EC500

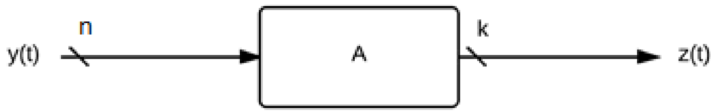
Design of Secure and Reliable Hardware

Lecture 7

Mark Karpovsky

Concurrent Checking of Linear Devices
A device is linear if it can be implemented by XOR gates and FFS only.

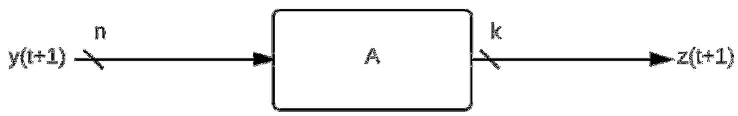
Linear combinational devices:



$z(t) = y(t) \cdot A$, all computations $mod\ 2$ (For q-ary devices all the computations mod q)

Example: Network computing syndromes for $(M = 2^m - 1, 2^m - m - 1, d = 3)$ Hamming code

Linear Sequential devices:
$D(t+1) = D(t)A \oplus y(t+1)$ $\qquad$ $D(t), y(t), z(t)$ are n-bit vectors
$z(t+1) = D(t+1)$ $\qquad$ A is $(n \times n)$ binary matrix



$D(t)$ internal state

Select code C of length t to protect a linear device (C is a (t,k) code, $r = t - k$). A generating matrix of C can always be presented as $G = (I \,\vdots\, P)$, where I is the $(k \times k)$ identity matrix and P is a $k \times (t-k)$ matrix.

Example: C is (7,4) Hamming with check matrix $H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$

If $(v_1, v_2, v_3, v_4, v_5, v_6, v_7) \in C$, then
$$\begin{aligned} v_2 \oplus v_3 \oplus v_4 \oplus v_5 &= 0 & &\rightarrow & v_5 &= v_2 \oplus v_3 \oplus v_4 \\ v_1 \oplus v_3 \oplus v_4 \oplus v_6 &= 0 & &\rightarrow & v_6 &= v_1 \oplus v_3 \oplus v_4 \\ v_1 \oplus v_2 \oplus v_4 \oplus v_7 &= 0 & &\rightarrow & v_7 &= v_1 \oplus v_2 \oplus v_4 \end{aligned}$$

$$G = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \\ P \end{array} \Rightarrow P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$(v_1, \cdots, v_k, v_{k+1}, \cdots, v_t) \in C$ iff $R(v_1, \cdots, v_k) = (v_{k+1}, \cdots, v_t) = (v_1, \cdots, v_k)P$
↖ redundant bits in a codeword ↖ encoding

Example: For the above (7,4) code
$$(1 \quad 0 \quad 0 \quad 1) \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = (1 \quad 0 \quad 0)$$ and $(1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$ is a codeword.

Thus for combinational linear machines we have for redundant outputs, $R(z(t)) = y(t)AP = y(t)A'$ where A' is $A \cdot P$ and $z(t) = y(t)A$

Example: Let the original linear combinational device be defined as:
$$z(t) = y(t) \quad \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, n = 5,\ k = 3$$

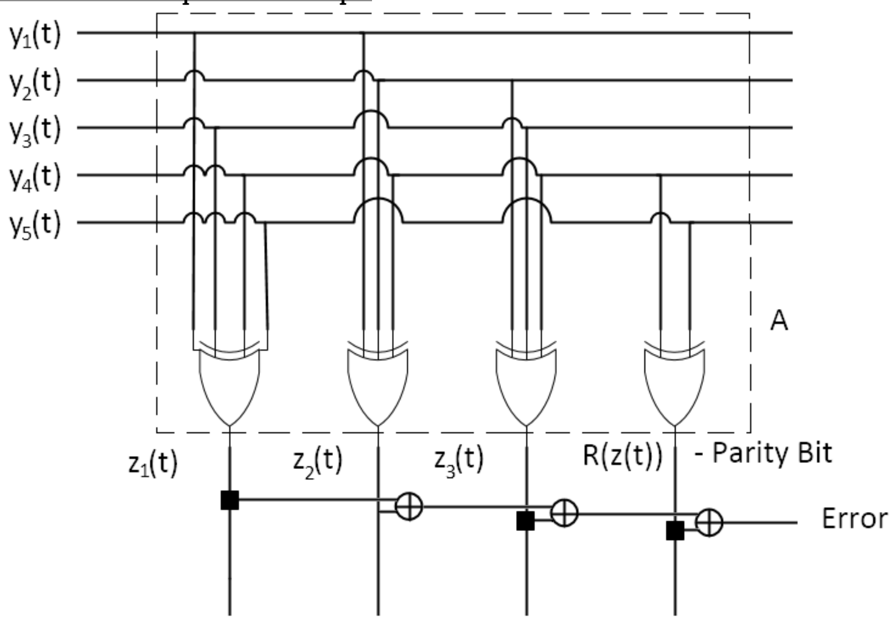
If we want to protect it with (4,3) 1-dim parity, then we have for the parity bit $P = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

$$R(z(t)) = y(t)A' = y(t) \cdot \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = y(t) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

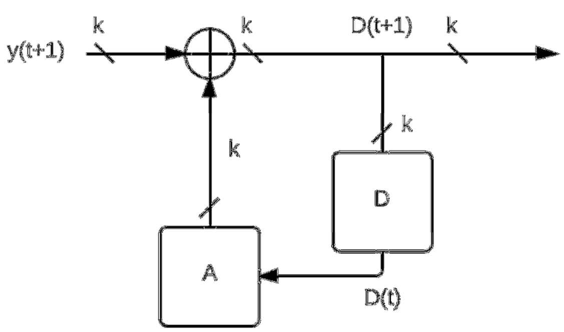
$$\nwarrow A'$$

If $y(t) = (y_1(t), y_2(t), y_3(t), y_4(t), y_5(t))$, then $R(z(t)) = y_4(t) \oplus y_5(t)$.

<u>Network for the previous example</u>



<u>Concurrent Checking of Linear Sequential Devices</u>



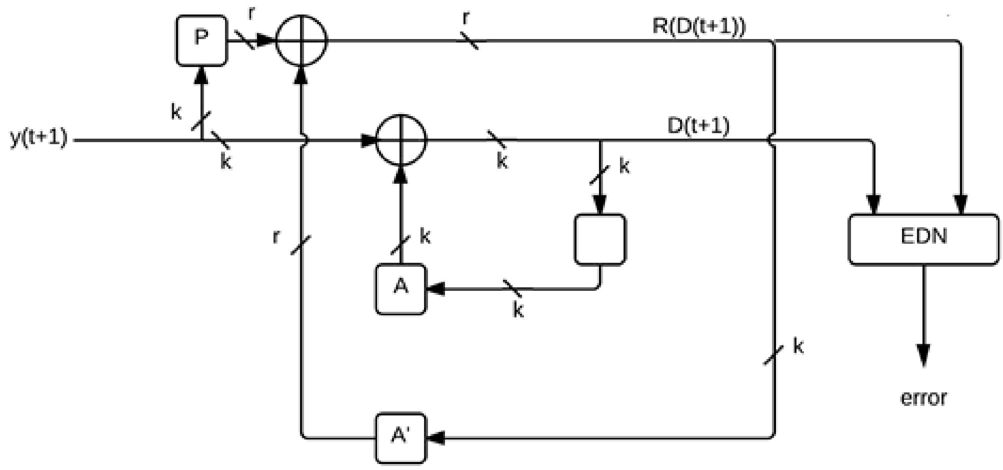
$D(t+1) = D(t)A \oplus y(t+1)$, $\qquad$ $D(t), y(t) \in GF(2^k)$

A is a $(k \times k)$ binary matrix.

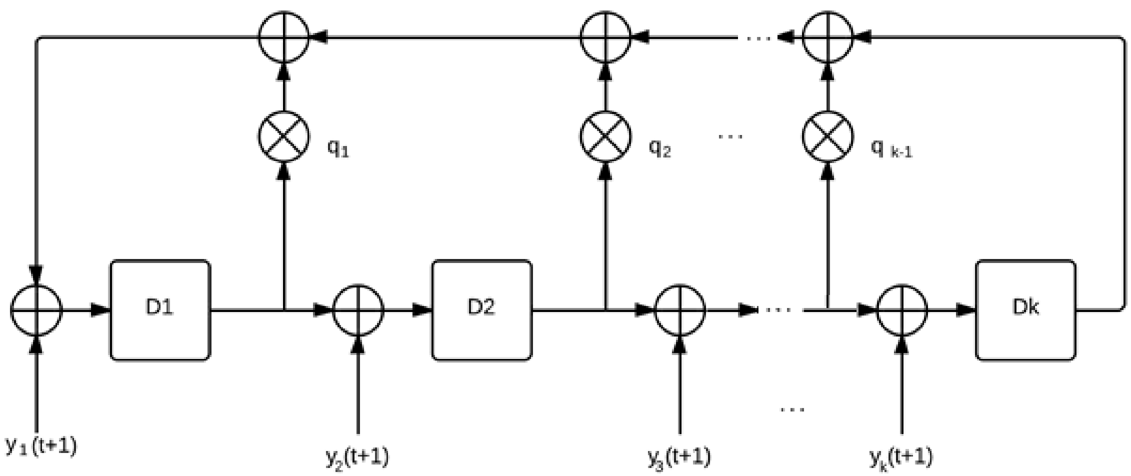
For concurrent checking by the code C with k information bits with $G = (I \vdots P)$, we have

$$\begin{aligned} R(D(t+1)) &= R(D(t)A) \oplus R(y(t+1)) \\ &= D(t)AP \oplus y(t+1)P \\ &= D(t)A' \oplus y(t+1)P, \qquad A' = A \cdot P \end{aligned}$$

Block Diagram for Concurrent Checking of Linear Sequential Devices



Example: Let the sequential device by MISR:



$$x^k + q_{k-1}x^{k-1} + \cdots + q_1 x + q$$

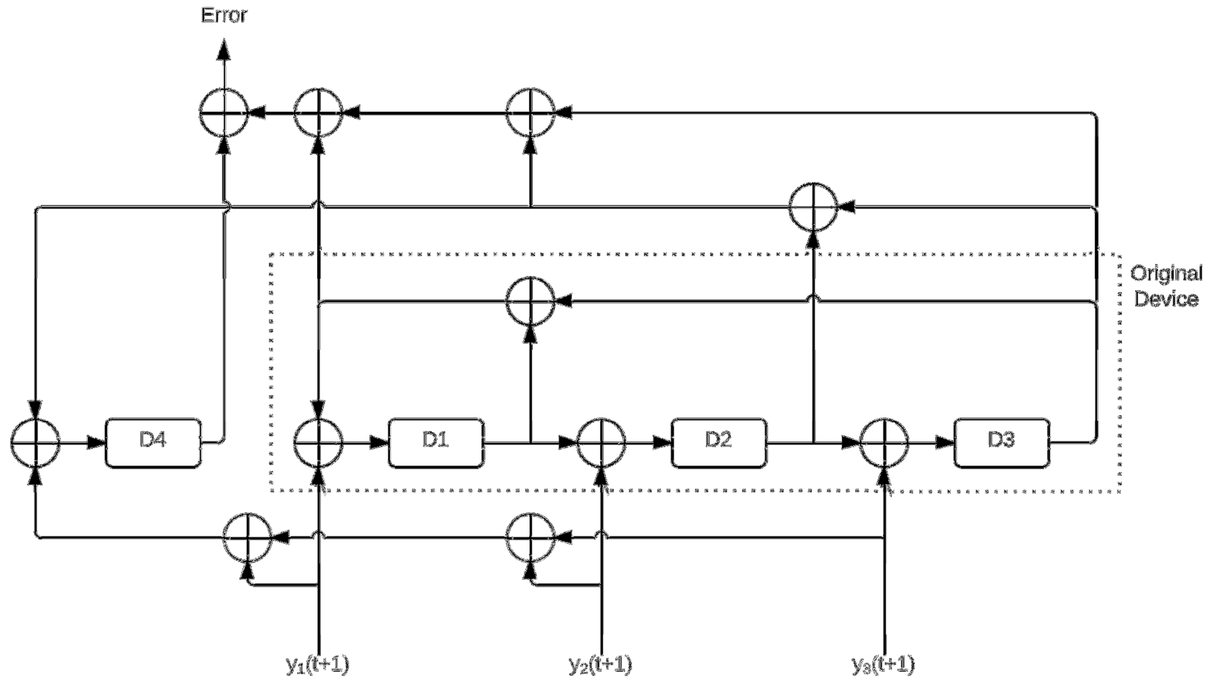
Then, $A = \underbrace{\left.\begin{bmatrix} q_1 & 1 & 0 & 0 & \cdots & 0 \\ q_2 & 0 & 1 & 0 & \cdots & 0 \\ q_3 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{k-1} & 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}\right\} k}_{k} = \begin{bmatrix} q_1 & & & \\ q_2 & & I_{k-1} & \\ \vdots & & & \\ q_{k-1} & & & \\ 1 & 0 & \cdots & 0 \end{bmatrix}$

Select $(k+1, k)$ code to protect this MISR, then $P = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ and $A' = AP = \begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \\ \vdots \\ \bar{q}_{k-1} \\ 1 \end{bmatrix}$, $\bar{q}_i = 1 - q_i$. For binary

$\bar{q}_i = 1 \oplus q_i$

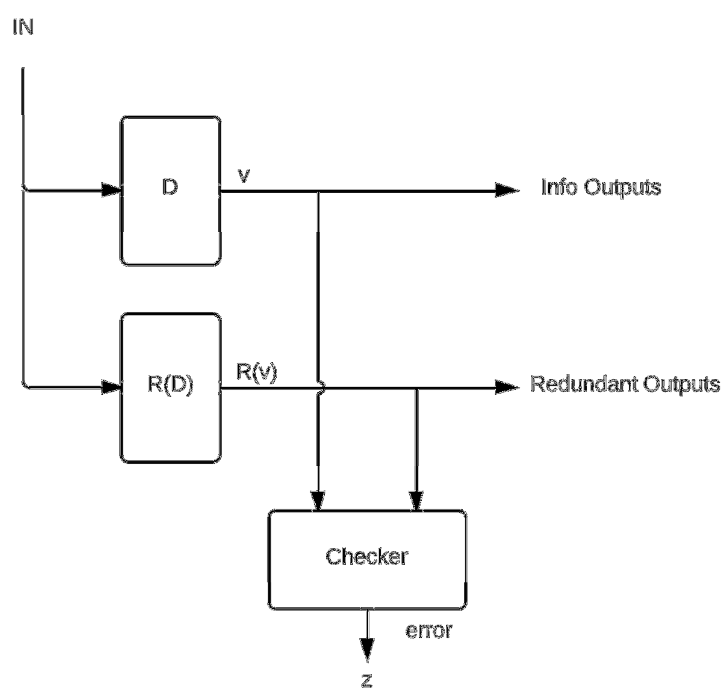
$y(t+1)P = \bigoplus_{i=1}^{k} y_i(t+1)$
$D(t)A' = \bigoplus_{i=1}^{k} D_i(t)\bar{q}_i \oplus D_k(t)$

Example: $k = 3$



Self Checking Checkers

(Self-checking decoders for error detecting/correcting codes)



$x = (v, R(v)) \in X -$ input code range of $z(x)$ for fault-free $v, R(v)$
$z(x) -$ output
Z domain of $z(x)$ for fault-free $v, R(v)$, z output code

Consider class F of faults in the checker
<u>Ex</u>. F is SSFS.

$Z(x) \xrightarrow{\quad\uparrow\quad} Z_f(x), \quad f \in F$

Fault f in the checker

Def 1. Checker is <u>fault-secure</u> iff for $\forall x \in X$ and $\forall f \in F$, $z_f(x) \notin Z \rightarrow$ fault is detected by the checker

Def 2. Checker is <u>self-testing</u> iff for $\forall f \in F\ \exists x \in X :\ z_f(x) \notin Z$

$\therefore$ there exists at least one fault-free input which provokes a fault $f \in F$ in the checker and distorts the outputs of the checker.

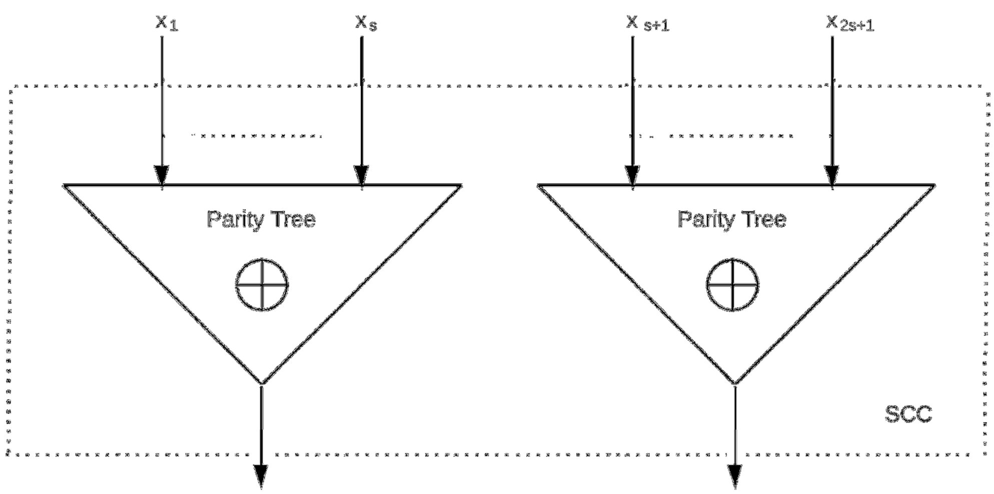
Def 3. A circuit is a <u>checker</u> iff for any $x \notin X$, $z(x) \notin Z$ and $x \in X$, $z(x) \in Z$ (This is also known as the <u>code-disjoint</u> property)

Def 4. A checker is <u>self-checking</u> iff fault-secure and self-testing

T1. A self-checking checker needs to have at least two output lines, each of which must take values 1 and 0 during normal operation.

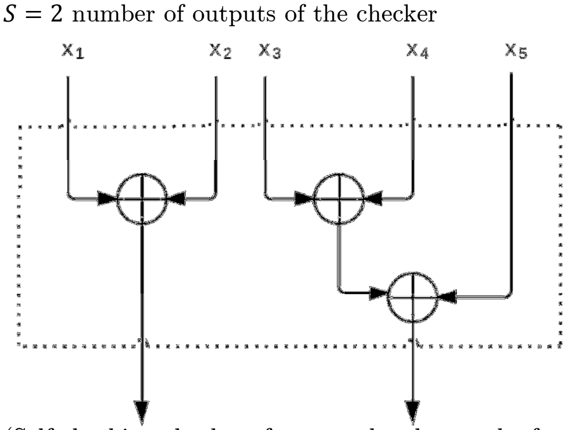
<u>Examples of Self Checking Checkers</u> (SCC)

X is an <u>odd</u> parity code of odd length $(x_1, \cdots, x_{2s+1}) \in X$ iff $x_1 \oplus x_2 \oplus \cdots \oplus x_{2s+1} = 1$



F is a set of SSFS. $\qquad Z = \{01{,}10\}$

$S = 2$ number of outputs of the checker

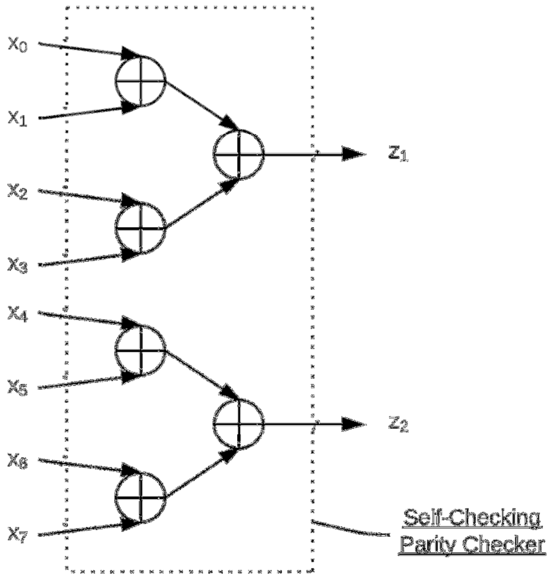


(Self-checking checkers for several codes can be found in: Pradhan, Fault Tolerant Computing Vol 1, Ch 5.)

Self-Testing Checker for Parity Verification

Verify $x_0 \oplus x_1 \oplus \cdots \oplus x_{m-1} = 0 \Leftrightarrow x \in V$

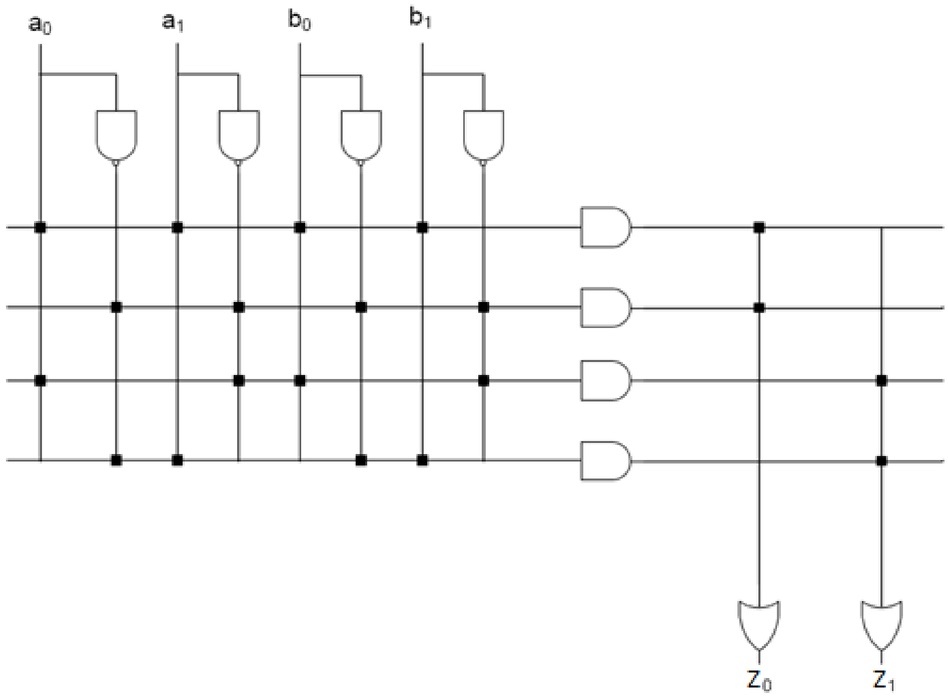
Example: $m = 8$



1) If $x \in V$ and no faults in the checker $\Rightarrow$ $z_1 = z_2$
2) If $x \notin V$ (odd number of ones in x) and no faults in the checker $\Rightarrow$ $z_1 \neq z_2$
3) If $x \in V$ and there is a SSF in the checker, then $z_1 \neq z_2$

Totally Self-Checking Match Detector

$k = 2,\ a = (a_0, a_1),\ b = (b_0, b_1)$
$a, b \in X = C_{In} \Leftrightarrow a = b,\ Z = C_{Out} = \{01{,}10\}$



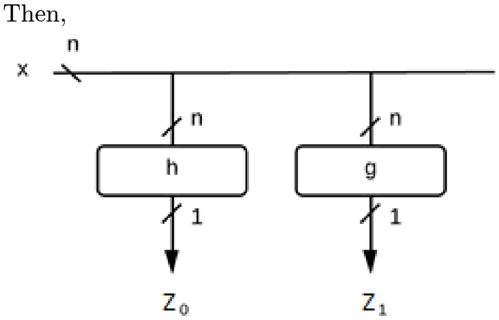
$Z_0 = 1,\ a_0 = b_0 = a_1 = b_1$
$Z_1 = 1,\ a_0 = b_0 \neq a_1 = b_1$

Design of Totally Self-Checking Checkers (TSCC)

Let $f(x) = 1 \Leftrightarrow x \in C_{In}$ – Input code
$x \in \{0{,}1\}^n$

Represent f as $h(x) \oplus g(x)$, then the following is a TSCC with $C_{Out} = \{01{,}10\}$ iff C_{In} is a test set for Single Stuck-at Faults (SSFS) in networks implementing h and g.

Then,



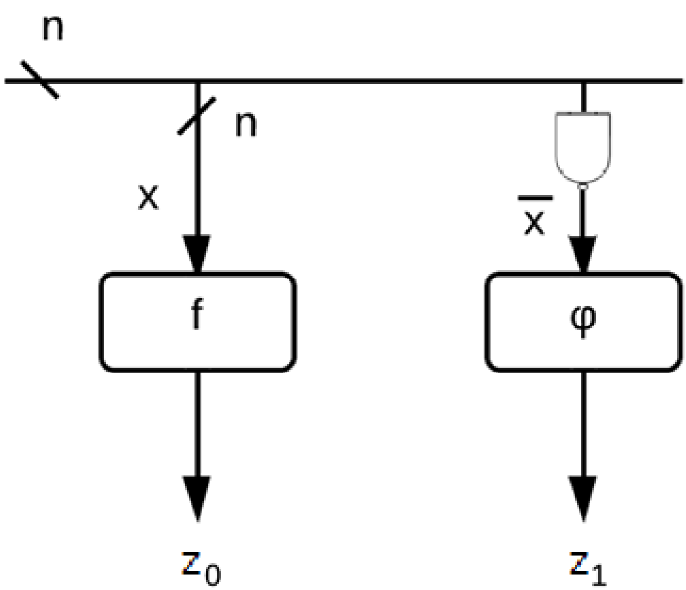
is a TSCC with $C_{Out} = \{01{,}10\}$ iff C_{In} is a test set for SSFS in networks implementing h and g.

Dual-Rail Design of TSCC with $\boldsymbol{Z = C_{Out} = \{01, 10\}}$

Let $f(x) = 1 \Leftrightarrow x \in C_{In}$, $\quad x \in \{0,1\}^n$, $X = C_{In} \subseteq \{0,1\}^n$

Denote $\varphi(x) = \bar{f}(\bar{x})$, $\quad \bar{x}$ – componentwise negation of x

$\varphi(x)$ is called dual to $f(x) \rightarrow \varphi(\bar{x}) = \bar{f}(x)$



$X = C_{In}$ is a test for SSFs in f and φ, f and φ have the same complexity.

This approach is better than replication (duplication) of f (duplication does not provide for self-testing)

Example: Totally Self Checking Checker for $(3,1,3)_2$ repetition code

$X = C_{In} = \{000,111\}$
$Z = C_{Out} = \{01,10\}$

$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2x_3 = \bar{x}_1\bar{x}_2\bar{x}_3 \oplus x_1x_2x_3$