamento ci si aspetterebbero costi computazionali per ciclo come mostrato in tabella 5.1, in realtà per verificare se due matrici

Procedura o funzione	**Costo computazionale per ciclo**
MatriciDiverse	$O(N^2)$
ProdottoLogicoMatriciale	$O(N^3)$

Tabella 5.1: Prima stima del costo computazionale

sono diverse è sufficiente scandirle, con due cicli nidificati, solo mentre si

trovano elementi identici $a_{ij} = b_{ij}$. Il confronto può infatti terminare appena si trovano due elementi diversi $a_{ij} \neq b_{ij}$.

È proprio così che ho realizzato la funzione *MatriciDiverse*, sfruttando il flag *diverse* di uscita dai cicli while che viene settato appena $a_{ij} \neq b_{ij}$.

Per il calcolo del prodotto logico matriciale $M(k+1) = M(k) * M(k)$ si può ridurre il costo computazionale ricordando che la matrice M delle connessioni è simmetrica. Di più, si può limitare il calcolo del prodotto al solo triangolo superiore (od inferiore) escludendo anche la diagonale principale che rimane, ad ogni iterazione, sempre piena di 1. Avremo così un costo pari ad un $O(N^2/2 - N)$.

Inoltre stiamo effettuando un prodotto logico matriciale per cui al posto del classico prodotto scalare riga i per colonna j:

$$a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{Ni}b_{Nj}$$

ora abbiamo l'espressione SP (Somme di Prodotti) associata:

$$a_{i1}\ AND\ b_{1j}\ OR\ a_{i2}\ AND\ b_{2j}\ OR\ \ldots\ OR\ a_{Ni}\ AND\ b_{Nj}$$

il calcolo del prodotto può allora terminare appena calcolato il primo mintermine $a_{ik}\ AND\ b_{kj} = 1$, ottenendo un costo computazionale minore rispetto a prima.

È proprio così che ho realizzato la procedura *ProdottoLogicoMatriciale*, sfruttando il flag *mintermineVero* per uscire dal ciclo più interno che calcola l'espressione SP (Somme di Prodotti), ed impiegando due for nidificati con indici $i = 0 \ldots N-1$ e $j = i+1 \ldots N$ per scandire il solo triangolo superiore.

Pertanto una stima più accurata del costo computazionale del programma è mostrata in tabella 5.2.

Procedura o funzione	**Costo computazionale per ciclo**
MatriciDiverse	$< O(N^2)$
ProdottoLogicoMatriciale	$O((N^2/2 - N)N)$

Tabella 5.2: Stima più accurata del costo computazionale

In realtà anche la procedura di stampa della matrice M delle connessioni ha il suo impatto in termini di costo computazionale, ma è stata inserita nel programma per mostrare l'evoluzione di M ad ogni iterazione. Ai fini del calcolo delle uscite può essere tranquillamente omessa, e pertanto non è stata considerata nel calcolo del costo computazionale.

Capitolo 6

Esempi di simulazione

Ecco alcuni esempi di simulazione.

6.1 Esempio 1 di simulazione

Il circuito in logica complementare considerato è quello di figura 6.1, che all'uscita del nodo 3 implementa la funzione $X\ AND\ (Y\ OR\ W)$.

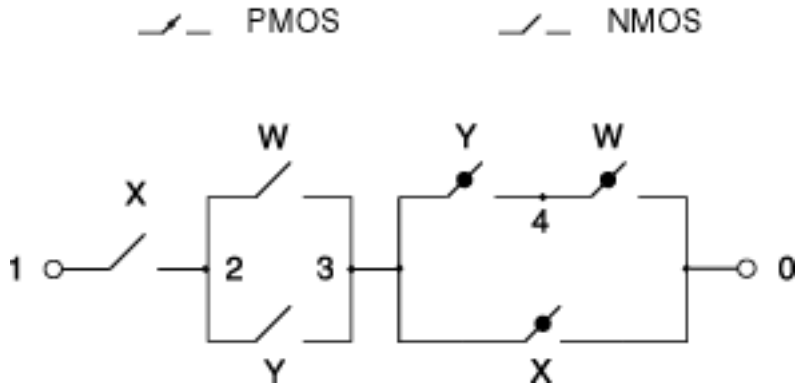

Figura 6.1: Circuito in logica complementare

File: circuit1.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a descrivere la topologia del circuito.

```
5
6
x       1       1       2
```

```
y       1       2       3
w       1       2       3
x       0       3       0
y       0       3       4
w       0       4       0

Questo file di testo rappresenta un circuito di switch ed e' formattato
come segue:

prima riga    numero di nodi nel circuito
seconda riga  numero totale di switch impiegati nel circuito

ogni altra riga contiene informazioni sugli switch ed e' formattata come
segue:

<Nome>tab<Logica>tab<Morsetto A>tab<Morsetto B>

dove i campi:

<Nome>
nome simbolico dello switch (1..5 car)

<Logica>
logica di funzionamento dello switch 1 Positiva NMOS, 0 Negativa PMOS

<Morsetto A>
numero del nodo del circuito cui e' collegato il pinA dello switch

<Morsetto B>
numero del nodo del circuito cui e' collegato il pinB dello switch

tab
carattere di tabulazione di separazione fra i campi
```

File: ingress1.txt

Il file contiene nella sua parte finale tutte le indicazioni utili ad impostare gli ingressi del circuito, ovvero lo stato iniziale dei transistor.

```
3
x       1
y       0
w       0

Questo file di testo rappresenta il vettore degli ingressi da applicare
al circuito ed e' formattato come segue:

prima riga    numero di ingressi

ogni altra riga contiene informazioni sul vettore degli ingressi ed e'
formattata come segue:

<Nome>tab<Valore>

dove i campi:

<Nome>
nome simbolico dell'ingresso (1..5 car)

<Valore>
```

```
valore dell'ingresso (1 o 0)

tab
carattere di tabulazione di separazione fra i campi
```

File: uscit1.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a prelevare le uscite del circuito.

```
5
U0        0
U1        1
U2        2
U3        3
U4        4

Questo file di testo rappresenta il vettore delle uscite da prelevare
dal circuito ed e' formattato come segue:

prima riga    numero di uscite

ogni altra riga contiene informazioni sul vettore delle uscite ed e'
formattata come segue:

<Nome>tab<Nodo>

dove i campi:

<Nome>
nome simbolico dell'uscita (1..5 car)

<Nodo>
numero del nodo del circuito da assumere come uscita

tab
carattere di tabulazione di separazione fra i campi
```

File: simulaz1.txt

Il risultato della simulazione viene salvato in questo file di testo.

```
Lettura file di descrizione del circuito: circuit1.txt ...completata!
Lettura file degli ingressi: ingress1.txt ...completata!
Lettura file delle uscite: uscit1.txt ...completata!
Simulazione circuitale a livello switch: Iniziata.


Stato iniziale degli switch a fronte dell'ingresso (x,y,w)=(1,0,0)

  Switch  Logica  PinA   PinB   Stato
  x       1 nmos  1      2        1
  y       1 nmos  2      3        0
  w       1 nmos  2      3        0
  x       0 pmos  3      0        0
  y       0 pmos  3      4        1
  w       0 pmos  4      0        1
```

```
------------------------------------------------------------ Iterazione 1 -
Matrice delle Connessioni M(0)
      0  1  2  3  4

  0   1  0  0  0  1
  1   0  1  1  0  0
  2   0  1  1  0  0
  3   0  0  0  1  1
  4   1  0  0  1  1

Matrice delle Connessioni M(1)=M(0)*M(0)
      0  1  2  3  4

  0   1  0  0  1  1
  1   0  1  1  0  0
  2   0  1  1  0  0
  3   1  0  0  1  1
  4   1  0  0  1  1


------------------------------------------------------------ Iterazione 2 -
Matrice delle Connessioni M(1)
      0  1  2  3  4

  0   1  0  0  1  1
  1   0  1  1  0  0
  2   0  1  1  0  0
  3   1  0  0  1  1
  4   1  0  0  1  1

Matrice delle Connessioni M(2)=M(1)*M(1)
      0  1  2  3  4

  0   1  0  0  1  1
  1   0  1  1  0  0
  2   0  1  1  0  0
  3   1  0  0  1  1
  4   1  0  0  1  1


I valori d'uscita sono:
  Uscita U0 = 0
  Uscita U1 = 1
  Uscita U2 = 1
  Uscita U3 = 0
  Uscita U4 = 0


Simulazione circuitale a livello switch: Terminata.
```

I valori di uscita calcolati dal simulatore sono direttamente verificabili in figura 6.2. In particolare al nodo 3 abbiamo un valore logico pari a 0 che è giustamente il risultato di $X\ AND\ (Y\ OR\ W)$ quando l'ingresso (X, Y, W) vale $(1, 0, 0)$.

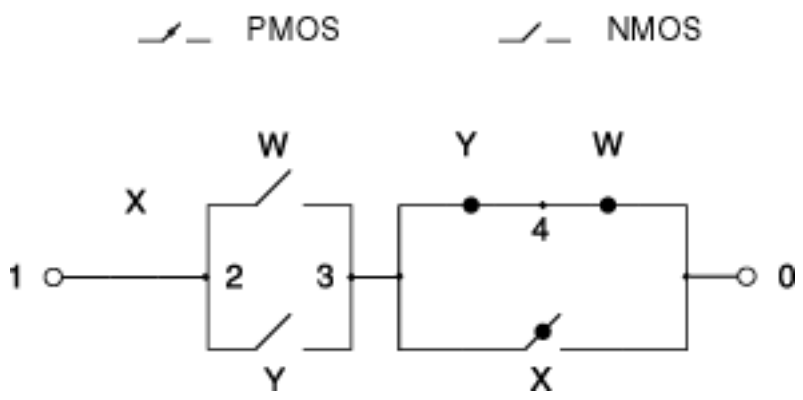

Figura 6.2: Circuito con ingresso (X, Y, W) = (1, 0, 0)

6.2 Esempio 2 di simulazione

Poiché la formattazione dei file e sempre la stessa, per brevità a partire da questa simulazione ometto di scrivere la parte finale dei file *circuit*, *ingress* ed *uscit*.

Il circuito in logica complementare considerato è sempre quello di figura 6.1 che implementa la funzione $X\ AND\ (Y\ OR\ W)$.

File: circuit2.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a descrivere la topologia del circuito. È identico al precedente *circuit1.txt*.

File: ingress2.txt

Il file contiene nella sua parte finale tutte le indicazioni utili ad impostare gli ingressi del circuito, ovvero lo stato iniziale dei transistor.

```
3
x       1
y       1
w       1
```

File: uscit2.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a prelevare le uscite del circuito. È identico al precedente *uscit1.txt*.

File: simulaz2.txt

Il risultato della simulazione viene salvato in questo file di testo.

```
Lettura file di descrizione del circuito: circuit1.txt ...completata!
Lettura file degli ingressi: ingress2.txt ...completata!
Lettura file delle uscite: uscit1.txt ...completata!
Simulazione circuitale a livello switch: Iniziata.


Stato iniziale degli switch a fronte dell'ingresso (x,y,w)=(1,1,1)

  Switch  Logica  PinA   PinB   Stato
  x       1 nmos  1      2        1
  y       1 nmos  2      3        1
  w       1 nmos  2      3        1
  x       0 pmos  3      0        0
  y       0 pmos  3      4        0
  w       0 pmos  4      0        0


------------------------------------------------------------ Iterazione 1 -
Matrice delle Connessioni M(0)
      0  1  2  3  4

  0   1  0  0  0  0
  1   0  1  1  0  0
  2   0  1  1  1  0
  3   0  0  1  1  0
  4   0  0  0  0  1

Matrice delle Connessioni M(1)=M(0)*M(0)
      0  1  2  3  4

  0   1  0  0  0  0
  1   0  1  1  1  0
  2   0  1  1  1  0
  3   0  1  1  1  0
  4   0  0  0  0  1


------------------------------------------------------------ Iterazione 2 -
Matrice delle Connessioni M(1)
      0  1  2  3  4

  0   1  0  0  0  0
  1   0  1  1  1  0
  2   0  1  1  1  0
  3   0  1  1  1  0
  4   0  0  0  0  1

Matrice delle Connessioni M(2)=M(1)*M(1)
      0  1  2  3  4

  0   1  0  0  0  0
  1   0  1  1  1  0
  2   0  1  1  1  0
  3   0  1  1  1  0
  4   0  0  0  0  1


I valori d'uscita sono:
```

```
  Uscita U0 = 0
  Uscita U1 = 1
  Uscita U2 = 1
  Uscita U3 = 1
  Uscita U4 = Z

Simulazione circuitale a livello switch: Terminata.
```

I valori di uscita calcolati dal simulatore sono direttamente verificabili in figura 6.3. In particolare al nodo 3 abbiamo un valore logico pari a 1 che è giustamente il risultato di $X\ AND\ (Y\ OR\ W)$ quando l'ingresso (X, Y, W) vale $(1, 1, 1)$.

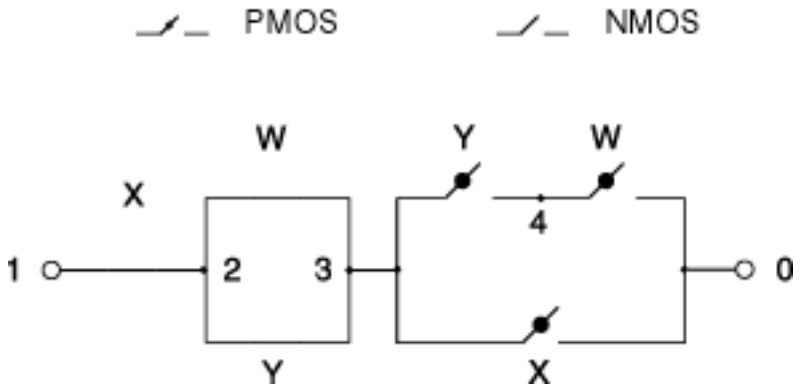

Figura 6.3: Circuito con ingresso (X, Y, W) = (1, 1, 1)

6.3 Esempio 3 di simulazione

Il simulatore può essere impiegato anche per testare una generica rete a switch come quella in figura 6.4, che non è realizzata in logica complementare in quanto non è possibile identificare una rete NMOS collegata all'alimentazione ed una rete PMOS collegata a massa.

File: circuit3.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a descrivere la topologia del circuito.

```
10
16
w       1       1       2
w       1       8       0
x       1       3       4
```

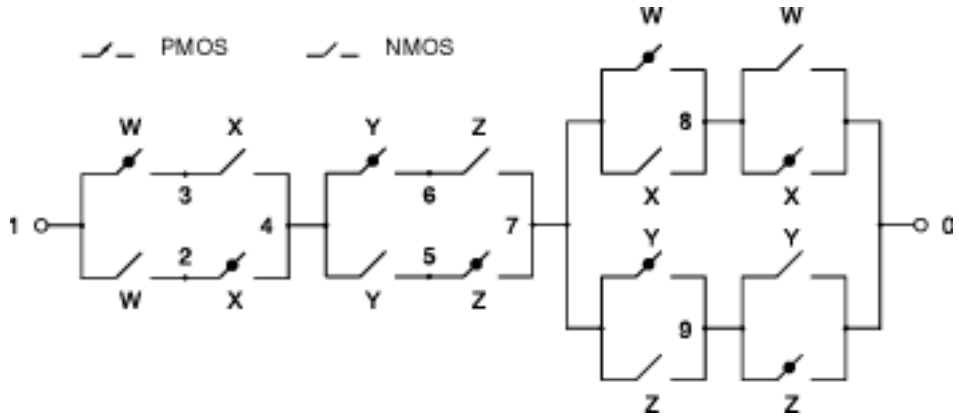

Figura 6.4: Circuito generico

```
x       1       7       8
y       1       4       5
y       1       9       0
z       1       6       7
z       1       7       9
w       0       1       3
w       0       7       8
x       0       2       4
x       0       8       0
y       0       4       6
y       0       7       9
z       0       5       7
z       0       9       0
```

File: ingress3.txt

Il file contiene nella sua parte finale tutte le indicazioni utili ad impostare gli ingressi del circuito, ovvero lo stato iniziale dei transistor.

```
4
w       0
x       1
y       1
z       0
```

File: uscit3.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a prelevare le uscite del circuito.

```
10
U0      0
U1      1
U2      2
U3      3
```

```
U4        4
U5        5
U6        6
U7        7
U8        8
U9        9
```

File: simulaz3.txt

Il risultato della simulazione viene salvato in questo file di testo.

```
Lettura file di descrizione del circuito: circuit2.txt ...completata!
Lettura file degli ingressi: ingress2.txt ...completata!
Lettura file delle uscite: uscit2.txt ...completata!
Simulazione circuitale a livello switch: Iniziata.


Stato iniziale degli switch a fronte dell'ingresso (w,x,y,z)=(0,1,1,0)

  Switch  Logica  PinA   PinB   Stato
  w       1 nmos  1      2        0
  w       1 nmos  8      0        0
  x       1 nmos  3      4        1
  x       1 nmos  7      8        1
  y       1 nmos  4      5        1
  y       1 nmos  9      0        1
  z       1 nmos  6      7        0
  z       1 nmos  7      9        0
  w       0 pmos  1      3        1
  w       0 pmos  7      8        1
  x       0 pmos  2      4        0
  x       0 pmos  8      0        0
  y       0 pmos  4      6        0
  y       0 pmos  7      9        0
  z       0 pmos  5      7        1
  z       0 pmos  9      0        1


------------------------------------------------------------ Iterazione 1 -
Matrice delle Connessioni M(0)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  0  0  0  0  0
  4   0  0  0  1  1  1  0  0  0  0
  5   0  0  0  0  1  1  0  1  0  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  0  0  0  0  1  0  1  1  0
  8   0  0  0  0  0  0  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1

Matrice delle Connessioni M(1)=M(0)*M(0)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  0  0  0  0  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  0  0  0
```

```
  4   0  1  0  1  1  1  0  1  0  0
  5   0  0  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  0  0  0  1  1  0  1  1  0
  8   0  0  0  0  0  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1


------------------------------------------------------------ Iterazione 2 -
Matrice delle Connessioni M(1)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  0  0  0  0  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  0  0  0
  4   0  1  0  1  1  1  0  1  0  0
  5   0  0  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  0  0  0  1  1  0  1  1  0
  8   0  0  0  0  0  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1

Matrice delle Connessioni M(2)=M(1)*M(1)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  1  0  1  0  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  1  1  0
  4   0  1  0  1  1  1  0  1  1  0
  5   0  1  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  1  0  1  1  1  0  1  1  0
  8   0  0  0  1  1  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1


------------------------------------------------------------ Iterazione 3 -
Matrice delle Connessioni M(2)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  1  0  1  0  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  1  1  0
  4   0  1  0  1  1  1  0  1  1  0
  5   0  1  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  1  0  1  1  1  0  1  1  0
  8   0  0  0  1  1  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1

Matrice delle Connessioni M(3)=M(2)*M(2)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  1  0  1  1  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  1  1  0
  4   0  1  0  1  1  1  0  1  1  0
  5   0  1  0  1  1  1  0  1  1  0
```

```
  6   0  0  0  0  0  0  1  0  0  0
  7   0  1  0  1  1  1  0  1  1  0
  8   0  1  0  1  1  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1


------------------------------------------------------------ Iterazione 4 -
Matrice delle Connessioni M(3)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  1  0  1  1  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  1  1  0
  4   0  1  0  1  1  1  0  1  1  0
  5   0  1  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  1  0  1  1  1  0  1  1  0
  8   0  1  0  1  1  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1

Matrice delle Connessioni M(4)=M(3)*M(3)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  0  1
  1   0  1  0  1  1  1  0  1  1  0
  2   0  0  1  0  0  0  0  0  0  0
  3   0  1  0  1  1  1  0  1  1  0
  4   0  1  0  1  1  1  0  1  1  0
  5   0  1  0  1  1  1  0  1  1  0
  6   0  0  0  0  0  0  1  0  0  0
  7   0  1  0  1  1  1  0  1  1  0
  8   0  1  0  1  1  1  0  1  1  0
  9   1  0  0  0  0  0  0  0  0  1


I valori d'uscita sono:
  Uscita U0 = 0
  Uscita U1 = 1
  Uscita U2 = Z
  Uscita U3 = 1
  Uscita U4 = 1
  Uscita U5 = 1
  Uscita U6 = Z
  Uscita U7 = 1
  Uscita U8 = 1
  Uscita U9 = 0


Simulazione circuitale a livello switch: Terminata.
```

I valori di uscita calcolati dal simulatore sono direttamente verificabili in figura 6.5.

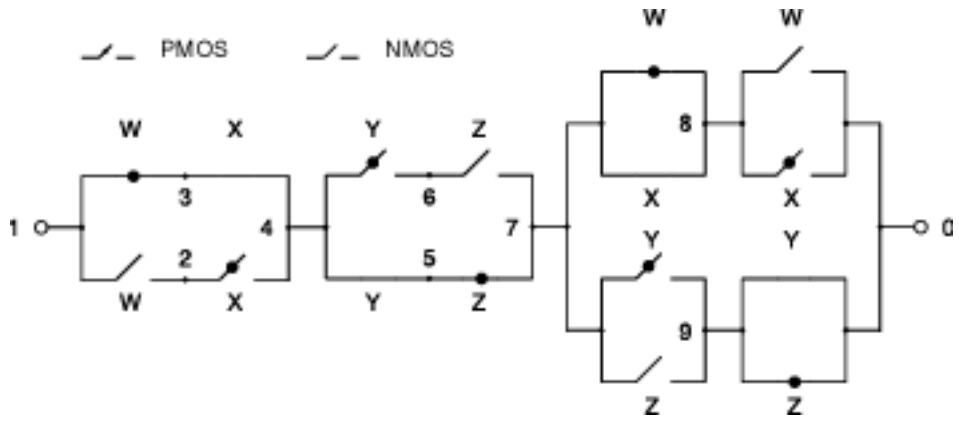

Figura 6.5: Circuito con ingresso (W, X, Y, Z) = (0, 1, 1, 0)

6.4 Esempio 4 di simulazione

File: circuit4.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a descrivere la topologia del circuito. È identico al precedente *circuit3.txt*.

File: ingress4.txt

Il file contiene nella sua parte finale tutte le indicazioni utili ad impostare gli ingressi del circuito, ovvero lo stato iniziale dei transistor.

```
4
w       0
x       0
y       1
z       1
```

File: uscit4.txt

Il file contiene nella sua parte finale tutte le indicazioni utili a prelevare le uscite del circuito. È identico al precedente *uscit3.txt*.

File: simulaz4.txt

Il risultato della simulazione viene salvato in questo file di testo.

```
Lettura file di descrizione del circuito: circuit2.txt ...completata!
Lettura file degli ingressi: ingress2.txt ...completata!
Lettura file delle uscite: uscit2.txt ...completata!
Simulazione circuitale a livello switch: Iniziata.
```

```
Stato iniziale degli switch a fronte dell'ingresso (w,x,y,z)=(0,0,1,1)

  Switch  Logica  PinA   PinB   Stato
  w       1 nmos  1      2        0
  w       1 nmos  8      0        0
  x       1 nmos  3      4        0
  x       1 nmos  7      8        0
  y       1 nmos  4      5        1
  y       1 nmos  9      0        1
  z       1 nmos  6      7        1
  z       1 nmos  7      9        1
  w       0 pmos  1      3        1
  w       0 pmos  7      8        1
  x       0 pmos  2      4        1
  x       0 pmos  8      0        1
  y       0 pmos  4      6        0
  y       0 pmos  7      9        0
  z       0 pmos  5      7        0
  z       0 pmos  9      0        0


------------------------------------------------------------ Iterazione 1 -
Matrice delle Connessioni M(0)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  0  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  0  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  0  0  1  1  0  0  0  0
  6   0  0  0  0  0  0  1  1  0  0
  7   0  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  0  1  1  0
  9   1  0  0  0  0  0  0  1  0  1
Matrice delle Connessioni M(1)=M(0)*M(0)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  1  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  1  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  1  0  1  1  0  0  0  0
  6   0  0  0  0  0  0  1  1  1  1
  7   1  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  1  1  1  1
  9   1  0  0  0  0  0  1  1  1  1


------------------------------------------------------------ Iterazione 2 -
Matrice delle Connessioni M(1)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  0  1  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  1  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  1  0  1  1  0  0  0  0
  6   0  0  0  0  0  0  1  1  1  1
```

```
  7   1  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  1  1  1  1
  9   1  0  0  0  0  0  1  1  1  1

Matrice delle Connessioni M(2)=M(1)*M(1)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  1  1  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  1  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  1  0  1  1  0  0  0  0
  6   1  0  0  0  0  0  1  1  1  1
  7   1  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  1  1  1  1
  9   1  0  0  0  0  0  1  1  1  1


------------------------------------------------------------ Iterazione 3 -
Matrice delle Connessioni M(2)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  1  1  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  1  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  1  0  1  1  0  0  0  0
  6   1  0  0  0  0  0  1  1  1  1
  7   1  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  1  1  1  1
  9   1  0  0  0  0  0  1  1  1  1

Matrice delle Connessioni M(3)=M(2)*M(2)
      0  1  2  3  4  5  6  7  8  9

  0   1  0  0  0  0  0  1  1  1  1
  1   0  1  0  1  0  0  0  0  0  0
  2   0  0  1  0  1  1  0  0  0  0
  3   0  1  0  1  0  0  0  0  0  0
  4   0  0  1  0  1  1  0  0  0  0
  5   0  0  1  0  1  1  0  0  0  0
  6   1  0  0  0  0  0  1  1  1  1
  7   1  0  0  0  0  0  1  1  1  1
  8   1  0  0  0  0  0  1  1  1  1
  9   1  0  0  0  0  0  1  1  1  1


I valori d'uscita sono:
  Uscita U0 = 0
  Uscita U1 = 1
  Uscita U2 = Z
  Uscita U3 = 1
  Uscita U4 = Z
  Uscita U5 = Z
  Uscita U6 = 0
  Uscita U7 = 0
  Uscita U8 = 0
  Uscita U9 = 0
```

```
Simulazione circuitale a livello switch: Terminata.
```

I valori di uscita calcolati dal simulatore sono direttamente verificabili in figura 6.6.

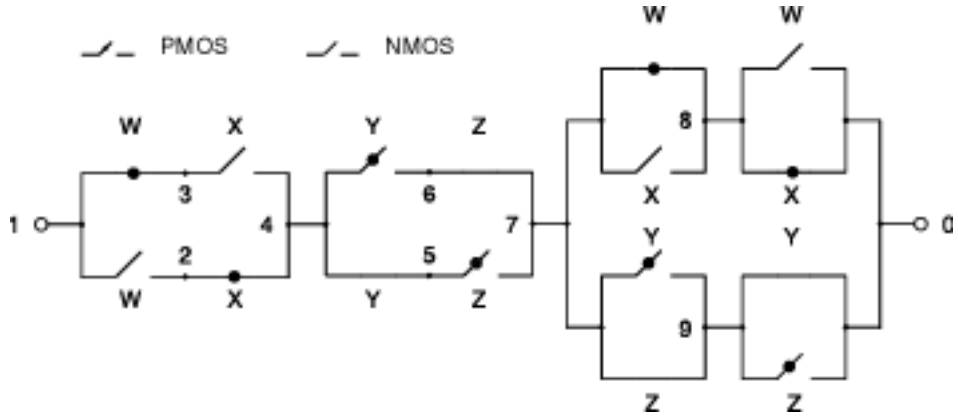

Figura 6.6: Circuito con ingresso (W, X, Y, Z) = (0, 0, 1, 1)

Appendice A

Codice sorgente del simulatore

Il codice del simulatore è contenuto nel file simcirc.cpp qui di seguito riportato.

```
/* Simulatore Circuitale a Livello di Switch - Ver. 1.0            */
/* Copyright (C) 2001 Tarin Gamberini                              */
/*                                                                 */
/* This program is free software; you can redistribute it and/or   */
/* modify it under the terms of the GNU General Public License     */
/* as published by the Free Software Foundation; either version 2  */
/* of the License, or (at your option) any later version.          */
/*                                                                 */
/* This program is distributed in the hope that it will be useful, */
/* but WITHOUT ANY WARRANTY; without even the implied warranty of  */
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the    */
/* GNU General Public License for more details.                    */
/*                                                                 */
/* You should have received a copy of the GNU General Public       */
/* License along with this program; if not, write to the Free      */
/* Software Foundation, Inc., 59 Temple Place - Suite 330,         */
/* Boston, MA 02111-1307, USA.                                     */


/* Simulatore Circuitale a Livello di Switch - Ver. 1.0            */
/* Copyright (C) 2001 Tarin Gamberini                              */
/*                                                                 */
/* Questo programma è software libero; è lecito redistribuirlo o   */
/* modificarlo secondo i termini della Licenza Pubblica Generica   */
/* GNU come è pubblicata dalla Free Software Foundation; o la      */
/* versione 2 della licenza o (a propria scelta) una versione      */
/* successiva.                                                     */
/*                                                                 */
/* Questo programma è distribuito nella speranza che sia utile, ma */
/* SENZA ALCUNA GARANZIA; senza neppure la garanzia implicita di   */
/* NEGOZIABILITÀ o di APPLICABILITÀ PER UN PARTICOLARE SCOPO. Si   */
/* veda la Licenza Pubblica Generica GNU per avere maggiori        */
/* dettagli.                                                       */
/*                                                                 */
/* Questo programma deve essere distribuito assieme ad una copia   */
/* della Licenza Pubblica Generica GNU; in caso contrario, se ne   */
/* può ottenere una scrivendo alla Free Software Foundation,       */
```

```
/* Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.    */


#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef struct tipoSwitch
{
 /* Nome simbolico dello switch */
 char nome[5];

 /* Logica dello switch: 1 Positiva NMOS, 0 Negativa PMOS */
 char logica;

 /* Indicano a quale nodo sono collegati i terminali dello switch */
 unsigned int  pinA,pinB;

 /* Chiuso o aperto a seconda della logica */
 char stato;
};


typedef struct tipoIO
{
 char nome[5];      /* Nome simbolico */
 int  valore;       /* Valore */
};


FILE *fileDescrittoreCircuito;
FILE *fileIngressi;
FILE *fileUscite;



unsigned int numNodiCircuito;  /* Dimensione matrice connessioni */
unsigned int numSwitch;    /* Dimensionevettore degli switch */
unsigned int numIngressi;    /* Numero di ingressi */
unsigned int numUscite;    /* Dimensione vettore delle uscite */


tipoSwitch *vetSwitch;    /* Vettore degli switch */
tipoIO   *vetIngressi;    /* Vettore degli ingressi */
tipoIO   *vetUscite;    /* Vettore delle uscite */
char   **matConnessioni;  /* Matrice delle connessioni */



void Inizializzazione(int argc, char **argv)
```

```
/* Controllo sintassi presenza file passati a linea di comando */
{

 /* Controllo sul numero di parametri */
 if (argc<4)
  {
   printf("Errore di sintassi: mancano alcuni parametri!\n");
   printf("Sintassi: simcirc <fileDescrizioneCircuito> <fileIngressi>
           <fileUscite>\n");
   exit(-1);
  }

 /* Verifica la presenza su disco del file di descrizione circuitale */
 if ((fileDescrittoreCircuito = fopen(argv[1],"rt")) == NULL)
   {
    fprintf(stderr, "Impossibile aprire %s\n",argv[1]);
    exit(-2);
   }
  else
    fclose(fileDescrittoreCircuito);

 /* Verifica la presenza su disco del file degli ingressi */
 if ((fileIngressi = fopen(argv[2],"rt")) == NULL)
   {
    fprintf(stderr, "Impossibile aprire %s\n",argv[2]);
    exit(-3);
   }
  else
    fclose(fileIngressi);

 /* Verifica la presenza su disco del file delle uscite */
 if ((fileUscite = fopen(argv[3],"rt")) == NULL)
   {
    fprintf(stderr, "Impossibile aprire %s\n",argv[3]);
    exit(-4);
   }
  else
    fclose(fileUscite);
}



void LeggiFileDescrittoreCircuito(char *nomeFileDescrittoreCircuito)
/* Carica il file di descrizione del circuito */
{
 unsigned int  i,j;

 fileDescrittoreCircuito = fopen(nomeFileDescrittoreCircuito,"rt");
 printf("Lettura file di descrizione del circuito: %s ...",
        nomeFileDescrittoreCircuito);

 /* Allocazione dinamica della matrice quadrata delle connessioni */
 fscanf(fileDescrittoreCircuito,"%i",&numNodiCircuito);
 matConnessioni=(char **) malloc(numNodiCircuito*sizeof(char *));
 for(i=0;i<numNodiCircuito;i++)
  matConnessioni[i]=(char *) malloc(numNodiCircuito*sizeof(char));

 /* Inizializzazione matrice delle connessioni "ad 1" */
 for (i=0; i<numNodiCircuito; i++)
  for (j=0; j<numNodiCircuito; j++)
    matConnessioni[i][j]=(i==j);
```

```
 /* Allocazione dinamica del vettore degli switch */
 fscanf(fileDescrittoreCircuito,"%i",&numSwitch);
 vetSwitch=(tipoSwitch *) malloc(numSwitch*sizeof(tipoSwitch));

 /* Carica nel vettore degli switch la descrizione del circuito */
 for (i=0; i<numSwitch; i++)
  fscanf(fileDescrittoreCircuito,"%s\t%i\t%i\t%i",vetSwitch[i].nome,
         &vetSwitch[i].logica,&vetSwitch[i].pinA,&vetSwitch[i].pinB);

 fclose(fileDescrittoreCircuito);
 printf("completata!\n");
}



void LeggiFileIngressi(char *nomeFileIngressi)
/* Carica il file degli ingressi */
{
 unsigned int  i;

 fileIngressi = fopen(nomeFileIngressi,"rt");
 printf("Lettura file degli ingressi: %s ...",nomeFileIngressi);
 fscanf(fileIngressi,"%i",&numIngressi);

 /* Allocazione dinamica del vettore degli ingressi */
 vetIngressi=(tipoIO *) malloc(numIngressi*sizeof(tipoIO));

 /* Carica nel vettore degli ingressi il file degli ingressi */
 for (i=0; i<numIngressi; i++)
  fscanf(fileIngressi,"%s\t%i",vetIngressi[i].nome,&vetIngressi[i].valore);

 fclose(fileIngressi);
 printf("completata!\n");
}



void LeggiFileUscite(char *nomeFileUscite)
/* Carica il file delle uscite */
{
 unsigned int  i;

 fileUscite = fopen(nomeFileUscite,"rt");
 printf("Lettura file delle uscite: %s ...",nomeFileUscite);
 fscanf(fileUscite,"%i",&numUscite);

 /* Allocazione dinamica del vettore delle uscite */
 vetUscite=(tipoIO *) malloc(numUscite*sizeof(tipoIO));

 /* Carica nel vettore delle uscite il file delle uscite */
 for (i=0; i<numUscite; i++)
  fscanf(fileUscite,"%s\t%i",vetUscite[i].nome,&vetUscite[i].valore);

 fclose(fileUscite);
 printf("completata!\n");
}
```

```
void StampaVetSwitch()
/* Stampa il vettore degli switch */
{
 unsigned int i;
 char   strVetIngr[255],strVal[2];

 clrscr();

 /* Costruisce la stringa del vettore di ingresso */
 strcpy(strVetIngr,"(");
 for (i=0; i<numIngressi; i++)
 {
  strcat(strVetIngr,vetIngressi[i].nome);
  (i==numIngressi-1 ? strcat(strVetIngr,")") : strcat(strVetIngr,","));
 }
 strcat(strVetIngr,"=(");
 for (i=0; i<numIngressi; i++)
 {
  itoa(vetIngressi[i].valore,strVal,10);
  strcat(strVetIngr,strVal);
  (i==numIngressi-1 ? strcat(strVetIngr,")") : strcat(strVetIngr,","));
 }

 /* Visualizza lo stato iniziale degli switch del circuito */
 printf("Simulazione circuitale a livello switch: Iniziata.\n\n\n");
 printf("Stato iniziale degli switch a fronte dell'ingresso %s\n\n",strVetIngr);
 printf(" Switch Logica PinA  PinB  Stato\n");
 for (i=0; i<numSwitch; i++)
 {
  printf(" %s",vetSwitch[i].nome);
  printf("    %i %s",vetSwitch[i].logica,(vetSwitch[i].logica ? "nmos" : "pmos"));
  printf(" %i",vetSwitch[i].pinA);
  printf("   %i",vetSwitch[i].pinB);
  printf("    %i\n",vetSwitch[i].stato);
 }
 printf("\n\n");
 getc(stdin);
}



void SettaStatoInizialeSwitch()
/* Setta lo stato degli switch in base alla loro logica di funzionamento ed
  in base al valore degli ingressi */
{
 unsigned int i,j;

 for (j=0; j<numIngressi; j++)
  for (i=0; i<numSwitch; i++)
   if (!strcmp(vetSwitch[i].nome,vetIngressi[j].nome))
    if (vetSwitch[i].logica)
      vetSwitch[i].stato=vetIngressi[j].valore;
     else
      vetSwitch[i].stato=!vetIngressi[j].valore;

 /* Stampa il vettore degli switch */
 StampaVetSwitch();
}
```

```
void SettaStatoInizialeCircuito()
/* Setta lo stato iniziale del circuito ponendo un 1 alle coordinate [i][j]
  e [j][i] della matrice delle connessioni solo se non vi era gia' un 1 e
  solo se esiste uno switch chiuso fra i nodi i e j della rete */
{
 unsigned int i;

 for (i=0; i<numSwitch; i++)
  if (matConnessioni[vetSwitch[i].pinA][vetSwitch[i].pinB]==0)
   {
    matConnessioni[vetSwitch[i].pinA][vetSwitch[i].pinB]=vetSwitch[i].stato;
    matConnessioni[vetSwitch[i].pinB][vetSwitch[i].pinA]=vetSwitch[i].stato;
   }
}
```

```
void ProdottoLogicoMatriciale(char **newMat, char **oldMat)
/* Calcola M(k+1)=M(k)*M(k) dove il "*" e' il prodotto logico matriciale */
{
 unsigned int i,j,k;
 char   prodottoLogico;   /* Flag di uscita dal ciclo piu' interno */

 for (i=0; i<numNodiCircuito-1; i++)
  for (j=i+1; j<numNodiCircuito; j++)
  {
   k=0;
   prodottoLogico=0;
   while (k<numNodiCircuito && !(prodottoLogico=oldMat[i][k]&&oldMat[k][j]))
    k++;
   newMat[i][j]=newMat[j][i]=prodottoLogico;
  }
}
```

```
char MatriciDiverse(char **matA, char **matB)
/* Ritorna 1 o 0 a seconda che A!=B o A=B */
{
 unsigned int i,j;
 char   diverse;   /* Flag di uscita dai cicli */

 diverse=0;
 i=0;
 while (i<numNodiCircuito-1 && !diverse)
 {
  j=i+1;
  while(j<numNodiCircuito && !(diverse=matA[i][j]!=matB[i][j]))
   j++;
  i++;
 }
 return(diverse);
}
```

```
void StampaMatrici(char **prodottoMatConnessioni, unsigned int k)
```

```
/* Visualizza la matrice delle connessioni M(k) ed M(k+1) */
{
 unsigned int i,j;

 printf("-------------------------------------
        ------------------- Iterazione %i -----\n",k);
 /* Visualizza la matrice delle connessioni al passo k-esimo M(k) */
 printf("Matrice delle Connessioni M(%i)\n",k-1);

 /* Visualizza il numero del nodo in cima alla matrice */
 printf("  ");
 for (j=0; j<numNodiCircuito; j++)
  printf(" %i",j);
 printf("\n\n");

 /* Visualizza la matrice delle connessioni */
 for (i=0; i<numNodiCircuito; i++)
 {
  printf(" %i ",i);
  for (j=0; j<numNodiCircuito; j++)
   printf(" %i",matConnessioni[i][j]);
  printf("\n");
 }

 /* Visualizza la matrice delle connessioni al passo k+1esimo M(k+1)=M(k)*M(k) */
 printf("\nMatrice delle Connessioni M(%i)=M(%i)*M(%i)\n",k,k-1,k-1);

 /* Visualizza il numero del nodo in cima alla matrice */
 printf("  ");
 for (j=0; j<numNodiCircuito; j++)
  printf(" %i",j);
 printf("\n\n");

 /* Visualizza la matrice M(k+1)=M(k)*M(k) */
 for (i=0; i<numNodiCircuito; i++)
 {
  printf(" %i ",i);
  for (j=0; j<numNodiCircuito; j++)
   printf(" %i",prodottoMatConnessioni[i][j]);
  printf("\n");
 }
 printf("\n\n");
 getc(stdin);
}




void AnalizzaCollegamentiCircuito()
/* Verifica i collegamenti del circuito effettuando iterativamente il
  calcolo M(k+1)=M(k)*M(k) e termnando solo quando M(k+1)=M(k) */
{
 unsigned int i,j;
 unsigned int k;    /* Conta il numero delle iterazioni M(k+1)=M(k)^2 */
 char   **prodottoMatConnessioni;     /* Memorizza M(k+1) */
 char   **tmpMat;     /* Sfruttata per scambiare le matrici */

 /* Allocazione dinamica della nuova matrice delle connessioni */
 prodottoMatConnessioni=(char **) malloc(numNodiCircuito*sizeof(char *));
 for(i=0;i<numNodiCircuito;i++)
  prodottoMatConnessioni[i]=(char *) malloc(numNodiCircuito*sizeof(char));
```

```
 /* Inizializzazione nuova matrice delle connessioni "ad 1" */
 for (i=0; i<numNodiCircuito; i++)
  for (j=0; j<numNodiCircuito; j++)
   prodottoMatConnessioni[i][j]=(i==j);


 k=0;
 /* M(k+1)=M(k)*M(k) */
 ProdottoLogicoMatriciale(prodottoMatConnessioni,matConnessioni);
 StampaMatrici(prodottoMatConnessioni,++k);
 while (MatriciDiverse(prodottoMatConnessioni,matConnessioni))
 {
  /* M(k)=M(k+1) */
  tmpMat=matConnessioni;
  matConnessioni=prodottoMatConnessioni;
  prodottoMatConnessioni=tmpMat;

  /* M(k+1)=M(k)*M(k) */
  ProdottoLogicoMatriciale(prodottoMatConnessioni,matConnessioni);
  StampaMatrici(prodottoMatConnessioni,++k);
 }
 free(prodottoMatConnessioni);
}



void CalcolaValoreDelleUsite()
/* Calcola il valore delle uscite {0,1,X,Z}
  vetUscite[i].valore contiene il numero simbolico del nodo del circuito
  che si vuole considerare come uscita */
{
 unsigned int i;

 printf("I valori d'uscita sono:\n");
 for (i=0; i<numUscite; i++)
 {
  if (matConnessioni[vetUscite[i].valore][0])
    if (matConnessioni[vetUscite[i].valore][1])
      /* Nodo collegato a massa e all'alimentazione: in cortocircuito */
      printf(" Uscita %s = X",vetUscite[i].nome);
     else
      /* Nodo collegato a massa */
      printf(" Uscita %s = 0",vetUscite[i].nome);
   else
    if (matConnessioni[vetUscite[i].valore][1])
      /* Nodo collegato all'alimentazione */
      printf(" Uscita %s = 1",vetUscite[i].nome);
     else
      /* Nodo ne' collegato a massa ne' all'alimentaz: alta impedenza */
      printf(" Uscita %s = Z",vetUscite[i].nome);
 printf("\n");
 }
 printf("\n\nSimulazione circuitale a livello switch: Terminata.\n");
 getc(stdin);
}



void Dealloca()
/* Libera lo spazio in memoria allocato per vettori e matrice dinamici */
```

```
{
 free(vetSwitch);
 free(vetIngressi);
 free(vetUscite);
 free(matConnessioni);
}



main(int argc, char **argv)
{
 Inizializzazione(argc,argv);

 LeggiFileDescrittoreCircuito(argv[1]);
 LeggiFileIngressi(argv[2]);
 LeggiFileUscite(argv[3]);

 SettaStatoInizialeSwitch();
 SettaStatoInizialeCircuito();

 AnalizzaCollegamentiCircuito();
 CalcolaValoreDelleUsite();

 Dealloca();
 return 0;
}
```

Appendice B

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as

"you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To

"Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.