Error-Correction & Crosstalk Avoidance in DSM Busses

Ketan Patel and Igor Markov

University of Michigan

Electrical Engineering & Computer Science

Outline

- Motivation
- Previous work
- Graph-based model & optimal codes
- Boundary shift codes
- Future work

Motivation

DSM busses increasingly susceptible to noise

- Crosstalk
- Radiation effects
- Power grid fluctuations

Goal: Avoid crosstalk & provide error-correction

Crosstalk Noise

Most detrimental switching pattern

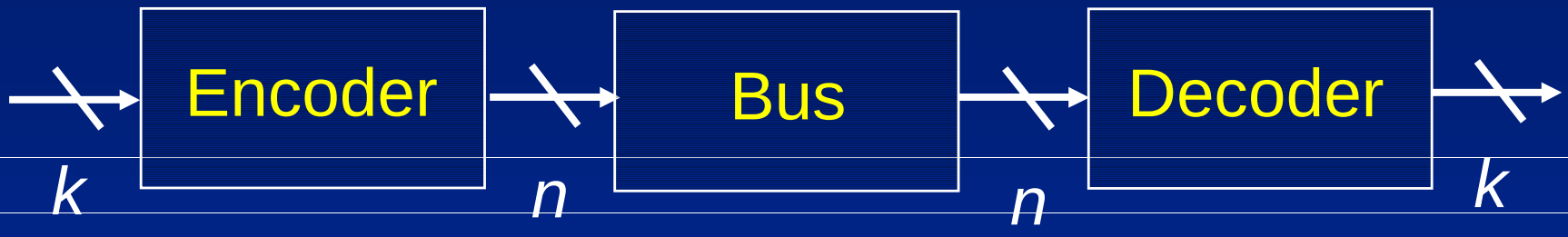
e.g.

Bus value (t=0) ⇒ 0 1 1 0 1 0 0 0

Bus value (t=1) ⇒ 1 1 0 1 1 0 1 1

We call this an invalid transition

Bus Encoding



- *k* bits encoded on *n* wires

  k = Rate

- Encoding disallows invalid transitions

  self-shielding code

Memory vs. Memoryless

Memoryless codes

- Encoding determined only by current bits
- Fixed codebook

Codes with memory

- Encoding may depend on previous codewords
- Dynamic codebook

Graph-based Model (Memoryless)

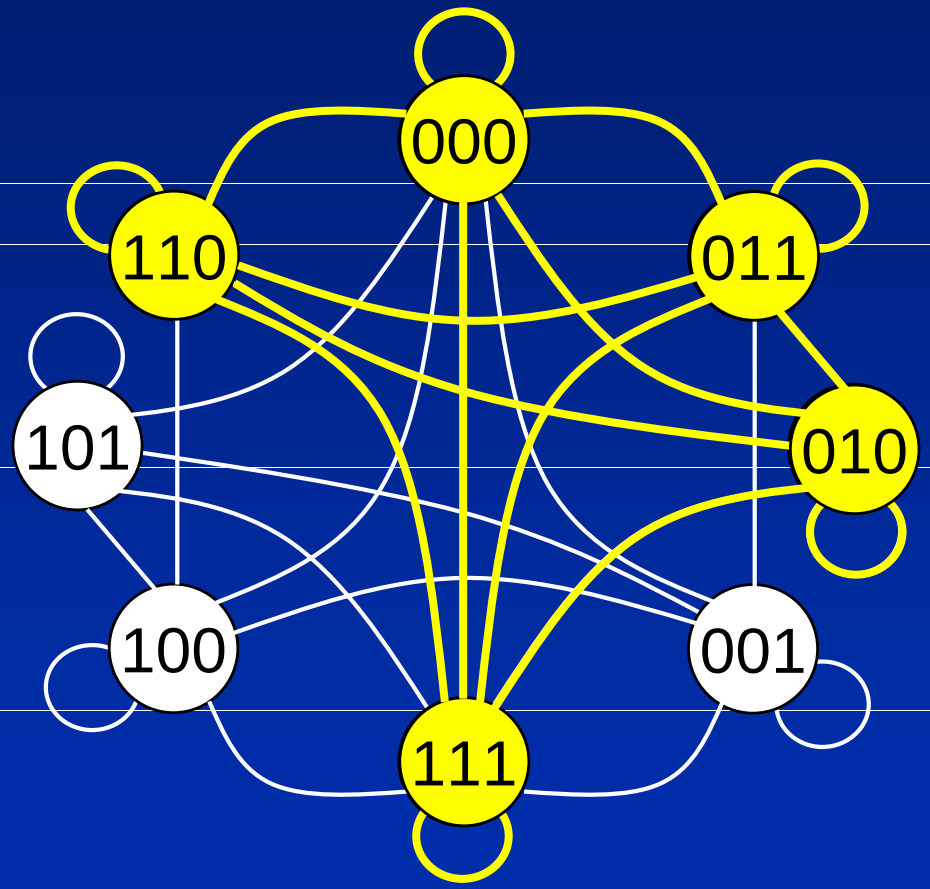
Vertices: Bus Values

Edges: Valid Transitions

optimal memoryless code ⟺ Max Clique

00 ⇨ 000
01 ⇨ 011
10 ⇨ 110
11 ⇨ 010
-- ⇨ 111

3-bit bus example



Error-Correction

Can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

$\Leftrightarrow$ **distance** $\geq$ d

e.g., to correct 1 error $\Leftrightarrow$ distance ≥ 3

Place edges if

- Valid transition $c_1 \Leftrightarrow c_2$ is valid
- Distance large enough $d(c_1,c_2) \geq d$

Codes with Memory

Two graphs:

$G_1 \Leftrightarrow$ crosstalk constraints

$G_2 \Leftrightarrow$ error-correction constraints

For rate $\log_2 M$ code

- Vertices connected to $\geq M$ vertices in G_1 forming clique in G_2

Can find optimal code using pruning algorithm

Optimal Codes

Drawbacks

- Algorithm becomes infeasible for large busses
- No practical encoder/decoder

Need codes that have:

- Scalable design
- Practical encoder/decoder

Dependent Boundaries

Dependent boundary: boundary with transition

0|1 1|0|1|0 0 0

Positions {1, 3, 4, 5}

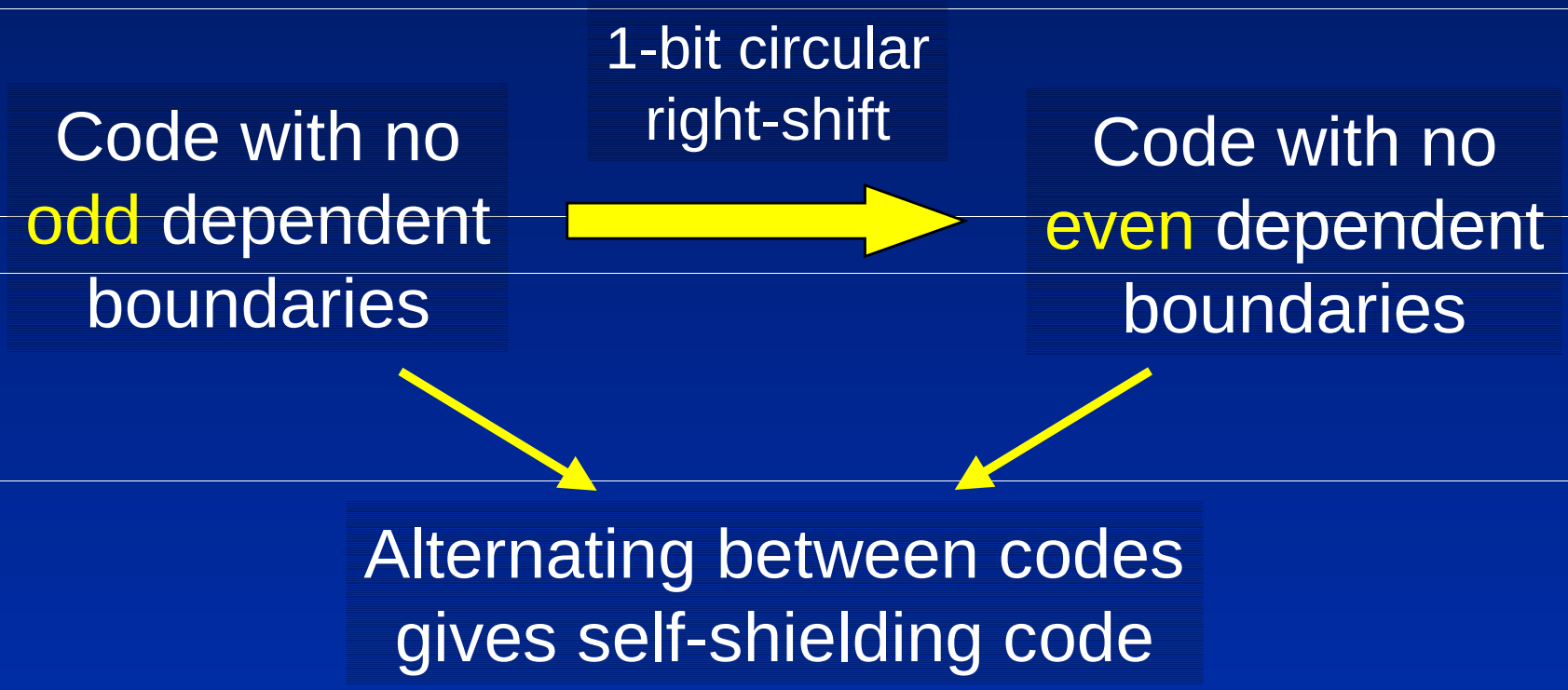
No overlapping dependent boundaries → No invalid transition

e.g.,

c_1 → 0|1 1|0 0|1 1 1 {1, 3, 5}

c_2 → 1 1|0 0|1 1 1|0 {2, 4, 7}

Boundary Shift Codes



Distance properties of original code preserved

General Construction

- Start with error-correcting code
- Duplicate all bits (no odd dependent boundaries)
- Possibly “puncture” last bit position
- ⟹ Code 1
- 1-bit circular right-shift ⟹ Code 2

Code 1 has no odd dependent boundaries

Code 2 has no even dependent boundaries

Single Error-Correcting Code

Use parity check code

e.g. [5,4,2] parity check code

$$x_1 \; x_2 \; x_3 \; x_4 \; \underbrace{(x_1 \oplus x_2 \oplus x_3 \oplus x_4)}$$

$$x_1 \; x_2 \; x_3 \; x_4 \; x_5$$

Duplicate bits and puncture

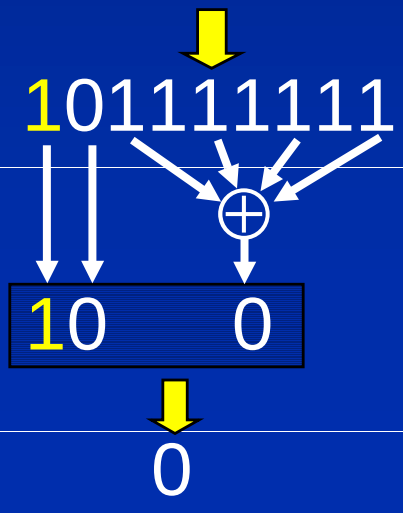
$$x_1 \; x_1 \; x_2 \; x_2 \; x_3 \; x_3 \; x_4 \; x_4 \; x_5 \; \cancel{x_5}$$

This is [9,4,3] single error-correcting code

Boundary Shift Code (Example)

Time	Input	Encoded Output
0	1010	110011000
1	0111	001111111
2	1000	110000001

Time	Received
1	110111111



1-bit left-shift

Decode by majority vote

Code Rates

Wires	Optimal Memoryless	Optimal	Boundary Shift Code
3	1	1	1
4	1	1	—
5	1.59	2	2
6	2.32	2.32	—
7	2.58	3.17	3
8	3.17	3.59	—
9	3.81	4.25	4

Single-error correcting self-shielding codes

Advantages/Drawbacks

Advantages

- Error-correction & crosstalk prevention
- Scalable construction
- Systematic (unencoded wires)

Drawbacks

- Encoding/decoding logic overhead
- Wire overhead
- Errors may cause invalid transitions

Future Work

- Generalize for more accurate fault models
- Evaluate using realistic simulations
- Generalize to include other constraint (e.g. power)