Modelli VHDL per simulazioni a basso livello

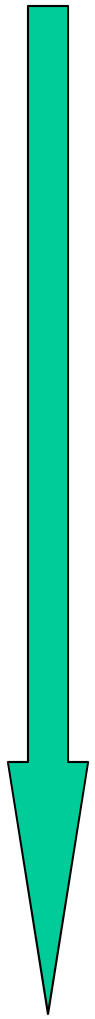
- Per simulare reti il cui comportamento non può essere descritto in termini puramente logici (0, 1) il VHDL mette a disposizione la possibilità descrivere insiemi di valori logici (“algebra” in maniera molto impropria) molto più complessi
- Il VHDL non dispone di primitive per simulare reti al livello switch

Modelli VHDL per simulazioni a basso livello

- Ci sono due livelli di problemi:
  - simulazioni in cui si possono individuare dei segnali pilotati da driver: estensione dell'insieme di valori logici e bus resolution function
  - simulazioni in cui non si hanno più elementi unidirezionali per controllare i segnali, ma il transistore (bidirezionale) diventa il componente fondamentale per determinare il valore dei segnali: simulazione livello switch

Modello “logico” dei segnali

- Valori logici: 0,1
- Segnali non inizializzati o conflitti: X
- Nodi in stati ad alta impedenza: Z
- Non si possono modellare bus o segnali il cui valore è determinato sulla base dell’esito di conflitti di conduttanze o che presentano capacità di hold
- Concetto di “drive strength” di un segnale



reti logiche

tristate bus

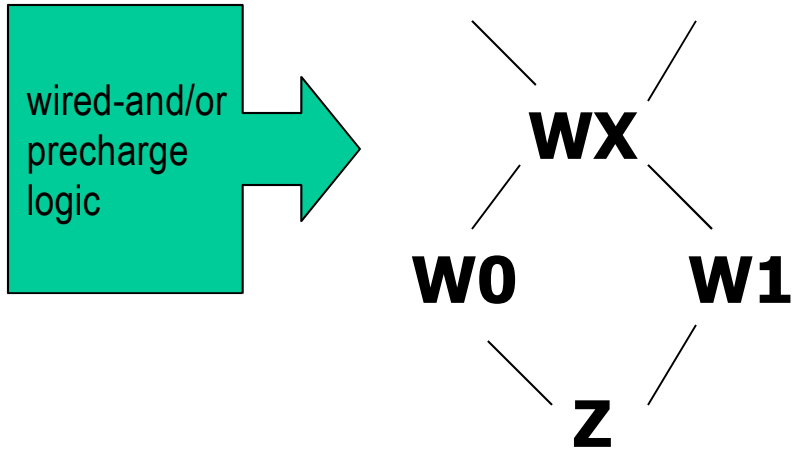
altri tipi di bus

Modello dei segnali basato su reticoli (lattice)

- Stato logico: A={0,1,X,Z}
- Drive strength: G={g0,g1, …, gn}
- Valore di un segnale V=AxG
- Si ottiene un insieme parzialmente ordinato
  - si dice che un elemento a di V ne copre un altro b se a>b e non esiste x tale che a>x>b

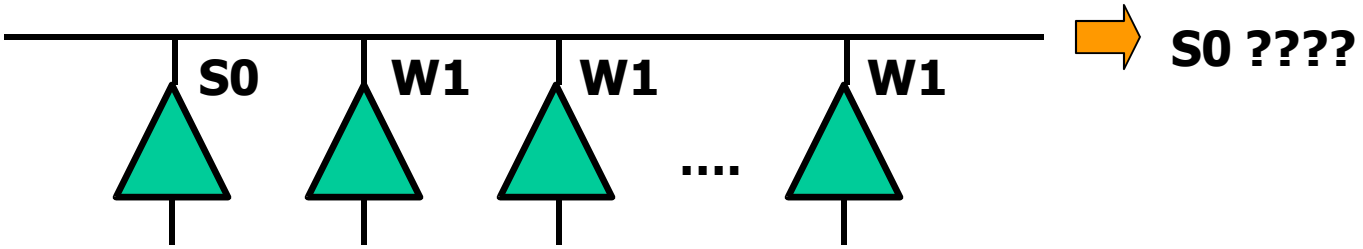
Modello dei segnali basato su reticoli (lattice)

- Diagramma di Hasse = rappresentazione grafica della relazione di copertura
- Algebra di Bryant:
  - A={0,1,X,Z}
  - G={S,W}



Applicazioni

- Il lattice è in pratica una bus resolution function per un bus
- Limitazioni:
  - transistori o driver di cui non si conosce la strength
  - non si cumulano gli effetti



- Simulazioni al livello switch

Package: **ieee_std_logic_1164.t_wlogic**

- Signal strengths:
  - Forced (F)
  - Strong resistive (R)
  - Weak resistive (W)
  - High impedance (Z)
  - Disconnect (D)

un valore di drive non noto può variare in qualsiasi intervallo di valori di strength

  - distinzione fra uno stato non valido e uno valido ma non noto

Package: **ieee_std_logic_1164.t_wlogic**

- Un valore di drive non noto per un buffer che pilota un valore noto è caratterizzato come:
  - b massima strength con cui si pilota K
  - d minima strength con cui si pilota K
  - K={0,1}

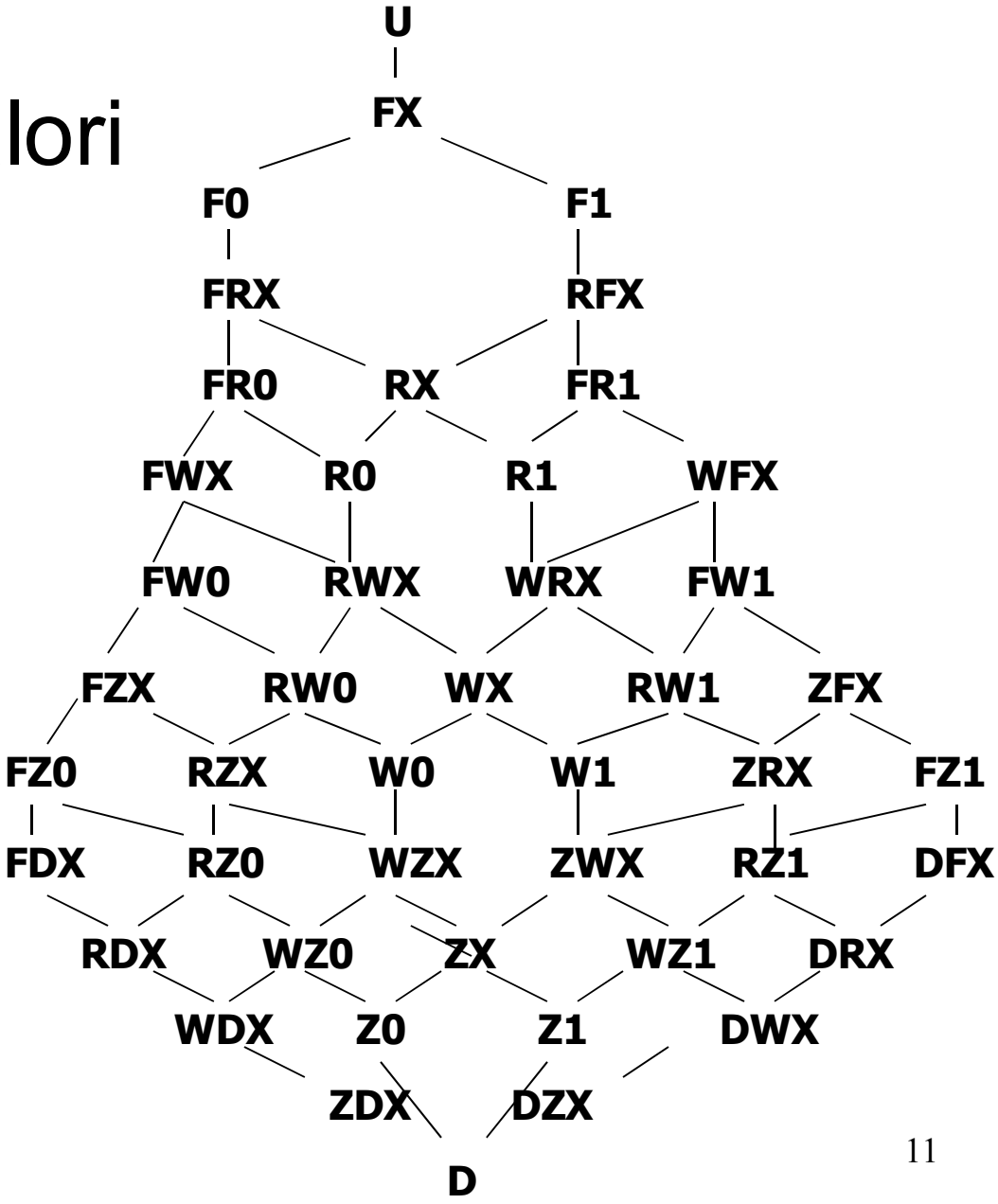
Package: **ieee_std_logic_1164.t_wlogic**

- Un valore di drive non noto per un buffer che pilota un valore anch'esso non noto è caratterizzato come:
  - p massima strength con cui si può pilotare uno 0
  - q massima strength con cui si può pilotare un 1
  - X

Package: **ieee_std_logic_1164.t_wlogic**

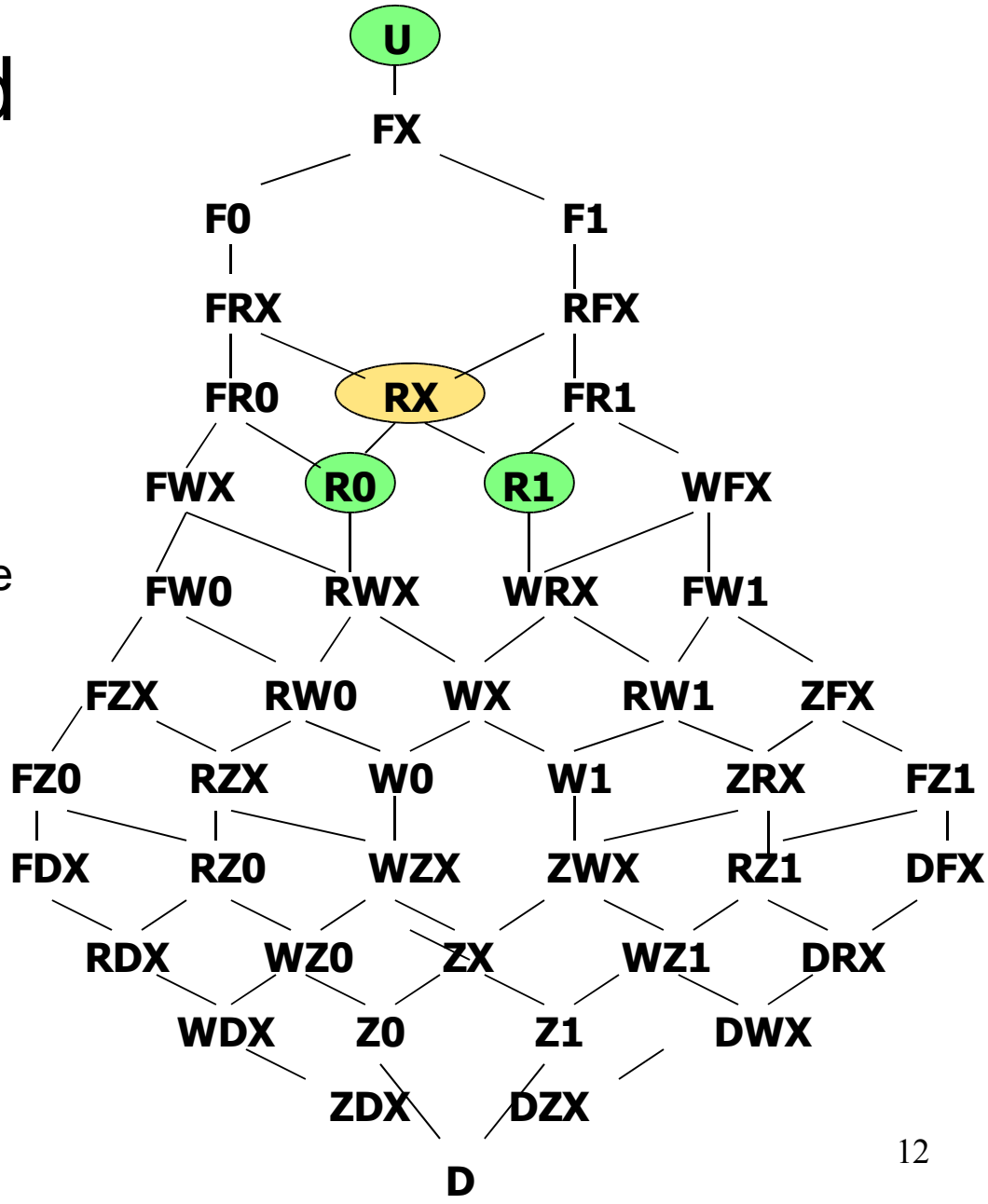
- Complessivamente si ottengono 46 valori
- Rimane la limitazione di non riuscire a cumulare gli effetti di diversi driver in parallelo

t_wlogic = 46 valori

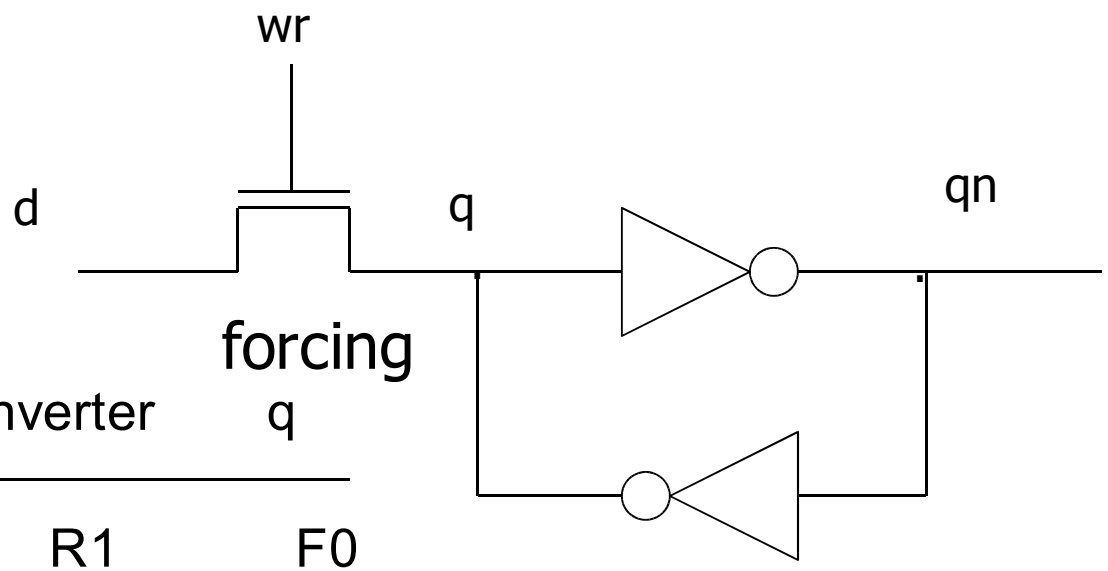


logica wired-and (nMOS ratioed)

In presenza di due valori, il valore in cui viene risolto il conflitto è dato dal least upper bound (ovvero il valore più piccolo fra quelli > dei due valori in conflitto)



Esempio - cella di RAM statica



pass transistor	inverter	q
F0	R1	F0
F0	R0	F0
F0	RX	F0
F1	R1	F1
F1	R0	F1
F1	RX	F1

Simulazione livello switch

- Motivazioni
  - non tutti i sistemi digitali sono ben descrivibili al livello gate
    - logiche dinamiche
    - logiche a pass transistor
    - logiche differenziali
  - il livello elettrico (analogico) risulta inefficiente nella simulazione di sistemi di grandi dimensioni

Modello switch del transistore MOS

- In molti sistemi digitali il comportamento dei transistori MOS può essere ben approssimato in maniera digitale
- Stati: ON/OFF
- In realtà sono necessarie maggiori informazioni riguardo conduttanza dei dispositivi e capacità dei nodi

Simulazione al livello switch

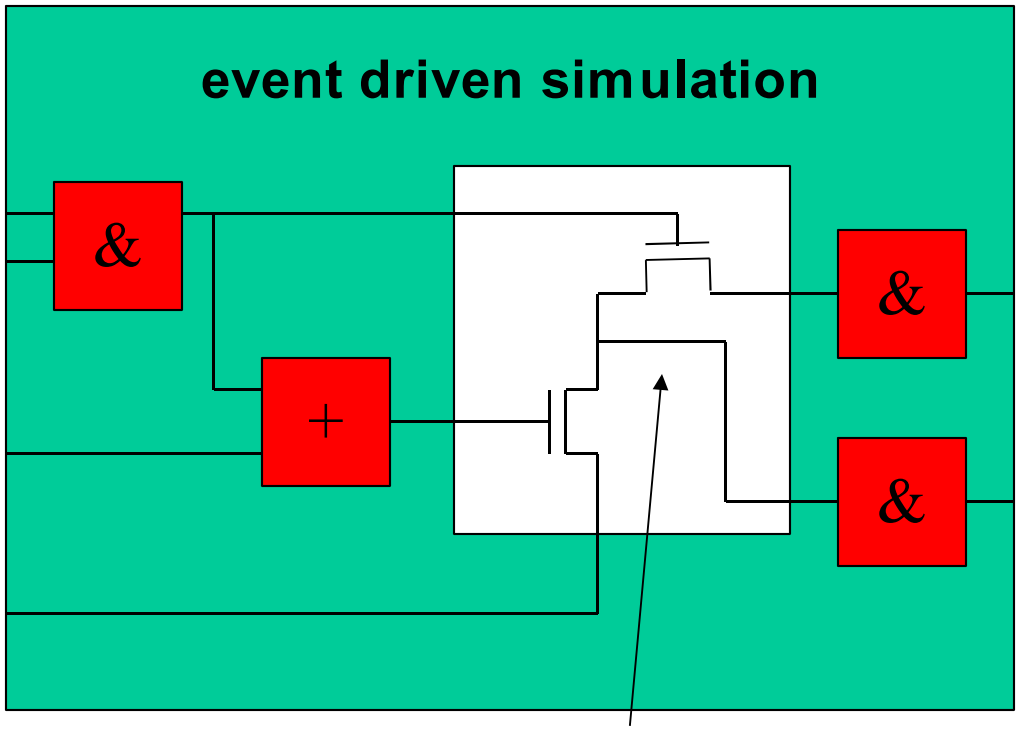
- Gate level
- Switch level
  - logic (0,1,X,Z)
  - discrete strengths
- Electrical level
  - RC approximate (relaxation methods)
  - SPICE

mixed-mode

Simulazione al livello switch in VHDL

- Il VHDL presenta rilevanti problemi nel trattare componenti bidirezionali come gli switch
- La valutazione dello stato di una rete di switch può richiedere diverse iterazioni per arrivare a un punto di lavoro fisso per il circuito in esame: **f(x)=x**
- Tali iterazioni non corrispondono a relazioni di causa effetto reali e quindi non devono comparire nella simulazione event-driven

Simulazione mixed-mode



Simulazione al livello switch in VHDL

- Il VHDL mette a disposizione un meccanismo per calcolare lo stato di un componente al livello switch senza coinvolgere la gestione degli eventi nel dominio dei tempi
- Questo è dato dal ritardo delta con il quale non è necessario separare la simulazione dei dispositivi MOS da quella event driven
- Le iterazioni necessarie per calcolare lo stato della rete al livello switch sono nascoste nei cicli delta
- Il VHDL non riesce a mettere in gioco considerazioni di tipo globale => metodi di rilassamento locali

Modello switch

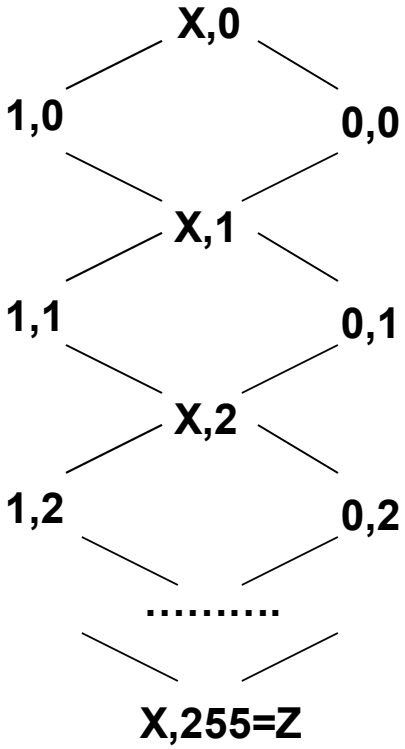
- Obbiettivo: non considerare le istanze a livello switch come casi particolari
- Nuove primitive per il simulatore
  - switch (p,n) (porte di tipo inout)
  - node (che sono equivalenti a segnali con risoluzione del VHDL)
- O in alternativa, la descrizione di processi che le implementano
- weakness values [0….255] => weakest
- Questi valori non sono trasmessi oltre i confini della rete al livello switch

Valutazione

- Non è diversa da quella al livello gate, tutto però è contenuto nei cicli delta
- Quando uno switch ON viene valutato trasmette il suo “voto” (in due direzioni) per il valore dei nodi cui è connesso
- Per simulare la resistività dei dispositivi reali, la weakness dei segnali viene incrementata a ogni dispositivo attraversato
- La funzione di risoluzione dei nodi aggiorna il valore di tali segnali
- La simulazione si ferma quando è stato raggiunto uno stato stabile

Risoluzione dei nodi

- Metodo least upper bound su un reticolo



Limitazioni: le resistenze in un circuito reale si compongono in maniera globale, il calcolo della weakness dei driver è del tutto locale (altrimenti non sarebbe simulabile in VHDL). La funzione di risoluzione non tiene conto del numero di driver.

Package

```
package bit_types is;
type bit_code is ('0', '1', 'X', 'Z');
type bit_code_vector is array (natural range <>) of weakness value;
type weakness_value is range 0 to 255;
type switch_state is (mos_on, mos_off, mos_x);
procedure lattice_lub(new_value: inout bit_code;
                      new_weakness: inout weakness_value;
                      value: in bit_code;
                      weakness: in weakness_value);
function new_node_wins(new_value: bit_code;
                       new_weakness: weakness_value;
                       old_value: bit_code;
                       old_weakness: weakness_value) return boolean;
end bit_types;
```

Switch (I)

```
use work.bit_types.all;

entity nswitch is
  generic (chann_degr: weakness_value := 1);
  port (gate: in bit code;
        source, drain: inout bit_code;
        source_weakness, drain_weakness: inout weakness_value);
end nswitch;

architecture arch of nswitch is
signal state: switch_state;
begin
process (gate,ource, drain, state)
begin
  if (gate'event) then
    case gate is
    when '0' => state <= mos_off;
    when '1' => state <= mos_on;
    when others => state <= mos_x;
    end case;
  end if;
```

Switch (II)

```
case state is
  when mos_off => drain <= 'X';
                  drain_weakness <= weakness_value'high;
                  source <= 'X';
                  source_weakness <= weakness_value'high;
  when mos_on => drain <= source;
                 drain_weakness <= source_weakness + channel_degr;
                 source <=drain;
                 source_weakness <= source_weakness + channel_degr;
  when mos_x => drain <= 'X';
                drain_weakness <= source_weakness + channel_degr;
                source <= 'X';
                source_weakness <= source_weakness + channel_degr;
  end case;
end process;
end arch;
```

Node (I)

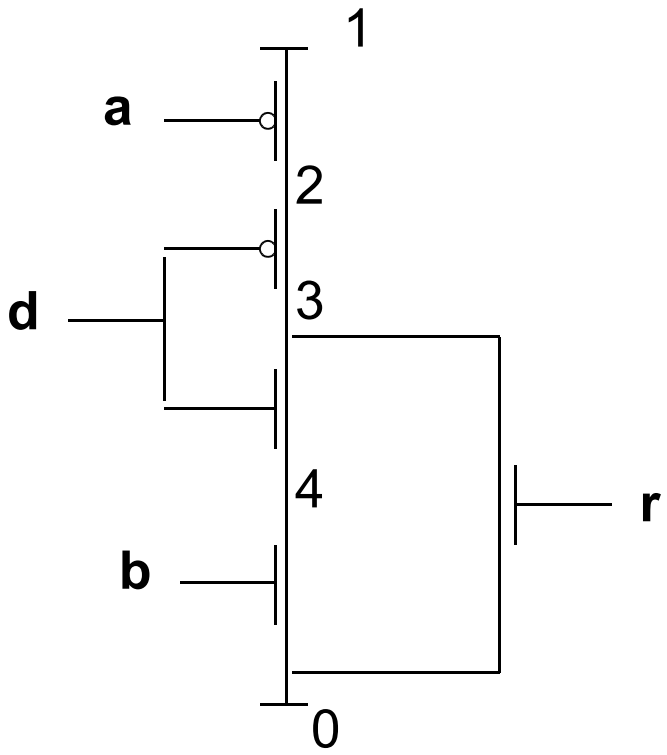
```
use work.bit_types.all;
entity ideal_node is
  port (states: inout bit_code_vector;
        strenghts: inout weakness_value_vector);
end ideal_node;

architecture arch of ideal_node is
begin
  process (states,strenghts)
  variable node_value, result_value: bit_code;
  variable node_weakness, result_weakness: weakness_value;
  begin
    result_value := 'X';
    result_value := weakness_value'high;

    for i in states'range loop
      lattice_lub(result_value,result_weakness,states(i),strenghts(i));
    end loop;
```

Node (II)

```
if new_node_wins(result_value,result_weakness,node_value,node_weakness)
    then
  for i in states'range loop
     if new_node_wins(result_value,result_weakness,
            states(i),strenghts(i)) then
       states(i) <= result_value;
       strength(i) <= result_weakness;
     end if;
  end loop;
 end if;
 node := result_value;
 node_weakness := result_weakness;
end process;
end arch;
```

Alternativa

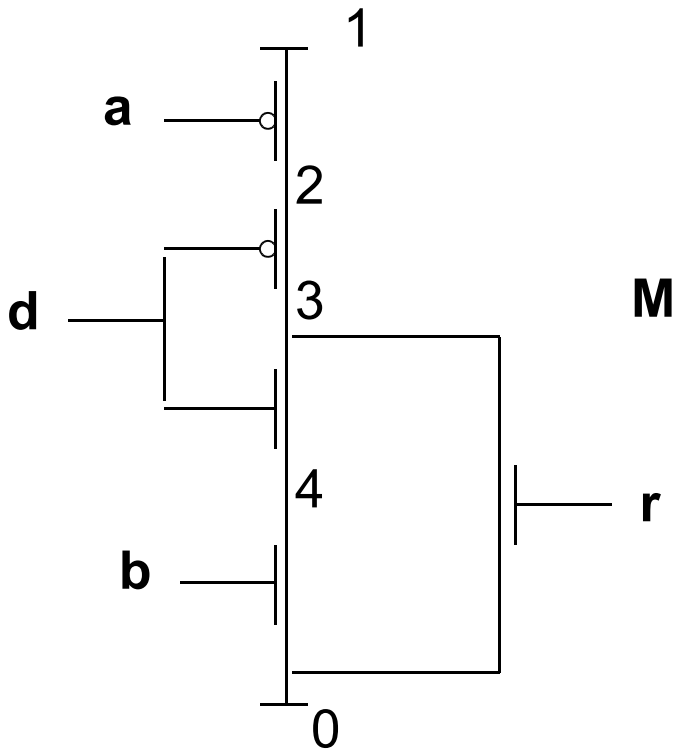
- Metodi di rilassamento simbolici
- Stato del sistema: matrice booleana di interconnessioni



$$M = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & r & b \\ 0 & 1 & a' & 0 & 0 \\ 0 & a' & 1 & d' & 0 \\ r & 0 & d' & 1 & d \\ b & 0 & 0 & d & 1 \end{pmatrix} \end{array}$$

Modello simbolico

- Applicazione della legge transitiva: **M(k+1) =MxM(k)**
- Fino a quando: **M(k+1)=M(k)**



M =

	0	1	2	3	4
0	1	0	0	bd+r	bd
1	0	1	a’	a’d’	0
2	d’r	a’	1	d’	0
3	bd+r	a’d’	d’	1	d
4	b	0	0	d	1

Modello VHDL