Test generation

M. Favalli

Engineering Department in Ferrara

Introduzione

Test generation

- Definizioni e strumenti per la test generation
- Misure di collaudabilitá
- Algoritmi

Collaudo funzionale vs collaudo strutturale (defect based)

- Collaudo funzionale esaustivo di un 64-bit adder: con 129 ingressi e 65 uscite
  - $2^{129} = 680,564,733,841,876,926,926,749,214,863,536,422,912$ test vectors
  - utilizzando un ATE a 1 GHz, potrebbe richiedere 2.15×10^{22} anni
- Collaudo strutturale defect based:
  - 64 bit slices contenenti ciascuna un full-adder
  - 27 guasti equivalenti per full-adder
  - $64 \times 27 = 1728$ guasti
  - richiede 0.000001728 su un ATE a 1 GHz

Definizione di ATPG (software)

- Un ATPG per circuiti digitali dati:
  - una descrizione del circuito a un qualche livello di astrazione
    - la struttura puó contenere accorgimenti di DFT
  - un insieme di modelli di guasto
- Genera una lista di guasti
- Per ciascuno di essi esplora lo spazio delle possibili sequenze di ingresso determinandone una che sia in grado di produrre in uscita una differenza di comportamento fra il circuito corretto e quello col guasto

Reti combinatorie a livello gate in presenza di stuck-at

- Caso piú semplice e diffuso sotto le ipotesi di guasto singolo e permeanente
- Lo spazio delle sequenze di ingresso si riduce a quello dei singoli vettori di collaudo
- Per ciascun guasto l'ATPG inietta il guasto (**fault injection**) e determina le condizioni che ne consentono la propagazione (**fault propagation**) degli effetti ai PO
- Queste operazioni richiedono la giustificazione di opportuni valori ai PI
- Da un punto di vista teorico viene esplorato un albero di decisione nel quale viene associato un valore a ciascun PI

Strumenti per l'ATPG

Descrizione compatta di circuito corretto e circuito guasto

simbolo	significato	fault-free	faulty	algebra
D	$1/0$	1	0	
$\overline{D}$	$0/1$	0	1	
1	$1/1$	1	1	Roth's algebra
0	$0/0$	0	0	
X	X/X	X	X	
$G1$	$1/X$	1	X	
$G0$	$0/X$	0	X	Muth's additions
$F1$	$X/1$	X	1	
$F0$	$X/0$	X	0	

Algebre di Roth e Muth

- Rappresentano due macchine nell'ATPG
  - good machine (primo valore)
  - bad machine (secondo valore)
- I valori della macchina fault-free e di quella guasta che precludono la rivelazione dei guasti si identificano subito
- Bisogni di un ATPG completo (se esiste un test lo trova supponendo di disporre di un tempo di CPU non limitato)
  - Combinatori: Multi-path sensitization, Roth algebra
  - Sequenziali: Muth Algebra - il circuito guasto puó essere inizializzato a valori diversi di quello fault-free

Generazione pseudo-casuale pesata

- Puó essere utilizzata per generare la parte iniziale della sequenza di collaudo con criteri statici o dinamici che la possono interrompere
  - arrivare a una copertura predeterminata 60-80%
  - fermarsi dopo che un certo numero di vettori di questo tipo non producono incrementi di copertura
- Eventualmente le probabilitá di avere il valore 1 possono essere cambiate durante la test generation

```
foreach i ∈ PI p_i(1) = 0.5
while (halt_criterion==FALSE)
  begin
    generate a random test vector
    perform fault simulation
    change input probabilities
  end
```

Algoritmi genetici

- Gli algoritmi genetici furono inizialmente sviluppati per l'ATPG di reti sequenziali
- Una popolazione di vettori (o sequenze) di test viene fatta evolvere avendo come obbiettivo la rivelazione del guasto
- Funzione di fitness: la minima distanza dai PO dei valori a D (osservabilitá) combinata con un indicatore di quanto si é vicini all'attivazione del guasto

Differenze Booleane

- Metodo simbolico utilizzabile se é possibile avere una rappresentazione funzionale compatta dell uscite (BDD)
- Rete combinatoria con n ingressi $\{x_1, x_2, ..., x_n\}$ e m uscite $\{y_1, y_2, ..., y_m\}$
- Si vuole rivelare gausto stuck-at-v ($v \in \{0, 1\}$) su un segnale h
- Ogni uscita primaria y_i viene espressa in funzione degli ingressi $x_1, x_2, ..., x_n$ e del segnale guasto: $y_i = \varphi_i(x_1, x_2, ..., x_n, h)$
- Il segnale guasto viene espresso in funzione degli ingressi: $h = \xi(x_1, x_2, ..., x_n)$

Differenze Booleane

- Per ciascuna uscita si calcola la condizione che rende osservabile h a tale uscita:

$$\frac{\partial \varphi_i}{\partial h} = \varphi_i(x_1, x_2, ..., x_n, 0) \oplus \varphi_i(x_1, x_2, ..., x_n, 1)$$

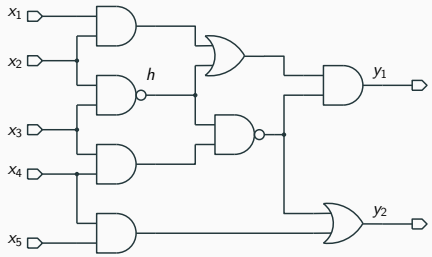
- Per attivare il guasto si calcola $\varepsilon = \xi(x_1, x_2, ..., x_n)$ se $v = 0$ (stuck-at-0) e $\varepsilon = \xi(x_1, x_2, ..., x_n)'$ se $v = 1$ (stuck-at-1)
- Il guasto é quindi rivelato da ogni vettore di ingresso che soddisfa la seguente condizione

$$\varepsilon(x_1, x_2, ..., x_n)\left(\bigvee_{\forall j \in PO} \frac{\partial \varphi_j}{\partial h}\right) = 1$$

- Ovvero il guasto deve essere attivato e reso osservabile ad almeno un uscita

Esempio - h stuck-at-0

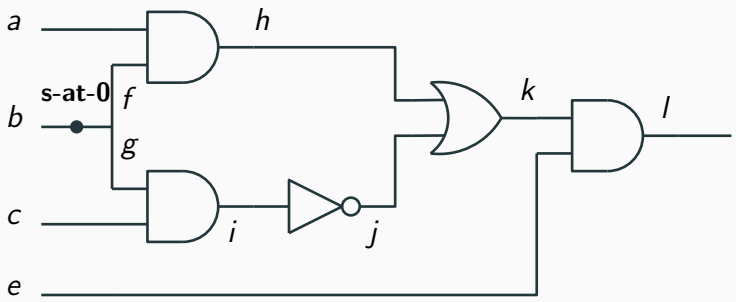
$$h = \xi(x_2, x_3) = x_2' + x_3'$$
$$y_1 = \varphi_1(x_1, x_2, x_3, x_4, h) = (x_1 x_2 + h)(x_3' + x_4' + h')$$
$$y_2 = \varphi_2(x_2, x_3, x_4, x_5, h) = x_3' + x_4' + h' + x_4 x_5$$
$$\frac{\partial \varphi_1}{\partial h} = (x_1 x_2) \oplus (x_3' + x_4') = x_1 x_2 x_3 x_4 + x_1' x_3' + x_2' x_4' + x_2' x_3' + x_2' x_4'$$
$$\frac{\partial \varphi_2}{\partial h} = 1 \oplus (x_3' + x_4' + x_4 x_5) = x_3 x_4 x_5'$$
$$\xi\left(\frac{\partial \varphi_1}{\partial h} + \frac{\partial \varphi_2}{\partial h}\right) = x_1' x_2 x_3' + x_2' x_3' + x_2' x_4' + x_2' x_5'$$



Sensibilizzazione di cammini

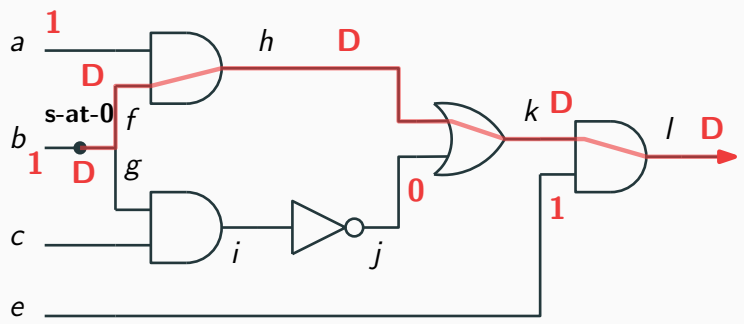
- Il primo algoritmo di ATPG sviluppato era la single path sensitization che non era completo perché in alcuni casi bisogna sensibilizzare cammini multipli
- Qui vedremo un semplice esempio di algoritmo che seleziona recursivamente un possibile cammino dal sito del guasto a un uscita cercando di propagare i valori D
- Per fare questo vengono assegnati alcuni valori ai segnali lungo il cammino che devono poi essere giustificati con opportuni assegnamenti agli ingressi

Esempio



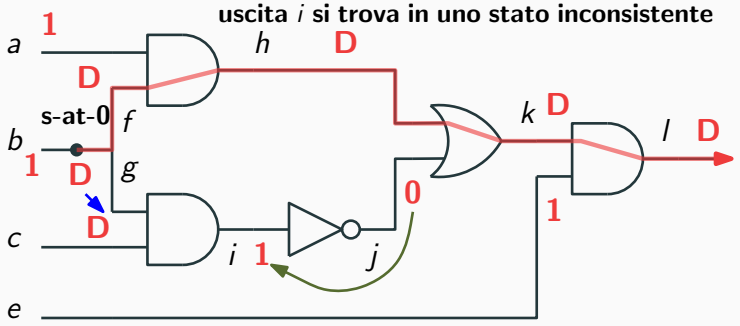
Esempio

sensibilizzazione del cammino



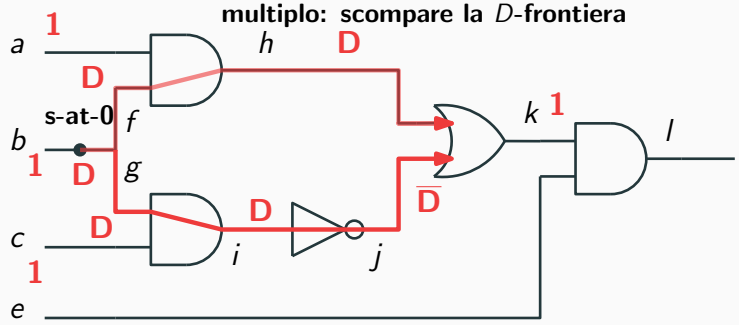
Esempio

propagazione e giustificazione, il gate con uscita i si trova in uno stato inconsistente



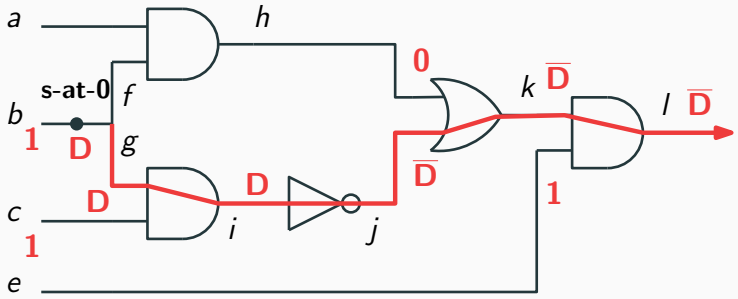
Esempio

propagazione attraverso un cammino multiplo: scompare la D-frontiera

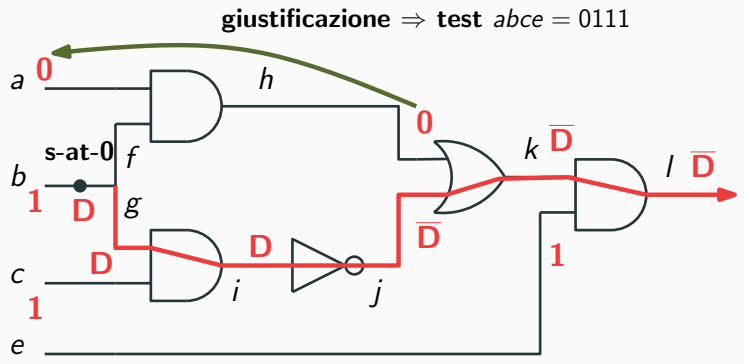


Esempio

cammino sensibilizzato



Esempio



Boolean satisfiability

- Descrive gli stati consistenti di un circuito utilizzando un espressione CNF
- L'espressione CNF della rete puó essere calcolata a partire da quelle dei singoli gate
- Esempio: CNF di un gate AND ($y = ab$)
  - $CNF = (a' \rightarrow y')(b' \rightarrow y')(ab \rightarrow y) = (a+y')(b+y')(a'+b'+y)$
- Problema di soddisfacibilitá: trovare un assegnamento delle variabili che renda vera l'espressione CNF
- Classificazione
  - problemi 2-SAT: le clausole hanno al piú 2 letterali e la soddisfacibilitá risulta calcolabile in tempo polinomiale
  - problemi 3-SAT: il numero di letterali per clausola é ≤ 3. In questo caso, il problema é NP-completo, anche se la soluzione puó essere calcolata in maniera piú efficiente rispetto al caso generale

Boolean satisfiability

- La Boolean satisfiability SAT trae vantaggio dall'esistenza di solver particolarmente efficienti
- Si tratta di rappresentare come una CNF il circuito fault-free, quello con il guasto e un miter che ne confronta le uscite
- A questi viene aggiunta la condizione di errore in uscita al miter
- Vedremo alcuni accorgimenti che possono rendere piú efficiente la soluzione del problema

Algoritmo di backtrack

- Utilizzato per trovare una soluzione ai problemi SAT
- Vengono prima propagate le clausole costituite da un solo letterale $(a + b)a(c + d) = a(c + d)$
- Poi l'algoritmo procede in maniera recursiva facendo delle scelte sui valori dei letterali
  - dopo ogni scelta vengono dedotti tutti gli assegnamenti resi necessari (guardando in particolare le clausole con 2 letterali non ancora assegnati)
  - se una clausola diviene non soddisfacibile, una procedura analizza le cause del conflitto ed aggiunge eventualmente una nuova clausola

Esempio

- $\varphi = (y + w + z)(x' + z')(v' + x' + w')$
- Assegnamenti correnti: $v = 1,\ y = 0,\ x = 1$
- $\varphi = (0 + w + z)(0 + z')(w' + 0 + 0) = 0$ (dopo unit clause propagation)
- Regola: $v.y'.x \rightarrow \varphi'$
- Nuova clausola da aggiungere alla CNF: $(v' + y + x')$

Recursive learning

- Identifica condizioni necessarie in comune fra le diverse clausole
- Esempio: $\varphi)(u + x + w')(x + y')(w + y + z')$
- Assegnamenti correnti:$z = 1,\ u = 0$
- $\varphi = (x + w')(x + y')(w + y) = x(w + y)$
- Quindi $x = 1$ é reso necessario da tali assegnamenti: $zu' \rightarrow x$
- Si aggiunge la clausola $(z' + u + x)$ alla CNF

Boolean satisfiability per ATPG

- Gli algoritmi SAT possono essere migliorati per l'ATPG
- Valori dei segnali a X o a Z
  - sono necessarie piú variabili booleane per segnale

is_0	is_1	valore
0	0	Z
0	1	1
1	0	0
1	1	X

funzione AND ($y = ab$)

$$is_0(y) = is_0(a) + is_0(b) + is_1(a)' + is_1(b)'$$
$$is_1(y) = is_0(a)'is_0(b)' + is_0(a)'is_1(b) + $$
$$+ is_1(a)is_0(b)' + is_1(a)is_1(b)$$

- Nella CNF delle due reti e del miter, servono solo le CNF dei gate nel circuito guasto che sono contenute nel fan-out transitivo del gate guasto

Complessitá computazionale degli ATPG

- Il problema é NP-completo
- Nelle condizioni di caso peggiore deve essere esplorato uno spazio dato dal numero di configurazioni di ingresso $2^{\#PI}$ per il numero delle configurazioni dei flip-flop $4^{\#FF}$
- Quindi la complessitá computazionale é $O(2^{\#PI} \times 4^{\#FF})$
- Il problema ha visto la comparsa di numerosi algoritmi con miglioramenti nelle prestazioni, ottenendo uno speed-up normalizzato di piú di 2 ordini di grandezza rispetto ai primi algoritmi
- Nel caso delle reti combinatorie il problema si puó considerare come risolto

Misure di collaudabilitá (testability measures)

- Si tratta di misure, tipicamente approssimate, che servono da aiuto nel collaudo con le seguenti motivazioni:
  - aiutare la test generation deterministica
    - scelte piú convenienti per propagare gli effetti di un guasto o giustificare un valore
  - aiutare la test generation pseudorandom
    - determinare la detection probability dei guasti e da questa la lunghezza di una sequenza pseudorandom che raggiunga un certo obbiettivo di copertura di guasto
  - aiutare le metodologie di DFT
    - identificare quali segnali rendere maggiormente controllabili o osservabili per migliorare la copertura di guasto

Misure di collaudabilitá per ATPG deterministico

- SCOAP (Sandia Controllability and Observability Analysis Program)
- Misure combinatorie
  - difficoltá nel controllare a 1 una linea del circuito ($CC1$)
  - difficoltá nel controllare a 0 una linea del circuito ($CC0$)
  - difficoltá nell'osservare una linea del circuito (CO)
- Le misure combinatorie sono proporzionali al numero di linee che devono essere assegnate per controllare o osservare una linea del circuito
- Ci sono anche misure di tipo sequenziale

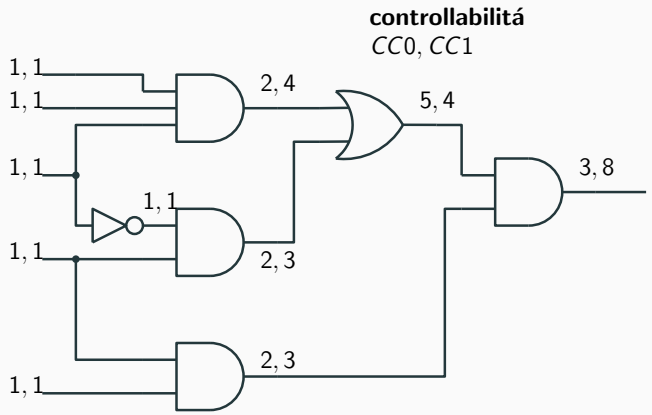
SCOAP: controllabilitá

- Calcolo di $CC1$ e $CC0$
- Sono compresi nell'intervallo 1 (facile), ∞ (difficile)
- Per i PI $CC0 = CC1 = 1$
- AND gate:
  - $CC0(out) = 1 + \min_{\forall in} CC0(in)$
  - $CC1(out) = 1 + \sum_{\forall in} CC1(in)$

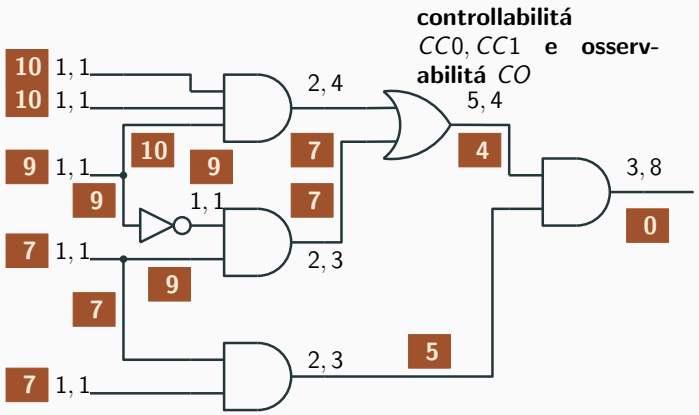
SCOAP: osservabilitá

- Calcolo di $\boldsymbol{CO}$
- Compresa nell'intervallo $\mathbf{0}$ (facile), ∞ (difficile)
- Per i PO $\boldsymbol{CO = 0}$
- AND gate (input $\boldsymbol{i}$):
  - $\boldsymbol{CO(i) = CO(out) + \sum_{\forall in \neq i} CC1(in) + 1}$
- fan-out stem
  - $\boldsymbol{CO(s) = min_{\forall i \in FOB} CO(i)}$

Esempio



Esempio



Misure di collaudabilitá probabilistiche

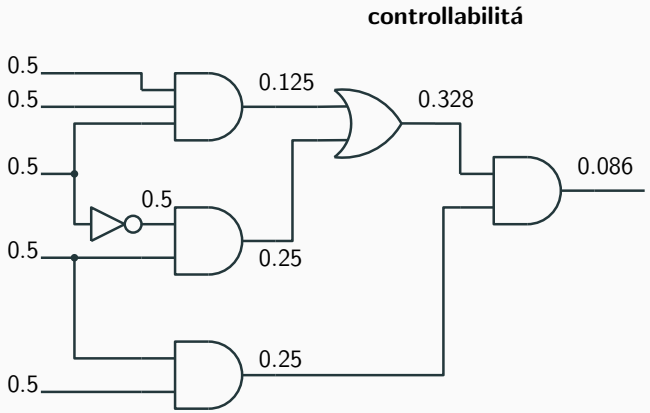
- Consentono di stimare la probabilitá (detectability) di rivelare un guasto per una sequenza casuale di ingresso
- Nel caso piú semplice (COP) viene calcolata
  1. la controllabilitá di ciascuna linea del circuito, ovvero la probabilitá di avere il valore 1
  2. l'osservabilitá di ciascuna linea di un circuito, ovvero la probabilitá che un cambiamento di valore di una linea si rifletta come errore sui PO
- La detectability di un guasto stuck-at-1 su una linea i viene calcolato come il prodotto della controllabilitá a 0 per l'osservabilitá di i (per lo stuck-at-0 si usa il caso duale)
- Errori piuttosto rilevanti dovuti al trascurare le correlazioni fra i segnali e quelle fra controllabilitá e osservabilitá

Controllabilitá e osservabilitá

- Il caloclo della cotrollabilitá procede dagli ingressi verso le uscite di una rete combinatoria calcolando la probabilitá di avere 1 all'uscita di ciascun gate in funzione di quella degli ingressi
  - $\forall i \in PI,\ p(i) = Prob\{i = 1\} = 0.5$
  - gate AND: $p(out) = \prod_{\forall in} p(in)$
  - gate OR: $p(out) = 1 - \prod_{\forall in}(1 - p(in))$
  - gate NOT: $p(out) = 1 - p(in)$
- In un secondo passo si procede dalle uscite verso gli ingressi calcolando l'osservabilitá
  - $\forall j \in PO,\ obs(j) = Prob\{observe\ j\} = 1.0$
  - gate AND con insieme F di fan-in
    $\forall j \in F,\ obs(j) = obs(out) \cdot \prod_{\forall k \neq j \in F} p(k)$
  - gate OR con insieme F di fan-in
    $\forall j \in F,\ obs(j) = obs(out) \cdot \prod_{\forall k \neq j \in F}(1 - p(k))$

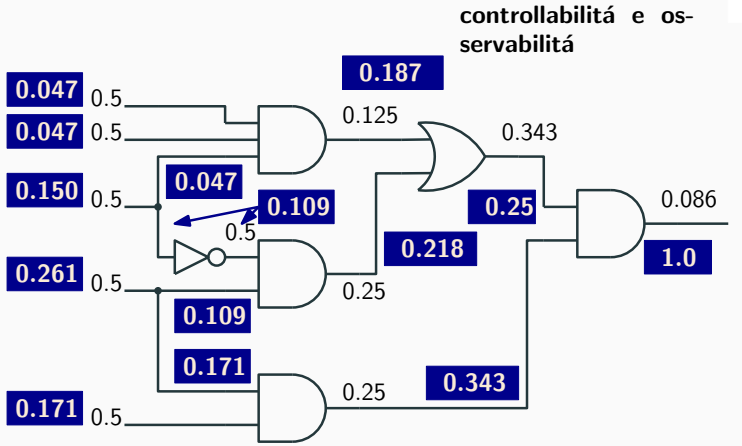
Detectability

- Alla fine si calcola la detectability:
  - stuck-at-1: $det(\text{j stuck-at-1}) = (1 - p(j)) \times obs(j)$
  - stuck-at-0: $det(\text{j stuck-at-0}) = p(j) \times obs(j)$
- Esempio di stima di controllability, observability (blu), detectability di stuck-at-0/stuck-at-1 (rosso):



Detectability

- Alla fine si calcola la detectability:
  - stuck-at-1: $det(\text{j stuck-at-1}) = (1 - p(j)) \times obs(j)$
  - stuck-at-0: $det(\text{j stuck-at-0}) = p(j) \times obs(j)$
- Esempio di stima di controllability, observability (blu), detectability di stuck-at-0/stuck-at-1 (rosso):



Detectability

- Alla fine si calcola la detectability:
  - stuck-at-1: $det(\text{j stuck-at-1}) = (1 - p(j)) \times obs(j)$
  - stuck-at-0: $det(\text{j stuck-at-0}) = p(j) \times obs(j)$
- Esempio di stima di controllability, observability (blu), detectability di stuck-at-0/stuck-at-1 (rosso):