Institut
Mines-Télécom

Reliability Improvement Techniques

Embedded Systems

Lirida Alves de Barros-Naviner
Master Program

Digital Design for Reliability

How to design reliable processors on unreliable devices?

- Defect tolerance
- Fault tolerance

- Prevention
- Masking
- Detection
- Correction

Defect tolerance

Based on hardening the devices

- More strict design rules at mask-level
- More expensive manufacturing (area)

Based on programmability

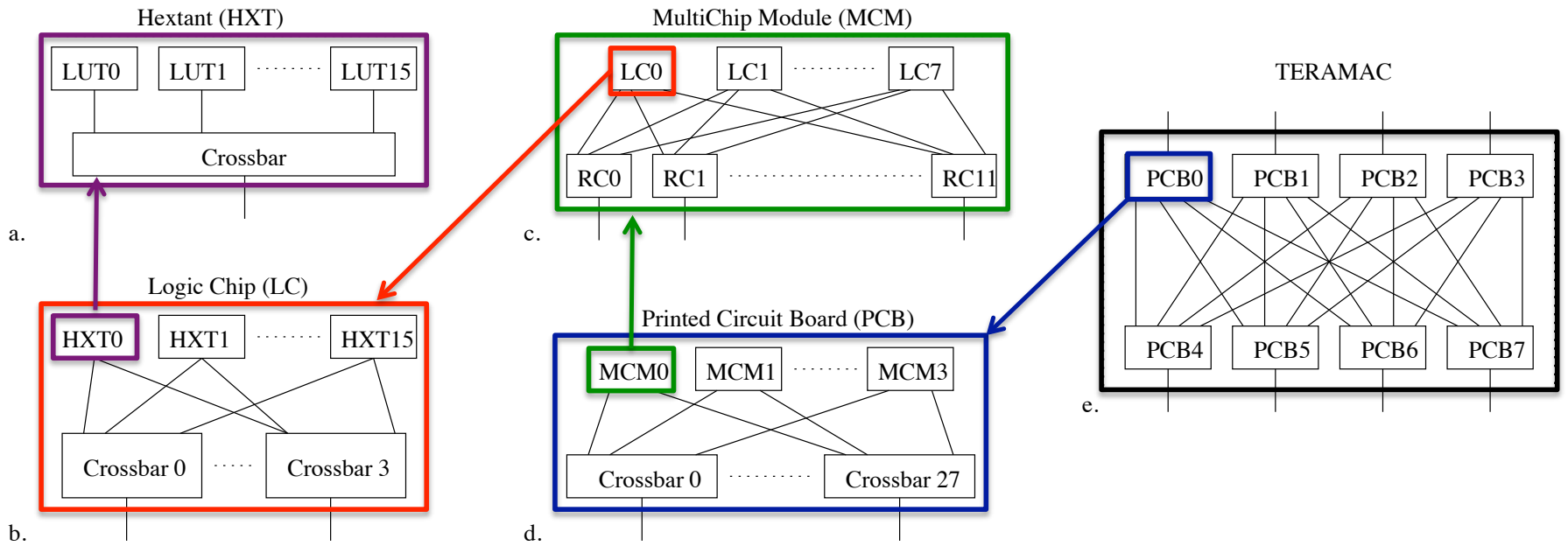
- Defectuous parts of the circuit are isolated
- Defect-free parts are used to implement the target function

Based on coding

- Very popular for memories
- Information redundancy

Teramac Design Hierarchy

- Defect-tolerance based on wires, switches, memory

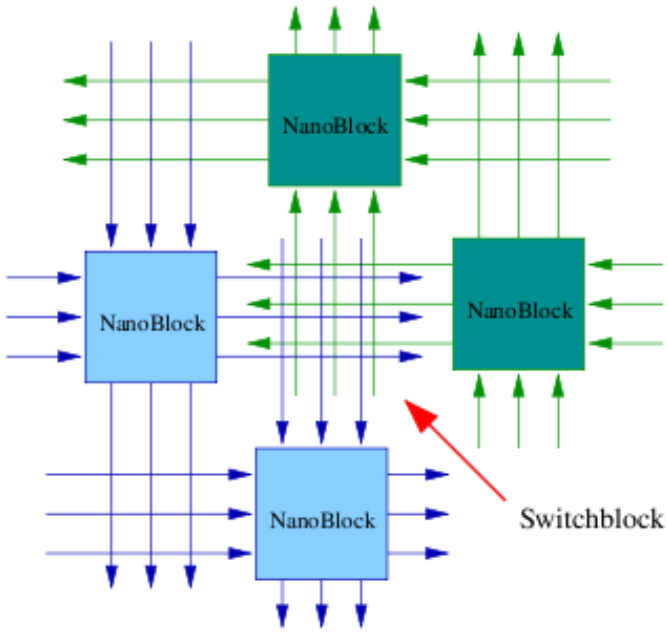


Heath et al. (1998)

864 FPGAs (647 with kind of defect), 3% of defective resources

Nanofabric Organization

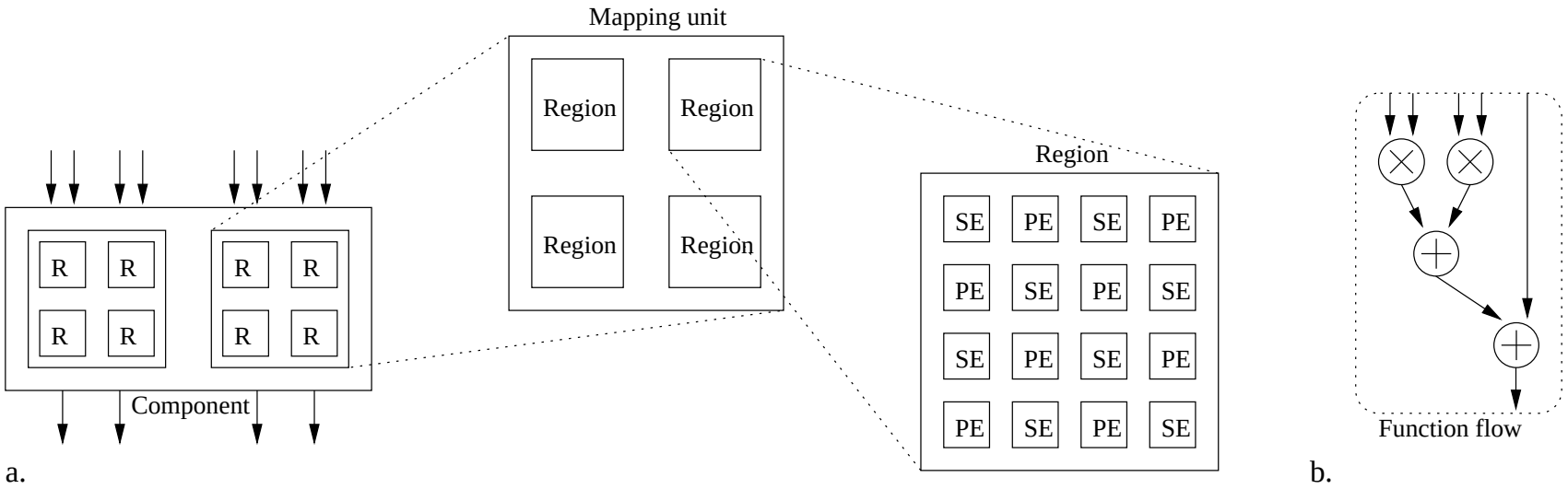
- Dense regular structures with reconfigurable capabilities



Goldstein and Budiu (2001)

Hierarchic Nanofabric

- Reconfiguration at a coarser grain
  - PEs can perform 8-bit arithmetic and logic operations
  - Tries to minimize time-consuming task of testing an mapping all of the nanofabric resources

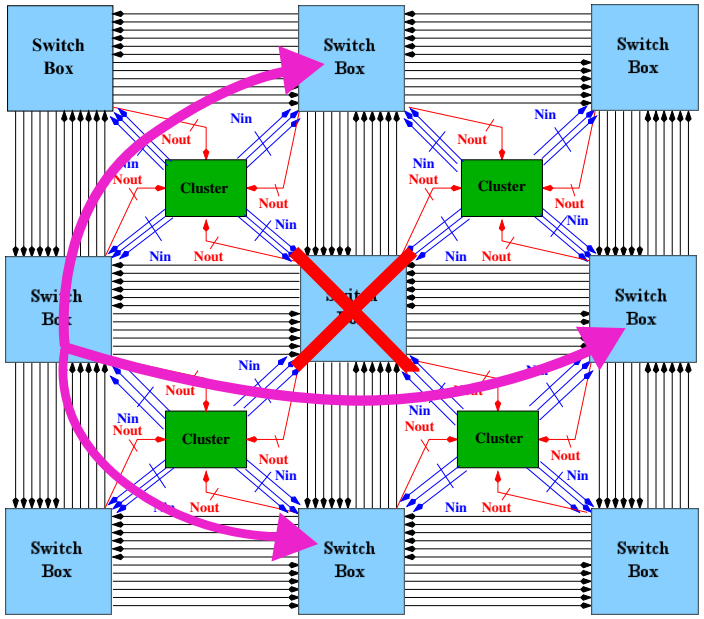


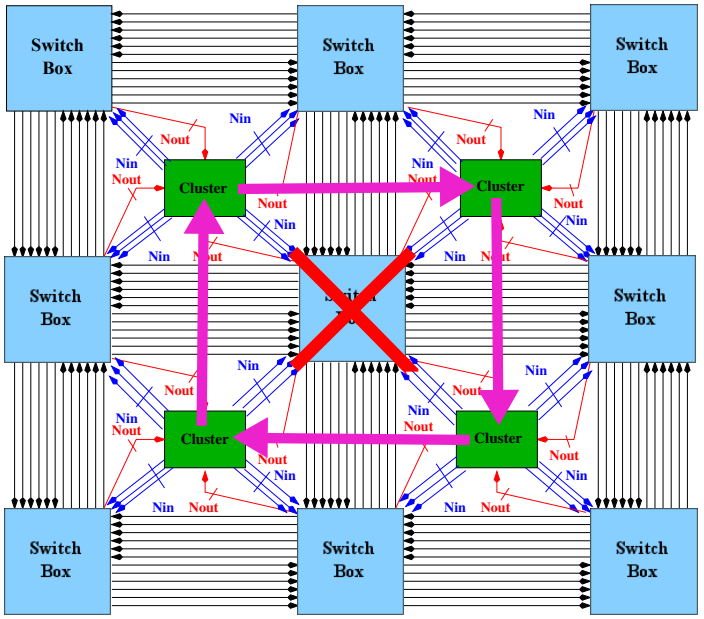
Chen et al. (2005)

SE= switch element, PE=processing element

SRAM based FPGA

Defect tolerance with additional connections and redundant basic blocks

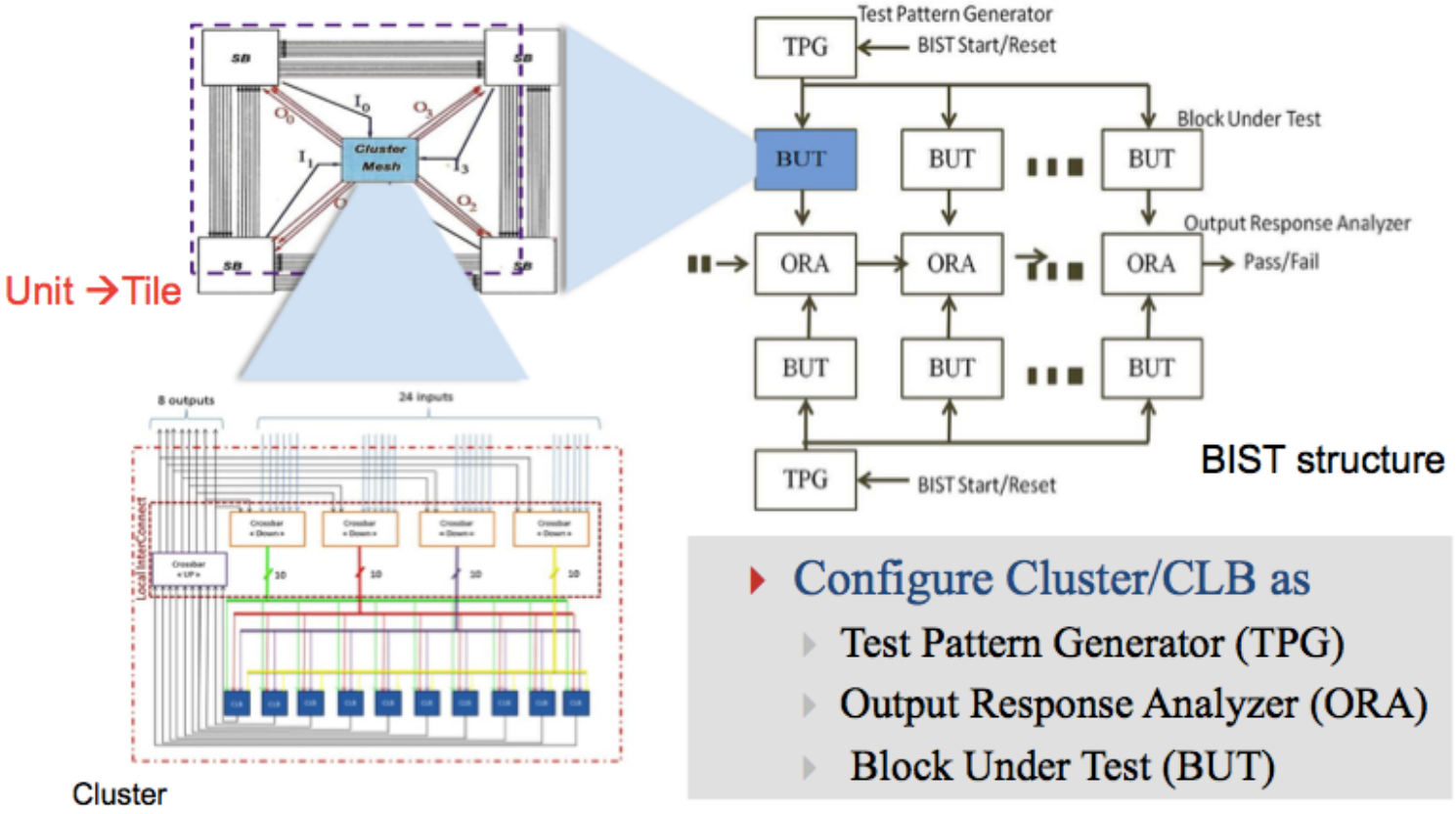




ANR RobustFPGA Project (2014)

SRAM based FPGA (cont.)

Defaults Cartography



Test strategy: BIST, ANR RobustFPGA Project (2014)

Outline

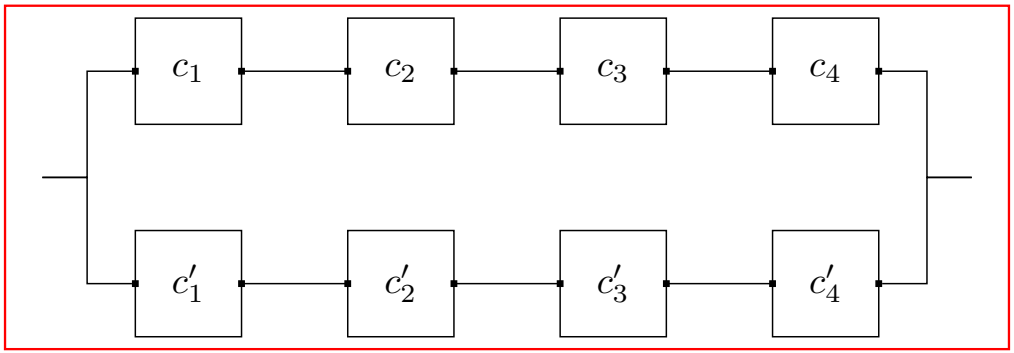
Introduction

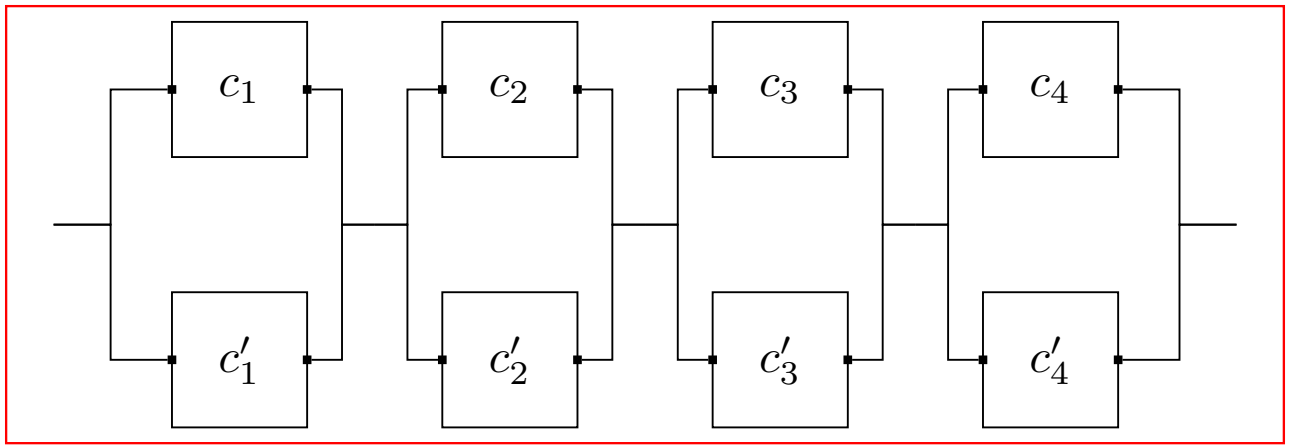
Manufacturing Defect Tolerance

Fault Tolerance
- Hardware redundancy
- Time redundancy
- Information redundancy

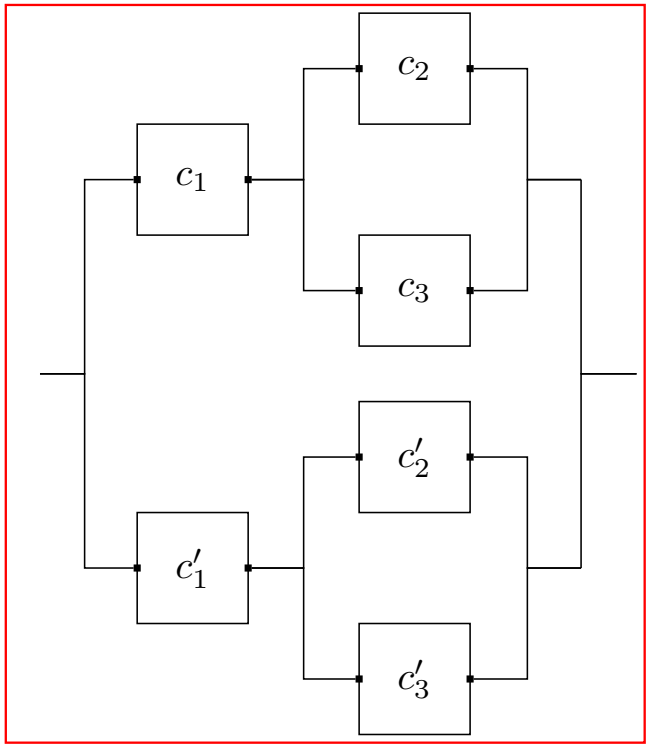
Conclusion

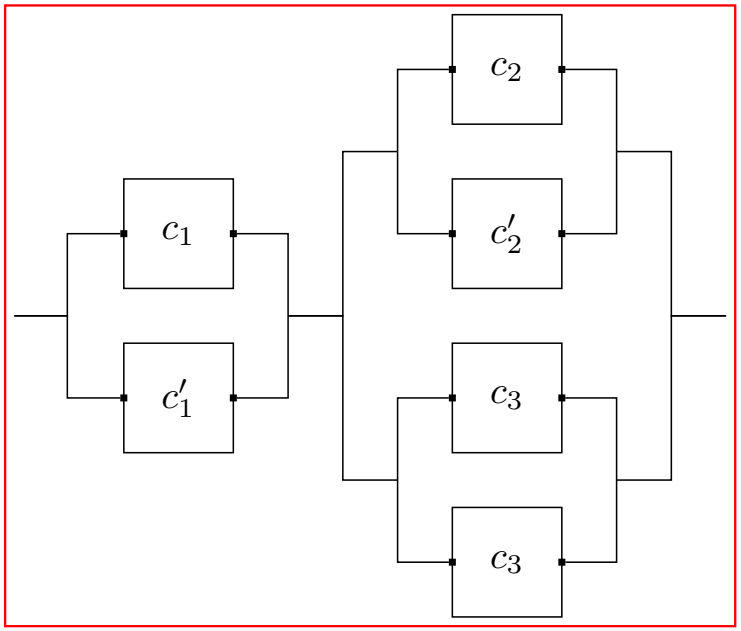
Coarse & Fine Grain Redundancy



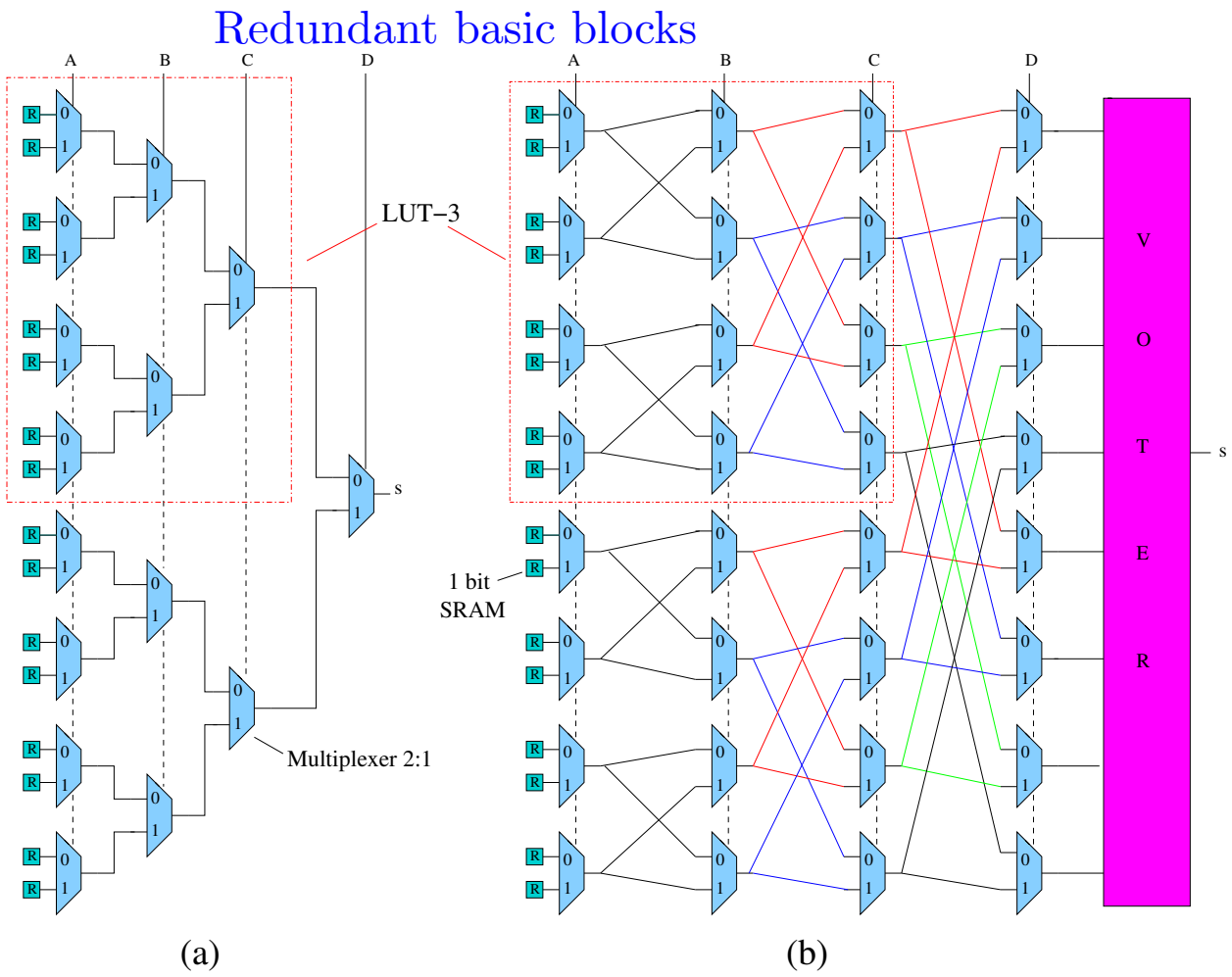


Coarse & Fine Grain Redundancy (cont.)



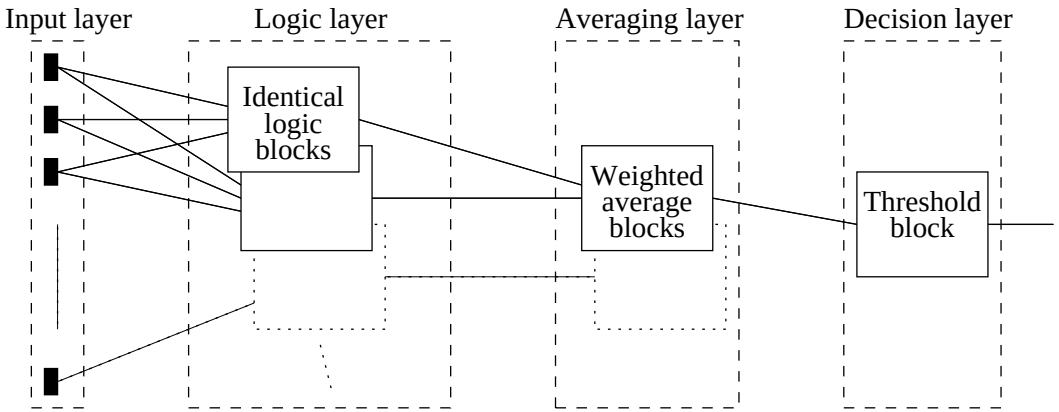


RobustFPGA Approach

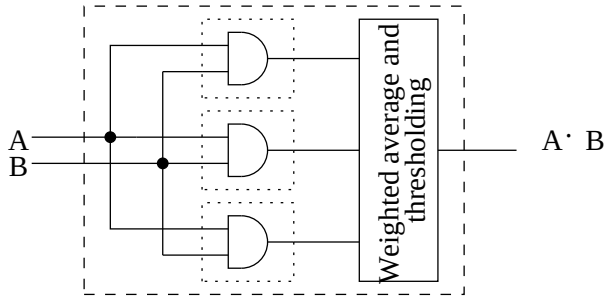


Multiple-valued Logic Approach

- Use of bit stream operators



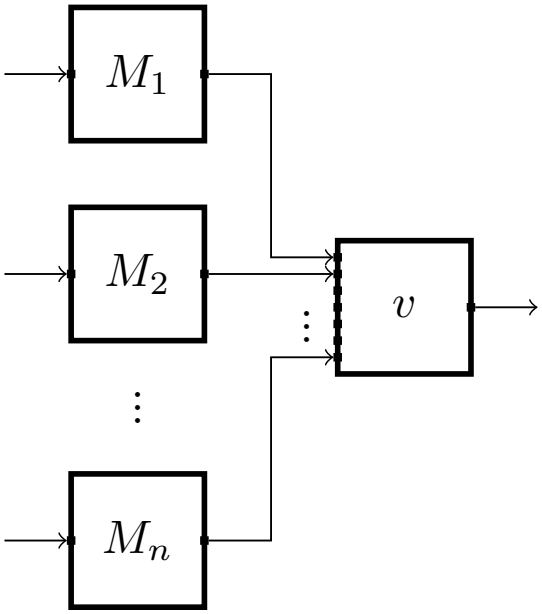
a.



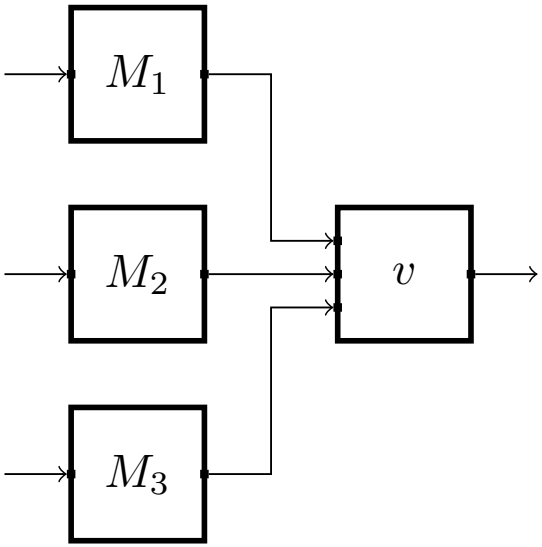
b.

Schmid and Leblebici (2004)

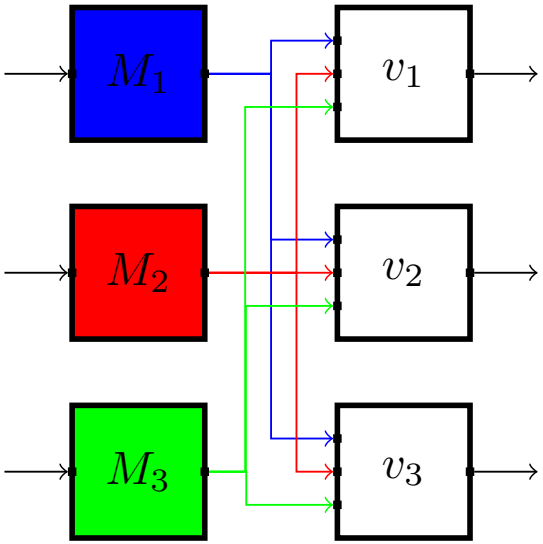
Modular Redundancy



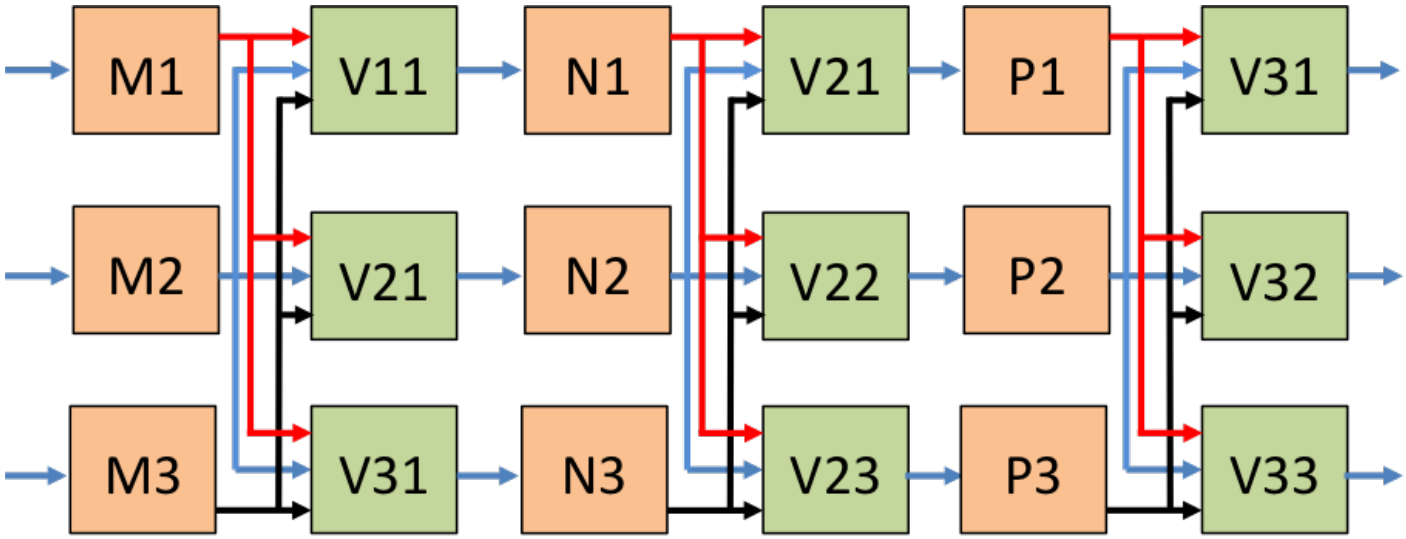
Triple Modular Redundancy



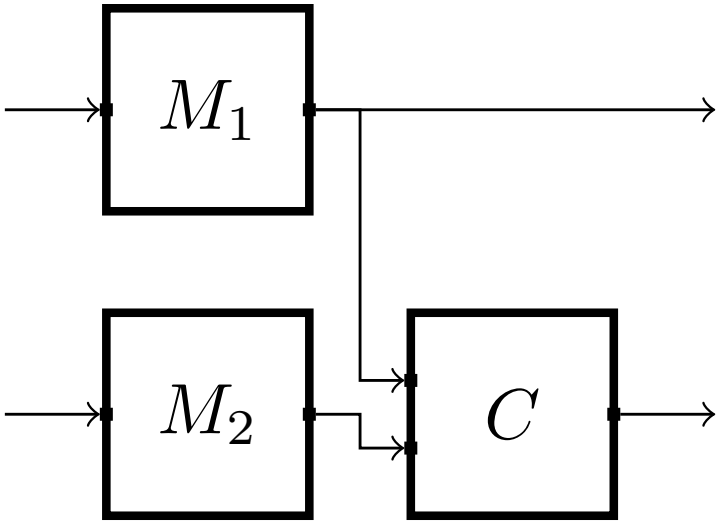
Triple Modular Redundancy (cont.)



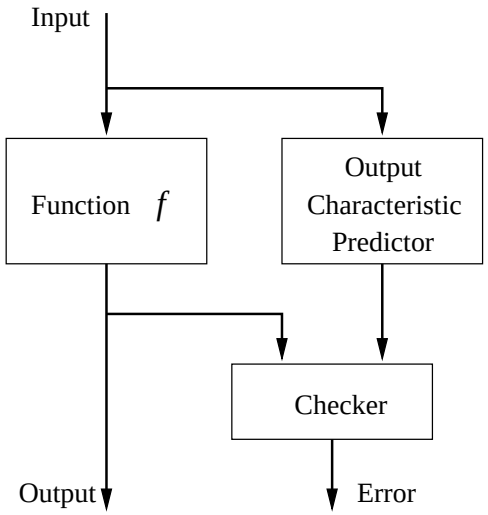
Triple Modular Redundancy (cont.)



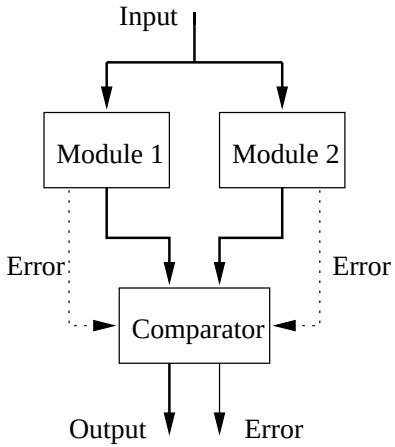
Duplication With Comparison



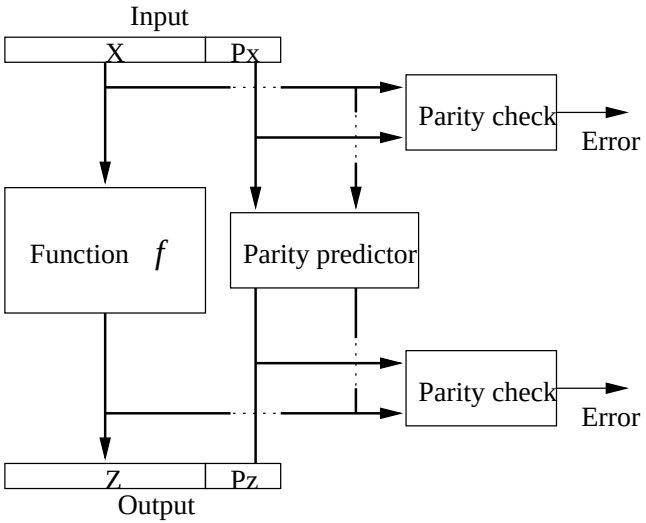
Concurrent Error Detection



a.

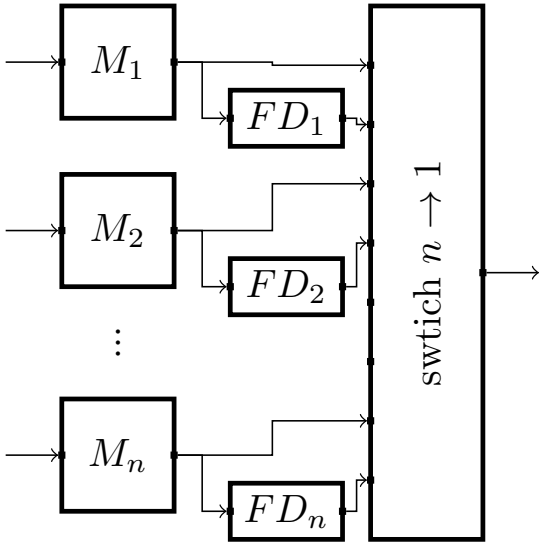


b.



c.

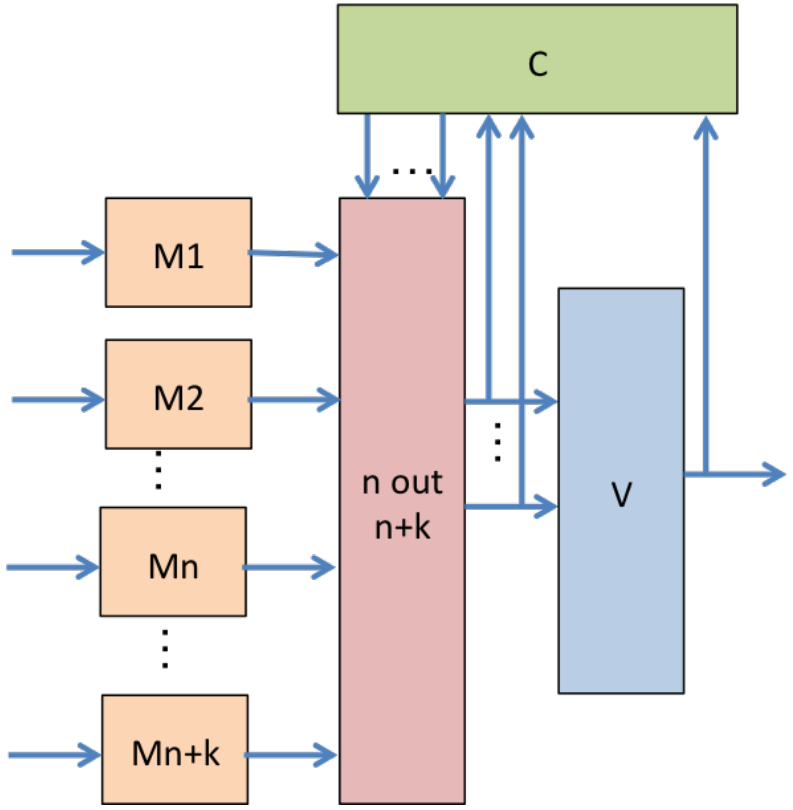
Standby Redundancy



- Only one block is active at a time.
- The **FD** blocks implement fault detection (often code based).

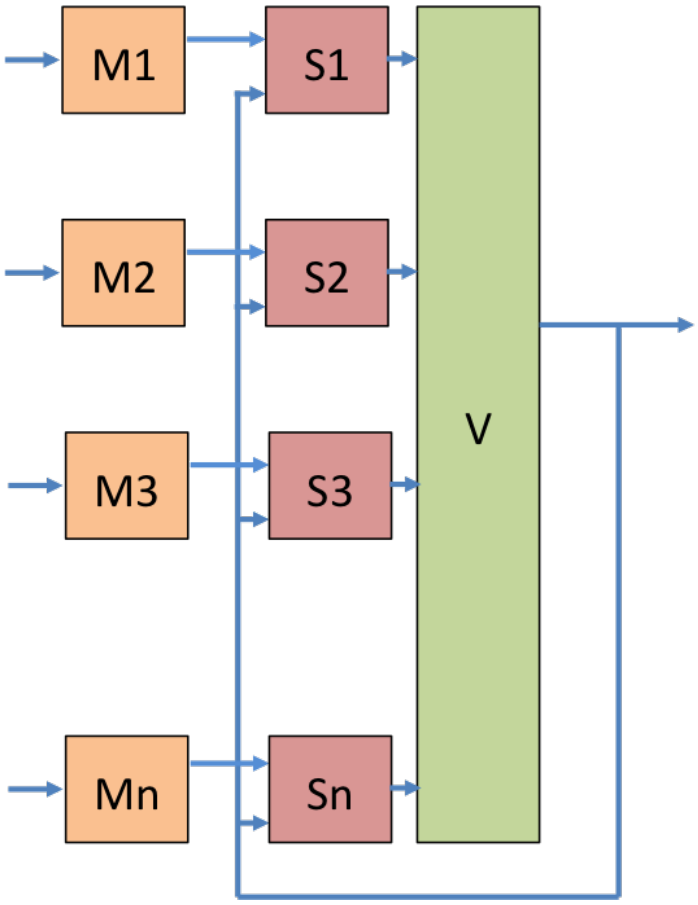
- Pair and Spare: similar to standby redundancy, but with two active modules

NMR with Spare



- n blocks are active at a time.
- The block **C** detects faulty blocks. It controls the action of the n-out-$n+k$ selector
- A faulty block is replaced by a spare one.

NMR with Purging



- All blocks are active in the beginning.
- The voter considers weighted inputs.
- S blocks use voter's output to change modules weights ($w = 0$ for a faulty block).

Outline

Introduction

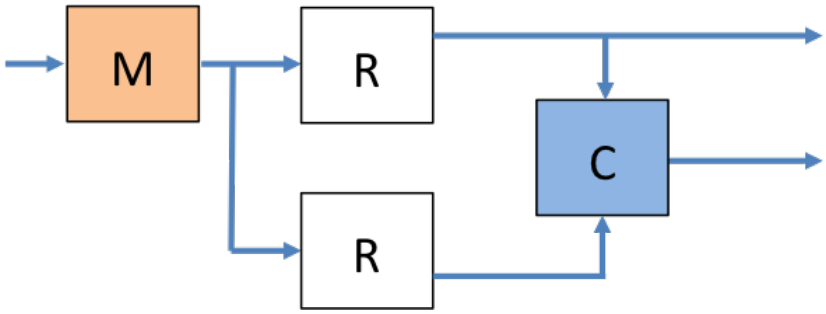
Manufacturing Defect Tolerance

Fault Tolerance
- Hardware redundancy
- Time redundancy
- Information redundancy

Conclusion

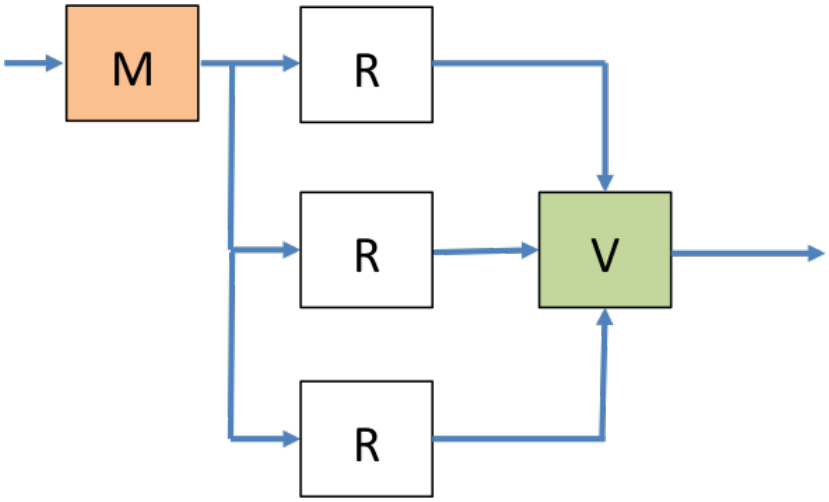
Time Redundancy & Recomputing

Transient faults: Duplication



Fault detection

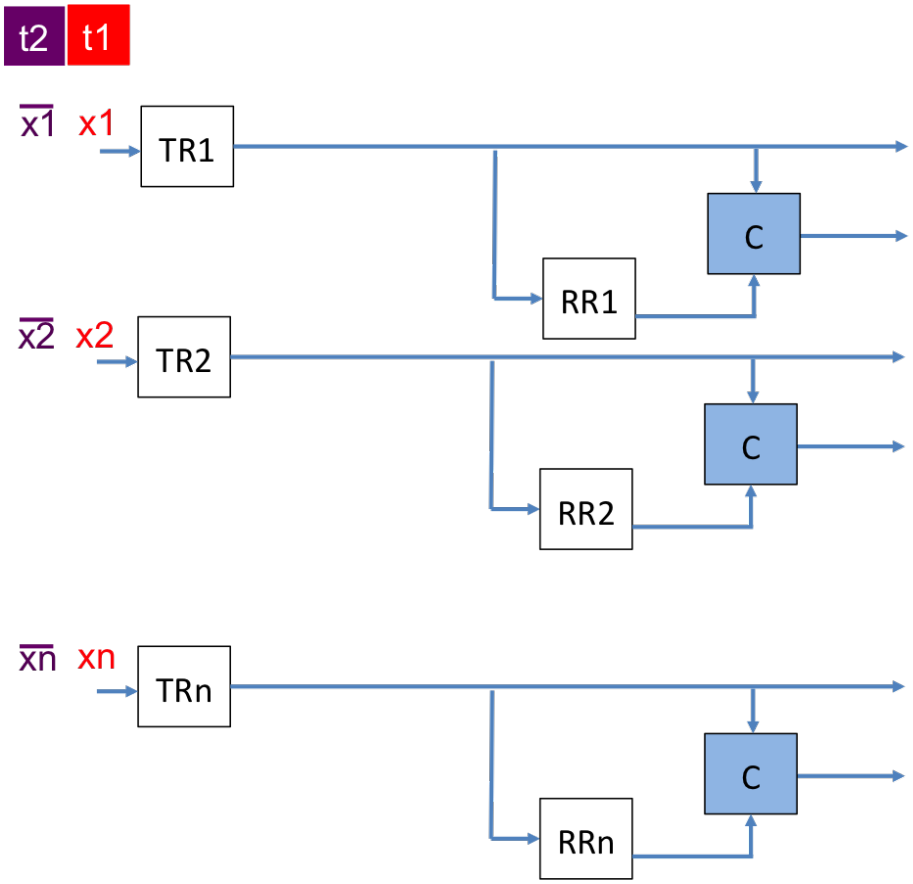
Transient faults: Triplication



Fault masking

Recomputing: Alternating Logic

Useful for permanent faults



Permanent fault detection

Recomputing: Operands Shifting

Useful for permanent faults

-	x_{n-1}	$\cdots$	x_{i+1}	x_i	$\cdots$	x_1	x_0
-	y_{n-1}	$\cdots$	y_{i+1}	y_i	$\cdots$	y_1	y_0
-	z_{n-1}	$\cdots$	z_{i+1}	z_i	$\cdots$	z_1	z_0

time $= k$

x_{n-1}	x_{n-2}	$\cdots$	x_i	x_{i-1}	$\cdots$	x_0	-
y_{n-1}	y_{n-2}	$\cdots$	y_i	y_{i-1}	$\cdots$	y_0	-
z_{n-1}	z_{n-2}	$\cdots$	z_i	z_{i-1}	$\cdots$	z_0	-

time $= k + 1$

c_{n-1}	c_{n-2}	$\cdots$	c_i	c_{i-1}	$\cdots$	c_0	-

check result

Recomputing: Duplication & Comparison

- Split the inputs into two parts:
  $A = A_H \| A_L$ and $B = B_H \| B_L$
- Time t_1:
  $f(A^L, B^L)$, with $A^L = A_L \| A_L$ and $B^L = B_L \| B_L$
- Time t_2:
  $f(A^H, B^H)$, with $A^H = A_H \| A_H$ and $B^H = B_H \| B_H$

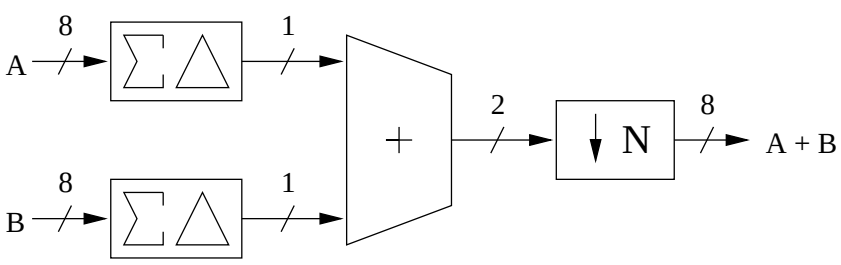
Recomputing: Swapped Operands

- Split the inputs into two parts:
  $A = A_H \| A_L$ and $B = B_H \| B_L$
- Time t_1:
  $f(A, B)$
- Time t_2:
  $f(A^s, B^s)$, with $A^s = A_L \| A_H$ and $B^s = B_L \| B_H$

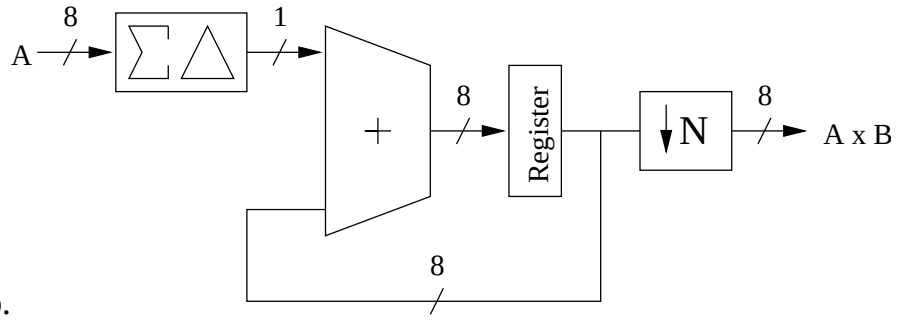
Time Redundancy

Sigma Delta Modulation Approach

- Use of bit stream operators



a.



b.

Schuller et al (2005)

Outline

Introduction

Manufacturing Defect Tolerance

Fault Tolerance
- Hardware redundancy
- Time redundancy
- Information redundancy

Conclusion

Information Redundancy

- Add redundant bits to the information representation
- Mainly used for memories (data storage)
  - Error correcting coding (ECC)
- Hardware/time redundancy can be viewed as information redundancy

Information Coding

- A **code** is a form of information representation satisfying some rules.
  - A binary code with **length** n is a set of binary n-tuples respecting the code rules.
  - The set $\{0,1\}^n$ of all 2^n n-tuples is named **codespace**
  - A n-tuple of the codespace that respects the code rules is named **codeword**
- **Encoding** is the process of changing a binary data k-tuple into a codeword
  - A $n-k$ is the number of **check** bits

Minimal Hamming Distance d_{min}

- Hamming distance between two codewords

$$d(\vec{x}, \vec{y}) = |\ \{i\ |0 \leq i \leq n-1, x_i \neq y_i\}|$$

  where $\vec{x}, \vec{y}$ are codewords

- Minimal Hamming distance is related to number of detectable/correctable errors
- n, k and d_{min} parameters define a (n, k, d_{min}) code.

Parity Coding

- We consider a $n-$bits code
  - $k = n - 1$ bits of information
  - 1 check bit
- Example:
  - Even parity: $\vec{x} = 1010 \Rightarrow \vec{c} = 10100$
  - Odd parity: $\vec{x} = 1010 \Rightarrow \vec{c} = 10101$

Parity Coding (cont.)

x_2	x_1	x_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	1	1	1	1

Even parity coding

Berger Codes

- Check bits represent the number w of $1's$ in the data word
- Code length is $n = k + m$, where $m = \lfloor \log_2(k+1) \rfloor$
- Check bits are given by the complement of w's binary representation

$$\vec{x} = 0010 \Rightarrow \vec{c} = 0010\mathbf{110}$$

Berger Codes (cont.)

x_2	x_1	x_0	c_4	c_3	c_2	c_1	c_0
0	0	0	0	0	0	1	1
0	0	1	0	0	1	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	1	1	1	0	0

(n, k) Linear Code

- Defined over the finite field $\mathbb{F}_q$
  - $\mathbb{F}_2$ is the set $\{0, 1\}$ with $\oplus$ and $\cdot$ operations
- k-dimensional subspace $\mathbb{C}_k$ of a vector space $\mathbb{V}_n$
  - $\mathbb{V}_n$ is a subset of $\mathbb{F}_q^n$ with with addition and multiplication by scalar operations
    $\mathbb{F}_2^n$ is the set of all n-uples containing elements of $\mathbb{F}_2$.
- $\mathbb{C}_k$ subspace is based on k linearly independent vectors.
  - Any codeword of a (n, k) linear code can be written as a linear combination of the k basis vectors $\{\vec{v}_0, \vec{v}_1 ..., \vec{v}_{k-1}\}$ of subspace $\mathbb{C}_k$

$$\vec{c} = \sum_{j=0}^{k-1} \vec{x}_i \vec{v}_i$$

Matrix Representation

$$\vec{c} = \vec{x} \cdot \mathbf{G}$$

- $\vec{c}$ is the codeword
- $\vec{x}$ is the information word
- $\mathbf{G}$ generator matrix
  - The rows of $\mathbf{G}$ are k vectors which are a basis of C.
  - $\mathbf{G}$ has n columns.

Example

- Code (5,3) over $\mathbb{F}_2$
- Generator matrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x_2 & x_1 & x_0 \end{pmatrix} \cdot \mathbf{G} = \begin{pmatrix} c_4 & c_3 & c_2 & c_1 & c_0 \end{pmatrix}$$

$$\begin{pmatrix} c_4 & c_3 & c_2 & c_1 & c_0 \end{pmatrix} = \begin{pmatrix} x_2 & x_1 & x_0 & x_2 \oplus x_0 & x_1 \oplus x_0 \end{pmatrix}$$

Example (cont.)

x_2	x_1	x_0	c_4	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1
0	1	0	0	1	0	0	1
0	1	1	0	1	1	1	0
1	0	0	1	0	0	1	0
1	0	1	1	0	1	0	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	0	0

$(5, 3)$ linear code

Parity Check Matrix H

- $\mathbf{H}$ is a $(n-k) \times k$ matrix used to detect errors

$$\mathbf{H}.\mathbf{G}^T = 0$$

- The product of $\mathbf{H}$ by a vector produces a syndrome vector

$$\mathbf{H}.\vec{r}^T = \vec{s} \text{ where } \vec{s} = \vec{0} \text{ if } \vec{c} \text{ is a codeword}$$

- If $\mathbf{G} = [\mathbf{I}_k \mathbf{A}]$, then $\mathbf{H} = \left[\mathbf{A}^T \mathbf{I}_{n-k}\right]$
- $d_{\mathbb{C}} \geq d \Leftrightarrow$ all subsets of $d-1$ columns of $\mathbf{H}$ are linearly independent
  - Singleton bound: $n \geq d_{\mathbb{C}} + k - 1$

Hamming Code

$$\vec{c} = \vec{x}\mathbf{G} = (1010) \left(\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array} \right) = (1011010)$$

Code (7,4):
generation matrix **G**,
syndrome matrix **H**

No error $\Rightarrow \vec{s} = \vec{c}\mathbf{H}^{\mathbf{T}} = \vec{0}$

$$\vec{s} = \vec{c}\mathbf{H}^{\mathbf{T}} = (1011010) \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) = (000)$$

H is in a lexicographic form

Error Vector	Syndrome $\mathbf{s} = \mathbf{cH^T}$
1000000	100
0100000	010
0010000	001
0001000	110
0000100	101
0000010	011
0000001	111

Hamming Code

$$\vec{c} = \vec{x}\mathbf{G} = (1010) \left(\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right) = (1011010)$$

Code (7,4):
generation matrix **G**,
syndrome matrix **H**

No error $\Rightarrow \vec{s} = \vec{c}\mathbf{H}^{\mathbf{T}} = \vec{0}$

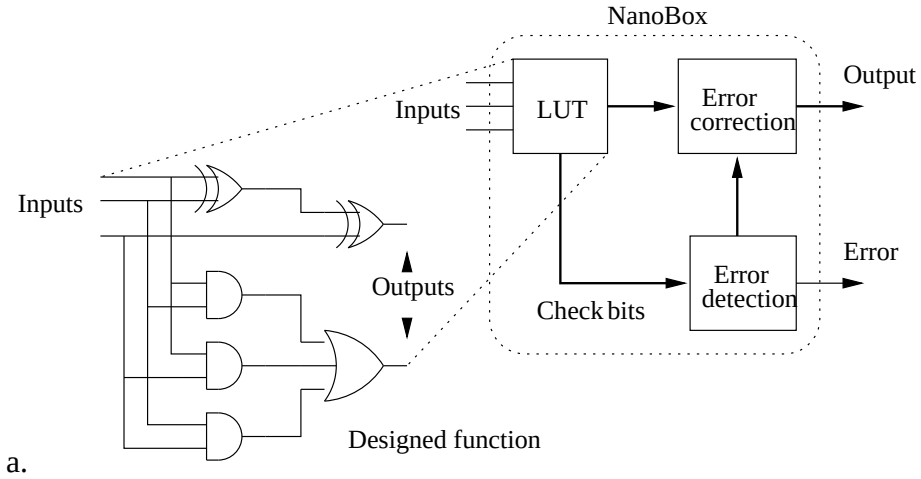
$$\vec{s} = \vec{c}\mathbf{H}^{\mathbf{T}} = (1011110) \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right) = (101)$$

H is in a lexicographic form

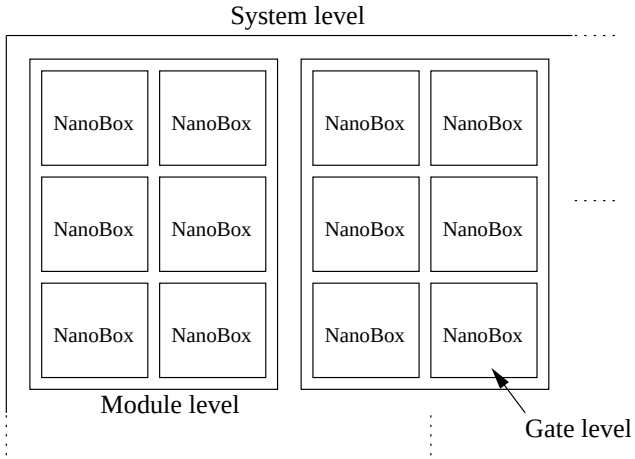
Error Vector	Syndrome $\mathbf{s} = \mathbf{cH^T}$
1000000	100
0100000	010
0010000	001
0001000	110
0000100	101
0000010	011
0000001	111

NanoBox Approach

- Dense regular structures with reconfigurable capabilities



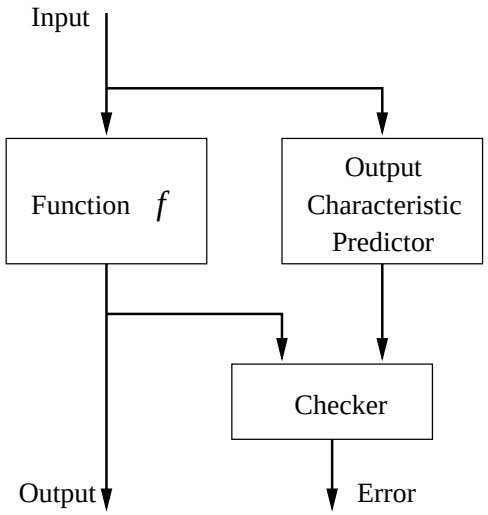
a.



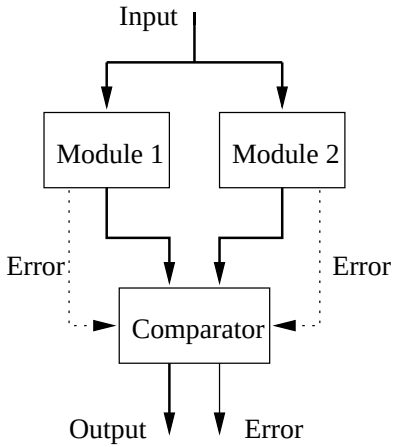
b.

Kleinosowski et al (2004)

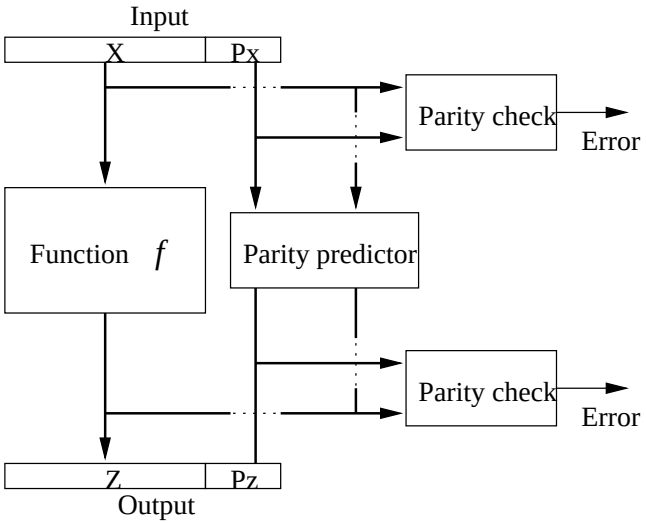
Concurrent Error Detection & Coding



a.

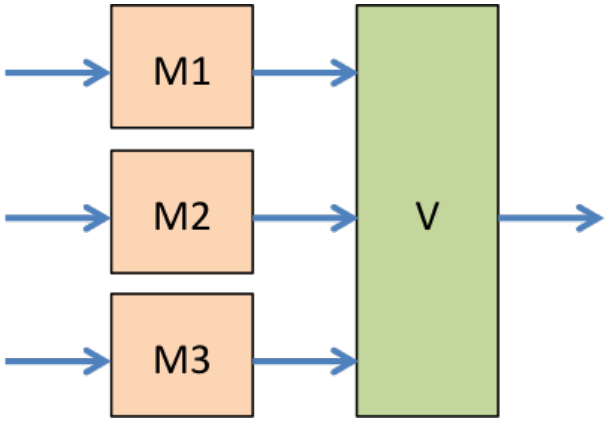


b.

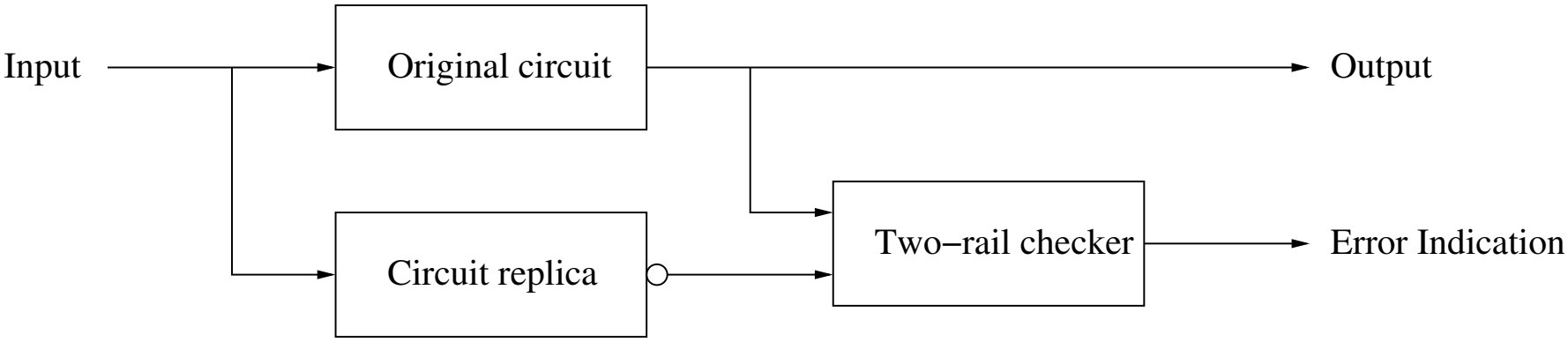


c.

TMR & Repetition Coding

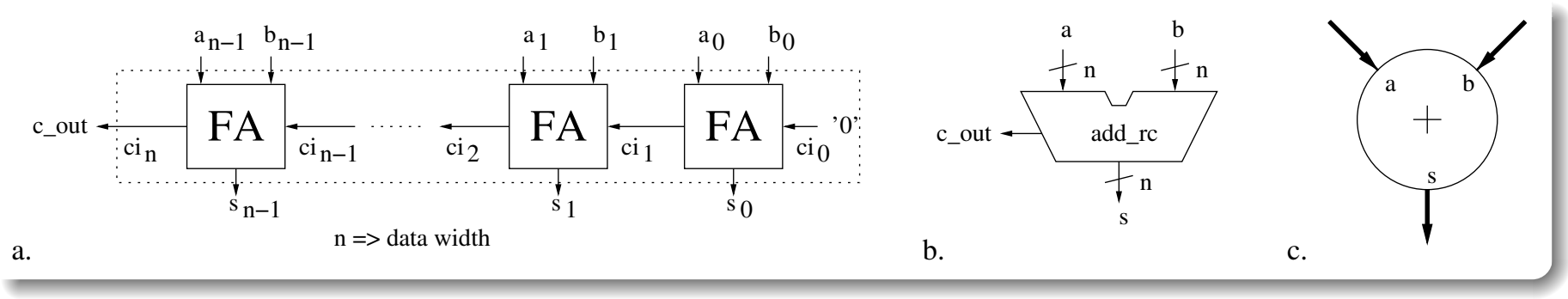


Repetition Coding

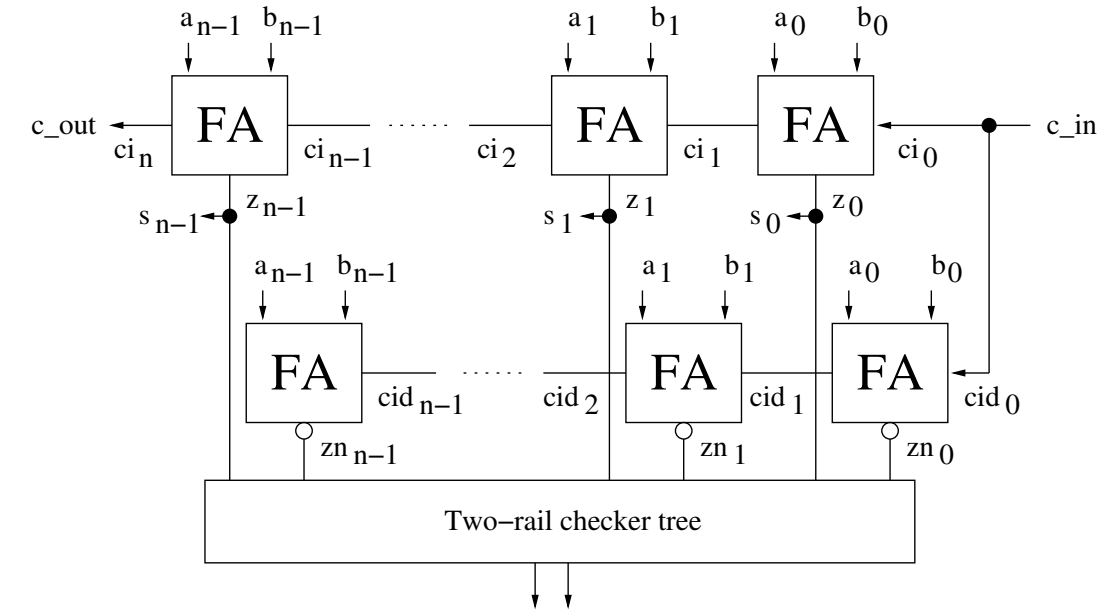


- Self-dual functions: $f(\overline{x_0}, \overline{x_1}, \cdots, \overline{x_n}) = \overline{f}(x_0, x_n, \cdots, x_n)$

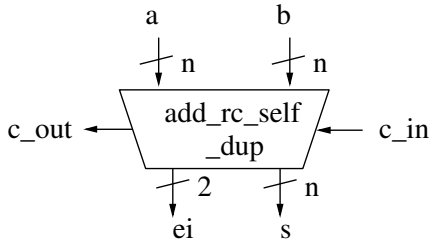
Ripple Carry Adder



Duplicated RC Adder



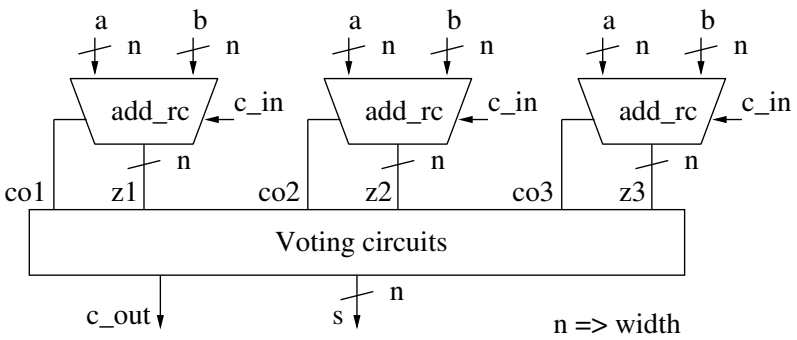
a.



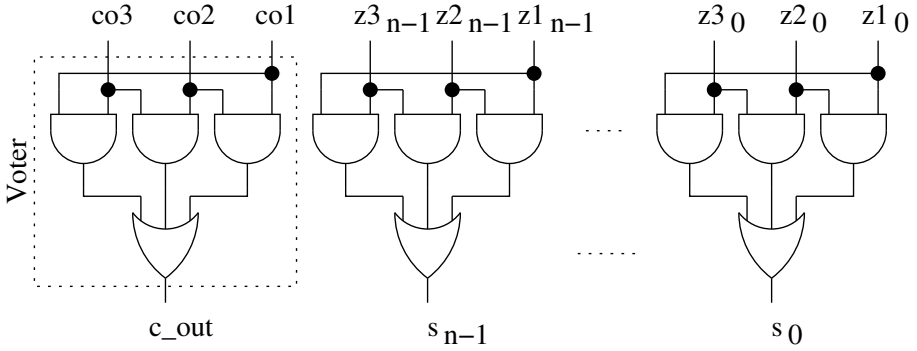
b.

Triple Modular Redundancy (TMR)

Ripple Carry (RC) Adder

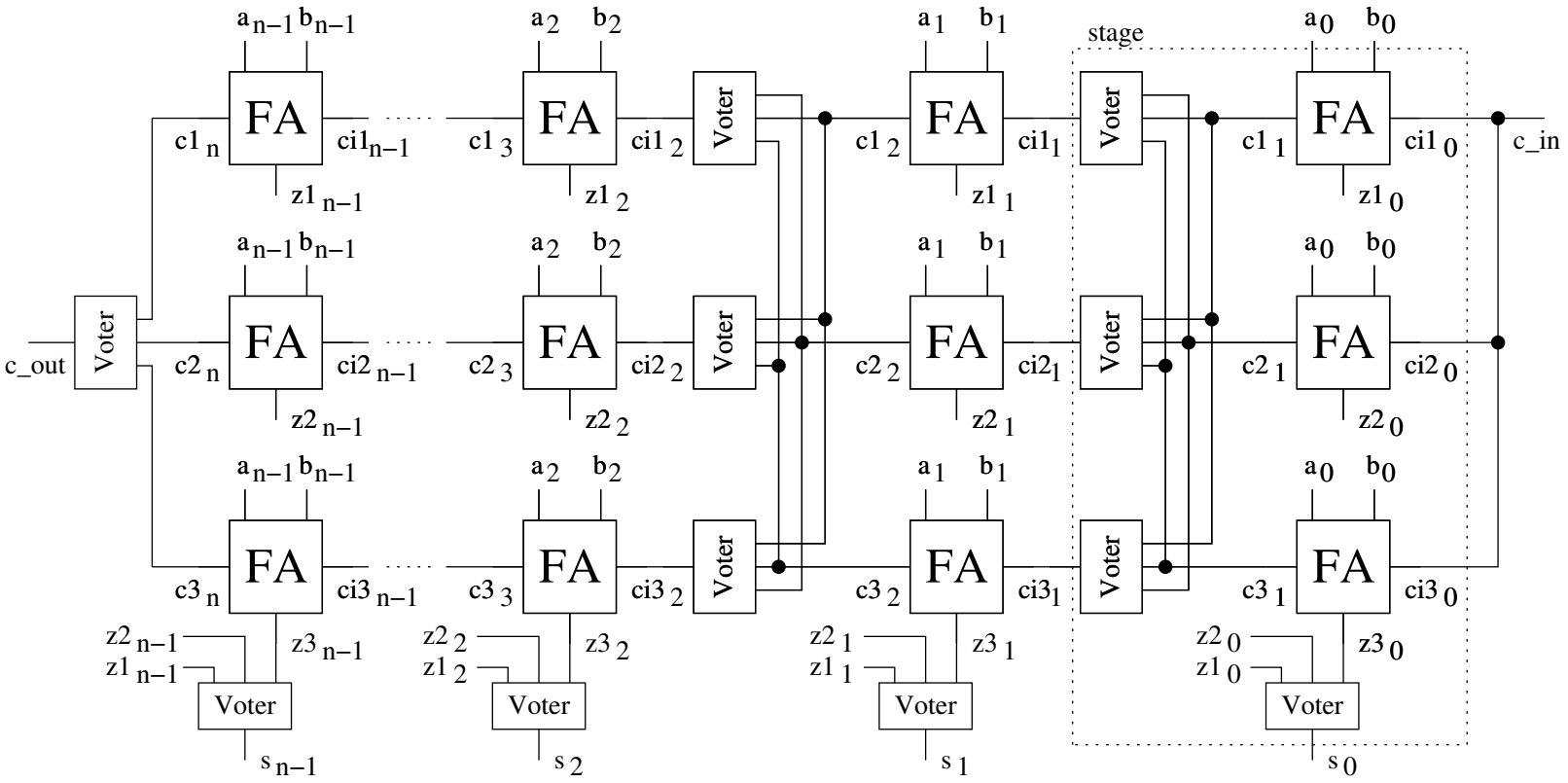


a.



b.

TMR RC – Redundant Voters



Conclusions

- This lesson dealt with different methods for reliability improvement
  - Detection and correction of errors
  - Solutions for transient and permanent errors
  - Strategies based on passive and active redundancy
- Reliability improvement leads to area and/or time penalties
- Next time, we will explore techniques for reliability assessment