On-line testing ed error recovery

M. Favalli

Engineering Department in Ferrara

Introduzione

- I costi della fault tolerance ottenuta mediante sistemi a ridondanza modulare sono piuttosto elevati
- Un approccio alternativo consiste nell'utilizzare on-line testing ed error recovery
- Compito dell'on-line testing é l'individuazione di errori non appena questi si manifestano
- Ció deve essere fatto senza interrompere le normali operazioni del sistema
- L'error recovery, una volta rivelato l'errore deve ristabilire le corrette operazioni del sistema

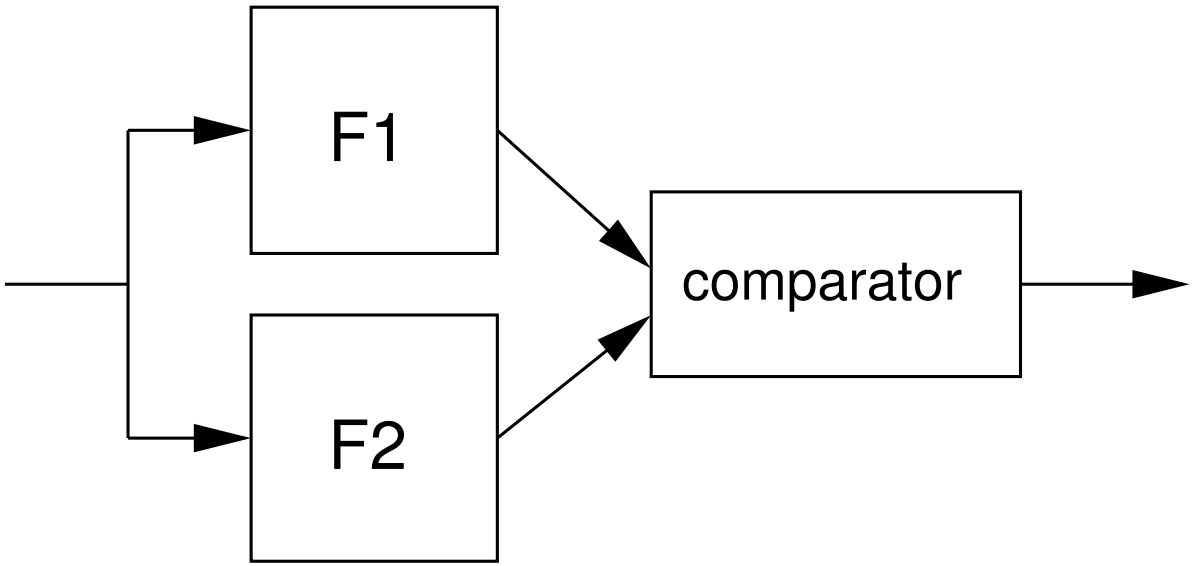
Caratteristiche rilevanti

- Gli errori vengono rilevati subito dopo la loro comparsa in modo da garantire l'integritá dei dati nel sistema
- É possibile gestire errori di tipo transitorio
- Chiaramente, nel caso di errori permanenti il contributo della tecnica di fault tolerance considerata si limita alla diagnosi, riparazione o riconfigurazione devono poi entrare in gioco

Tecniche di on-line testing

- Duplicazione e confronto
- Sistemi self-checking
  - definizioni
  - codici
  - checker
  - blocchi funzionali

Duplicazione e confronto



- É caratterizzata da una elevata facilitá di progetto
- Presenta diversi svantaggi:
  - area overhead
  - common mode failures

Sistemi self-checking

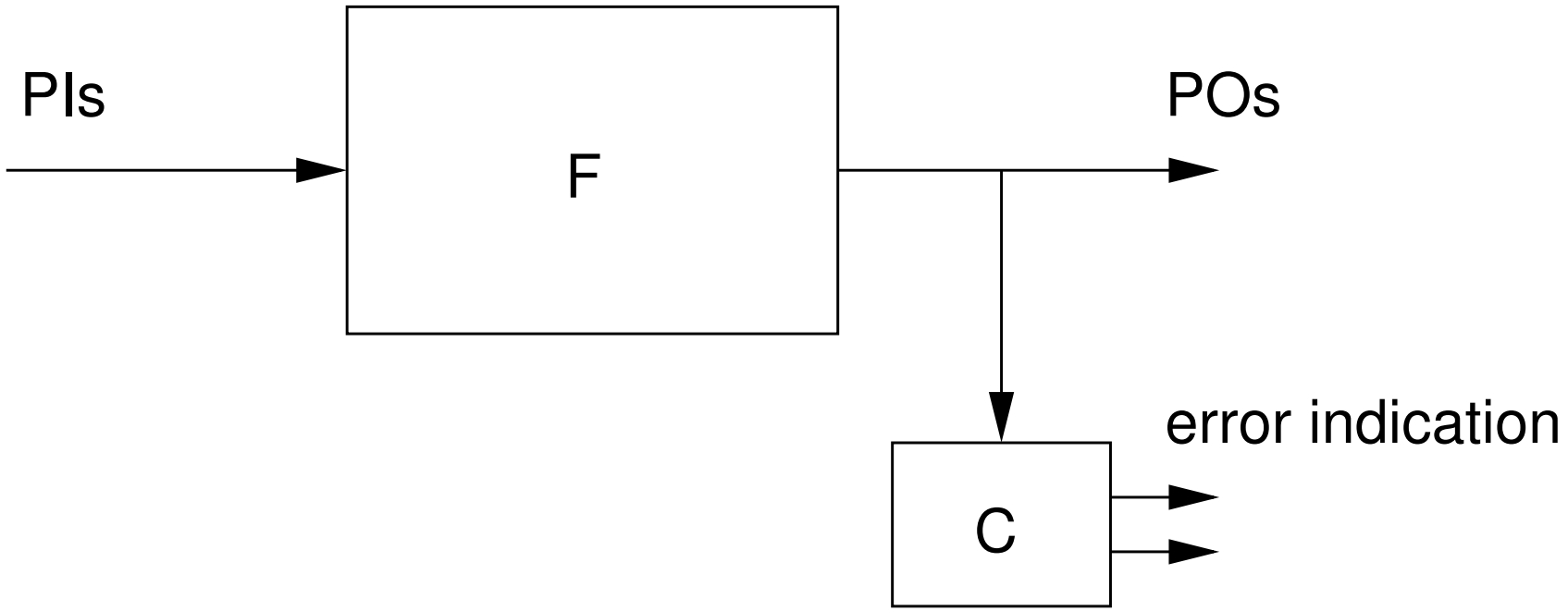
- Rispetto a duplicazione confronto, cercano di ottenere dei risparmi dal punto di vista dell'area
- Le metodologie di progetto sono comunque piú complesse
- Applicazioni:
  - sistemi ad alta affidabilitá (aerospazio)
  - alta disponibilitá (centrali telefoniche)
  - integritá dei dati

Circuiti self-checking

- Sono sistemi basati su una ridondanza di informazioni che implica poi anche una qualche ridondanza hardware
- Ingressi, uscite e stati interni di un sistema self-checking sono codificati utilizzando codici a rivelazione di errore
- Circuiti, detti checker, verificano l'appartenenza delle configurazioni di tali segnali ai codici a rivelazione d'errore

Struttura generale

- *F* é l'unitá funzionale (es. ALU, multiplier ....)
- *C* é il checker che verifica l'appartenenza delle uscite al codice utilizzato e contemporaneamente verifica la correttezza delle sue stesse operazioni



Criteri di progetto

- F deve dare luogo a parole di uscita che appartengono al codice in assenza di guasti e non di codice in presenza di guasti
- C verifica l'appartenenza al codice dei suoi ingressi dando un indicazione di errore se questo non accade
- C deve essere in grado di svolgere la sua funzione anche in presenza di guasti dando luogo a un errore in caso questo non sia piú possibile

Proprietá self-checking

- I precedenti criteri di progetto sono stati formalizzati nella proprietá self-checking (che é riferita a un insieme di guasti)
- F deve essere Totally Self-Checking (TSC) o Strongly Fault Secure rispetto a possibili guasti
- C deve essere Totally Self-Checking (TSC) o Strongly Code Disjoint (SCD) rispetto a possibili guasti

Definizioni

- F con ingressi x e uscite y sia il circuito del quale si vogliono verificare le proprietá
- Sia $\mathcal{F}$ l'insieme di guasti da considerare
- Sia $\Phi(x)$ la funzione svolta dal circuito in condizioni fault-free e sia $\Phi(x, f)$ quella svolta in presenza del gusto f
- Siano X e Y gli spazi delle possibili configurazioni degli ingressi e dell'uscita
- Siano $A \subseteq X$ e $B \subseteq Y$ gli spazi di parole di codice in ingressi e in uscita

Proprietá Fault-Secure

- Un circuito é Fault-Secure (FS) rispetto a un insieme di guasti $\mathcal{F}$ se, per ogni guasto $f \in \mathcal{F}$, il circuito non produce mai una parola di codice sbagliata quando in ingresso é presente una parola di codice
- Quindi:

$$\forall f \in \mathcal{F} \wedge \forall a \in A,\ \Phi(a, f) = \Phi(a) \vee \Phi(a, f) \notin B$$

- Nella pratica, non si devono produrre parole di codice errate perché queste non verrebbero rivelate dal checker

Proprietá Self-Testing

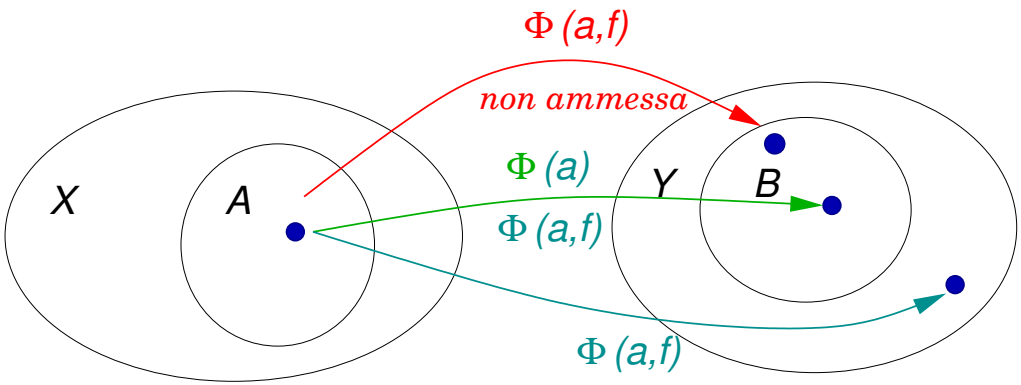
- Un circuito é Self-Testing rispetto a un insieme di guasti $\mathcal{F}$ se, per ogni guasto $f \in \mathcal{F}$, il circuito produce una parola non di codice per almeno una parola di codice in ingresso
- Quindi:

$$\forall f \in \mathcal{F}, \exists a \in A \mid \Phi(a, f) \notin B$$

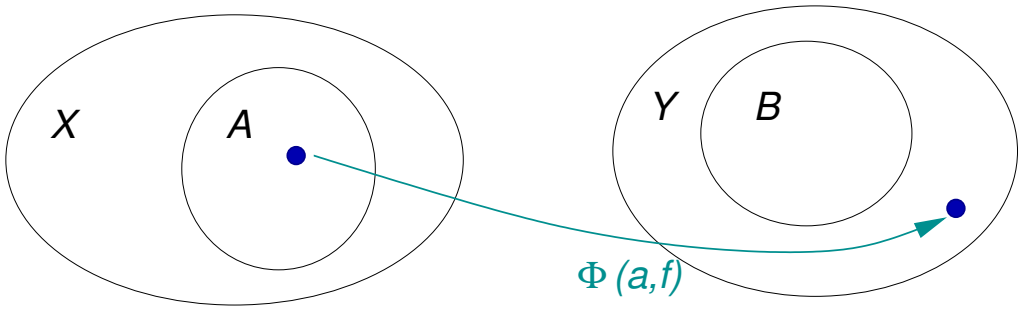
- L'idea é quella che ogni possibile guasto sia rivelabile dal checker

Proprietá FS e ST

Fault–secure



Self–Testing



Proprietá code-disjoint

- Un circuito é Code-Disjoint (CD) se mappa parole di ingresso di codice in parole di uscita di codice, e parole di ingresso non di codice in parole di uscita non di codice
- Quindi:

$$\forall a \in A, \Phi(a) \in B \wedge \forall x \notin A, \Phi(x) \notin B$$

Proprietá Totally Self-Checking

- Un blocco funzionale che é FS e ST rispetto a $\mathcal{F}$ é Totally Self-Checking (TSC) rispetto a $\mathcal{F}$
- Un checker che é CD, FS e ST rispetto a $\mathcal{F}$ é Totally Self-Checking (TSC) rispetto a $\mathcal{F}$
- Come si vedrá, queste proprietá possono essere difficili da raggiungere

Proprietá Strongly Fault-Secure

- Si tratta di una proprietá piú generale della FS e ST
- Un circuito é Strongly Fault-Secure (SFS) rispetto a un insieme di guasti $\mathcal{F}$ se, per ogni guasto $f \in \mathcal{F}$:
  1. il circuito é FS e ST o
  2. il circuito é FS e se si presenta un altro guasto $g \in \mathcal{F}$, o 1) o 2) é verificata per la sequenza di guasti

Proprietá Strongly Code-Disjoint

- Proprietá piú generale della ST e CD
- Un circuito é Strongly Code-Disjoint (SCD) rispetto a un insieme di guasti $\mathcal{F}$ se, per ogni guasto $f \in \mathcal{F}$:
  1. il circuito é ST o
  2. il circuito é mappa parole non di codice in ingresso in parole non di codice in uscita e, se un altro guasto $g \in \mathcal{F}$ si verifica, o 1) o 2) é verificata per la sequenza di guasti

Ipotesi

É evidente che la completezza dell'insieme di guasti considerato gioca un ruolo chiave nel definire la qualitá di un sistema self-checking.
Sono necessarie alcune ipotesi semplificative:

1. i guasti si verificano uno alla volta (non é possibile che compaiano contemporaneamente due o piú guasti)
2. il tempo fra l'occorrenza di due guasti é sufficientemente lungo da garantire l'applicazione di tutte le possibili parole di codice in ingresso

Guasti: stuck-at, bridging, delay, crosstalk faults e guasti transitori

Limitazioni

- É possibile che F produca solo un sottoinsieme delle parole di codice e che quindi il checker non verifichi l'ipotesi 2)
- Alcune parole possono essere molto rare e quindi l'ipotesi 2) non risulta realistica
- Il checker puó quindi non soddisfare né la proprietá SCD che quella TSC (embedded checker)

TSC goal

- Utilizzando queste ipotesi, la prima parola errata di un blocco funzionale TCS o SFS é una parola non di codice
- Che puó quindi essere immediatamente rilevata dal checker

TSC goal

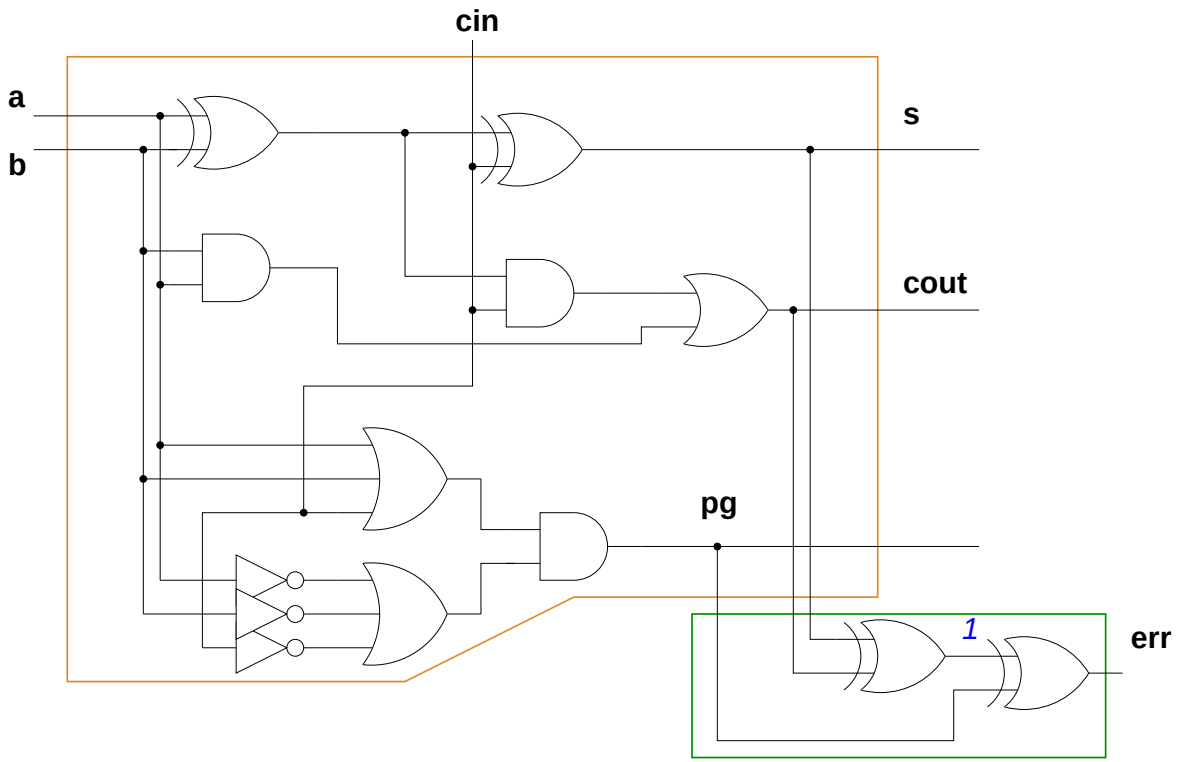
Questo obbiettivo garantisce di preservare l'integritá dei dati al livello di sistema

Progetto di un sistema self-checking

- Oltre ai tradizionali vincoli di area, prestazioni e consumo di potenza, si devono considerare i vincoli del TSC goal
- La scelta del tipo di codice a rivelazione d'errore, i metodi di sintesi dell'unitá funzionale e del checker sono intimamente correlati dagli effetti del tipo di guasti considerati
- In particolare, anche nel semplice caso di reti combinatorie, il progetto di checker e blocco funzionale non puó utilizzare tutte le metodologie comunemente utilizzate nella sintesi multilivello

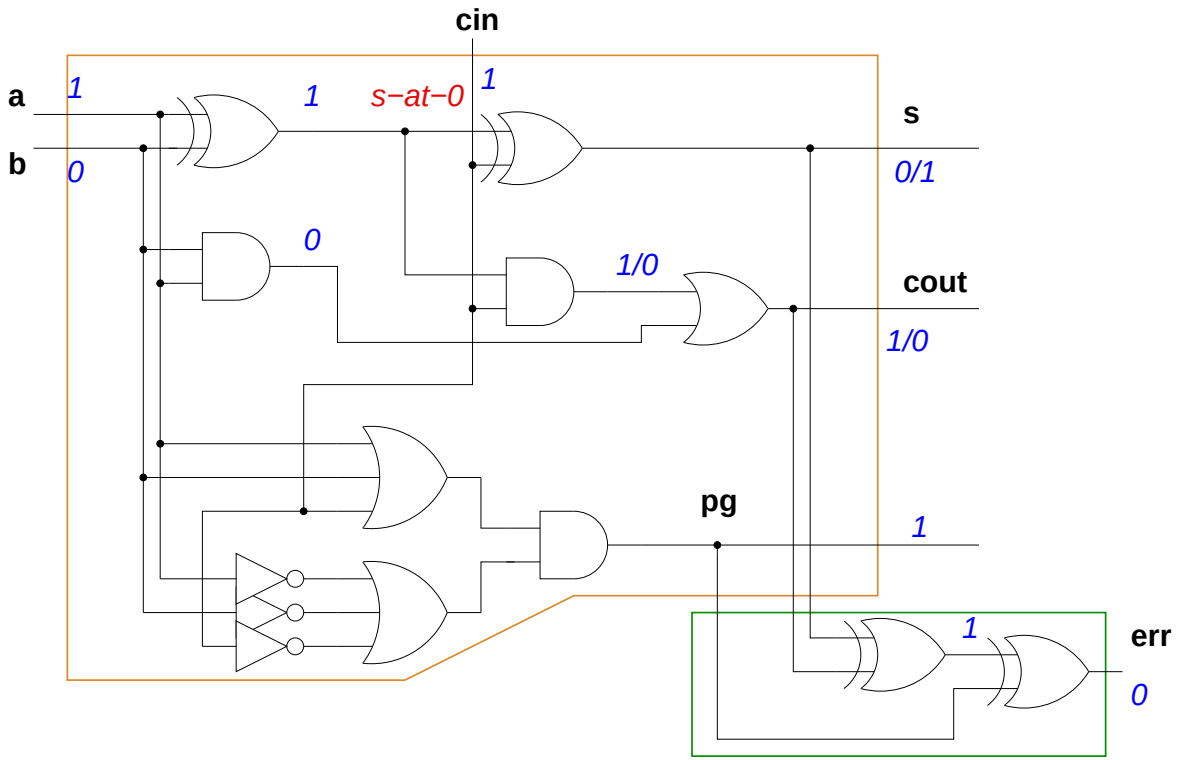
Esempio

- blocco funzionale: FA e parity generator
- codice (solo sulle uscite): parity checker



Esempio

Violazione della proprietá FS per il guasto indicato



Si ha anche una violazione della proprietá ST per il guasto *ERR* stuck-at-0

Considerazioni pratiche di progetto

- L'esempio precedente mostra alcuni aspetti importanti nel progetto di sistemi self-checking
- Il fan-out non puó essere sfruttato appieno (e quindi il costo di questi sistemi non puó essere ridotto in maniera efficiente)
- I checker non possono avere una singola uscita, di solito ne presentano due che forniscono 01 e 10 come indicazioni di funzionamento corretto e 00 e 11 come indicazioni di errore
- Le modalitá di errore giocano un ruolo fondamentale

Modalitá di errore

- Dipendono da: 1) struttura del circuito; 2) funzione logica; 4) modello di guasto; 5) temporizzazioni
- Molteplicitá
- Direzionalitá
  - un errore di dice unidirezionale se tutti i bit errati avevano lo stesso valore nel circuito fault-free
  - altrimenti, l'errore si dice non unidrezionale

Codici utilizzati nei sistemi self-checking

- Codici separabili: una parola di codice é composta di I bit di informazione e C di check
- Codici sistematici: codici separabili in cui compaiono tutti i possibili simboli ottenibili con i bit di informazione
- Codici non sistematici: i bit di informazione non possono separati da quelli di check

Codici utilizzati nei sistemi self-checking

- Solitamente (memorie etc.) i codici sono caratterizzati da:
  - tipologia di errori rilevati
  - ridondanza (numero di bit che é necessario aggiungere a una parola non codificata)
  - capacitá (numero di informazioni codificabili data la dimensione di una parola di codice)
  - facilitá di codifica/decodifica (codici separabili e sistematici)
- Nel caso dei sistemi self-checking, si aggiungono altri parametri
  - facilitá di progetto per blocco funzionale e checker
  - costo complessivo

Codici utilizzati nei sistemi self-checking

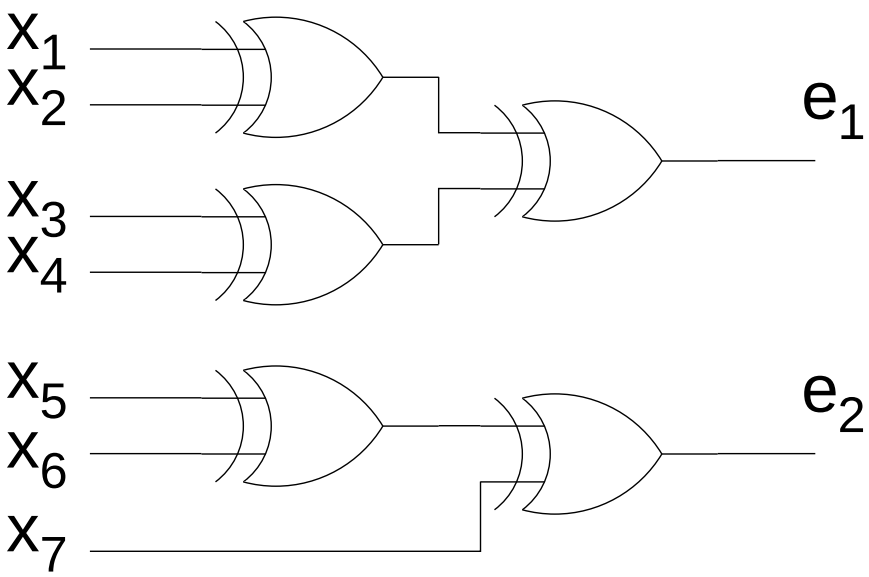
Codici caratterizzati da capacitá di rivelazione di errore e costi diversi

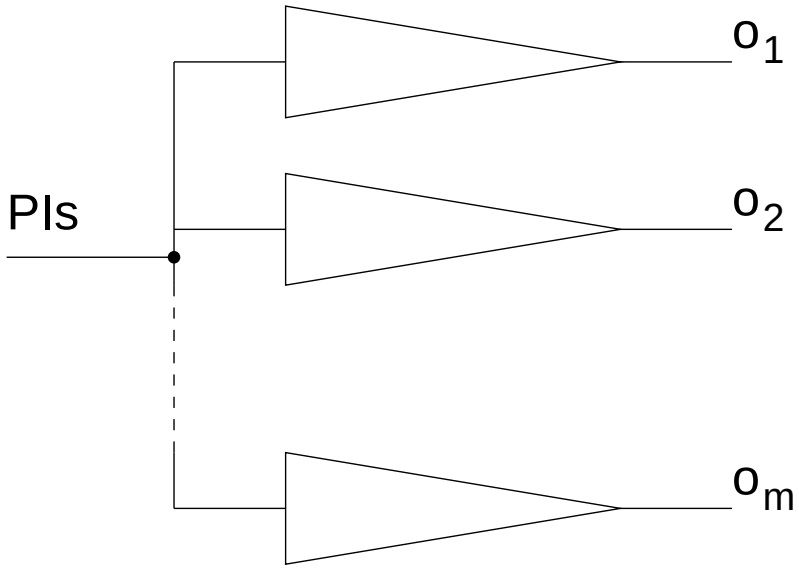
- Codice di paritá (sistematico)
- Codice di two-rail (sistematico e in grado di rivelare tutti gli errori unidirezionali)
- Codici in grado di rivelare tutti gli errori di tipo unidirezionale
  - codice *m*-out-of-*n* (non sistematico)
  - codice Berger (sistematico)

Codice di paritá

- Schema con paritá pari o dispari
- Largamente utilizzato nelle memorie, ma poco adatto per la logica
- Rivela errori singoli $\rightarrow$ per ottenere circuiti che soddisfino il TSC goal, si deve utilizzare un fan-out unitario (un cono di logica per ogni uscita)
- Il checker é costituito a due alberi di exor

Checker e blocco funzionale



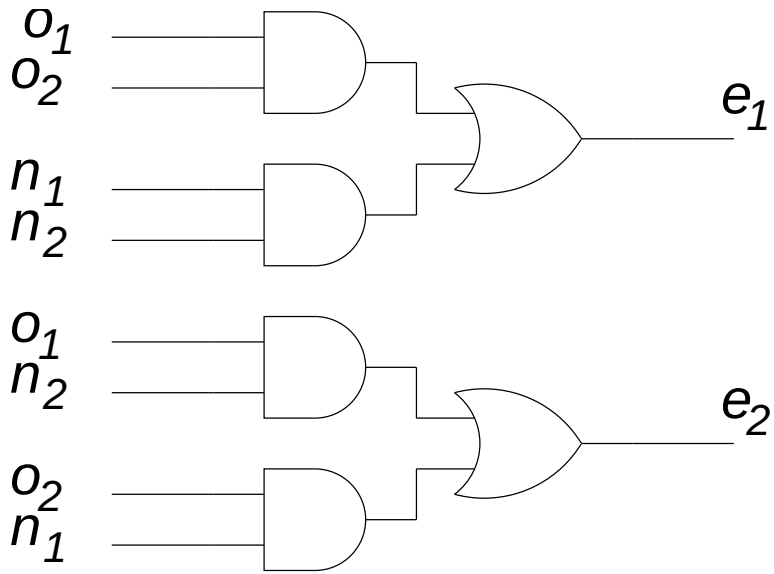


Codice two-rail

- Si realizzano due blocchi funzionali (che hanno in comune i soli ingressi), uno realizza le funzionalitá del sistema, l'altro, invece, produce le uscite complementate
- Nel caso di blocchi combinatori, il primo blocco realizza una funzione $F : \{0,1\}^n \rightarrow \{0,1\}^m$ e il secondo $F' : \{0,1\}^n \rightarrow \{0,1\}^m$
- Se l'uscita i-ma del primo blocco é data da $o_i = f(\mathbf{x})$, l'uscita i-ma nel secondo blocco é data da $n_i = f'(\mathbf{x})$
- Vantaggi: a) facilitá di progetto; b) una minore sensibilitá ai common mode failures rispetto ai sistemi duplex
- Svantaggi: costo

Two-rail checker

Two variable two-rail checker (TRC-2)

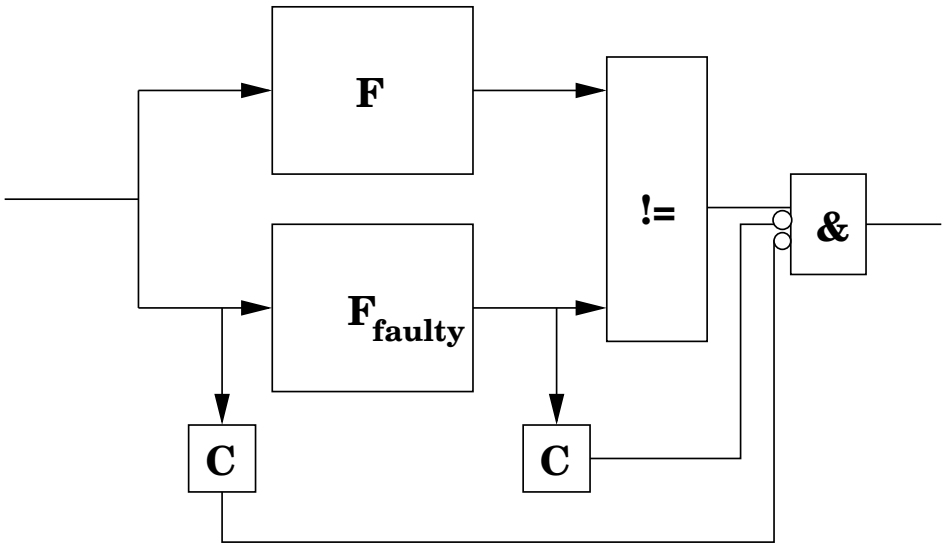


Codice *m*-out-of-*n*

- Checker
- Blocco funzionale

Progetto - 1

- Verifica della proprietá FS tramite boolean satisfiability



Progetto - 2

- Verifica della proprietá ST tramite boolean satisfiability