Algoritmi di test generation

M. Favalli

Engineering Department in Ferrara

Introduzione

Algoritmi di test generation

- Algoritmo D
- PODEM

Algoritmo D

- Storicamente il primo algoritmo completo ad essere stato sviluppato
- Lavora su una descrizione strutturale del circuito attraverso le fasi di:
  - propagazione
  - giustificazione
  - implicazione
- Queste fasi vengono gestite tramite il D-calculus e lo spazio delle possibili soluzioni viene esplorato tramite un push-down stack

D-calculus

- Viene gestito da operazioni descritte da tabelle ottenute a partire dalla descrizione dei gate nell'algebra di Roth

Esempio: gate di tipo AND

- tabella di forward implication

a \ b	0	1	D	$\overline{D}$	X
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	X
D	0	D	D	0	X
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	X
X	0	X	X	X	X

- singular cover (giustificazione): minimo numero di assegnamenti per giustificare un valore in uscita

a	b	out
0	X	0
X	0	0
1	1	1

D-calculus

- D-cube: propagazione

a	b	out
1	D	D
D	1	D
D	D	D
1	$\overline{D}$	$\overline{D}$
1	$\overline{D}$	$\overline{D}$
$\overline{D}$	$\overline{D}$	$\overline{D}$

- D-intersection: operazione di consistenza durante l'implicazione
- Queste operazioni sono applicate durante l'esecuzione dell'algoritmo

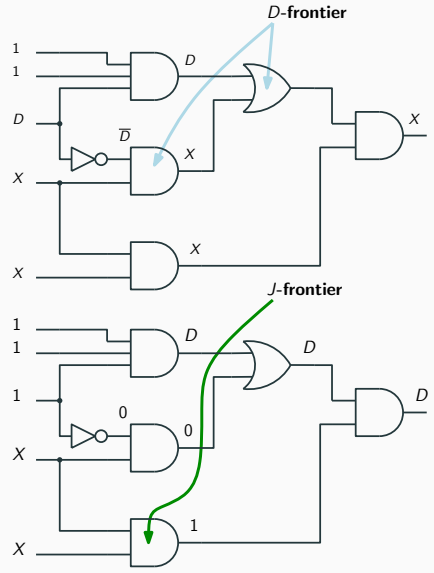
Algoritmo D: versione semplificata

- La versione che vedremo dell'algoritmo D é piú semplice di quella proposta da Roth ed é piú consistente con quella degli algoritmi seguenti
- Definizioni:
  - **D-frontier**: insieme dei gate con un valore a D in ingresso e il valore di uscita ancora a X
  - **J-frontier**: tutti i gate del circuito la cui uscita é a un valore noto (ovvero diverso da X), ma non é giustificato dai valori degli ingressi

Esempi

D-frontier

J-frontier



Algoritmo-D

1. Inietta il guasto portando a D o $\overline{D}$ il segnale guasto
2. Propaga il valore D verso le uscite primarie (D-drive) fino a quando un valore D (ovvero la D-frontiera) non raggiunge un PO
   - nel fare questo vengono fatte delle scelte pilotate dall'osservabilitá
   - contemporaneamente vengono assegnati dei valori sui side input dei gate della D-frontiera (sulla base dei D-cubi di propagazione) i gate che producono tali segnali vengono aggiunti alla J-frontiera

Algoritmo-D

3. L'algoritmo procede poi giustificando i valori della J-frontiera fino a portarla sugl ingressi
4. Durante ogni assegnamento viene verificata la consistenza dei valori dei segnali e l'esistenza della D-frontiera
5. Altrimenti viene fatta un operazione di backtrack in cui una delle scelte fatte viene cambiata. L'assenza di alternative porta a definire il guasto come non rivelabile

Algoritmo D

```
D_Algorithm(C,f,v) /* line f stuck-at-v */
{
  forall i in C
    out(i)='X';

  D-frontier=∅;
  J-frontier=∅;

  if (v=='0') /* inject the fault */
    out(f)='D';
  else if (v=='1')
    out(f)='nD';

  result=D-Algorithm-recursive(C);
  if (result==success)
    print (out(PI), out(PO));
  else
    print f stuck-at-v is untestable;
}
```


Algoritmo D

```
outcome D-Algorithm-recursive(C)
{
  if (conlict in assignment) or (D-frontier==∅)
    return failure;
  if (¬∃ PO out(PO)='D' or 'nD')
    while not all gates in the D-frontier have been tried
      {
        g=a gate in the D frontier that has not been tried;
        set all unassigned inputs of g to non-controlling values
           and add them to the J-frontier
        result=D-Algorithm-recursive(C);
        if (result==success)
          return success;
      }
  return failure;
  if (J-frontier==∅)
    return success;
}
```

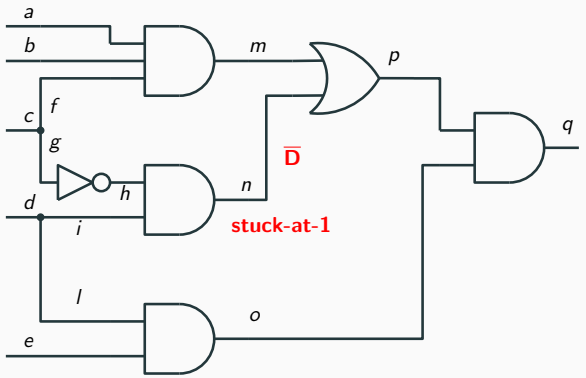
Algoritmo D

```
  g= a gate in the J-frontier;
  while g has not been justified
    {
      j=unassigned input of g;
      set j=1 and insert j in the J-frontier:
      result=D-Algorithm-recursive(C);

      if (result==success)
        return success;
      else
        out(j)='0';
      end if;
    }
  return failure;
}
```


Esempio

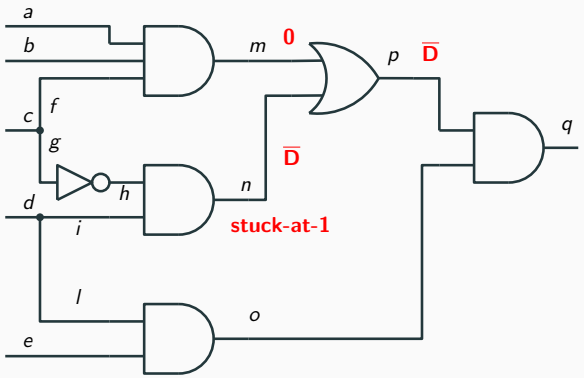
passo 1: iniezione di guasto



nota: i segnali non annotati con un valore si intendono a X

Esempio

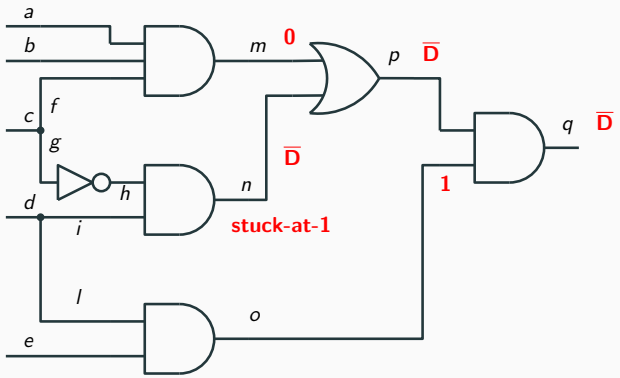
passo 2: *D*-**drive**



nota: i segnali non annotati con un valore si intendono a X

Esempio

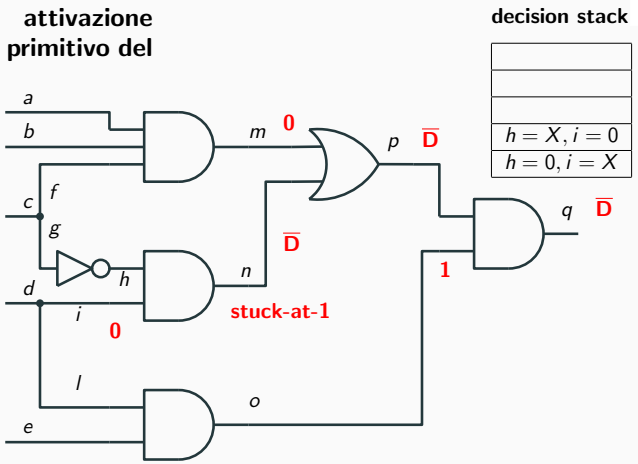
passo 3: *D*-**drive**



nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 4: attivazione (D-cubo primitivo del guasto)



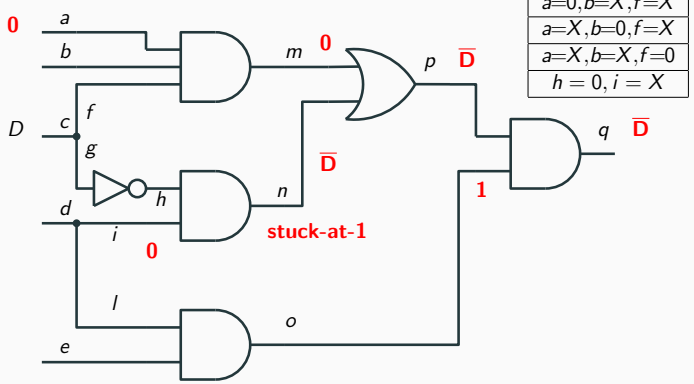
decision stack

$h = X, i = 0$
$h = 0, i = X$

nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 5: giustificazione (singular cover)



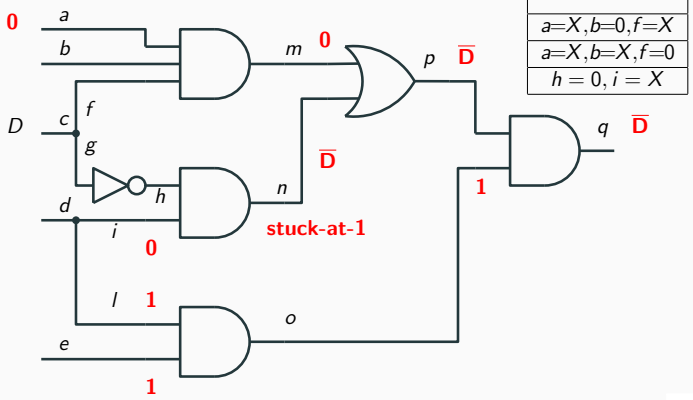
decision stack

a=0,b=X,f=X
a=X,b=0,f=X
a=X,b=X,f=0
$h = 0, i = X$

nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 6: giustificazione (singular cover)



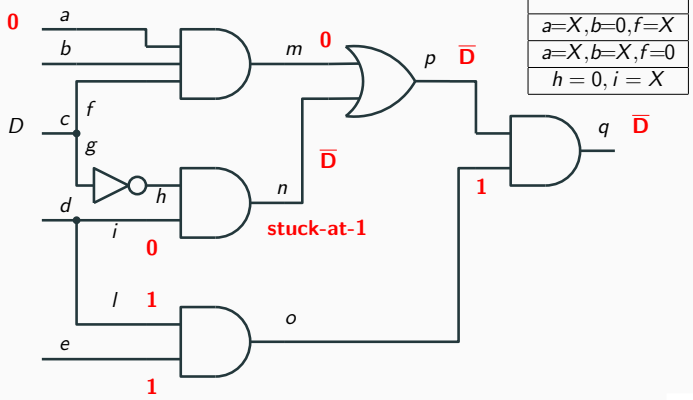
decision stack

a=X,b=0,f=X
a=X,b=X,f=0
$h = 0, i = X$

nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 7: consistenza $\Rightarrow$ conflitto su d



decision stack

a=X,b=0,f=X
a=X,b=X,f=0
$h = 0, i = X$

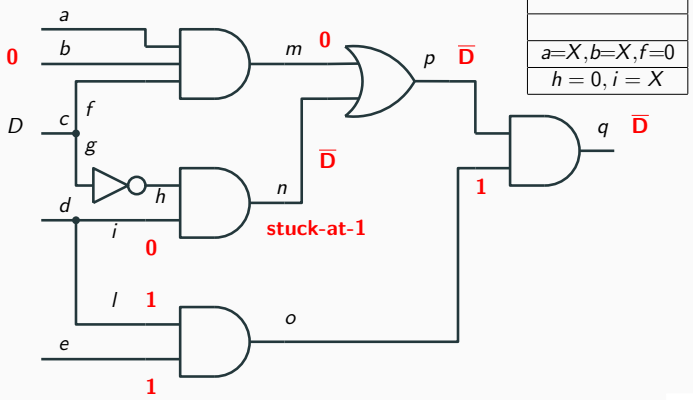
nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 8: backtrack

decision stack

$a{=}X, b{=}X, f{=}0$
$h = 0, i = X$



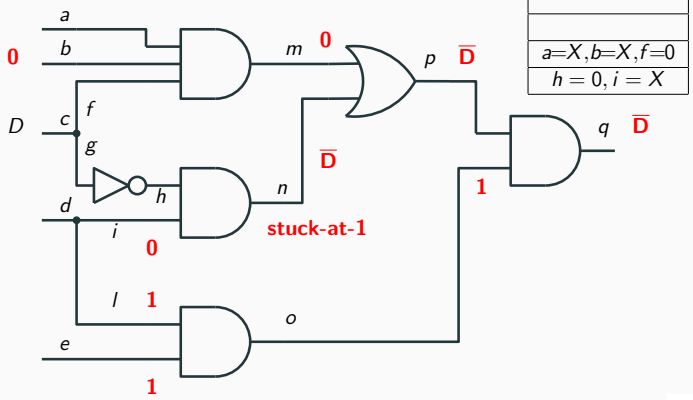
nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 9: consistenza, rimane il conflitto

decision stack

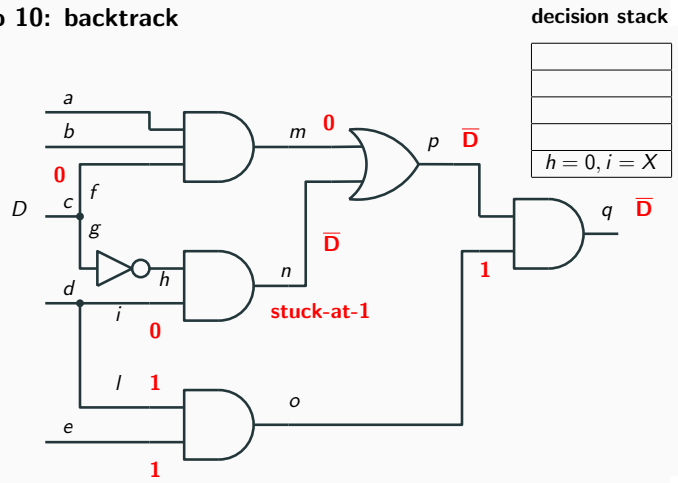
$a{=}X, b{=}X, f{=}0$
$h = 0, i = X$



nota: i segnali non annotati con un valore si intendono a X

Esempio

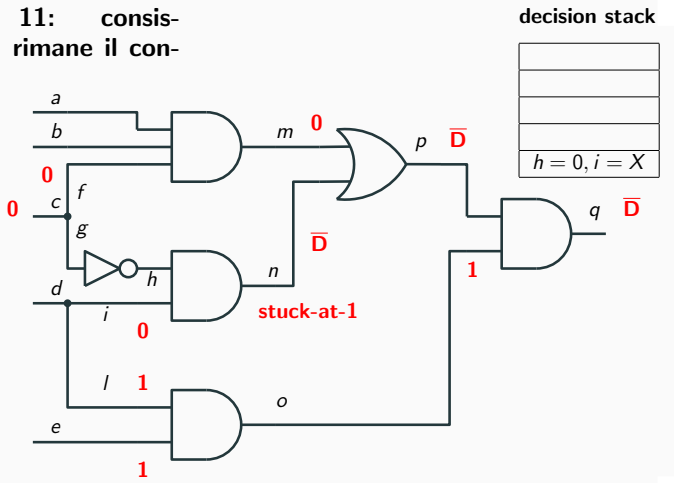
passo 10: backtrack



nota: i segnali non annotati con un valore si intendono a X

Esempio

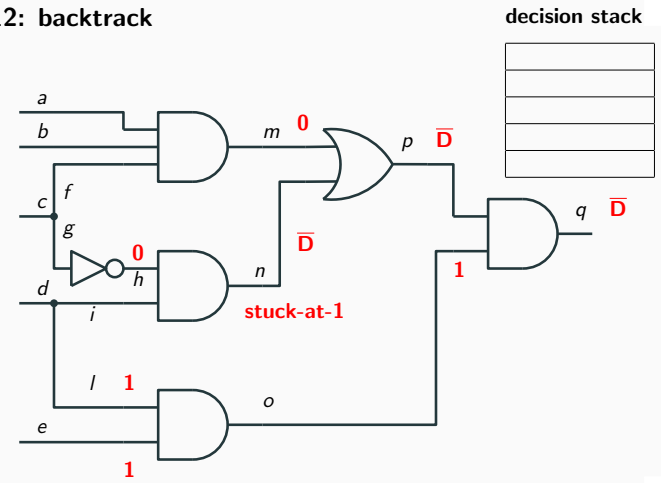
passo 11: consistenza, rimane il conflitto



nota: i segnali non annotati con un valore si intendono a X

Esempio

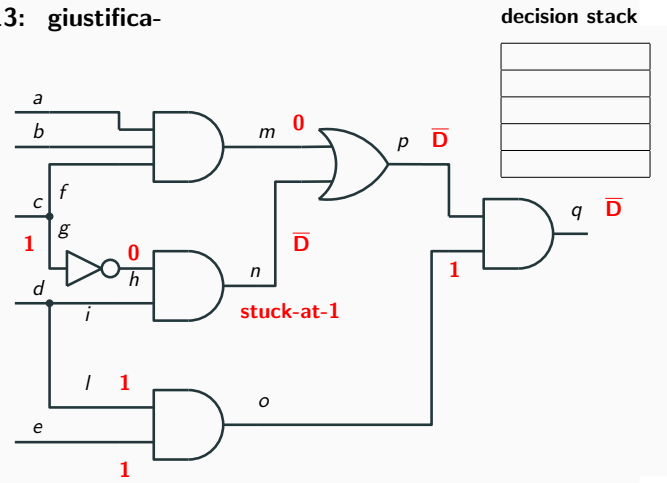
passo 12: backtrack



nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 13: giustificazione

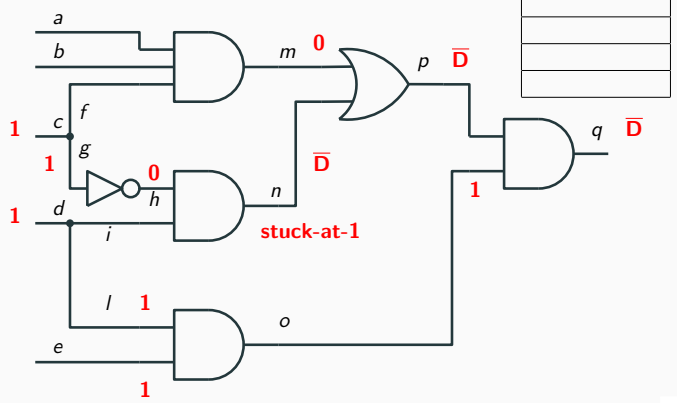


nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 14: consistenza su c e d

decision stack



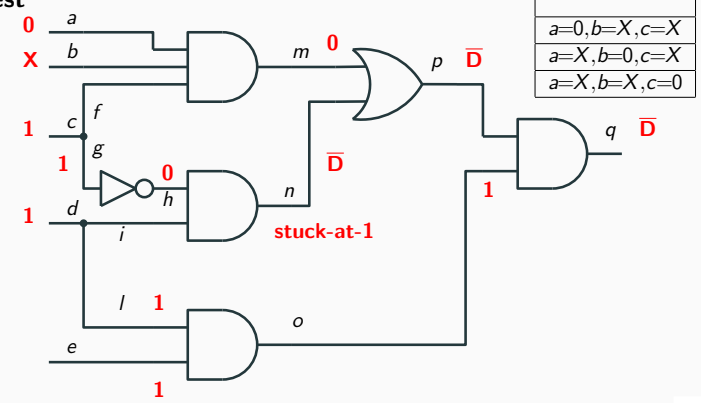
nota: i segnali non annotati con un valore si intendono a X

Esempio

passo 15: giustificazione, D frontiera e J-frontiera vuote $\Rightarrow$ test

decision stack

a=0,b=X,c=X
a=X,b=0,c=X
a=X,b=X,c=0



nota: i segnali non annotati con un valore si intendono a X

Commento

- Si osserva che per i gate del circuito vengono prese decisioni che fra loro possono risultare inconsistenti

PODEM

- Nell'algoritmo D ogni gate puó essere un punto di decisione
- Questo da luogo a un albero di decisioni che in alcuni casi (reti ECAT con alberi di EXOR) puó essere di dimensioni eccessive
- In realtá lo stato del circuito dipende solo dagli assegnamenti dei PI
- Restringendo lo spazio di decisione ai soli ingressi si ha un restringimento dell'albero di scelte da esplorare
- PODEM (Path Oriented Decision Makeing) si basa su questa considerazione
- Viene mantenuta la D-frontiera, ma non la J-frontiera perché i valori sono giustificati dalle scelte ai PI

PODEM

1. Se esiste una configurazione di ingressi non ancora esplorata assegna un ingresso non ancora assegnato, altrimenti esce e dichiara il guasto come non rivelabile
2. Determina tutte le implicazioni di tale assegnamento (simulazione)
3. Se la D-frontiera ha raggiunto i PO dichiara il guasto rivelato e produce il test
4. Altrimenti verifica se il guasto é stato attivato, se no va al punto 8
5. Se il guasto é attivato, verifica se esiste almeno un cammino, X-path, fra la D-frontiera e i PO con i segnali non ancora assegnati altrimenti non é possibile propagare gli effetti del guasto ai PO e va al punto 8
6. PODEM seleziona il migliore (misure di osservabilitá) X-path dalla D-frontiera ai PO
7. Torna al punto 1 con un nuovo obbiettivo
8. Backtrack: rimuove l'ultima decisione sui PI e va al punto 1

Algoritmo PODEM

```
PODEM\_Algorithm(C,f,v) /* line f stuck-at-v */
{
  forall i in C
    out(i)='X';

  D-frontier=∅;

  result=PODEM-recursion(C);
  if (result==success)
    print (out(PI), out(PO));
  else
    print f stuck-at-v is untestable;
}
```

Algoritmo PODEM

```
PODEM-recursion(C)
{
  if (∃ PO | out(PO) =′ D′orout(PO) =′ nD′)
    return(success);
  (g,v)=getObjective(C);
  (PI,u)=backtrace(g,v);
  LogicSimulate(PI,u);
  result=PODEM-recursion(C);
  if (result==success)
    return(success);
  /* backtrack */
  logicSimulate(PI,not(u));
  result=PODEM-recursion(C);
  if (result==success)
    return(success);
  /* must turn on a previous decision */
  logicSimulate(PI,'X');
  return(failure);
}
```


Algoritmo PODEM

```
getObjective(C)
{
  if fault is not excited
    return(g,not(v));

  d=a gate in the D-frontier;
  g=an input of d with value 'X';
  v=non controlling value of d;
  return (g,v);
}
```

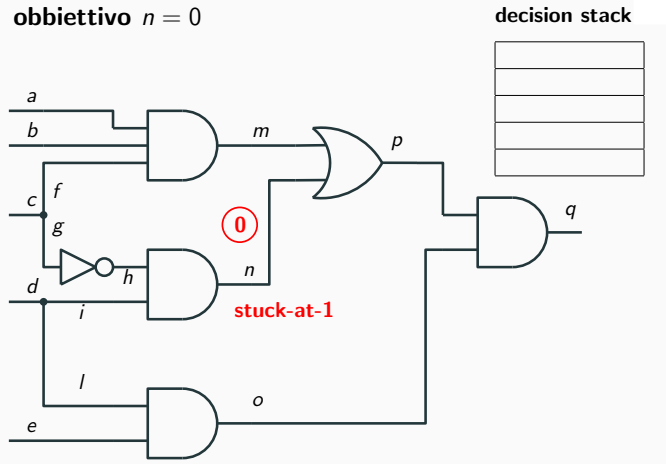
Algoritmo PODEM

```
backtrace(C)
{
  i=g;
  num_inversions=0;

  while i ≠ PI
    {
      i=an input of i with value 'X';
      if i is an inverted gate
        num_inversions++;
    }
  if num_inversions%2
    v=not(v);
  return (i,v);
}
```

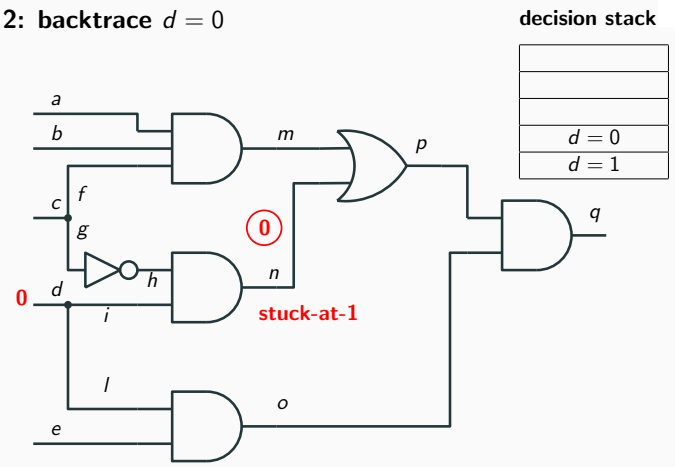

Esempio

passo 1: obbiettivo $n = 0$



Esempio

passo 2: backtrace $d = 0$

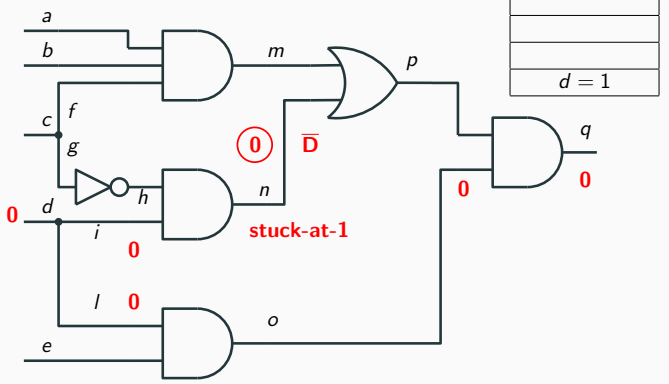


decision stack

$d = 0$
$d = 1$

Esempio

passo 3: imply

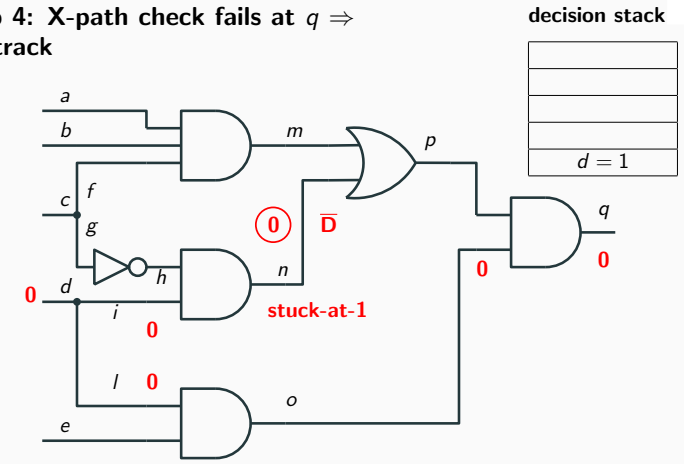


decision stack

$d = 1$

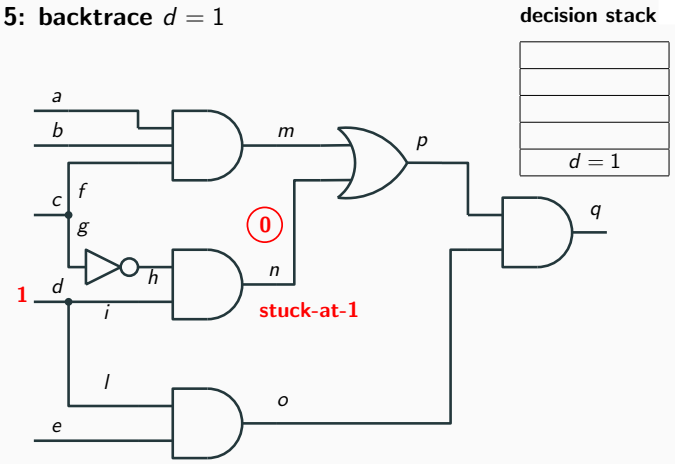
Esempio

passo 4: X-path check fails at $q \Rightarrow$ backtrack



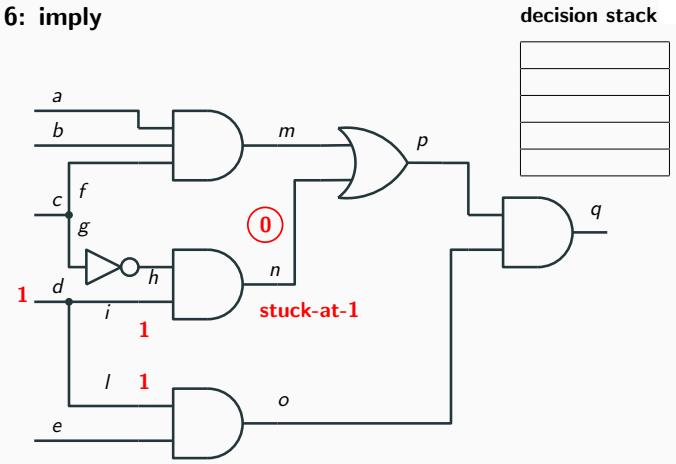
Esempio

passo 5: backtrace $d = 1$



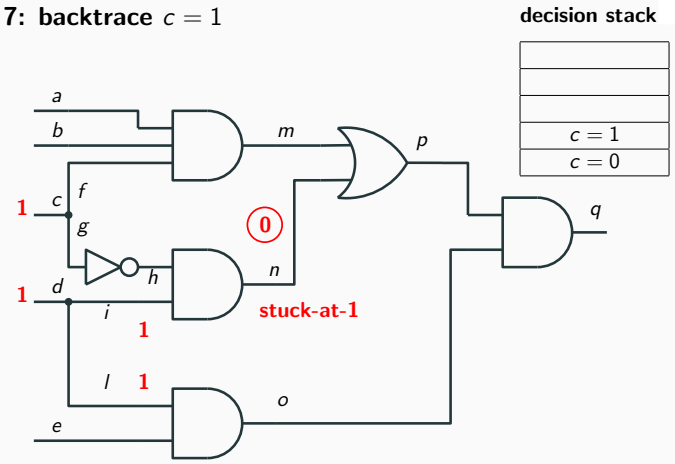
Esempio

passo 6: imply



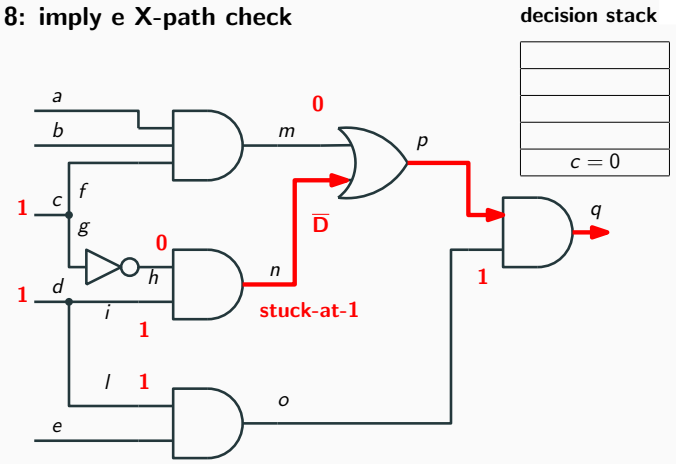
Esempio

passo 7: backtrace $c = 1$



Esempio

passo 8: imply e X-path check

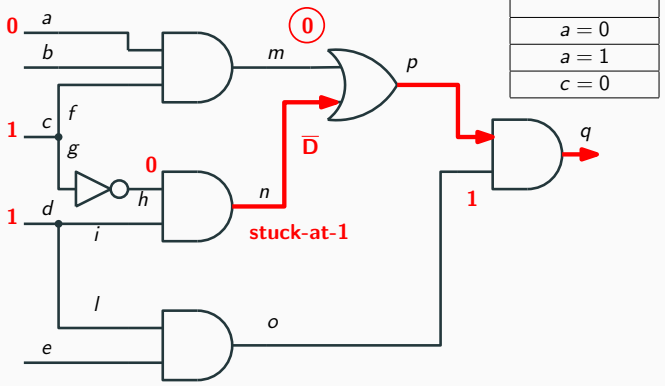


decision stack

$c = 0$

Esempio

passo 9: obbiettivo $m = 0$ e back-trace

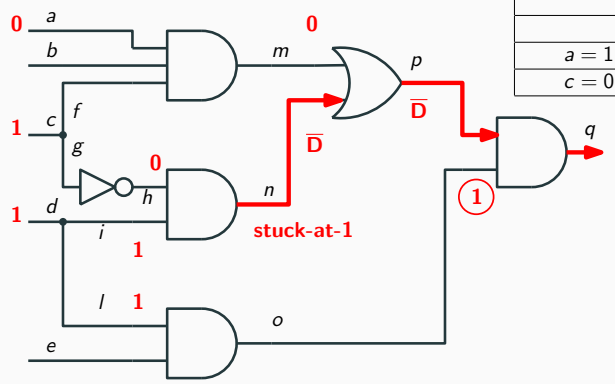


decision stack

$a = 0$
$a = 1$
$c = 0$

Esempio

passo 10: imply

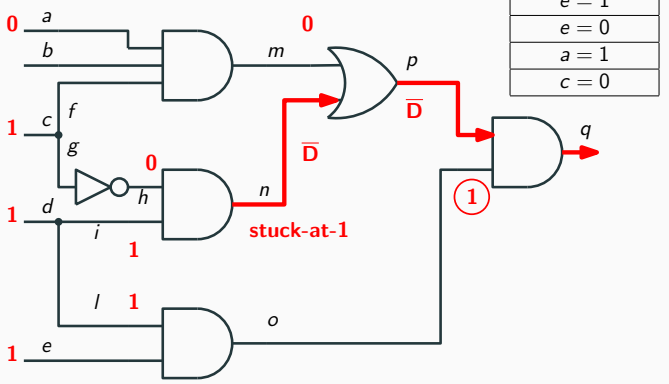


decision stack

$a = 1$
$c = 0$

Esempio

passo 11: obbiettivo $o = 1$ e back-trace



decision stack

$e = 1$
$e = 0$
$a = 1$
$c = 0$

Esempio

passo 12: imply, fault detected

decision stack

$e = 0$
$a = 1$
$c = 0$