Model Checking the FlexRay Physical Layer Protocol

Bernd Finkbeiner, Peter Faymonville, Michael Gerke

Reactive Systems Group
Saarland University
Germany

22.12.2011

If you hit the brakes ...

... and they don't work:

Brakes should work!

Drive-by-Wire

FlexRay

Drive-by-Wire

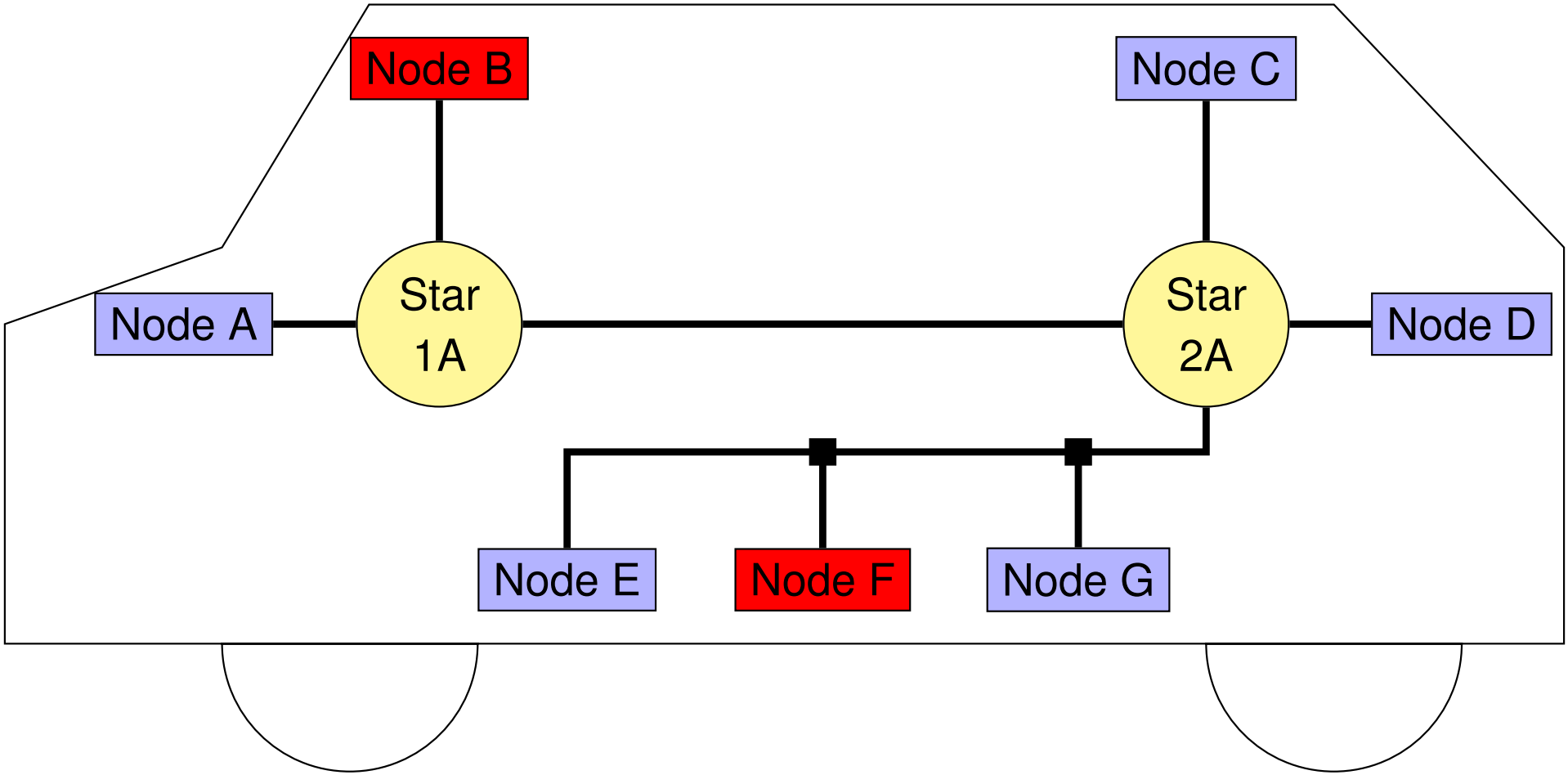
FlexRay

FlexRay Bus Protocol

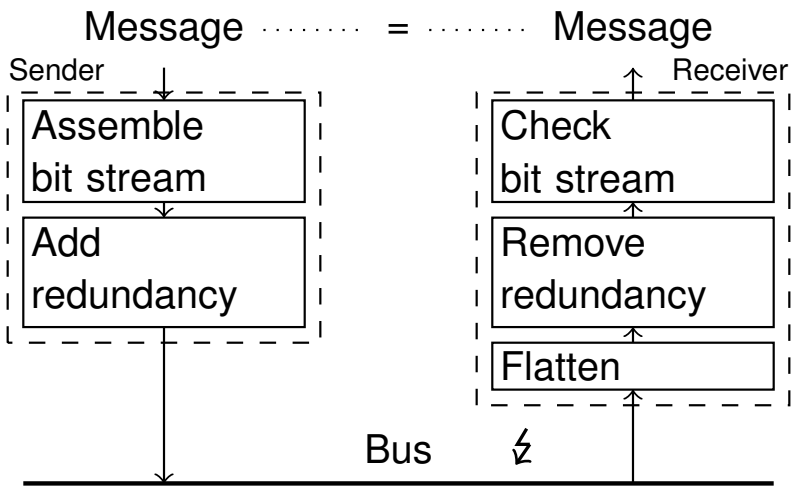
FlexRay

- communication protocol for distributed components in cars
- used in BMW X 5 and BMW's 7 series for X-by-wire
- developed by: BMW, Bosch, Daimler, Freescale, General Motors, NXP Semiconductors, Volkswagen, et al.

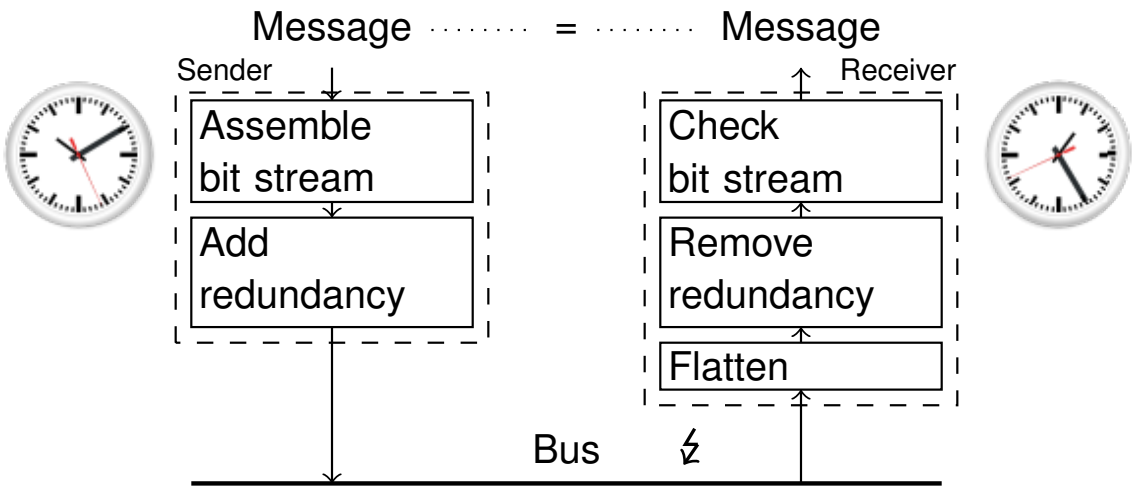
FlexRay Network



Case Study: FlexRay Physical Layer Protocol

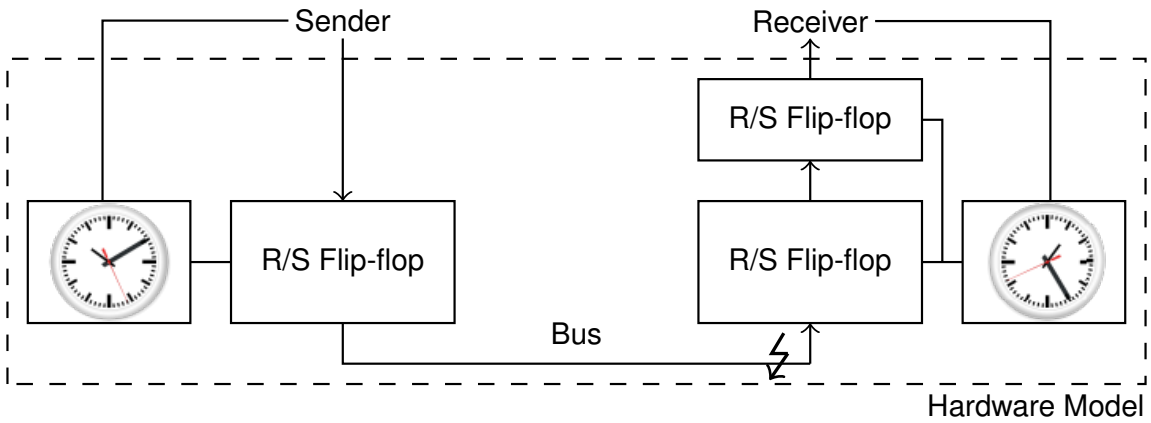


Case Study: FlexRay Physical Layer Protocol



The only (but important) source of time in the model: the two oscillators

Error Sources in Communication Scenario



Sources of error:

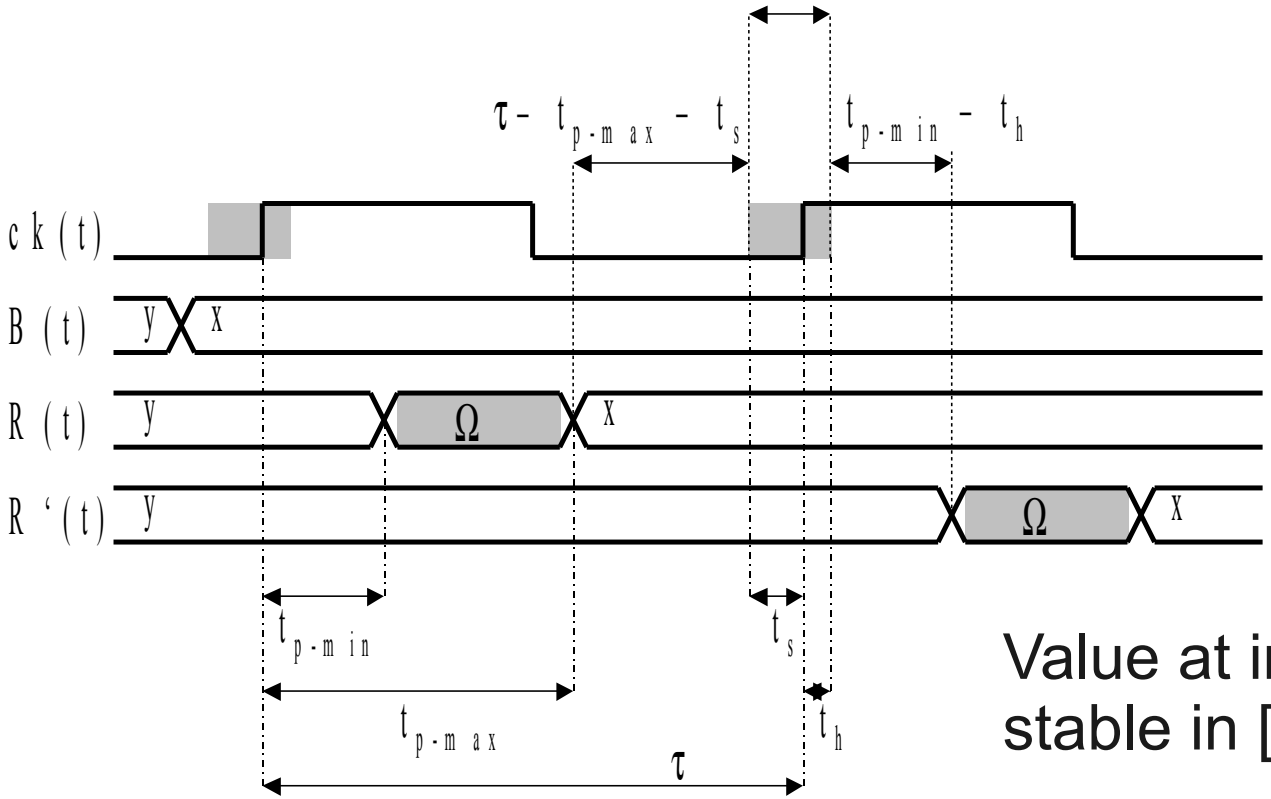
jitter:

- clock drift
- variance in delay
- unstable value in sampling interval (setup/hold)

glitches:

- bits flipped on the bus

Register Semantics



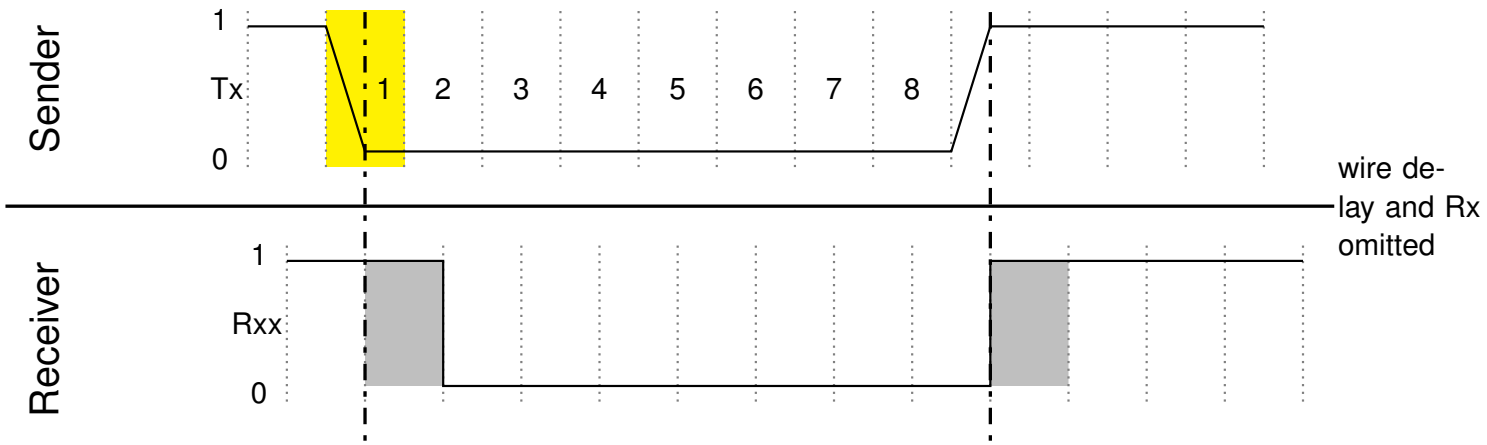
Value at input has to be stable in [ck-t_s,ck+t_h]

τ =cycle time

$t_{p\text{-}min/max}$ =delay of voltage change and signal propagation

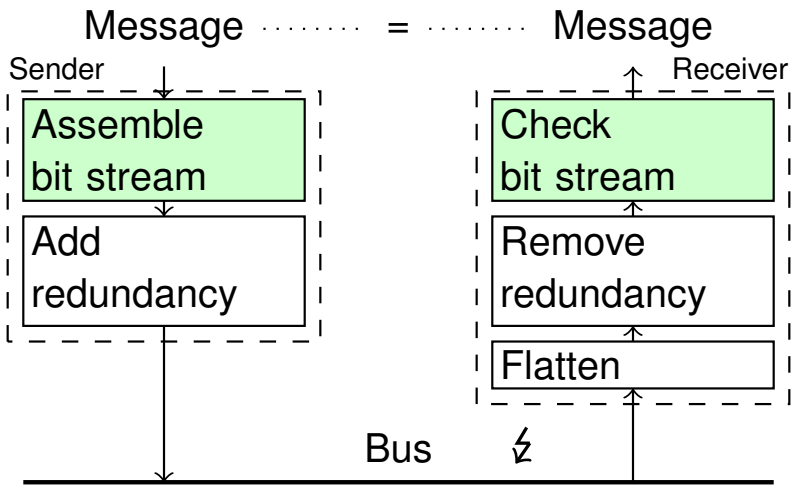
$t_{s/h}$ =hold and setup times of the register

Jitter Effects

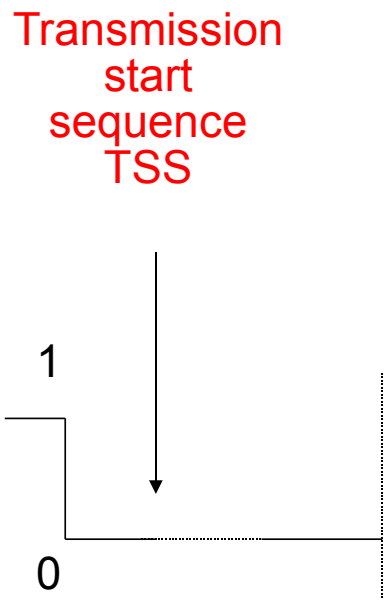


- clock drift $\rightarrow$ asynchronous communication
- signal changes are not instantaneous
- unstable value during setup / hold $\rightarrow$ error

Protocol Architecture

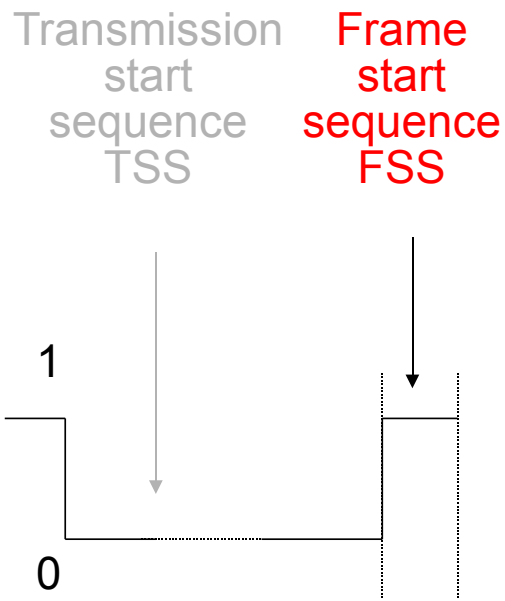


Encoding (ENC)
Frames



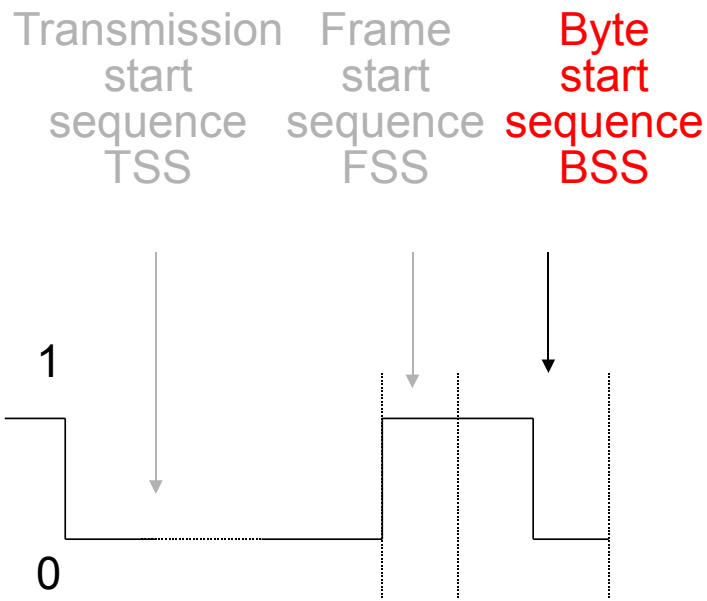
Frame coding in static segment

Encoding (ENC)
Frames



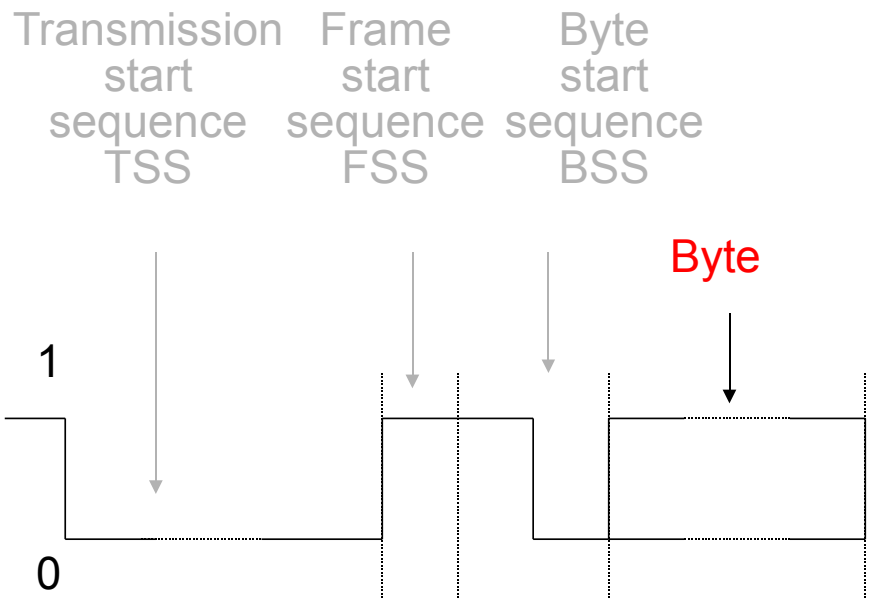
Frame coding in static segment

Encoding (ENC)
Frames



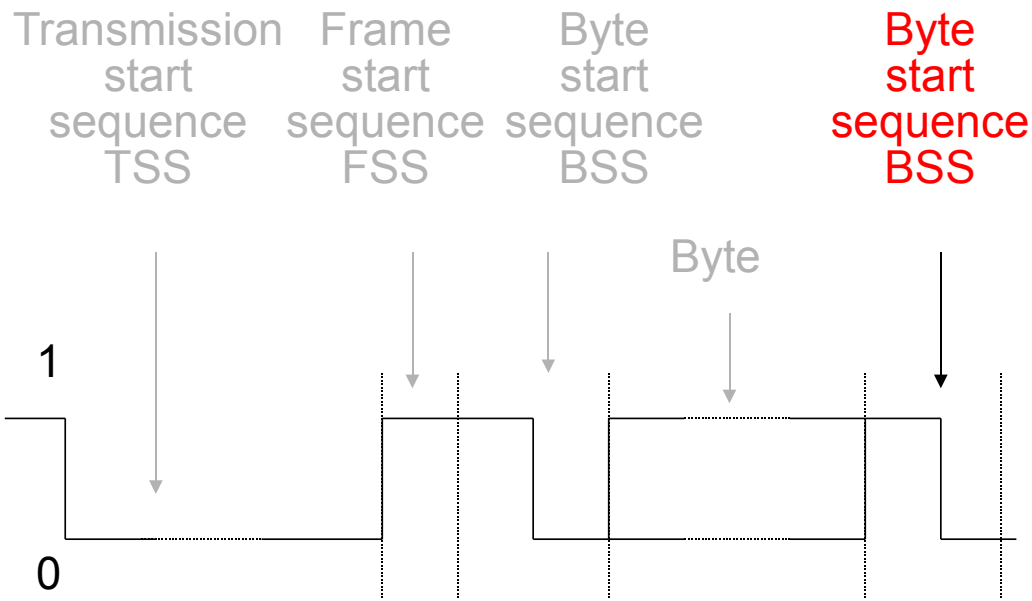
Frame coding in static segment

Encoding (ENC)
Frames



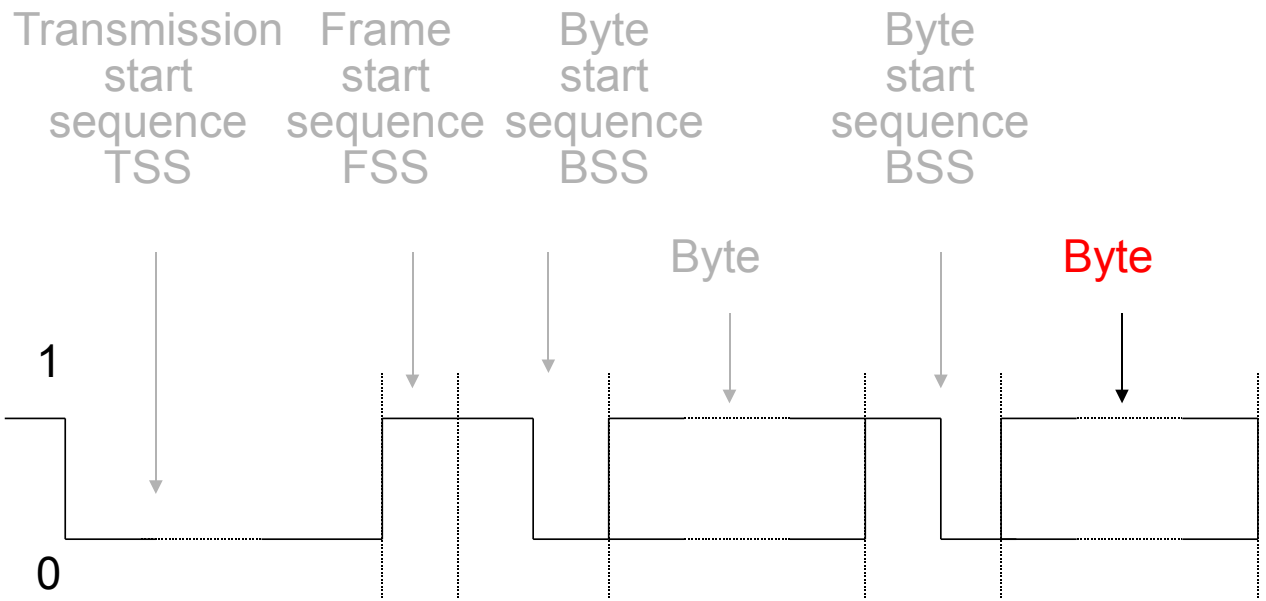
Frame coding in static segment

Encoding (ENC)
Frames



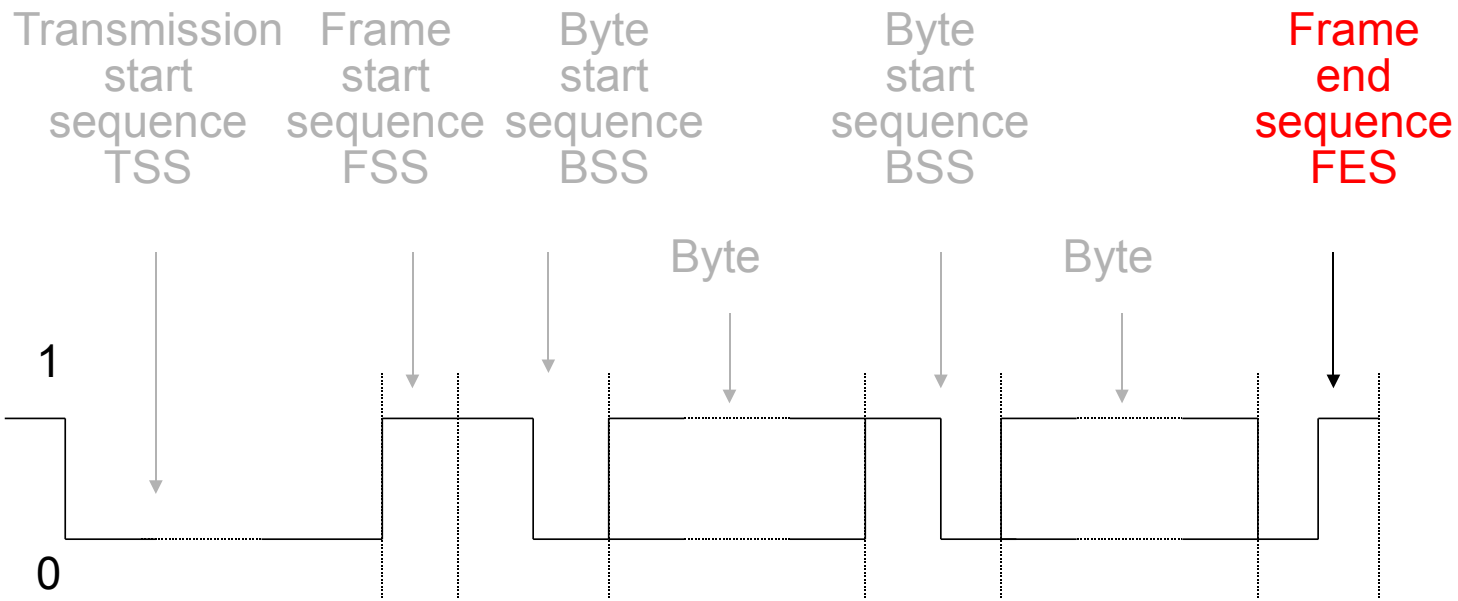
Frame coding in static segment

Encoding (ENC)
Frames



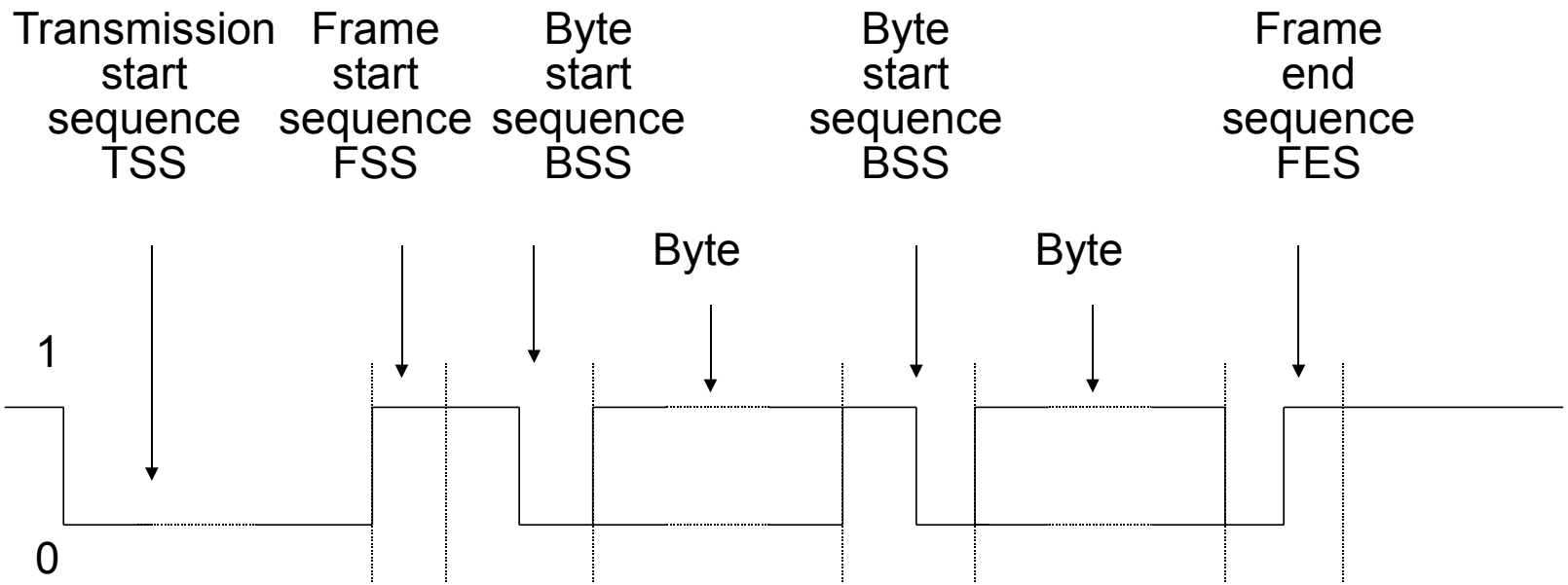
Frame coding in static segment

Encoding (ENC)
Frames



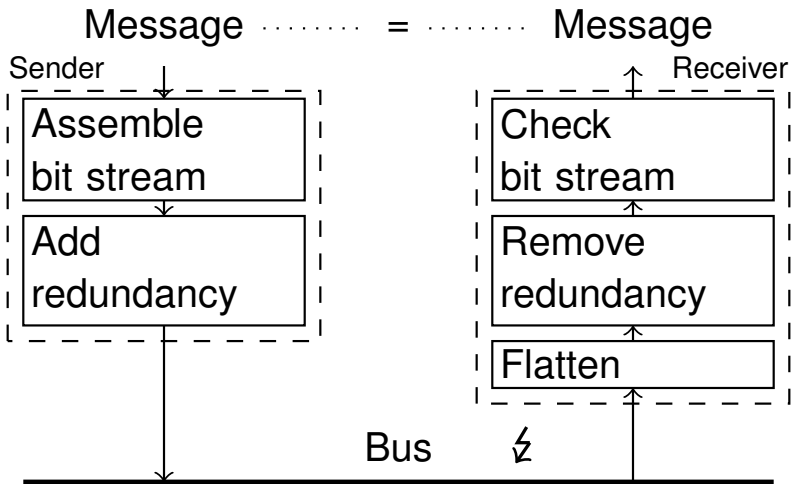
Frame coding in static segment

Encoding (ENC)
Frames



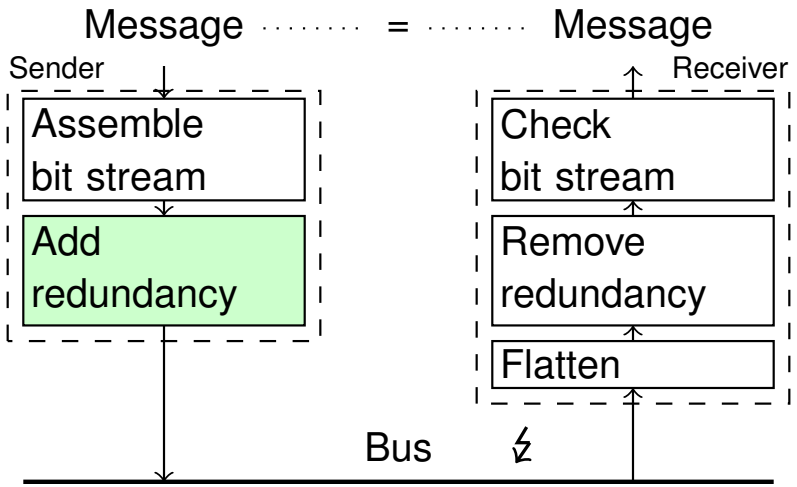
Frame coding in static segment

Protocol Architecture



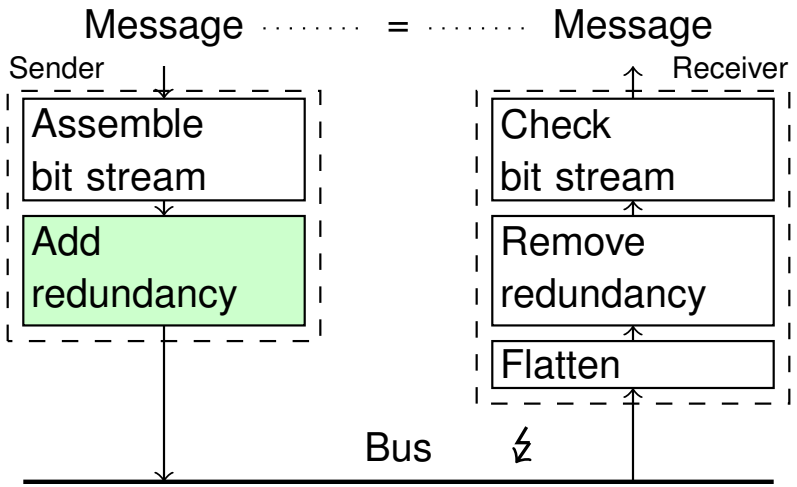
- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



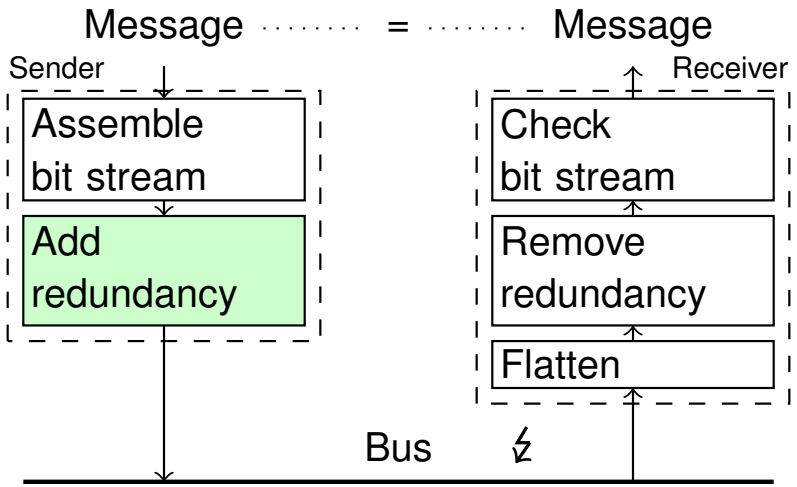
- Add redundancy: send each bit for 8 cycles (bit cell)

  Stream: 1 =Bus: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- Flatten: take majority of last 5 samples (voting window size 5)

- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



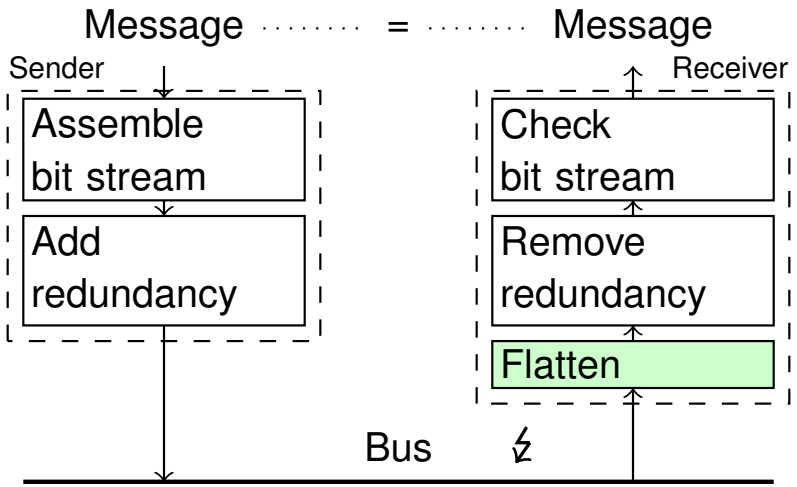
- Add redundancy: send each bit for 8 cycles (bit cell)

  Stream: 10 =Bus:

1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

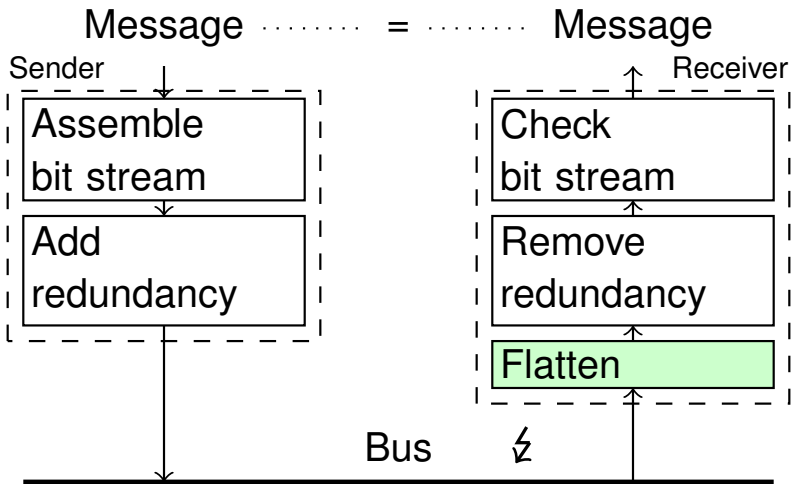
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

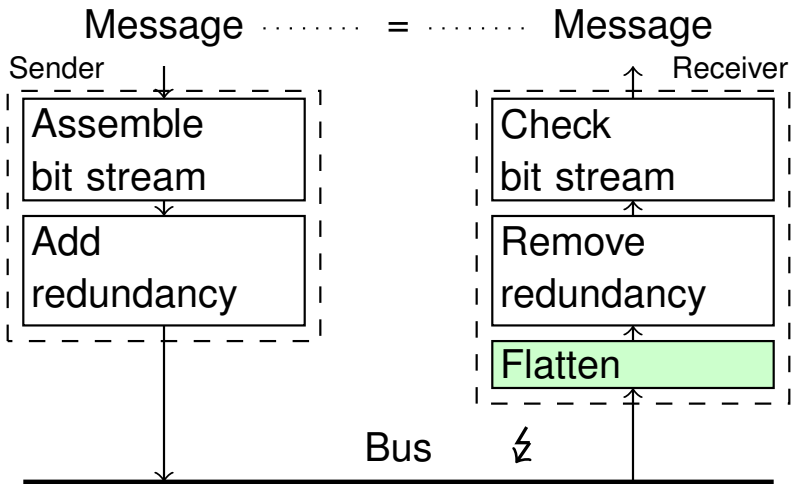
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | $= 1$
- Remove redundancy: pick 5^{th} voted value from each bit cell

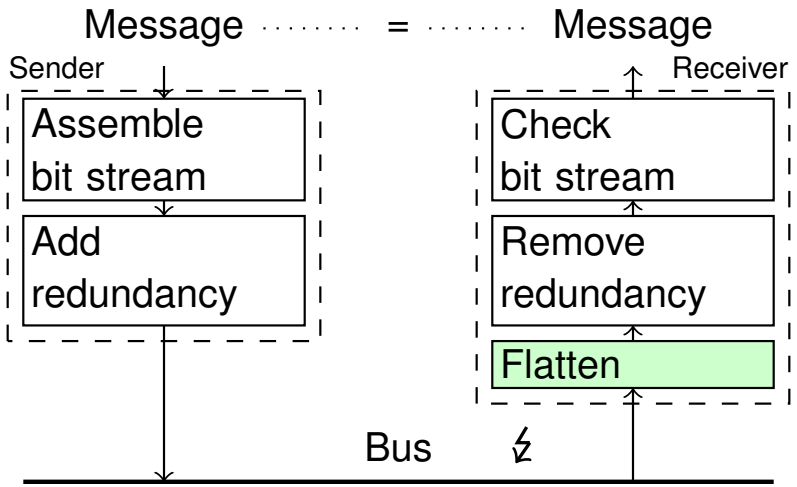
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | $= 1$
- Remove redundancy: pick 5^{th} voted value from each bit cell

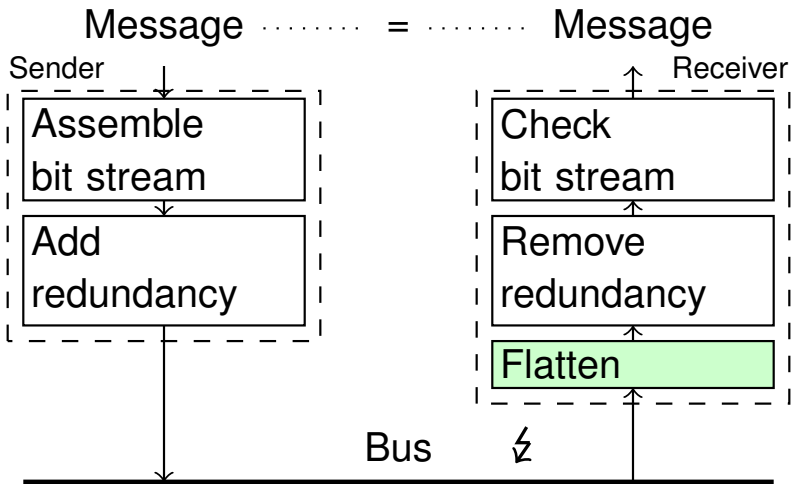
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | $= 1$
- Remove redundancy: pick 5^{th} voted value from each bit cell

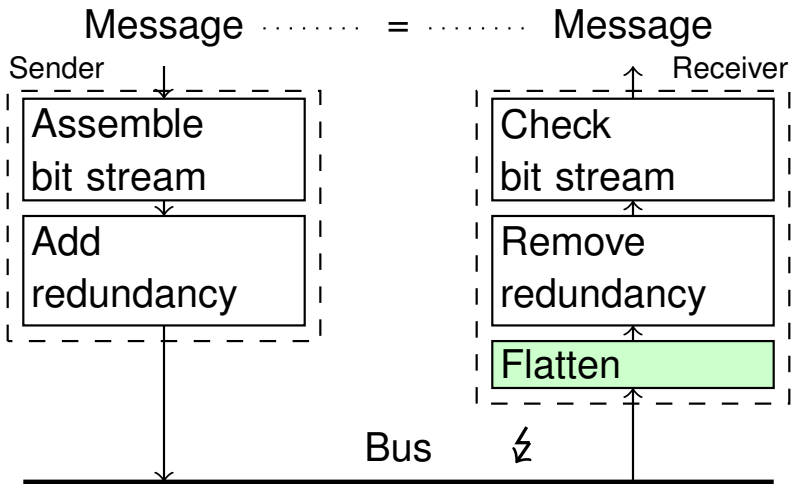
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

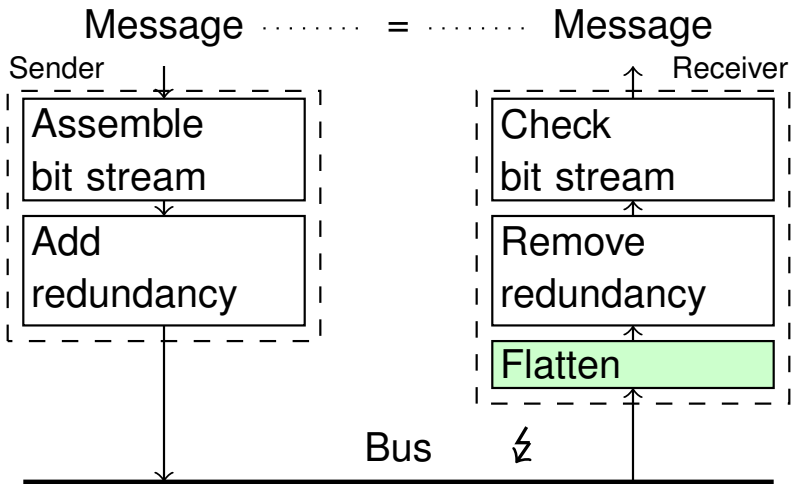
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

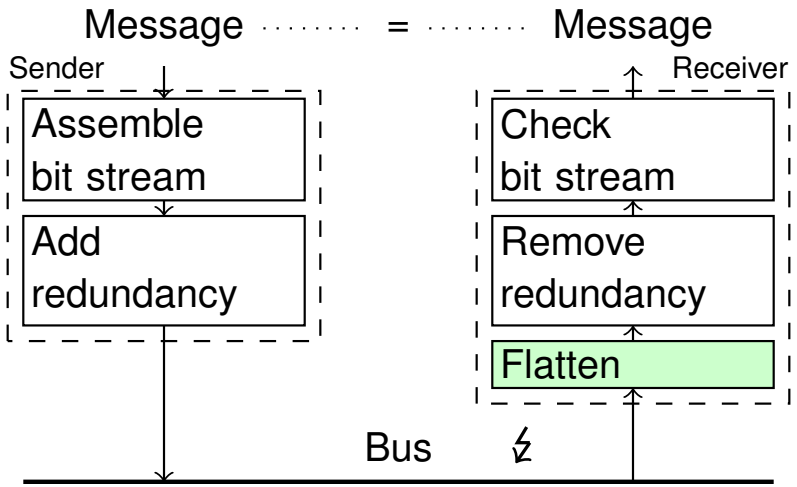
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

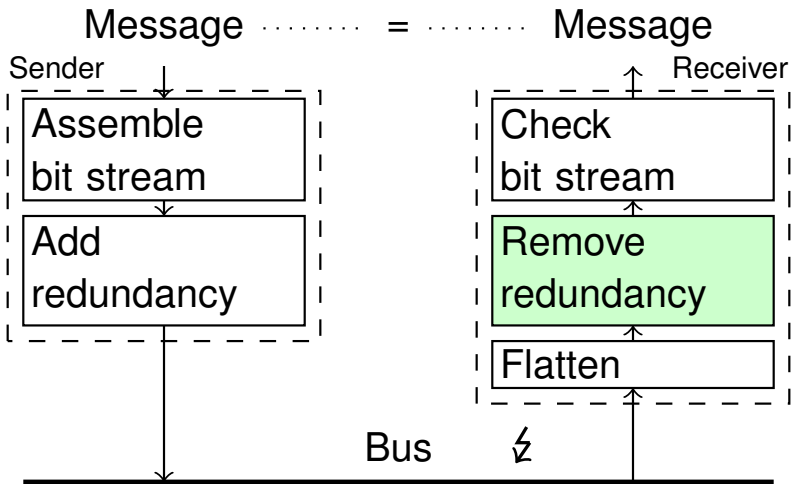
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

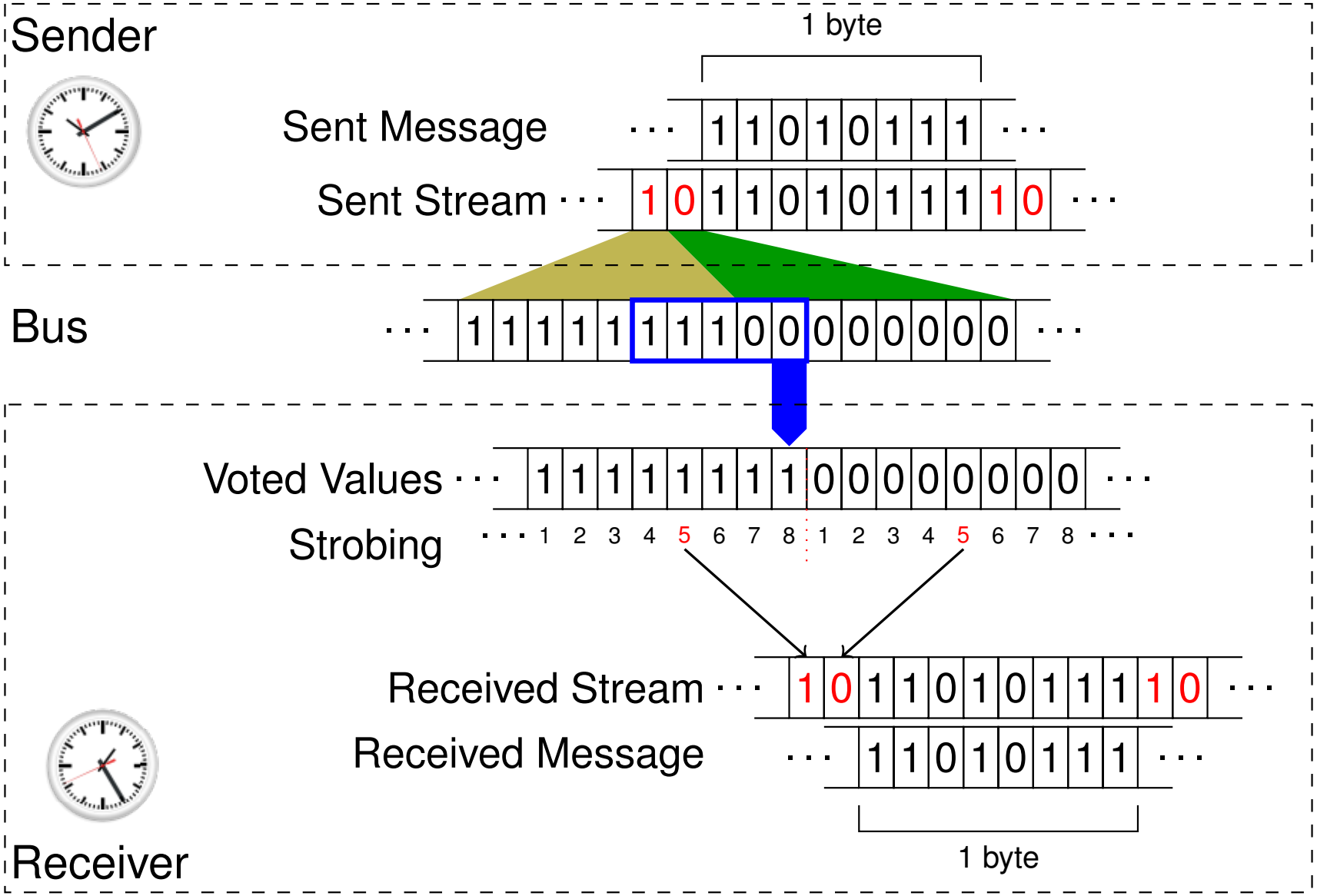
  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



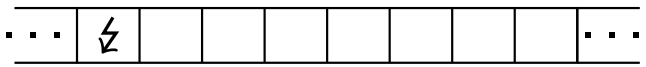
- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

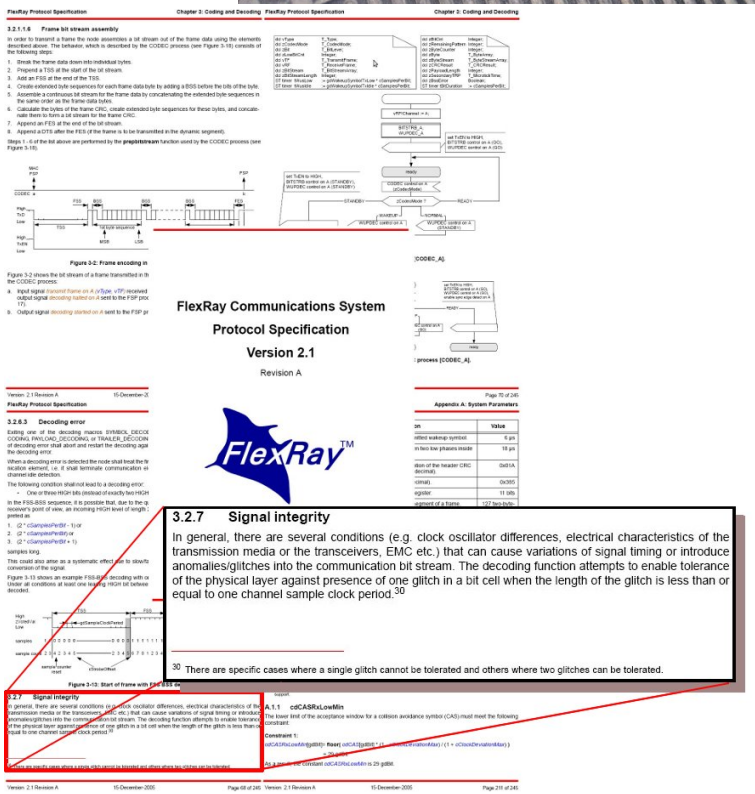
Example: FlexRay Physical Layer Protocol



FlexRay Error Resilience

FlexRay





FlexRay Communications System
Protocol Specification
Version 2.1
Revision A

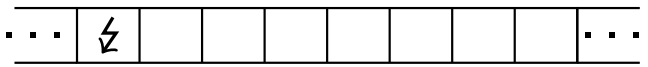
3.2.7 Signal integrity

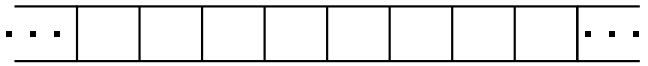
In general, there are several conditions (e.g. clock oscillator differences, electrical characteristics of the transmission media or the transceivers, EMC etc.) that can cause variations of signal timing or introduce anomalies/glitches into the communication bit stream. The decoding function attempts to enable tolerance of the physical layer against presence of one glitch in a bit cell when the length of the glitch is less than or equal to one channel sample clock period.[30]

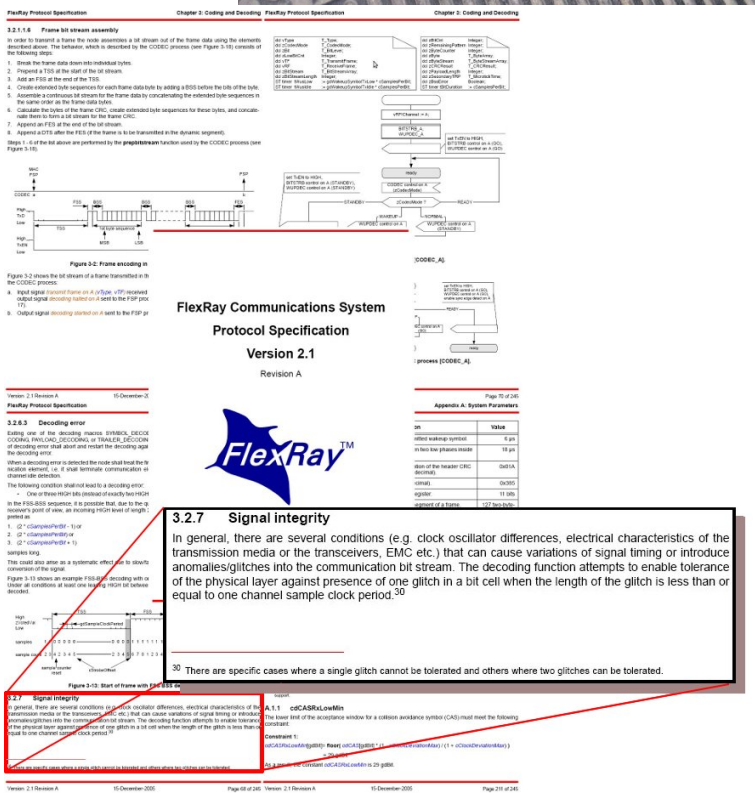
[30] There are specific cases where a single glitch cannot be tolerated and others where two glitches can be tolerated.

FlexRay Error Resilience

FlexRay



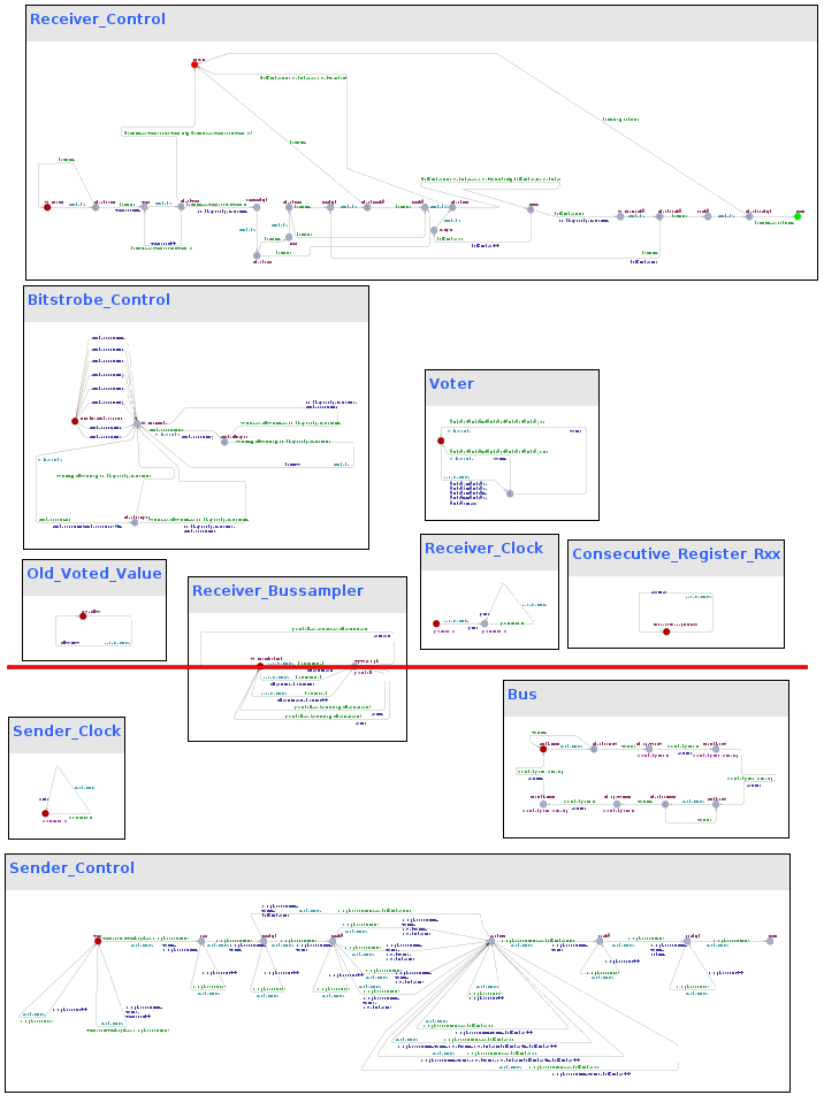




Modeling: Sender / Receiver Partition

partition model into sender / receiver

$\rightarrow$ just 2 clocks needed



Modeling: Sender / Receiver Partition

partition model into sender / receiver
$\rightarrow$ just 2 clocks needed

model hardware and protocol
independently
$\rightarrow$ separation of concerns, better
adaptability

Modeling: Sender / Receiver Partition

partition model into sender / receiver
$\rightarrow$ just 2 clocks needed

model hardware and protocol independently
$\rightarrow$ separation of concerns, better adaptability

keep hardware parameterized
$\rightarrow$ better adaptability, exploration of hardware requirements

Modeling: Sender / Receiver Partition

partition model into sender / receiver
$\rightarrow$ just 2 clocks needed

model hardware and protocol independently
$\rightarrow$ separation of concerns, better adaptability

keep hardware parameterized
$\rightarrow$ better adaptability, exploration of hardware requirements

$> 10^{600}$ different FlexRay messages
$\rightarrow$ use nondeterminism for state space reduction

Contribution: Glitch Position Counts

Position of glitch more important than number of glitches

- easy to destroy message using just 3 meanly placed glitches
- message survives hundreds of nicely placed glitches
- environment independent of protocol $\rightarrow$ glitches independent of protocol
- *pattern*: relative position of glitches to each other
- pattern of glitches matters

Contribution: Glitch Position Counts

Position of glitch more important than number of glitches

- easy to destroy message using just 3 meanly placed glitches
- message survives hundreds of nicely placed glitches
- environment independent of protocol $\rightarrow$ glitches independent of protocol
- *pattern*: relative position of glitches to each other
- pattern of glitches matters

Contribution: Guaranteed Glitch Tolerance

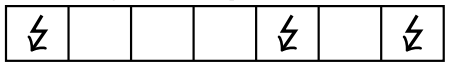
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

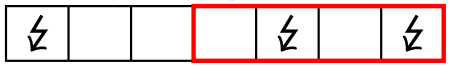
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

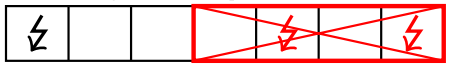
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

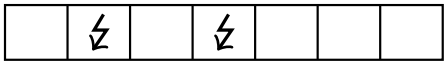
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Contribution: Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)
  E.g.: ...  ...

Note: one message $\approx$ 21.000 samples

Contribution: Parameter Exploration

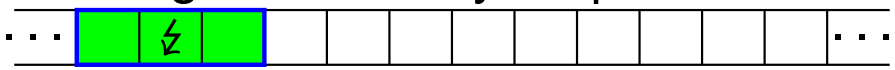
Parameter exploration using binary search:
boundaries for variation of a single parameter

glitch tolerance	delay variance
(1 out of 4)	$1.435ns \rightarrow 7.6075ns$
(2 out of 88)	$1.435ns \rightarrow 7.6075ns$
(1 at most)	$1.435ns \rightarrow 12.020ns$

glitch tolerance	deviation of clock from standard rate
(1 out of 4)	$0.15\% \rightarrow 0.46\%$
(2 out of 88)	$0.15\% \rightarrow 0.46\%$
(1 at most)	$0.15\% \rightarrow 1.09\%$
(no glitches)	$0.15\% \rightarrow 1.74\%$

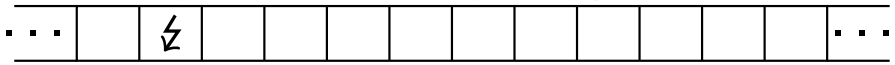
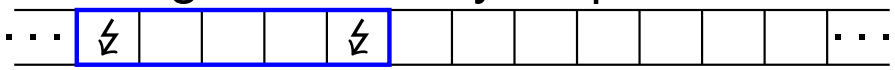
Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.:
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

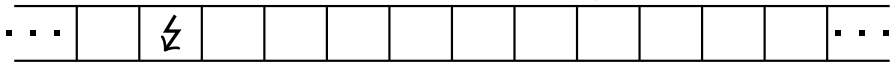
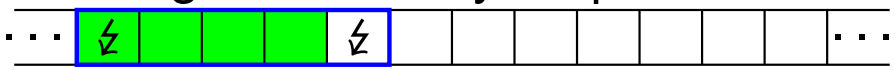
Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.: 
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

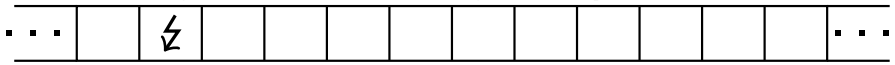
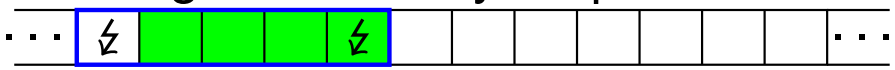
Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.: 
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

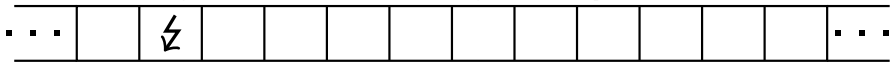
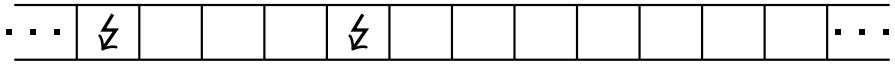
Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.: 
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

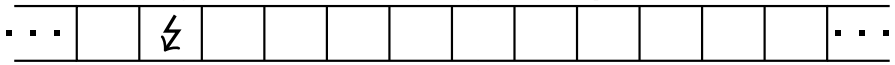
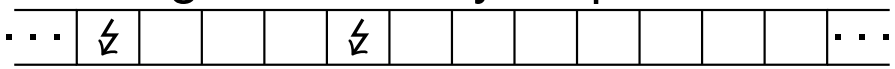
Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.: 
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

Contribution: Glitch Tolerance vs. Voting Window Size

Voting window size of

- 3, tolerates 1 glitch in every sequence of 3 consecutive samples.
  E.g.: 
- 5, tolerates 1 glitch in every sequence of 4 consecutive samples.
  E.g.: 
- 7, tolerates 1 glitch in every sequence of 5 consecutive samples.
- 9, tolerates 1 glitch in every sequence of 6 consecutive samples.

Uppaal: Semi-Symbolic Approach

Loc		Clock zone
l_0	$\mapsto$	$0 \leq x < 4$
l_1	$\mapsto$	$8 \leq x < 15 \ \vee \ 16 < y - x \leq 23$
l_2	$\mapsto$	$x = 42 \ \vee \ y = 45$
$\ldots$		$\ldots$

UPPAAL - A Tool Suite for Automatic Verification of Real-Time Systems
Bengtsson, Larsen, Larsson, Pettersson, Yi – 1995

The Tool KRONOS
Daws, Olivero, Tripakis, Yovine – 1995

Difference Bound Matrix (DBM)

The clock zone

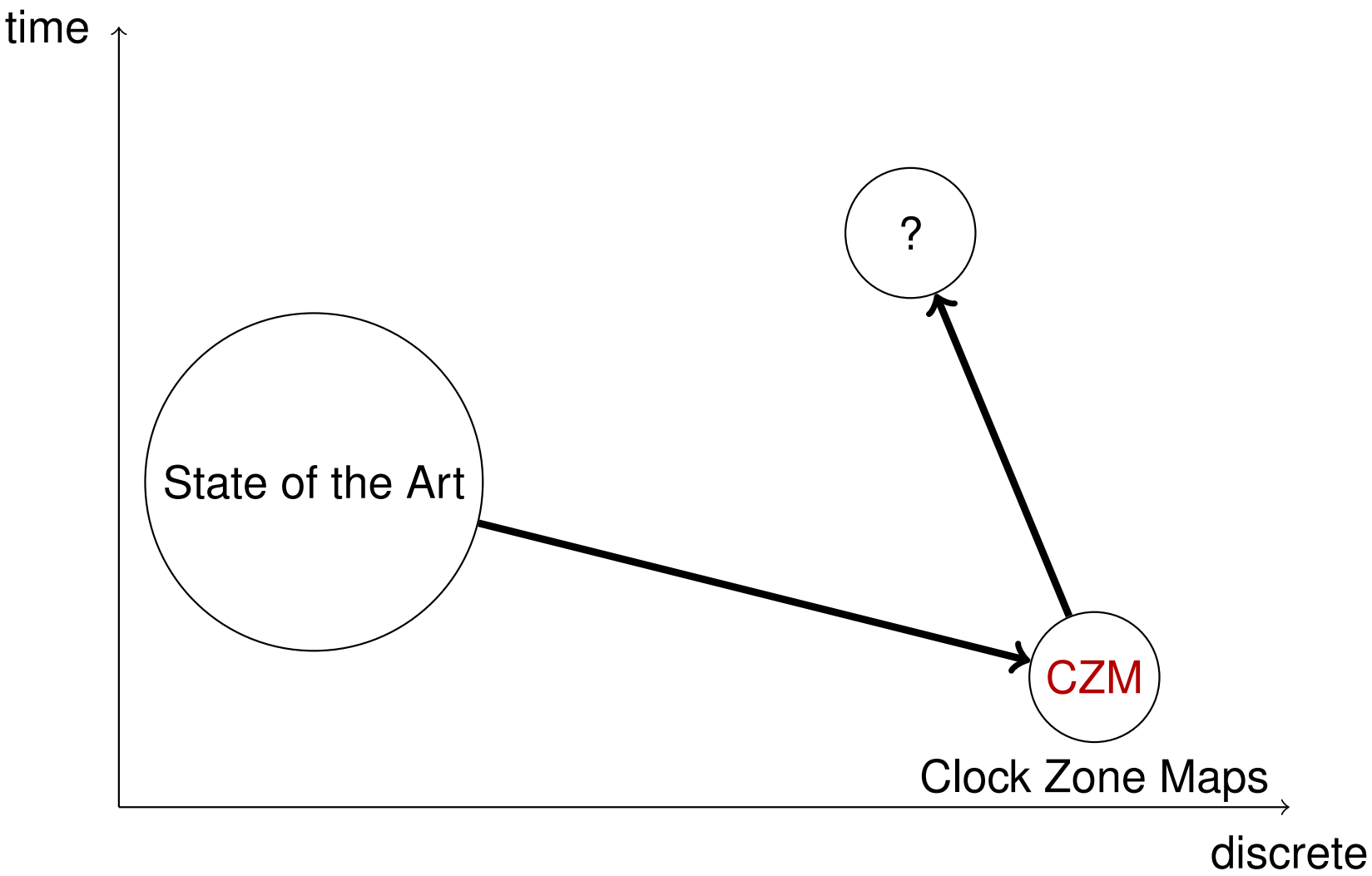
$$x_1 - x_2 \leq 1 \wedge 2 \leq x_1 \leq 3 \wedge x_2 < 3$$

can be represented by the following canonical DBM:

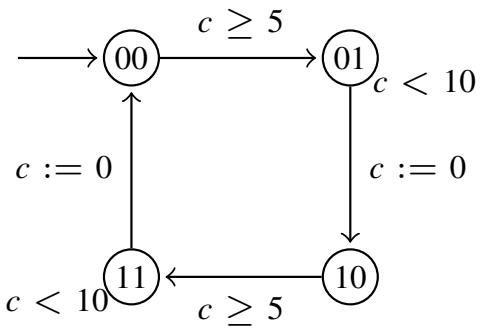
$+\backslash -$	x_0	x_1	x_2
x_0	$(0, \leq)$	$(-2, \leq)$	$(-1, \leq)$
x_1	$(3, \leq)$	$(0, \leq)$	$(1, \leq)$
x_2	$(3, <)$	$(1, <)$	$(0, \leq)$

Timing Assumptions and Verification of Finite-State Concurrent Systems
Dill – 1990

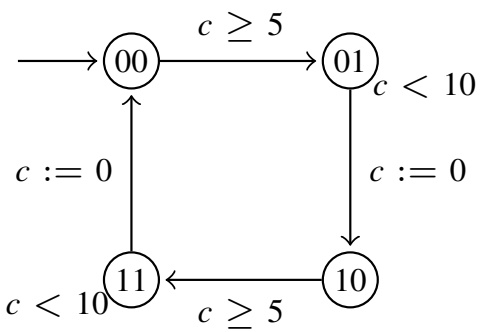
New Data Structure: Clock Zone Maps



Clock Zone Maps



Clock Zone Maps

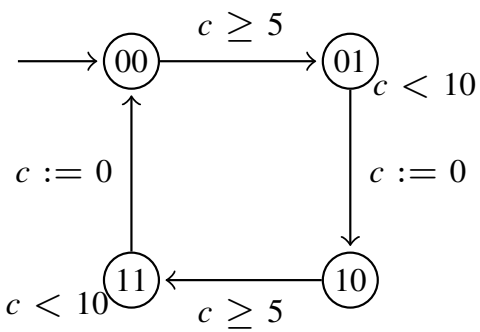


location$\mapsto$zones

l_1	l_0	clock zone
0	0	true
0	1	$c \geq 5 \wedge c < 10$
1	0	true
1	1	$c \geq 5 \wedge c < 10$

Clock Zone Maps

Novel state space representation:
mapping from clock zones (DBMs) to ROBDDs



location$\mapsto$zones

l_1	l_0	clock zone
0	0	true
0	1	$c \geq 5 \wedge c < 10$
1	0	true
1	1	$c \geq 5 \wedge c < 10$

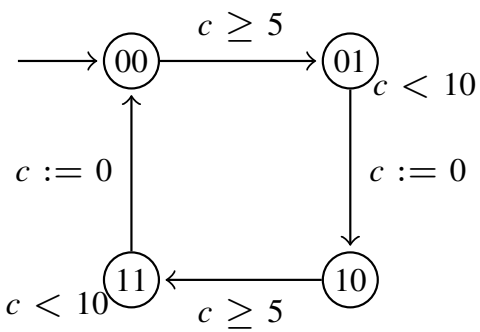
zone$\mapsto$locations

clock zone	l_1	l_0
true	0	0
	1	0
$c \geq 5 \wedge c < 10$	0	1
	1	1

Making the Right Cut in Model Checking Data-Intensive Timed Systems
Ehlers, Gerke, Peter
12th International Conference on Formal Engineering Methods (ICFEM 2010)

Clock Zone Maps

Novel state space representation:
mapping from clock zones (DBMs) to ROBDDs



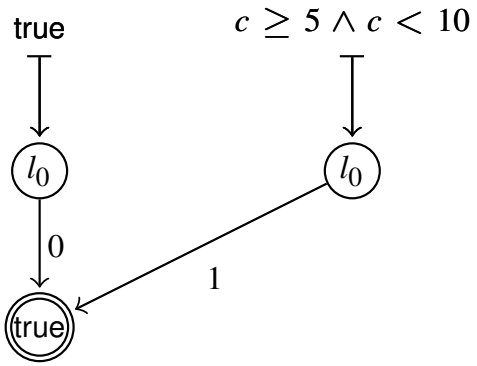
location$\mapsto$zones

l_1	l_0	clock zone
0	0	true
0	1	$c \geq 5 \wedge c < 10$
1	0	true
1	1	$c \geq 5 \wedge c < 10$

zone$\mapsto$locations

clock zone	l_1	l_0
true	0	0
	1	0
$c \geq 5 \wedge c < 10$	0	1
	1	1

Clock zone map (CZM) of the reachable state space:



CZM Fixed Point Algorithm

Preparation

For every guard/reset pair, encode the discrete transitions as a BDD.

Let R_i be the CZM representing the states reachable in $\leq i$ steps.

Fixed Point Algorithm

$i := 0$

Do repeat

$R_{i+1} := R_i$

For every reachable clock zone z (i.e., $\forall$ keys from R_i)

For every guard/reset pair,

$L_i := BDD(transition) \wedge R_i(z)$

For every combination of invariants,

$R_{i+1} := R_{i+1} \cup \left(BDD(invariants) \wedge L_i\right)$

$i := i + 1$

Until $R_i = R_{i-1}$

CZM: Summary

State Space Representation:

Map clock zones to location sets

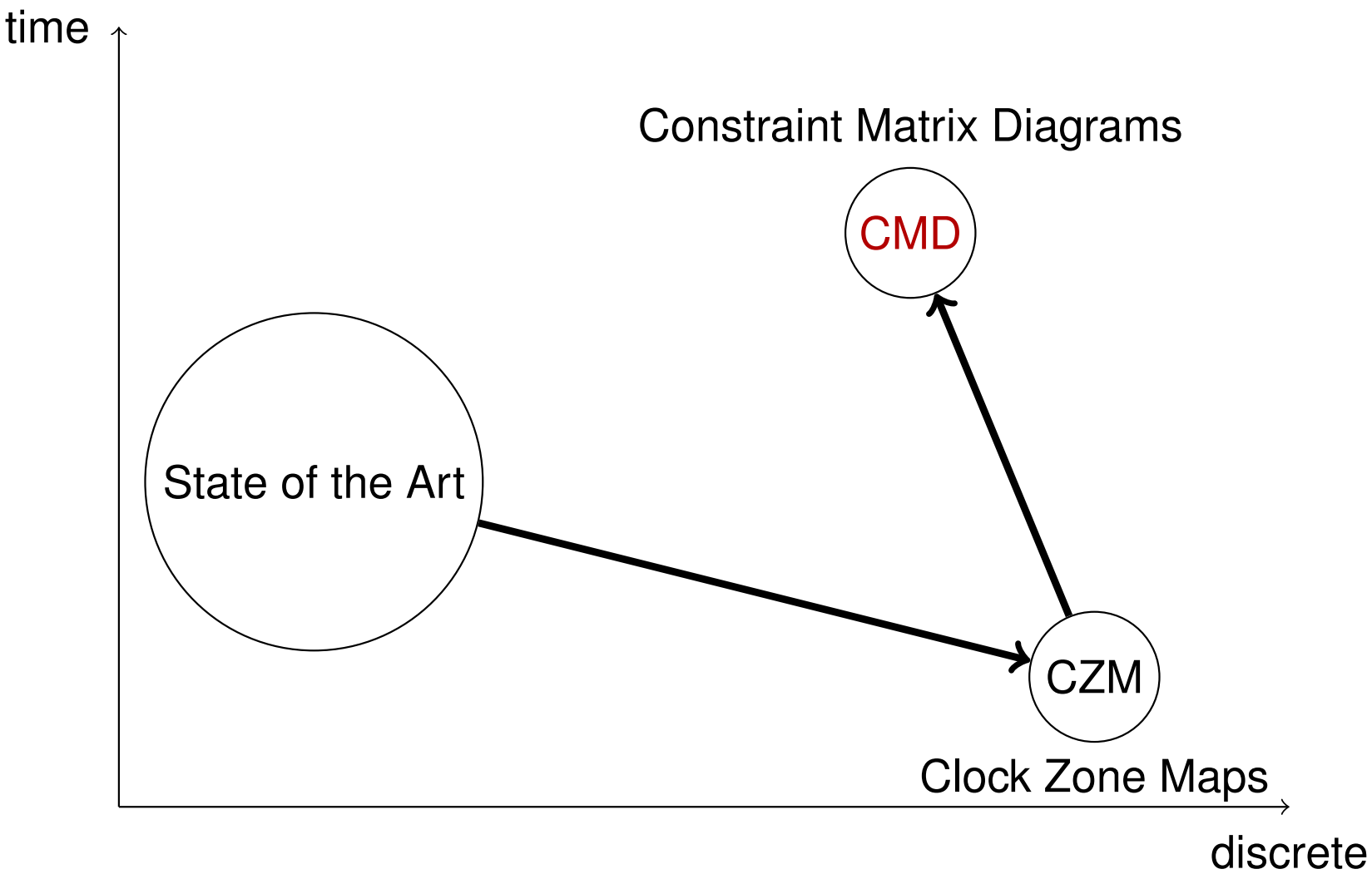
Reachability Model Checking:

Group transitions by guard/reset pairs;
Group locations by combinations of invariants

Promising Experimental Results:

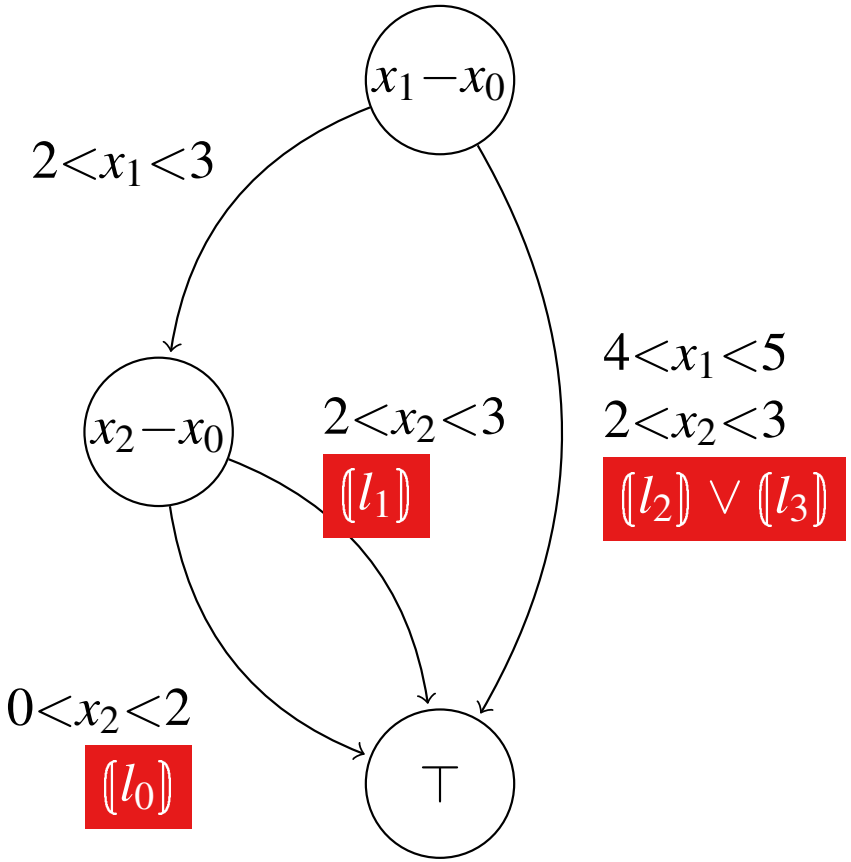
Data-intensive systems with many locations,
but little timing-related behavior

New Data Structure: Constraint Matrix Diagrams



Constraint Matrix Diagrams (CMDs)

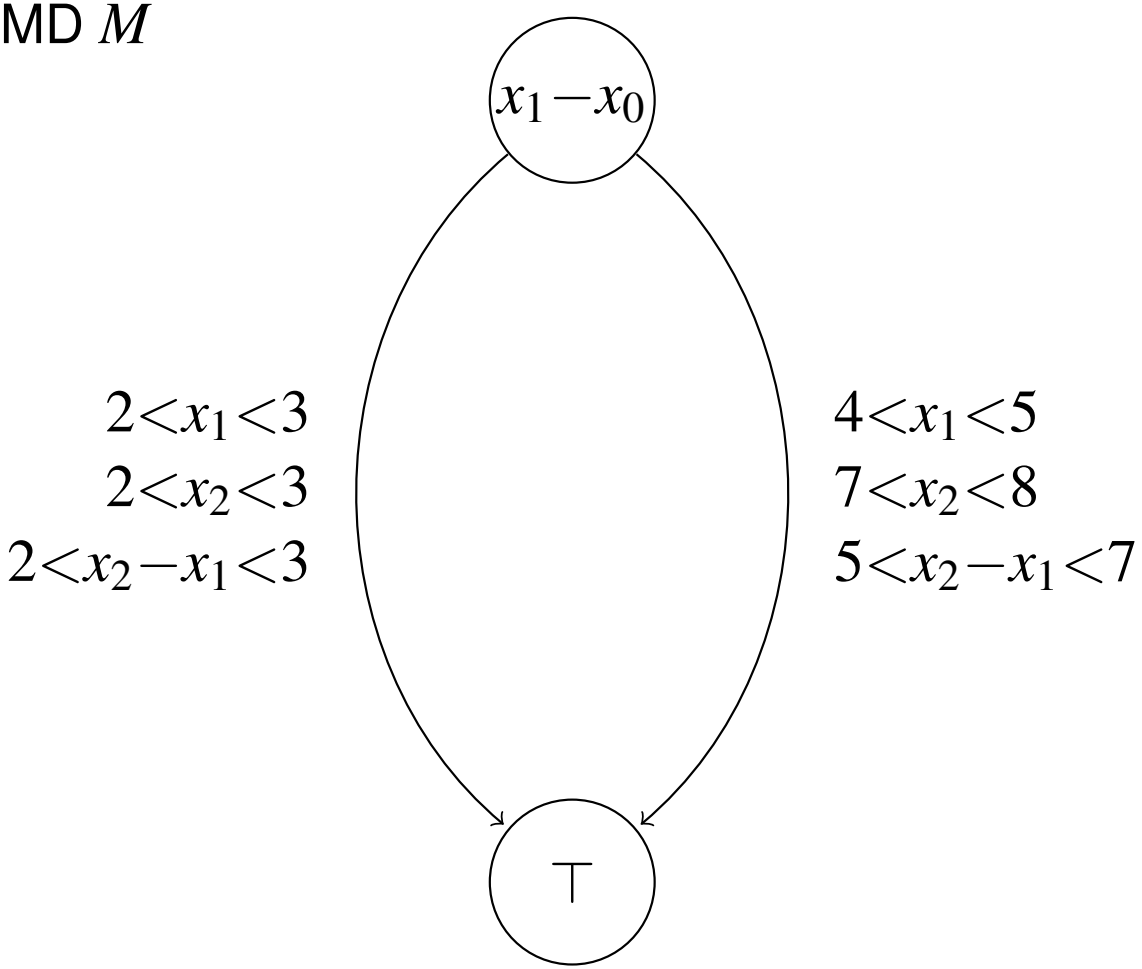
Locations are represented as boolean constraints (boolean functions) represented by **ROBDDs** in the constraint matrices



Fully Symbolic Timed Model Checking using Constraint Matrix Diagrams
Ehlers, Fass, Gerke, and Peter
31st IEEE Real-Time Systems Symposium (RTSS 2010)

$M \vee z$

CMD M

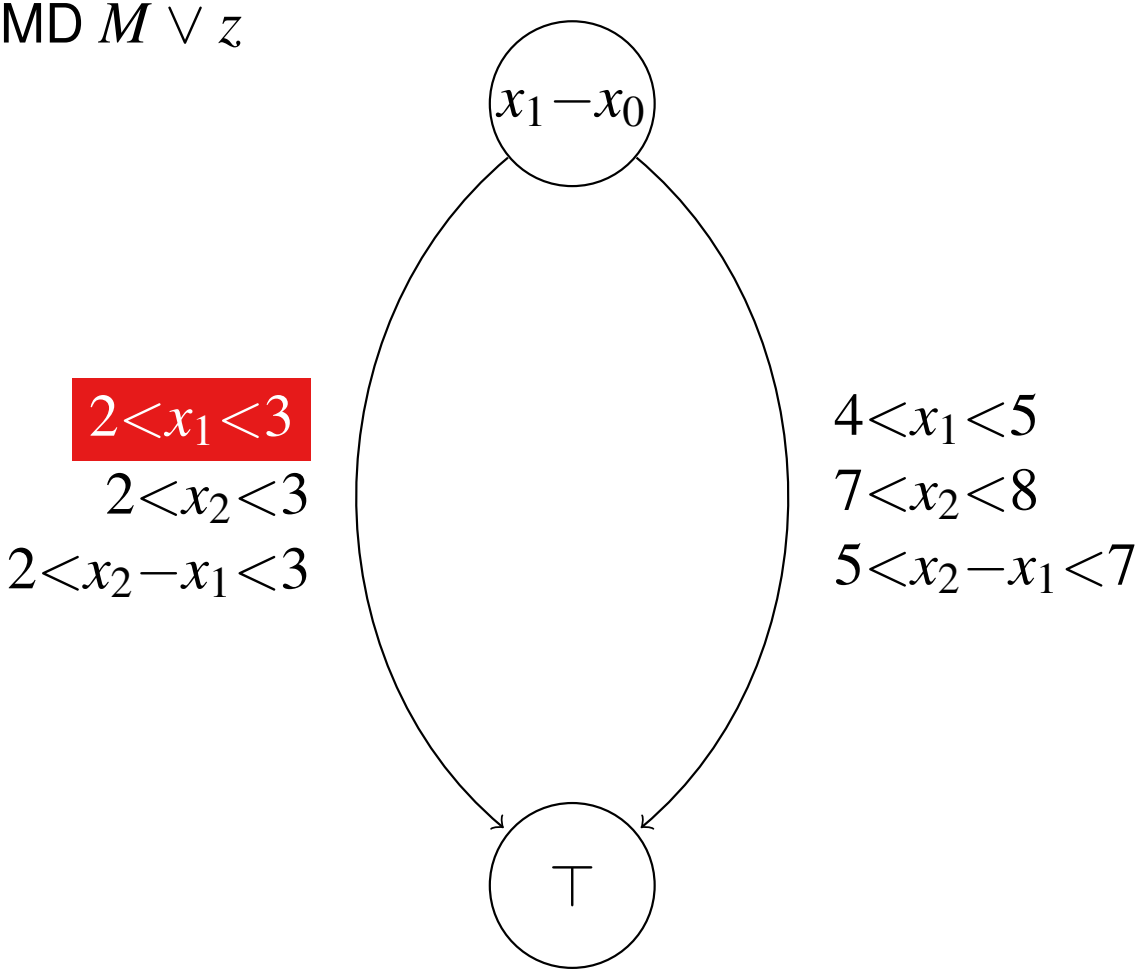


Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$
$$4 < x_2 < 5 \quad \wedge$$
$$5 < x_2 - x_1 < 7$$

$M \vee z$

CMD $M \vee z$

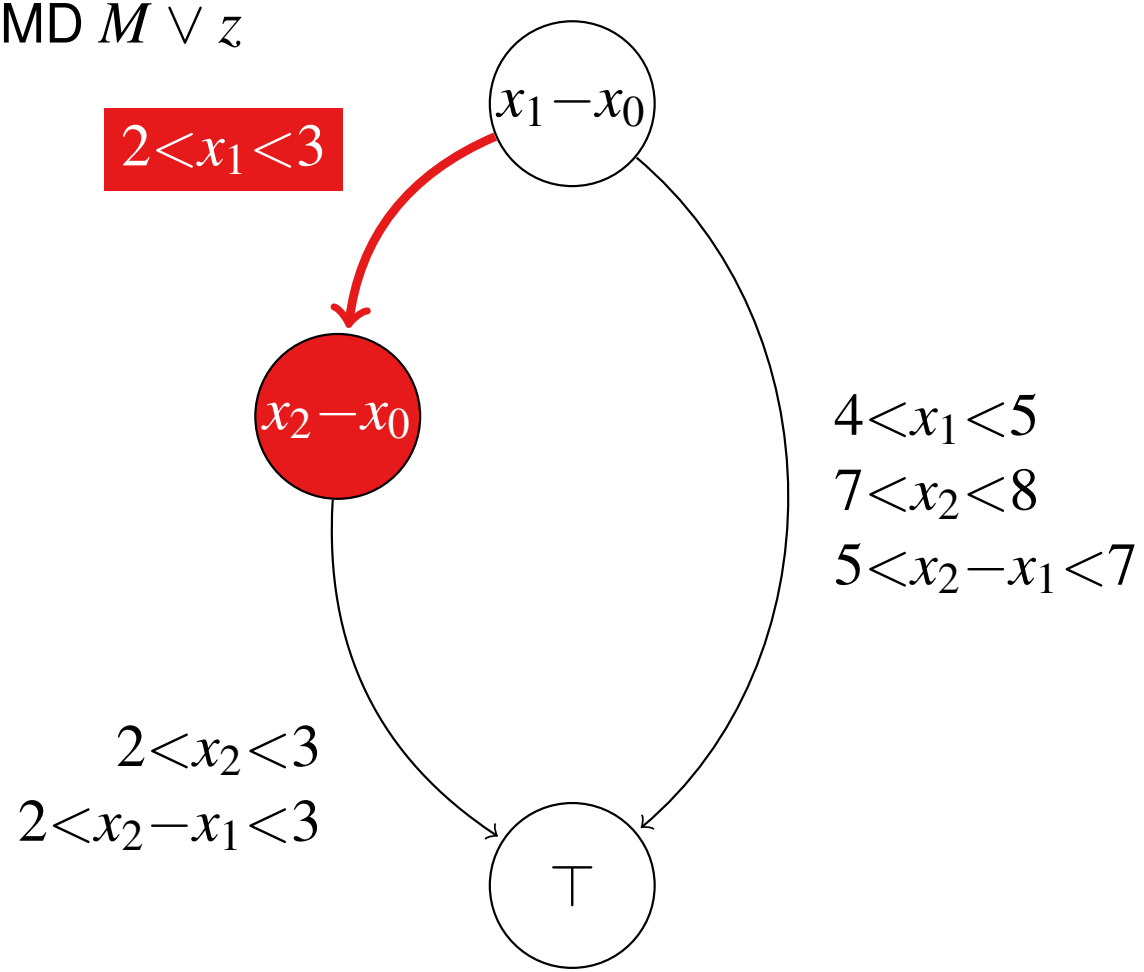


Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$
$$4 < x_2 < 5 \quad \wedge$$
$$5 < x_2 - x_1 < 7$$

$M \vee z$

CMD $M \vee z$

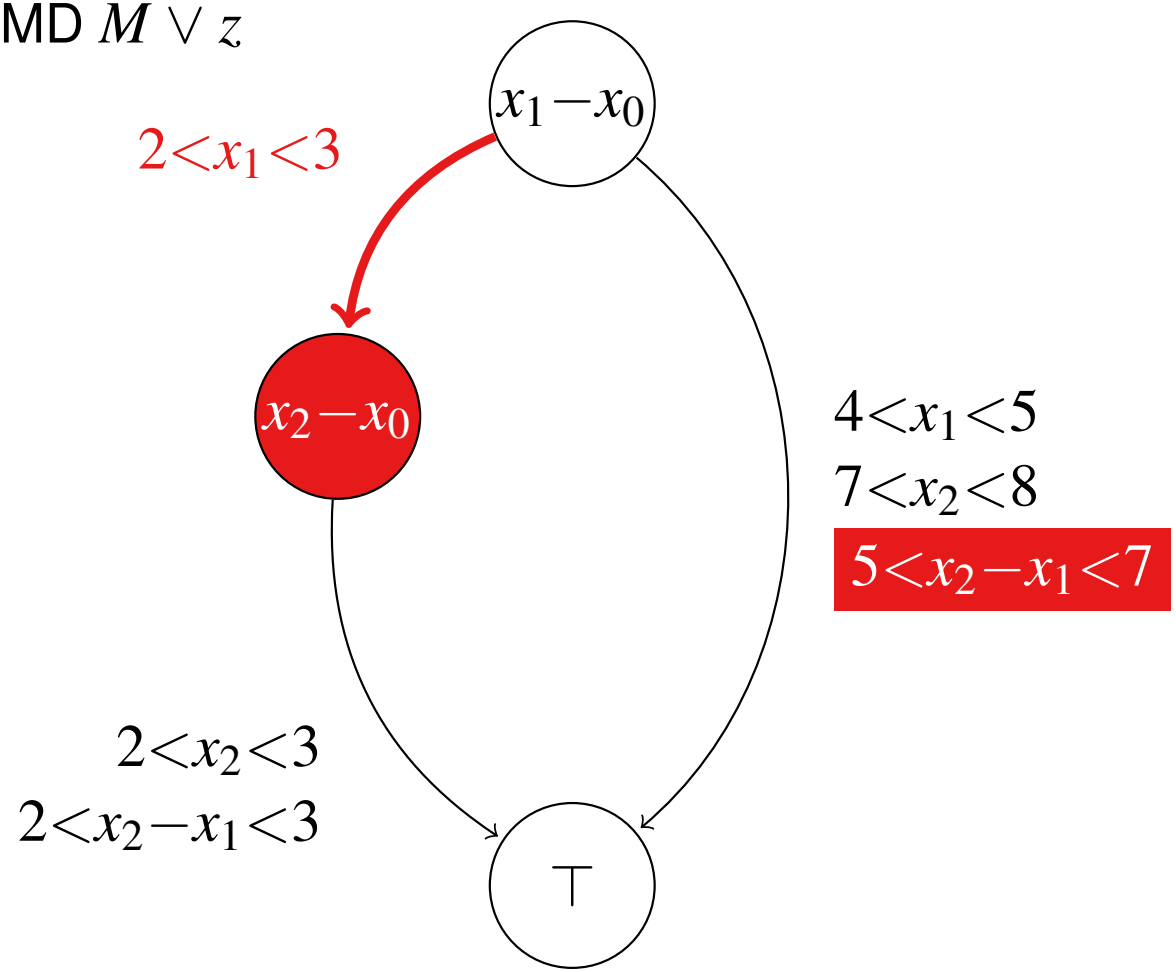


Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$
$$4 < x_2 < 5 \quad \wedge$$
$$5 < x_2 - x_1 < 7$$

$M \vee z$

CMD $M \vee z$

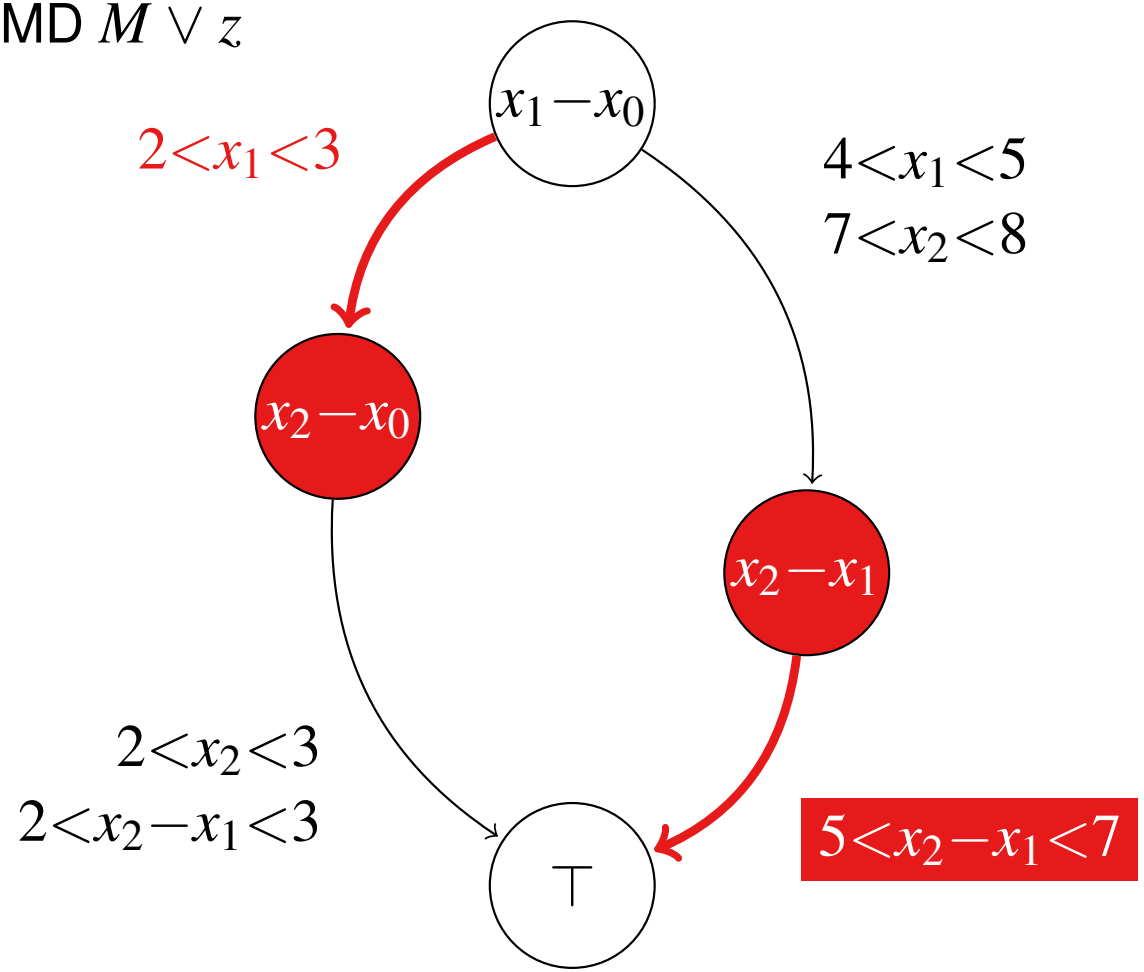


Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$
$$4 < x_2 < 5 \quad \wedge$$
$$5 < x_2 - x_1 < 7$$

$M \vee z$

CMD $M \vee z$

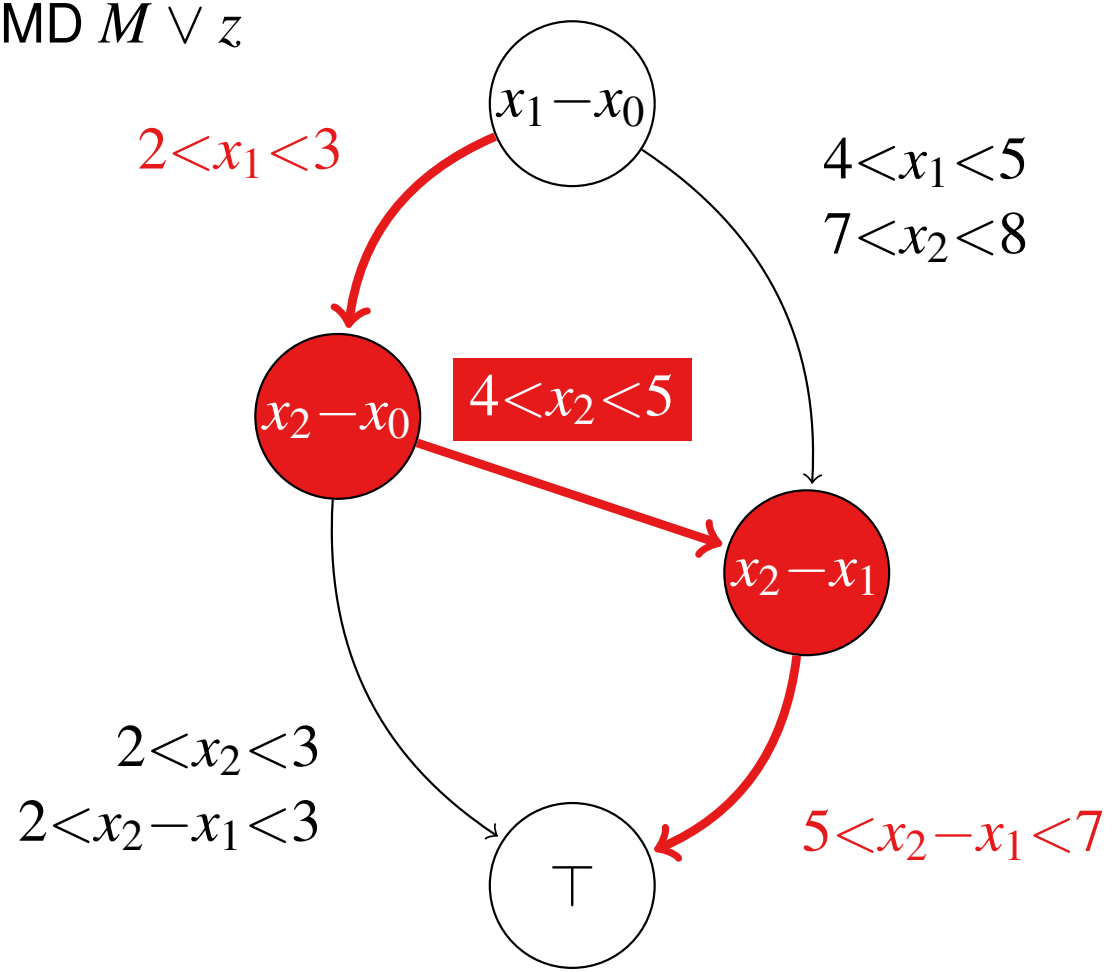


Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$
$$4 < x_2 < 5 \quad \wedge$$
$$5 < x_2 - x_1 < 7$$

$M \vee z$

CMD $M \vee z$



Convex clock zone z

$$2 < x_1 < 3 \quad \wedge$$

$$4 < x_2 < 5 \quad \wedge$$

$$5 < x_2 - x_1 < 7$$

CMD: Summary

Constraint Matrix Diagrams

- enable a fully symbolic timed reachability analysis
- combine diagram- and matrix-based approaches
- show promising results