

# **Coded Systems and Dynamic Approaches to Error Detection and Correction**

**Christoforos Hadjicostis**

Decision and Control Laboratory  
Coordinated Science Laboratory and Dept. of Electrical and Computer Engineering  
University of Illinois at Urbana-Champaign

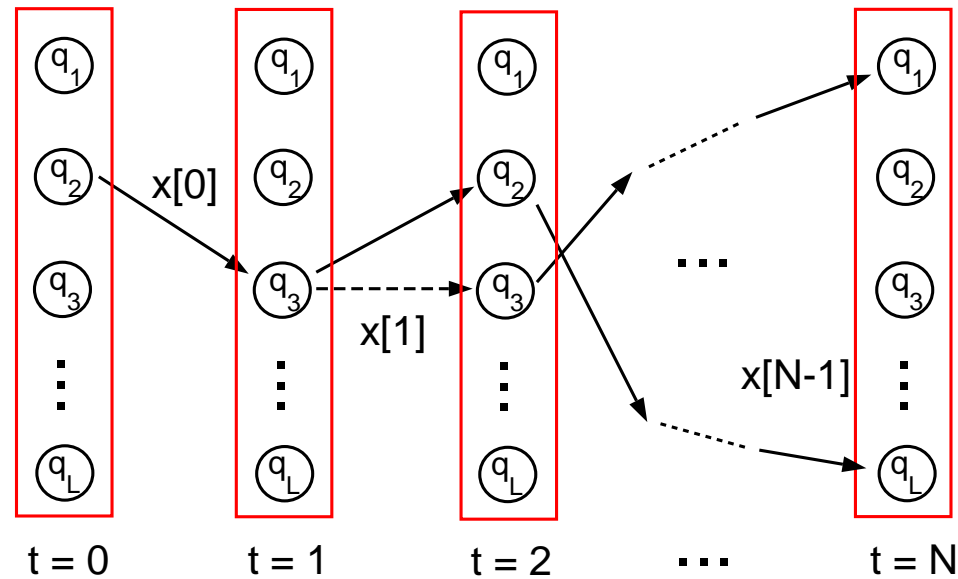
## C. HADJICOSTIS: MAIN RESEARCH THRUSTS (1)

- **Analysis of complex networked systems** (NSF ITR, NSF ECS)
  - Error control, monitoring, testing, verification
  - Infrastructure constraints (e.g., link delays, packet drops, distributivity)
  - Architectural constraints (hardware/communication/algorithmic overhead)
  - Fundamental limitations, robustness to incomplete/erroneous sensor data
  - Other issues: Model uncertainty, detection delay, sensor/actuator allocations
  - **Applications: Networked control systems, remotely controlled systems, traffic systems, manufacturing systems**
- **Path diversity in networked systems** (Lucent)
- **Soft-decision decoding** (Motorola)
- **Fault-tolerant operation of energy processing systems** (NSF EPNES, ONR)

## C. HADJICOSTIS: MAIN RESEARCH THRUSTS (2)

- **Fault-tolerant dynamic systems** (NSF Career, AFOSR URI)
  - Role of system dynamics and structure, coding for protection
  - Special-purpose architectures (e.g., communication, signal processing)
  - **Applications: Digital sequential systems, digital filters, embedded systems**
- **Error control and noise-tolerance in digital circuits** (NSF ITR)
  - Implications to speed, cost, power consumption
  - Novel digital designs
  - **Applications: Reliable systems from unreliable components**
- **Graduate course in Fall 2003: ECE 497CH**
  - Coding Approaches to Reliable System Design

## PREVIEW: NON-CONCURRENT (DYNAMIC) PROTECTION OF FSMs



**State-transition fault:** Transient fault during  $t = 1$  affects state evolution

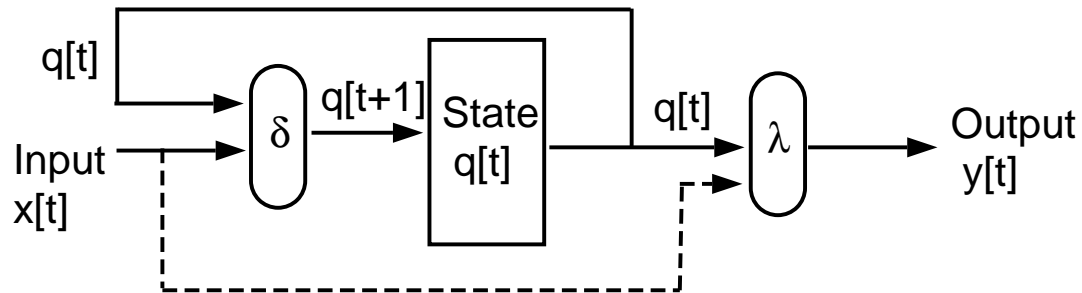
**Non-concurrent checking:** Efficient use of *redundancy* so that we can

- |  |   |   |
|--|---|---|
| (i) Detect errors  | } | Based on state at time step $N$           |
| (ii) Systematically “diagnose” errors<br>(what went wrong, how and when) |   |   |
|  |   | (a) <b>Reduce overhead</b>                |
|  |   | (b) <b>Relax reliability requirements</b> |

## RELATED WORK ON FAULT TOLERANCE AND DIAGNOSIS

- **Communication systems (channel noise, error-correcting codes)**
- **Computational circuits (hardware faults, modular redundancy)**
  - Each component fails with *constant* probability
  - Earlier work by von Neumann, Shannon, Winograd, Elias and others; later work by Pippenger, Gács, Hajek, Feder, Reischuk
- **Special-purpose systems (Algorithm-Based Fault Tolerance)**
  - Protect against *fixed* number of faults, fault-free corrector
  - Recent work by Abraham, Redinbo, Musicus, Beckmann
- **Fault diagnosis in discrete event systems**
  - Observability limitations, distributivity constraints, complexity of diagnoser
  - Work by Wonham, Teneketzis, Lafortune, Kumar, Benveniste, Giua
- **Fault tolerance and monitoring in finite-state machines (concurrent check)**
  - Work by Reed, Redinbo, Kinney, Shen, Leveugle

## DISCRETE-TIME DYNAMIC SYSTEMS



State Evolution:  $q[t + 1] = \delta(q[t], x[t])$

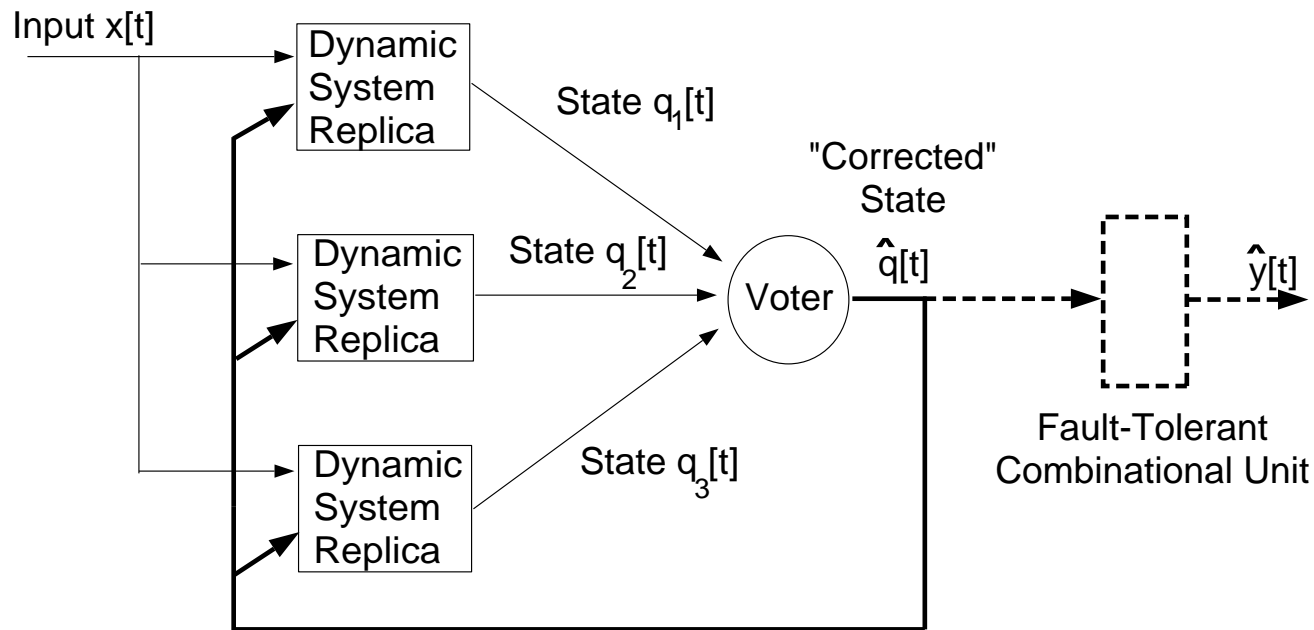
Output Equation:  $y[t] = \lambda(q[t], x[t])$

**Examples:** Digital filters, encoders/decoders, FSMs, algorithmic computations

**Transient fault in state transition mechanism:** Error propagates in future steps

**Transient fault in output mechanism:** Output corrupted at that particular step

## UNIVERSAL APPROACH: MODULAR REDUNDANCY (VON NEUMANN)

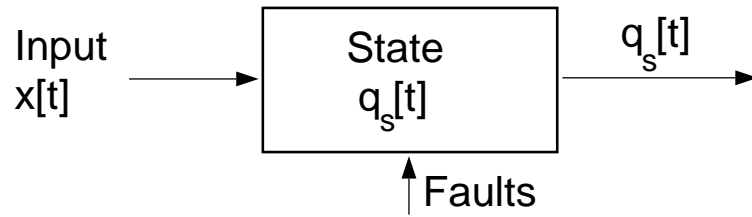


### Problems with modular redundancy

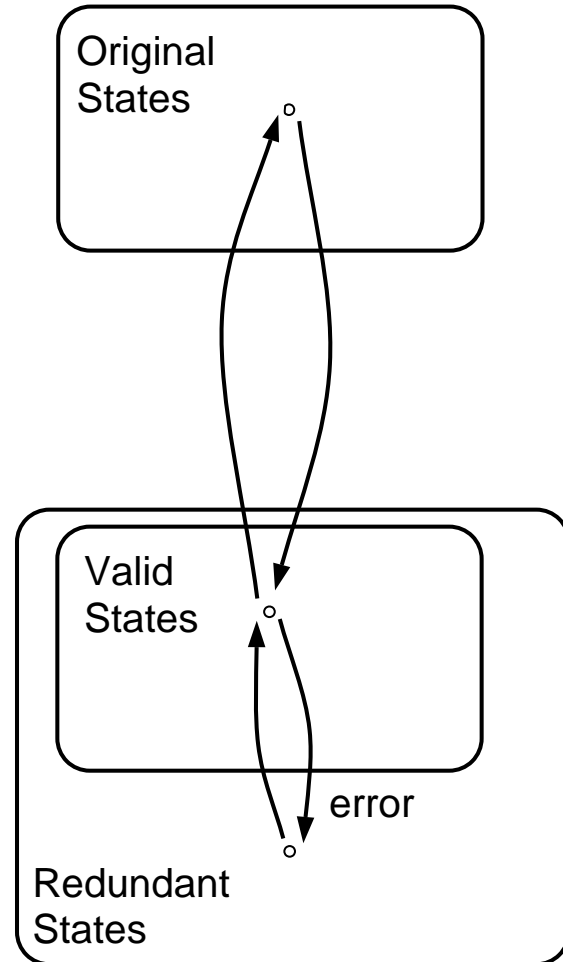
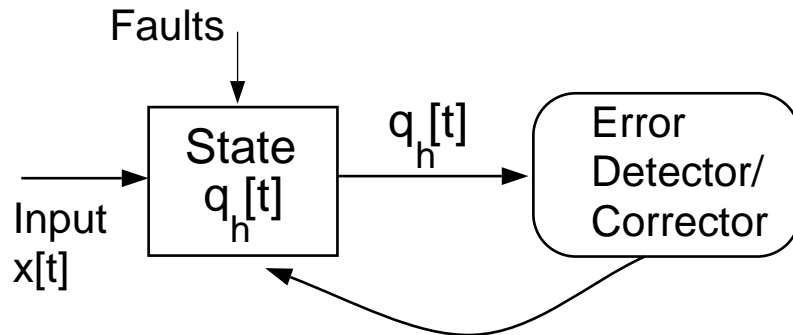
- Replication
- Checking overhead/slowdown
- Reliability of checking mechanism

} Our work address these issues!

# REDUNDANT IMPLEMENTATIONS



Replace with larger dynamic system:



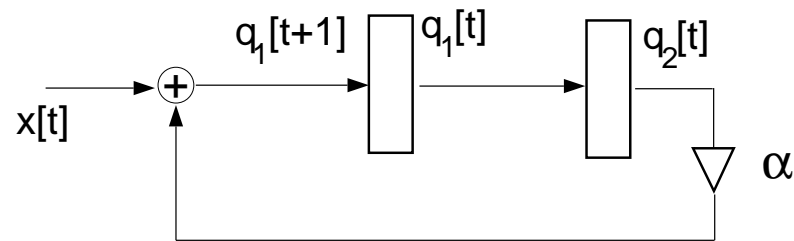
## LINEAR FINITE-STATE MACHINES (LFSMS)

**State evolution:**  $\mathbf{q}[t + 1] = \mathbf{A}\mathbf{q}[t] \oplus \mathbf{B}\mathbf{x}[t]$

- $\mathbf{A}$ ,  $\mathbf{b}$ ,  $\mathbf{q}[\cdot]$  and  $\mathbf{x}[\cdot]$  have entries in  $GF(q)$  ( $q = p^m$  for  $p$  prime,  $m \geq 1$ )
- Addition and multiplication in  $GF(q)$

**Examples:** Sequence enumerators, random number generators, encoders/decoders, linear feedback shift registers, linear cellular automata

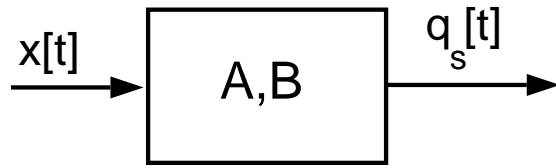
**One possible implementation:** Use adders, multipliers and memory elements



$$\mathbf{q}[t] \equiv \begin{bmatrix} q_1[t] \\ q_2[t] \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & \alpha \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

## FAULT DETECTION AND CORRECTION IN LFSMS

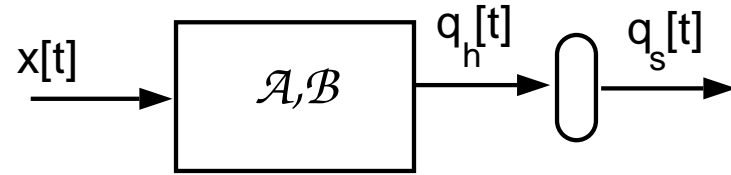
### Original LFSM



$$\mathbf{q}_s[t + 1] = \mathbf{A}\mathbf{q}_s[t] \oplus \mathbf{B}\mathbf{x}[t]$$

$\mathbf{q}_s$  is  $d$ -dimensional

### Redundant LFSM



$$\mathbf{q}_h[t + 1] = \mathcal{A}\mathbf{q}_h[t] \oplus \mathcal{B}\mathbf{x}[t]$$

$\mathbf{q}_h$  is  $\eta$ -dimensional,  $\eta = d + s, s > 0$

- **Concurrent simulation:**  $\mathbf{q}_s[t] = \mathbf{L}\mathbf{q}_h[t]$
  - **Encoding constraints:**  $\mathbf{q}_h[t] = \mathbf{G}\mathbf{q}_s[t]$
- } **Linear in GF(q) (not necessary)**
- **Fault detection:** If  $\mathbf{q}_h[t]$  is *not* in the column space of  $\mathbf{G}$ , or
- $$\mathbf{P}\mathbf{q}_h[t] \neq \mathbf{0}, \quad \mathbf{P}\mathbf{G} = \mathbf{0}$$

## CHARACTERIZATION OF REDUNDANT LFSM IMPLEMENTATIONS

<p><u>Original LFSM</u></p> $\mathbf{q}_s[t + 1] = \mathbf{A}\mathbf{q}_s[t] \oplus \mathbf{B}\mathbf{x}[t]$ <p><math>\mathbf{q}_s</math> is <math>d</math>-dimensional</p>	$\left. \begin{array}{l} \xrightarrow{\mathbf{q}_h[t]=\mathbf{G}\mathbf{q}_s[t]} \\ \xleftarrow{\mathbf{q}_s[t]=\mathbf{L}\mathbf{q}_h[t]} \end{array} \right\}$	<p><u>Redundant LFSM</u></p> $\mathbf{q}_h[t + 1] = \mathcal{A}\mathbf{q}_h[t] \oplus \mathcal{B}\mathbf{x}[t]$ <p><math>\mathbf{q}_h</math> is <math>\eta</math>-dimensional (<math>\eta = d + s</math>)</p>
---	--	---

**Standard redundant implementations (Hadjicostis & Verghese 2002):**

$(\mathcal{A}, \mathcal{B})$  is a redundant implementation for  $(\mathbf{A}, \mathbf{B})$  iff  $(\mathcal{A}, \mathcal{B})$  is similar to the following standard form:

$$\mathbf{q}_\sigma[t + 1] = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}}_{\mathcal{A}_\sigma} \mathbf{q}_\sigma[t] \oplus \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\mathcal{B}_\sigma} \mathbf{x}[t]$$

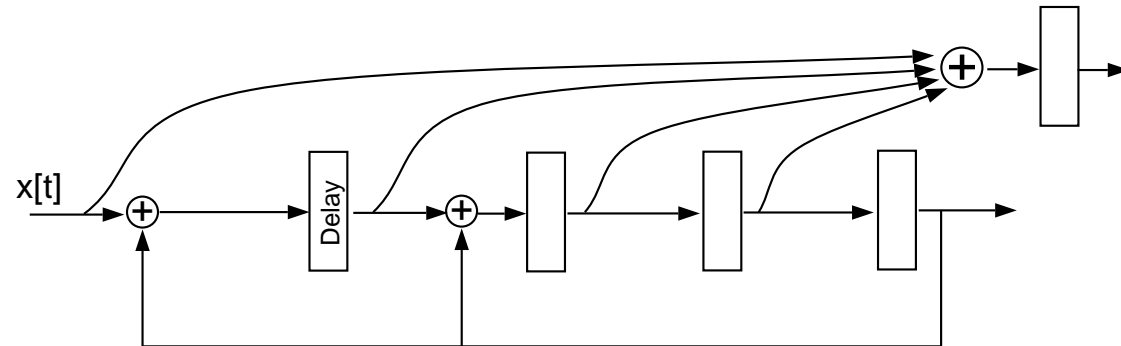
for some matrices  $\mathbf{A}_{12}, \mathbf{A}_{22}$

**Specifically:** Invertible  $\mathcal{T}$  such that  $\mathcal{A}_\sigma = \mathcal{T}^{-1}\mathcal{A}\mathcal{T}$ ,  $\mathcal{B}_\sigma = \mathcal{T}^{-1}\mathcal{B}$

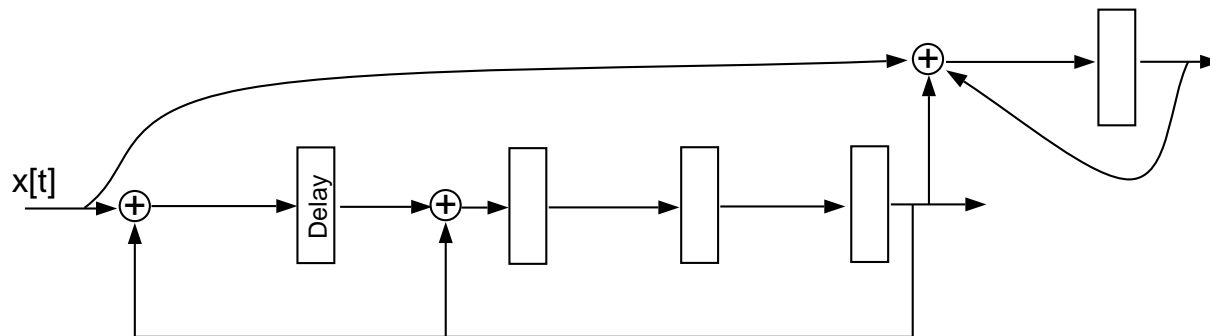
**Related concepts:** Inclusion principle (Šiljak), bisimulation (Pappas)

# DIFFERENT REDUNDANT IMPLEMENTATIONS FOR CHECKSUM SCHEME IN $GF(2)$

**Traditionally (Chatterjee, Abraham, Reed):**



**Using previous theorem:**



## CONCURRENT FAULT DETECTION AND IDENTIFICATION

**Fault model:** Single fault corrupts  $i$ th state variable

$$\mathbf{q}_f[t] = \underbrace{\mathbf{q}_h[t]}_{\text{fault-free}} \oplus v \mathbf{e}_i$$

**Justification:** Very general model (can be relaxed)

For instance, model is valid if a single fault results in a single bit (variable) error

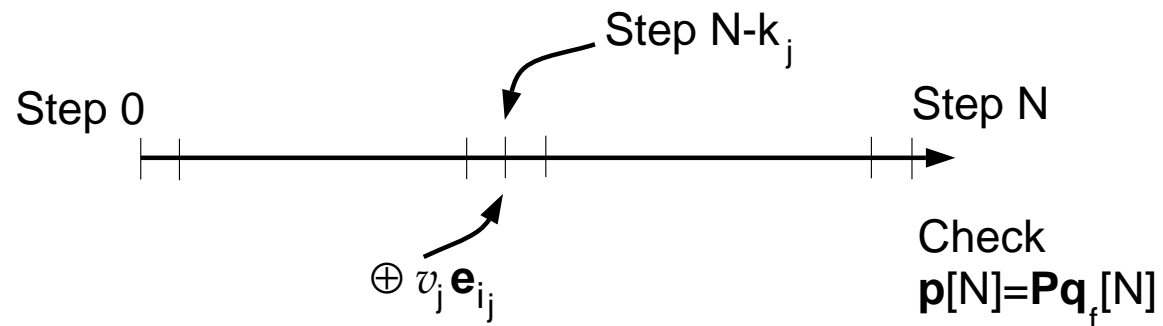
**Concurrent error detection (Abraham, Chatterjee, Hadjicostis, ...):**

At end of *each* time step, perform the *parity check*

$$\mathbf{p}[t] \equiv \mathbf{P} \mathbf{q}_f[t] = \mathbf{P} v \mathbf{e}_i \stackrel{?}{=} \mathbf{0}$$

**Error detection/correction capabilities:** Constraints on matrix  $\mathbf{P}$ , coding theory

## NON-CONCURRENT CHECKING (1)



**Goal:** Design redundant implementation so that knowledge of  $\mathbf{p}[N]$  allows detection and identification of error(s) in interval  $[0, N]$

**Motivation:** Relax checking requirements (e.g., periodic checking)

**Need:** For each fault ( $j$ ), identify

- Value ( $v_j$ )
  - State variable ( $\mathbf{e}_{i_j}$ )
  - Step ( $N - k_j$ )
- } For error correction, one may have to reset past states/outputs

## NON-CONCURRENT CHECKING (2)

**Error model:** Initially, fault  $j$  during step  $N - k_j - 1$  causes

$$\mathbf{q}_f[N - k_j] = \mathbf{q}_h[N - k_j] \oplus v_j \mathbf{e}_{i_j}$$

**Error propagation:** At step  $N$ ,

$$\mathbf{q}_f[N] = \mathbf{q}_h[N] \oplus \mathcal{A}^{k_j} v_j \mathbf{e}_{i_j}$$

**Parity check:** At step  $N$ ,

$$\mathbf{p}[N] = \mathbf{P} \mathbf{q}_f[N] = v_j \mathbf{P} \mathcal{A}^{k_j} \mathbf{e}_{i_j}$$

**Multiple errors result in:**

$$\mathbf{p}[N] = \sum_{j=1}^D v_j \mathbf{P} \mathcal{A}^{k_j} \mathbf{e}_{i_j}$$

**Task:** Ensure that  $D$  errors can be detected/identified

## SYNDROME GENERATION

**Observation:** Syndrome  $\mathbf{p}[N]$  is a linear combination of  $D$  columns of

$$\mathbf{S} = \left[ \mathbf{P} \quad \mathbf{PA} \quad \mathbf{PA}^2 \quad \dots \quad \mathbf{PA}^{N-1} \right]$$

**Lemma 1:** Detection of  $D$  errors *if and only if*  
all sets of  $D$  columns of  $\mathbf{S}$  are linearly independent  
 $\Rightarrow$  Need at least  $D$  additional variables ( $s \geq D$ )

**Lemma 2:** Identification of  $D$  errors *if and only if*  
all sets of  $2D$  columns of  $\mathbf{S}$  are linearly independent  
 $\Rightarrow$  Need at least  $2D$  additional variables ( $s \geq 2D$ )

**Theorem:** The syndrome matrix  $\mathbf{S}$  can be expressed as

$$\mathbf{S} = \left[ \mathbf{P} \quad \mathbf{A}_{22}\mathbf{P} \quad \mathbf{A}_{22}^2\mathbf{P} \quad \dots \quad \mathbf{A}_{22}^{N-1}\mathbf{P} \right]$$

## MAIN OBSERVATION EXPLOITED IN NON-CONCURRENT SCHEMES

**Vandermonde matrix:**  $\mathbf{V}(x_1, x_2, \dots, x_r) = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_r \\ x_1^2 & x_2^2 & \dots & x_r^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{2D-1} & x_2^{2D-1} & \dots & x_r^{2D-1} \end{bmatrix}$

**Fact:** Any  $2D$  columns of  $\mathbf{V}$  are linearly independent if parameters  $\{x_i\}$  are distinct

**Diagonal matrix:**  $\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & x & 0 & \dots & 0 \\ 0 & 0 & x^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & x^{2D-1} \end{bmatrix}$

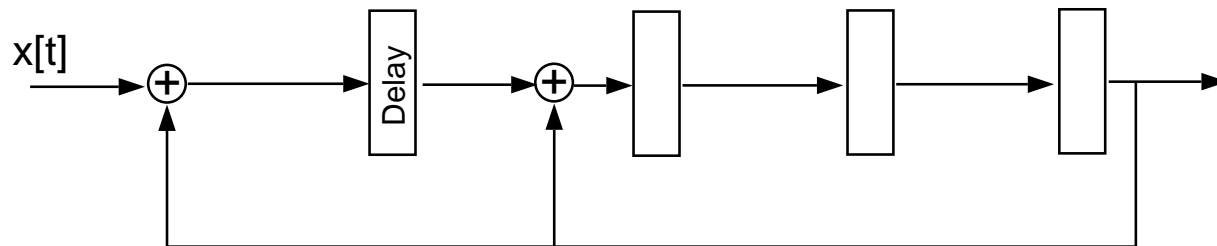
**Then:**  $\mathbf{\Lambda}^k \mathbf{V}(x_1, x_2, \dots, x_r) = \mathbf{V}(x_1 x^k, x_2 x^k, \dots, x_r x^k)$

**Conclusion:** Any  $2D$  columns of  $[\mathbf{V} \ \mathbf{\Lambda V} \ \dots \ \mathbf{\Lambda}^k \mathbf{V}] = \mathbf{V}(x_1, \dots, x_r, \dots, x_1 x^k, \dots, x_r x^k)$  are linearly independent if all parameters involved are distinct

## RESULTING CONSTRUCTION

- Redundant implementation uses minimal number of additional state variables ( $s = 2D$ )
- Allows non-concurrent
  - Identification of  $D$  errors
  - Detection of  $2D$  errors
- **Design requirement:**  $x_i x^k$  are unique,  $1 \leq i \leq \eta$ ,  $0 \leq k \leq N - 1$
- **Necessary condition:**  $GF(q)$  needs at least  $\eta N$  nonzero entries  $q - 1 \geq \eta N$
- **Related:** MDS convolutional codes (Rosenthal & York 1999)

## EXAMPLE: NON-CONCURRENT IDENTIFICATION OF TWO ERRORS (1)



State evolution (in  $\text{GF}(41)$ ) for original system:

$$\begin{aligned} \mathbf{q}_s[t+1] &= \mathbf{A}\mathbf{q}_s[t] + \mathbf{b}x[t] \\ &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{q}_s[t] + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} x[t] \end{aligned}$$

**Goal:** Detect and identify two errors  $\Rightarrow$  Use 4 additional variables

**Note:**  $N = 5, \eta = 8 \Rightarrow N\eta = 40$

## EXAMPLE: NON-CONCURRENT IDENTIFICATION OF TWO ERRORS (2)

**Primitive element in GF(41):**  $g = 7$  generates all nonzero elements

$$\{7, 7^2, 7^3, 7^4, 7^5, 7^6, 7^7, 7^8, 7^9, 7^{10}, \dots\} = \{7, 8, 15, 23, 38, 20, 17, 37, 13, 9, \dots\}$$

**Choose:**

$$\mathbf{M} = \mathbf{V}(38, 20, 17, 37) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 38 & 20 & 17 & 37 \\ 9 & 31 & 2 & 16 \\ 14 & 5 & 34 & 18 \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & x^2 & 0 \\ 0 & 0 & 0 & x^3 \end{bmatrix}, \quad x = 37$$

**Set:**

$$\mathbf{C} = -\mathbf{M}^{-1}\mathbf{V}(7, 8, 15, 23) = \begin{bmatrix} 37 & 20 & 29 & 28 \\ 38 & 23 & 12 & 34 \\ 7 & 21 & 25 & 21 \\ 40 & 17 & 15 & 39 \end{bmatrix}$$

$$\mathbf{A}_{22} = \mathbf{M}^{-1}\mathbf{\Lambda}\mathbf{M} = \begin{bmatrix} 21 & 16 & 36 & 18 \\ 7 & 18 & 40 & 7 \\ 23 & 18 & 12 & 37 \\ 32 & 31 & 36 & 21 \end{bmatrix}$$

## EXAMPLE: NON-CONCURRENT IDENTIFICATION OF TWO ERRORS (3)

**State evolution in  $GF(41)$  for redundant implementation:**

$$\mathbf{q}_h[t+1] = \left[ \begin{array}{cccc|cccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 24 & 25 & 9 & 21 & 16 & 36 & 18 \\ 37 & 16 & 27 & 26 & 7 & 18 & 40 & 7 \\ 38 & 33 & 5 & 29 & 23 & 18 & 12 & 37 \\ 7 & 9 & 25 & 17 & 32 & 31 & 36 & 21 \end{array} \right] \mathbf{q}_h[t] + \left[ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ \hline 37 \\ 38 \\ 7 \\ 40 \end{array} \right] x[t]$$

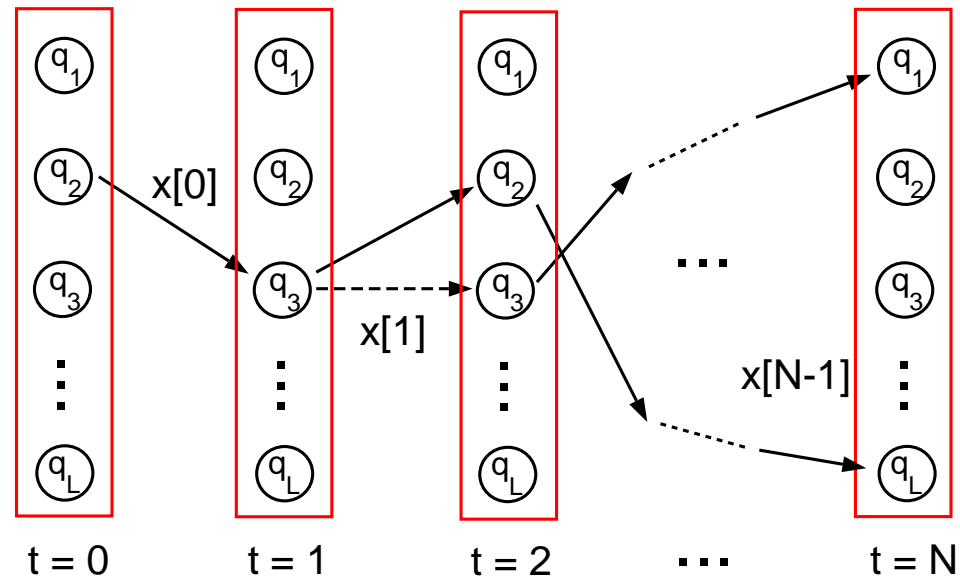
**Syndrome matrix:**

$$\mathbf{S} = \mathbf{M}^{-1} \underbrace{\left[ \mathbf{S}_0 \quad \mathbf{S}_1 \quad \mathbf{S}_2 \quad \mathbf{S}_3 \quad \mathbf{S}_4 \right]}_{\mathbf{Q}}$$

where  $\mathbf{S}_k = \mathbf{V}(7x^k, 8x^k, 15x^k, 23x^k, 38x^k, 20x^k, 17x^k, 37x^k)$ ,  $x = 37$

**Efficient decoding:** Slight modification allows the use of PGZ algorithm

## NON-CONCURRENT PROTECTION OF FSMs



**State-transition fault:** Transient fault during  $t = 1$  affects state evolution

### Objectives:

- Capture bounded number of state-transition faults
- Minimize overhead in redundant implementation
- Minimize overhead in error detecting/correcting stage

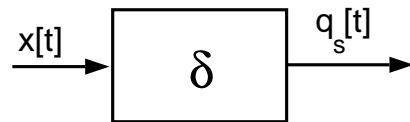
## FINITE-STATE MACHINE NOTATION



- **State Set:**  $Q_s = \{q_1, q_2, \dots, q_L\}$       **Input Set:**  $X = \{x_1, x_2, \dots, x_K\}$
- **States viewed as vectors:**  $Q_s = \{\mathbf{q}_s^{(1)}, \mathbf{q}_s^{(2)}, \dots, \mathbf{q}_s^{(L)}\}$ 
  - E.g.,  $b$ -dimensional vectors in  $GF(2)$  (where  $b \geq \lceil \log_2 L \rceil$  so that  $2^b \geq L$ )
  - Will use:  $d$ -dimensional vectors in  $GF(q)$  (where  $q^d \geq L$ )
- **Next-State Function:**  $\delta_x(\mathbf{q}_s^{(j)}) = \delta(\mathbf{q}_s^{(j)}, x)$ , defined for each  $x \in X$

## REDUNDANT IMPLEMENTATIONS OF FSMs

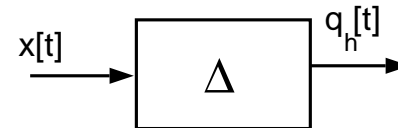
### Original FSM



$$\mathbf{q}_s[t + 1] = \delta(\mathbf{q}_s[t], x[t])$$

$\mathbf{q}_s[t]$  is  $d$ -dimensional in  $GF(q)$

### Redundant FSM



$$\mathbf{q}_h[t + 1] = \Delta(\mathbf{q}_h[t], x[t])$$

$\mathbf{q}_h[t]$  is  $\eta$ -dimensional in  $GF(q)$ ,  $\eta = d + s$

- **Concurrent simulation:**  $\mathbf{q}_s[t] = \mathbf{L}\mathbf{q}_h[t]$ ,  $\mathbf{L} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \end{bmatrix}$
  - **Encoding constraints:**  $\mathbf{q}_h[t] = \mathbf{G}\mathbf{q}_s[t]$ ,  $\mathbf{G} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{C} \end{bmatrix}$
- } **Linear constraints**
- **Concurrent fault detection:** Verify that  $\mathbf{p}[t] \equiv \mathbf{P}\mathbf{q}_h[t] = \mathbf{0}$ ,  $\mathbf{P} = \begin{bmatrix} -\mathbf{C} & \mathbf{I}_s \end{bmatrix}$

**Task:** Appropriate  $\Delta_x$  for  $x \in X$  such that  $\mathbf{G}\delta_x(\mathbf{q}_s^{(j)}) = \Delta_x(\mathbf{G}\mathbf{q}_s^{(j)})$  for all  $\mathbf{q}_s^{(j)} \in \mathcal{Q}_s$

## SIMPLE EXAMPLE OF REDUNDANT FSM IMPLEMENTATION (1)

**Notation:**  $\mathbf{q}_h[t] = \begin{bmatrix} \mathbf{q}_{hs}[t] \\ \mathbf{q}_{hr}[t] \end{bmatrix}$  ( $\mathbf{q}_{hs}[t]$  is  $d$ -dimensional,  $\mathbf{q}_{hr}[t]$  is  $s$ -dimensional)

**Under normal conditions:** (i)  $\mathbf{q}_{hs}[t] = \mathbf{q}_s[t]$  (original state)

(ii)  $\mathbf{q}_{hr}[t] = \mathbf{C}\mathbf{q}_s[t]$  (constraints)

**Possible next-state transition mapping:** For each input  $x \in X$

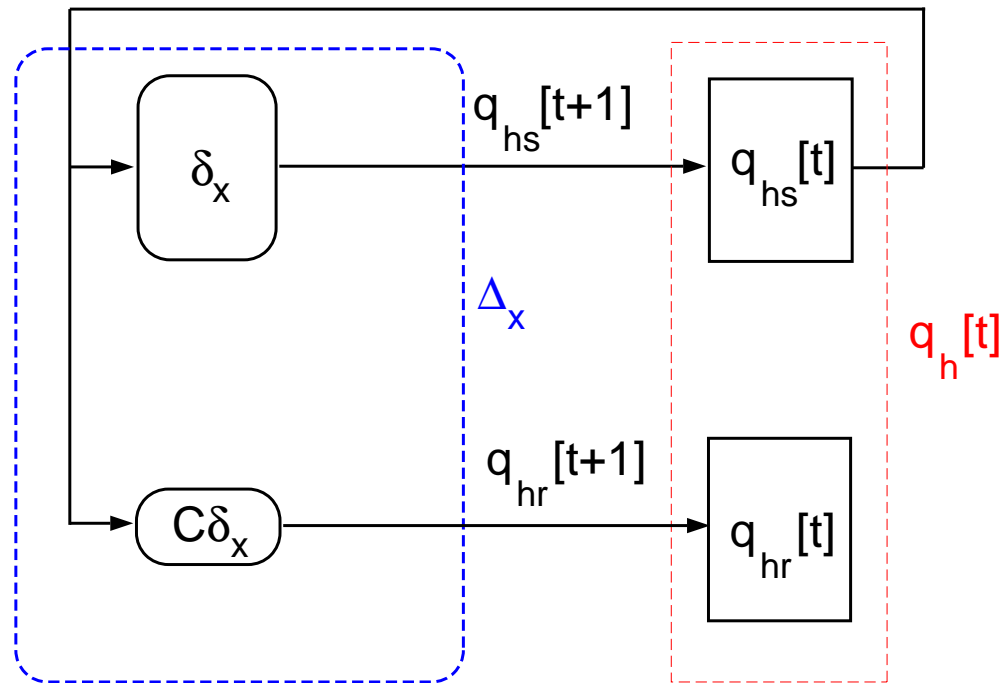
$$\Delta_x(\mathbf{q}_h[t]) = \begin{bmatrix} \delta_x(\mathbf{q}_{hs}[t]) \\ \mathbf{C}\delta_x(\mathbf{q}_{hs}[t]) \end{bmatrix}$$

**Verify:** Under fault-free conditions (assuming appropriate initialization)

$$\mathbf{q}_h[t] = \mathbf{G}\mathbf{q}_s[t] = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{C} \end{bmatrix} \mathbf{q}_s[t] \quad \Rightarrow \quad \mathbf{q}_h[t+1] = \mathbf{G}\mathbf{q}_s[t+1]$$

## SIMPLE EXAMPLE OF REDUNDANT FSM IMPLEMENTATION (2)

**Mapping  $\Delta_x$  (defined for each input  $x \in X$ ):**



**“Memoryless” construction:**  $q_{hr}[t]$  not used in next-state generation

## MORE GENERAL CLASS OF REDUNDANT FSM IMPLEMENTATIONS

**Same notation:**  $\mathbf{q}_h[t] = \begin{bmatrix} \mathbf{q}_{hs}[t] \\ \mathbf{q}_{hr}[t] \end{bmatrix}$

**Next-state transition mapping:** For each input  $x \in X$

$$\Delta_x(\mathbf{q}_h[t]) = \begin{bmatrix} \delta_x(\mathbf{q}_{hs}[t]) \ominus \mathbf{A}_{12_x} \mathbf{C} \mathbf{q}_{hs}[t] \oplus \mathbf{A}_{12_x} \mathbf{q}_{hr}[t] \\ \mathbf{C} \delta_x(\mathbf{q}_{hs}[t]) \ominus (\mathbf{C} \mathbf{A}_{12_x} \mathbf{C} \oplus \mathbf{A}_{22_x} \mathbf{C}) \mathbf{q}_{hs}[t] \oplus (\mathbf{C} \mathbf{A}_{12_x} \oplus \mathbf{A}_{22_x}) \mathbf{q}_{hr}[t] \end{bmatrix}$$

**Verify:** Under fault-free conditions (assuming appropriate initialization)

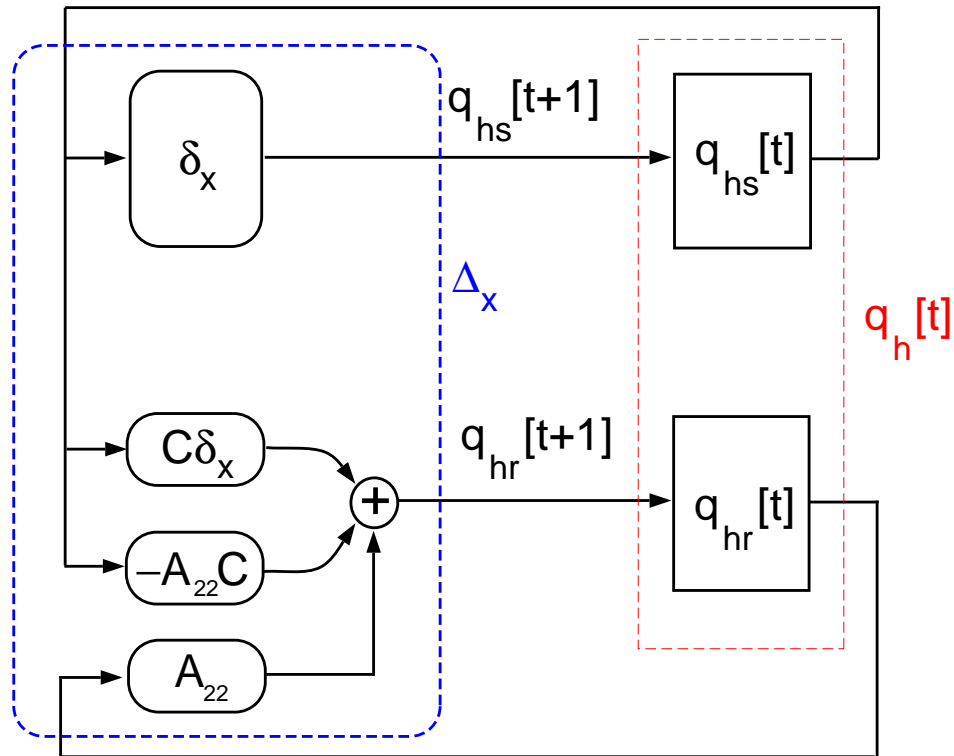
$$\mathbf{q}_h[t] = \mathbf{G} \mathbf{q}_s[t] = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{C} \end{bmatrix} \mathbf{q}_s[t] \quad \Rightarrow \quad \mathbf{q}_h[t+1] = \mathbf{G} \mathbf{q}_s[t+1]$$

## BLOCK DIAGRAM DESCRIPTION OF REDUNDANT FSM IMPLEMENTATION

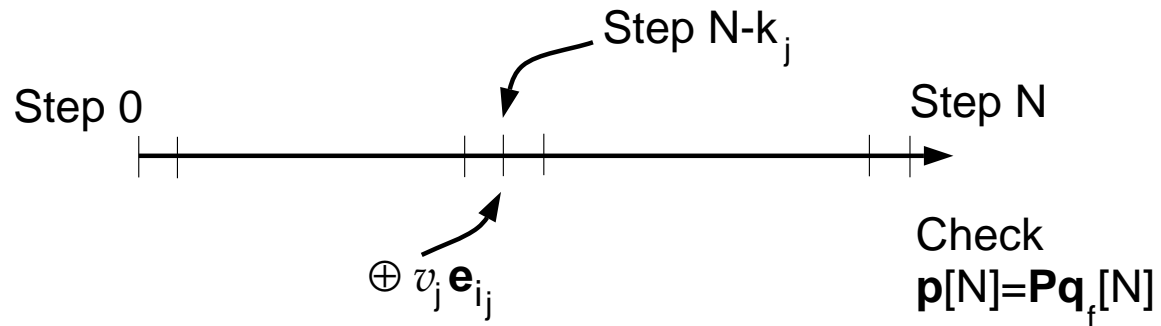
**Simplifications:** (i)  $A_{12_x} = \mathbf{0}$  for all  $x \in X$

(ii)  $A_{22_x} = A_{22}$  for all  $x \in X$

**Mapping  $\Delta_x$  (defined for each input  $x \in X$ ):**



## NON-CONCURRENT CHECKING IN FSMs (1)



**Goal:** Design redundant FSM implementation so that knowledge of  $p[N]$  allows us to detect and identify error(s) in interval  $[0, N]$

**Need:** For each fault ( $j$ ), identify

- Value ( $v_j$ )
  - State variable ( $e_{i_j}$ )
  - Step ( $N - k_j$ )
- } Error correction involves resetting past states/outputs

## NON-CONCURRENT CHECKING IN FSMs (2)

**Error model:** Fault  $j$  corrupts the  $i_j$ th state variable by  $v_j$  during step  $N - k_j$

$$\mathbf{q}_f[N - k_j] = \underbrace{\mathbf{q}_h[N - k_j]}_{\text{fault-free state}} \oplus v_j \mathbf{e}_{i_j}$$

**Error propagation:** At step  $N$ ,  $\mathbf{q}_f[N] = \mathbf{q}_h[N] \oplus \mathbf{e}$

**Nonlinear machine:** Error  $\mathbf{e}$  hard to characterize

**Theorem:** Syndrome  $\mathbf{p}[N] = v_j \mathbf{A}_{22}^{k_j} \mathbf{P} \mathbf{e}_{i_j}$

**Generalizes to:**

$$\text{Syndrome } \mathbf{p}[N] = \sum_{j=1}^D v_j \mathbf{A}_{22}^{k_j} \mathbf{P} \mathbf{e}_{i_j}$$

**Again:** Syndrome  $\mathbf{p}[N]$  is a linear combination of columns of

$$\mathbf{S} = \left[ \mathbf{P} \quad \mathbf{A}_{22} \mathbf{P} \quad \mathbf{A}_{22}^2 \mathbf{P} \quad \cdots \quad \mathbf{A}_{22}^{N-1} \mathbf{P} \right]$$

## RESULTING REDUNDANT FSM IMPLEMENTATIONS AND FUTURE WORK

- **Encoded embeddings of FSMs:**

- Reflection of hardware faults through error models
- Generalization of modular redundancy and checksum schemes
- Non-concurrent, periodic checking

- **Related future work:**

- Rollback-based correction, resource-efficiency issues
- Extensions to verification, branch prediction
- Extensions to distributed/decentralized settings
- Other applicable choices of coding constraints and redundant dynamics
- Flexibility in choosing  $A_{12}$ , or input-varying  $A_{12_x}$  and/or  $A_{22_x}$

## FUTURE WORK

Overall objective: Understand and manage complex dynamic systems

- **Applications: monitoring, testing, verification, control**
- Distributed embedded systems
- Real-time systems
- Layered control architectures

⇒ **Applicability of error detection/correction techniques**

⇒ **Implications to hardware redundancy, architectural constraints**