Optimization of Robust Asynchronous Circuits by Local Input Completeness Relaxation

Cheoljoo Jeong Steven M. Nowick

Computer Science Department
Columbia University

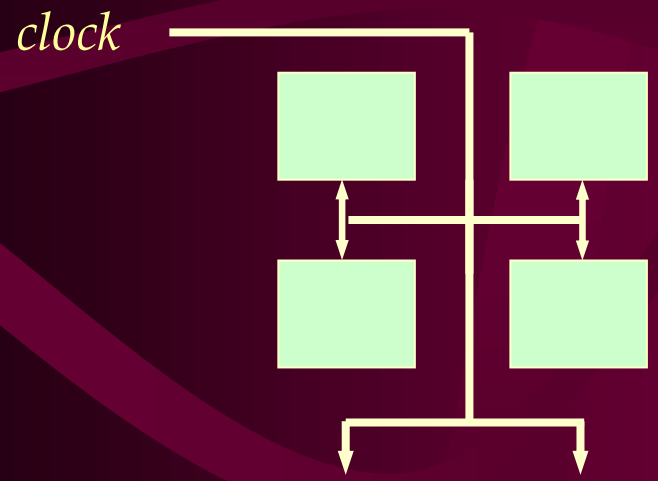
Outline

1. Introduction
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

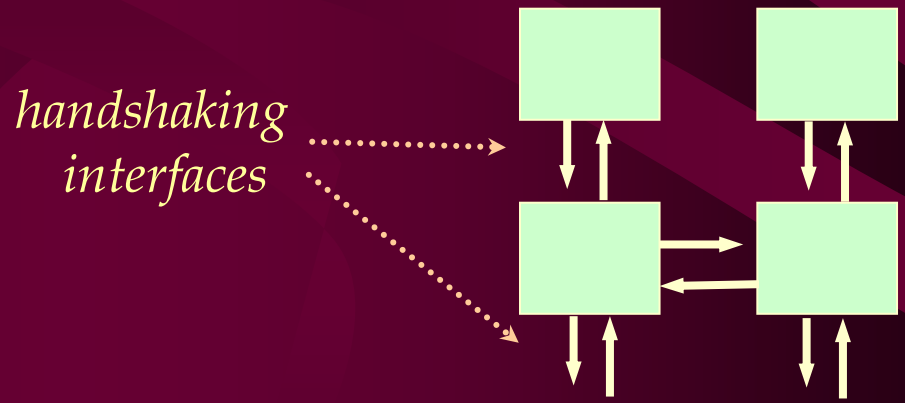
Asynchronous Circuits

- Synchronous vs. Asynchronous Systems

Synchronous System	Asynchronous System
global clock	*no global clock*
entire system operates *at fixed rate*	components operate at *varying rates*
centralized control	*distributed control* (*communicate locally* via *handshaking*)



Synchronous System



Asynchronous System

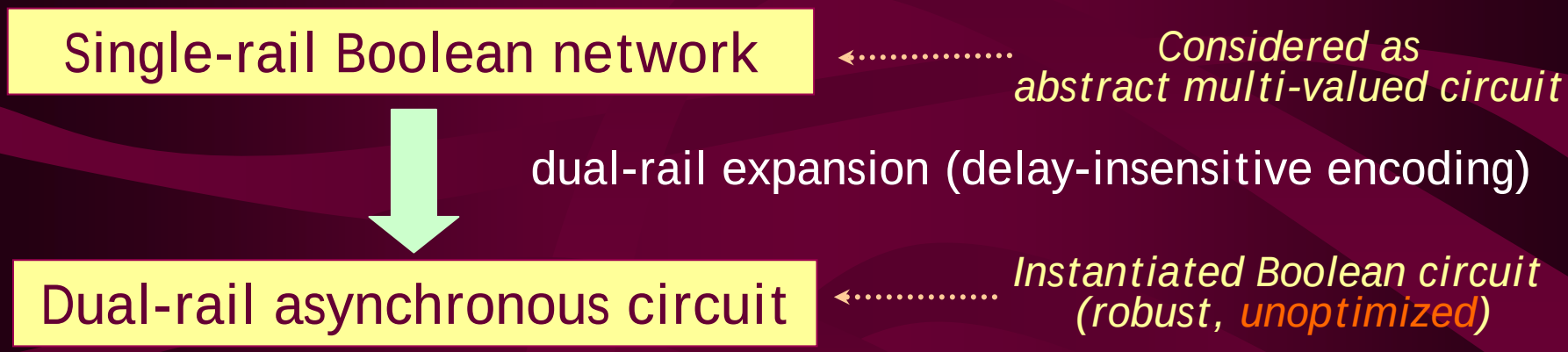
Asynchronous Circuits (cont.)

- Benefits of Asynchronous Circuits
  - Robustness to process variation
  - Mitigates: timing closure problem
  - Low power consumption, low EMI
  - Modularity

- Challenges in Asynchronous Circuit Design
  - Lack of CAD tools
  - Robust design is required: hazard-freedom
  - Area overhead
  - Lack of systematic optimization techniques

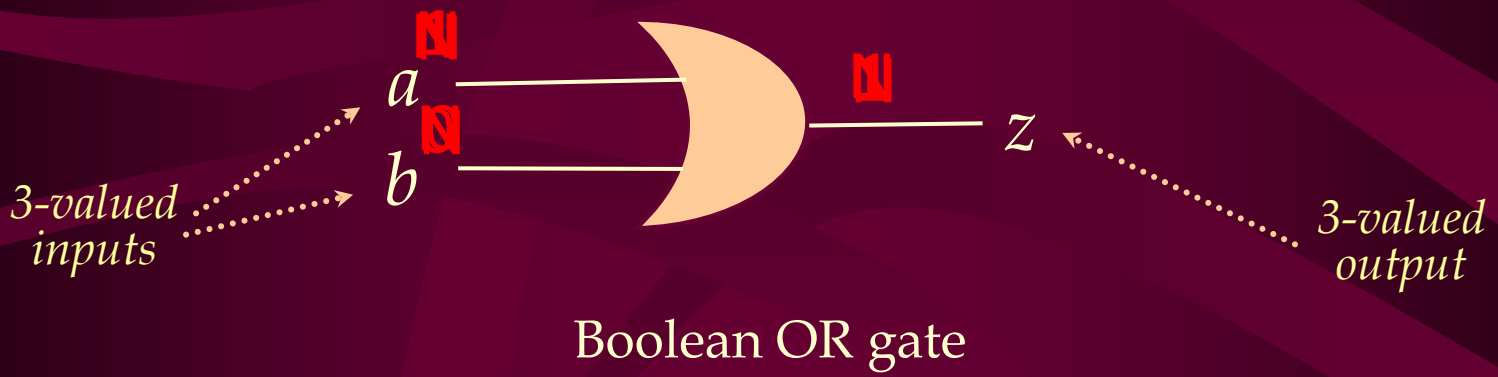
Unoptimized Asynchronous Synthesis Flow

- Existing Synthesis Flow:



Single-Rail Boolean Networks

- Boolean Logic Network: *Starting point for dual-rail circuit synthesis*
  - Uses three-valued logic with {0, 1, NULL}
    - 0/1 = data values
    - NULL = no data (invalid data)
  - Computation alternates between *DATA and NULL phases*

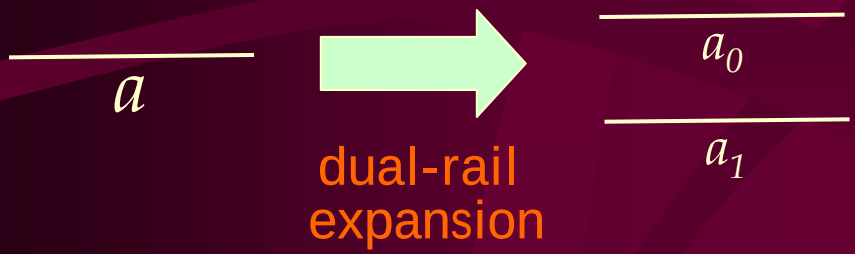


Boolean OR gate

  - DATA (Evaluate) phase:
    - outputs have DATA values only after all inputs have DATA values
  - NULL (Reset) phase:
    - outputs have NULL values only after all inputs have NULL values.

Delay-Insensitive Encoding

- Approach:
  - Single Boolean signal is represented by two wires (0-rail and 1-rail)
  - Goal: map abstract Boolean netlist to robust dual-rail asynchronous circuit



a_1	a_0	a
0	0	NULL
0	1	0
1	0	1
1	1	*Not allowed*

spacer

valid data

invalid

Encoding table

- Motivation: robust data communication

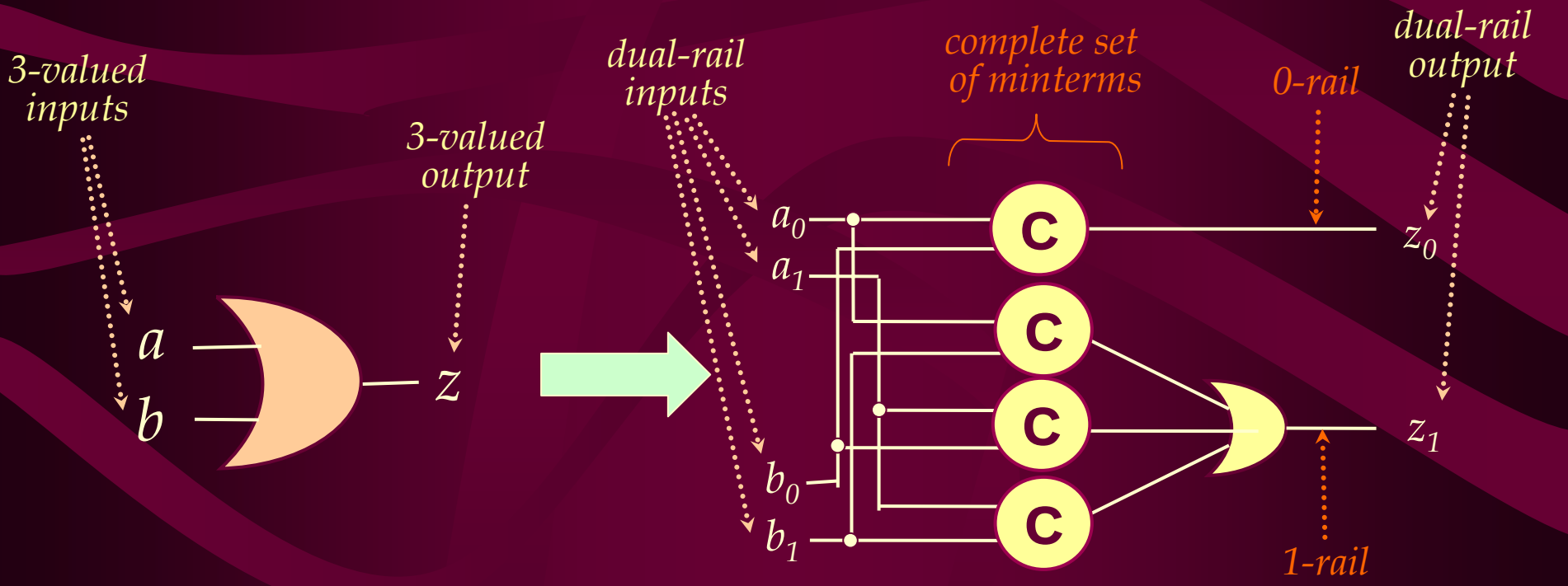
Dual-Rail Expansion

- Approach:
  - Goal: To obtain dual-rail circuit
    - dual-rail implementation of Boolean network
  - Single Boolean signal: is represented by two wires (0-rail / 1-rail)
  - Single Boolean gate: expanded into small dual-rail network

- Two common dual-rail circuit styles
  - DIMS (Delay-Insensitive Minterm Synthesis)
  - NCL (Null Convention Logic)

Dual-Rail Asynchronous Circuits

- DIMS-Style Dual-Rail Expansion:
  - Single Boolean gate: expanded into 2-level network

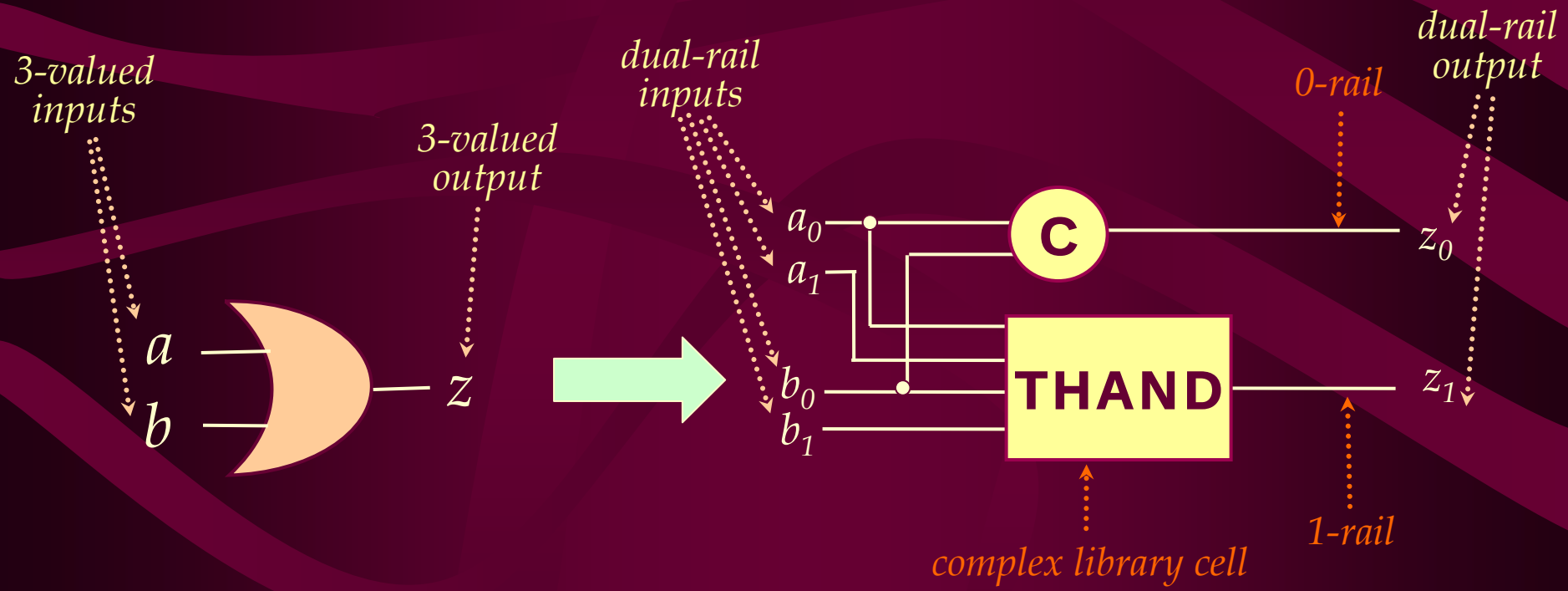


Boolean OR gate

DIMS-style dual-rail OR circuit

Dual-Rail Asynchronous Circuits (cont.)

- NCL-Style Dual-Rail Expansion *(Theseus Logic)*:
  - Single Boolean gate: expanded into two NCL gates
  - Allows more optimized mapping (to custom library)

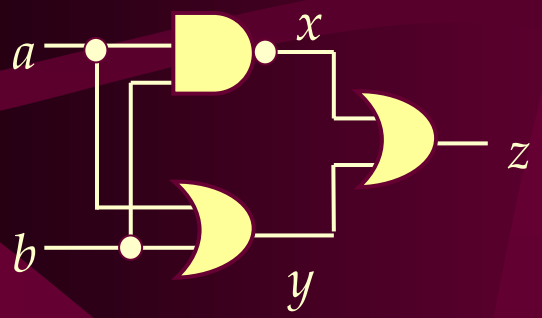


Boolean OR gate

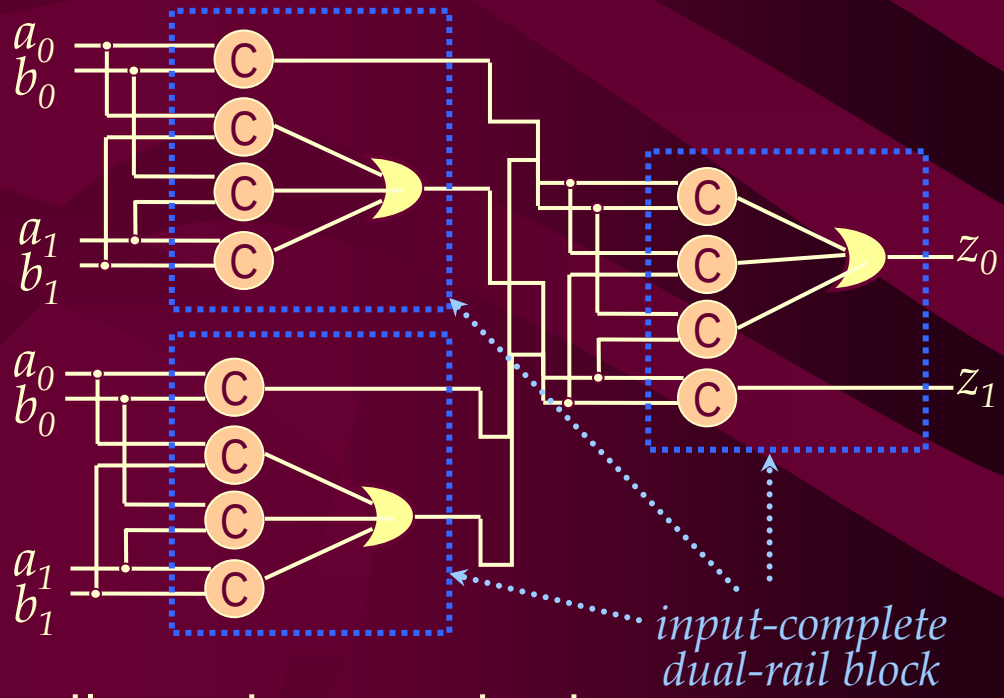
NCL-style dual-rail OR circuit

Summary: Existing Synthesis Approach

- Starting Point: single-rail abstract Boolean network (3-valued)
- Approach: performs dual-rail expansion of each gate
  - Use 'template-based' mapping (DIMS-style, NCL-style)
- End Point: (unoptimized) dual-rail asynchronous circuit
- Result: timing-robust asynchronous netlist



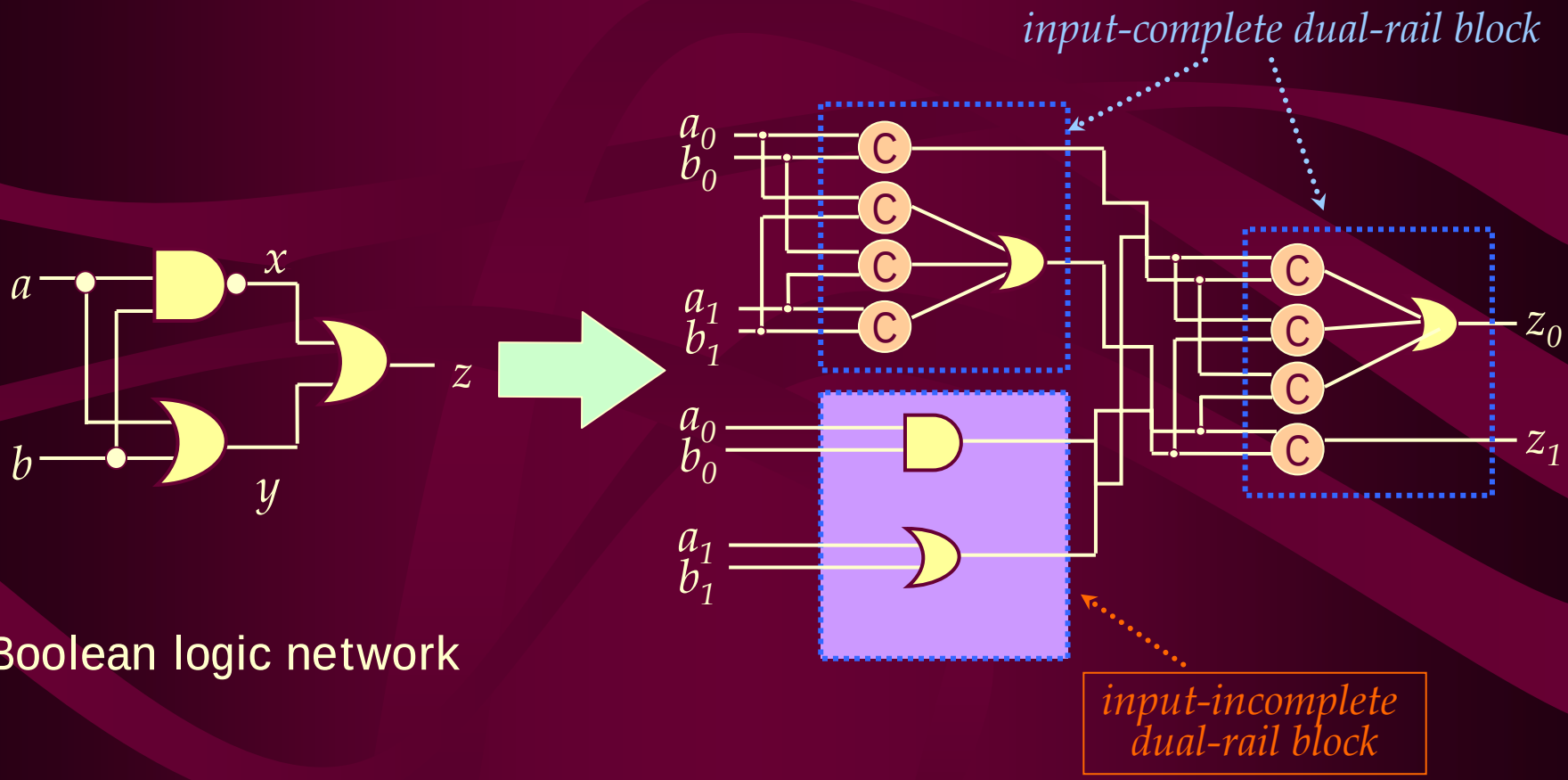
Boolean logic network



Dual-rail asynchronous circuit

Summary: Proposed Optimized Approach

- Dual-rail circuit example (with *gate relaxation*) :



Relaxed (= "eager") dual-rail asynchronous circuit

Outline

1. Introduction
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

Hazard Issues

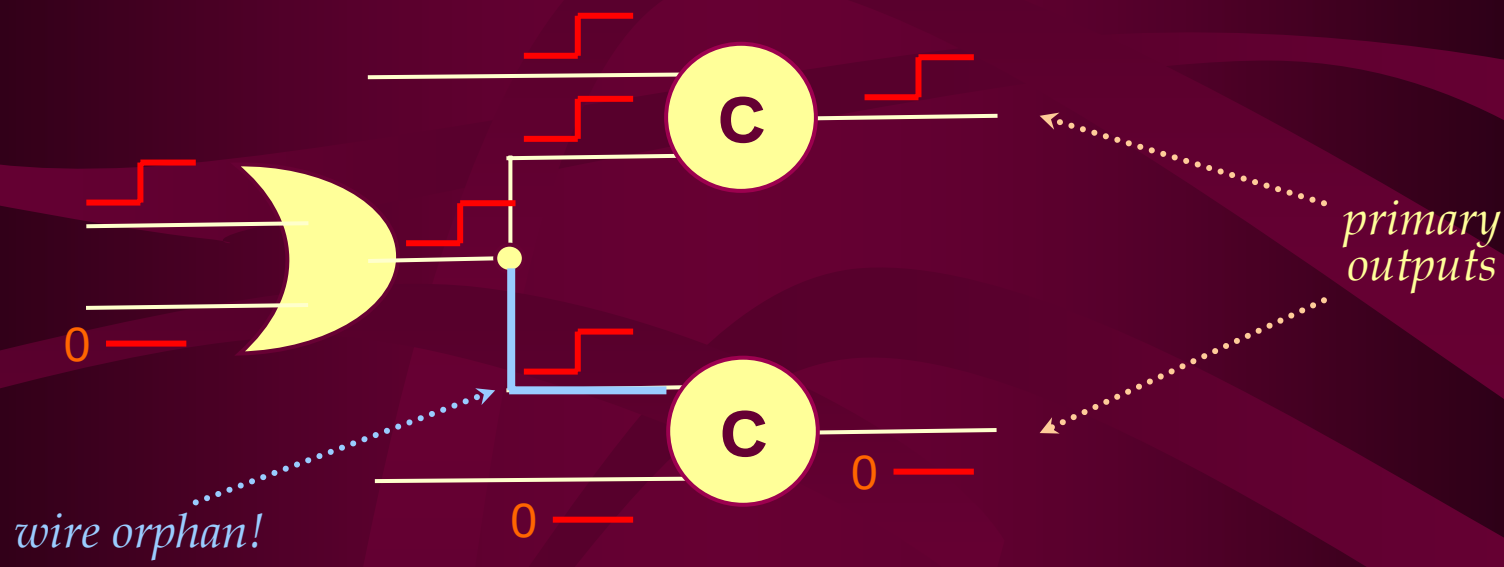
- Delay-Insensitivity (= Delay Model)
  - Assumes arbitrary gate and wire delay
    - circuit operates correctly under all conditions
  - Most robust design style

- "Orphans": Hazards to Delay-Insensitivity
  - *"Ineffective" signal transition sequences*

    (= unobservable paths)
  - *Wire orphans*: timing requirements on wires at fanout
  - *Gate orphans*: timing requirements on paths at fanout

Hazard Issues (cont.)

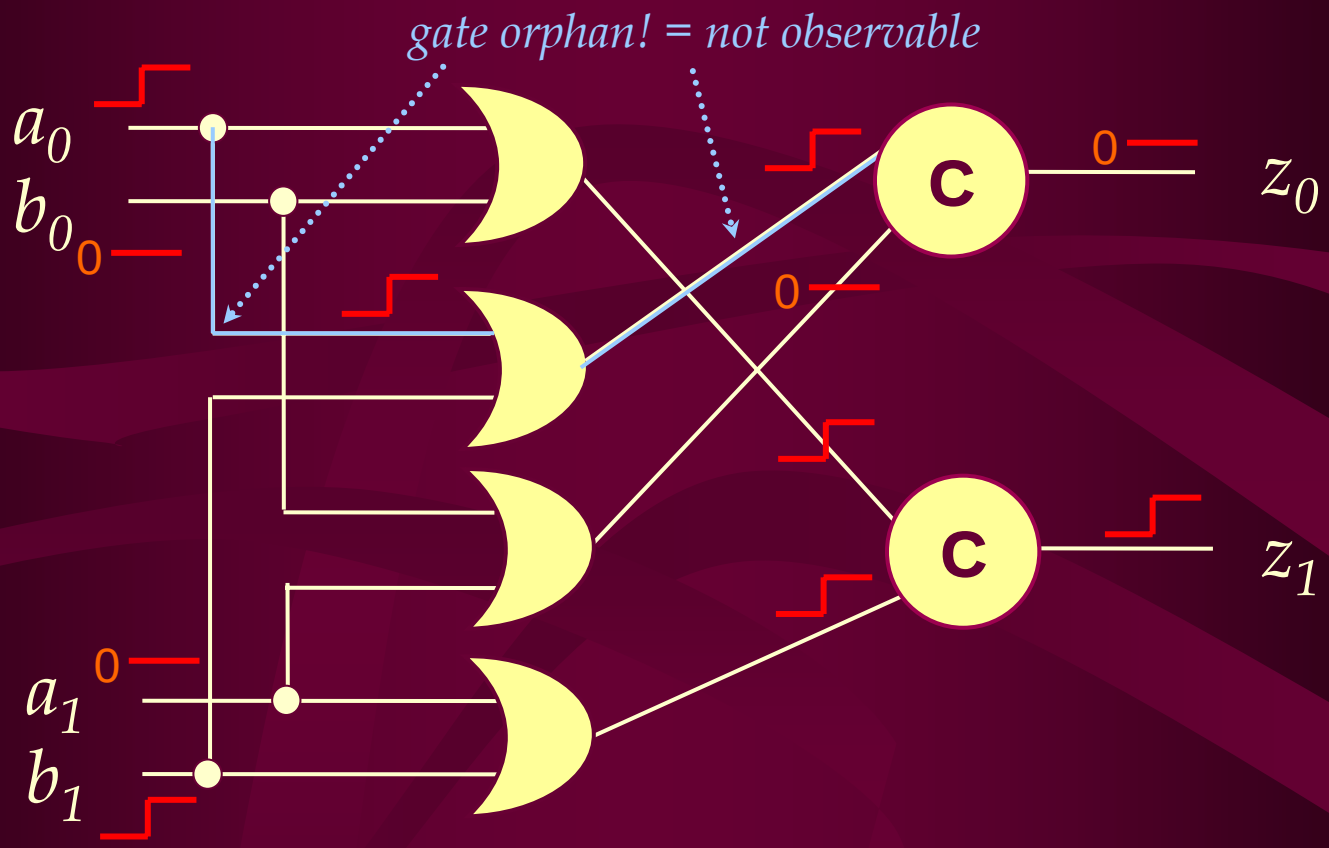
- Wire Orphan Example:



Wire orphan example

Hazard Issues (cont.)

- Gate Orphan Example:



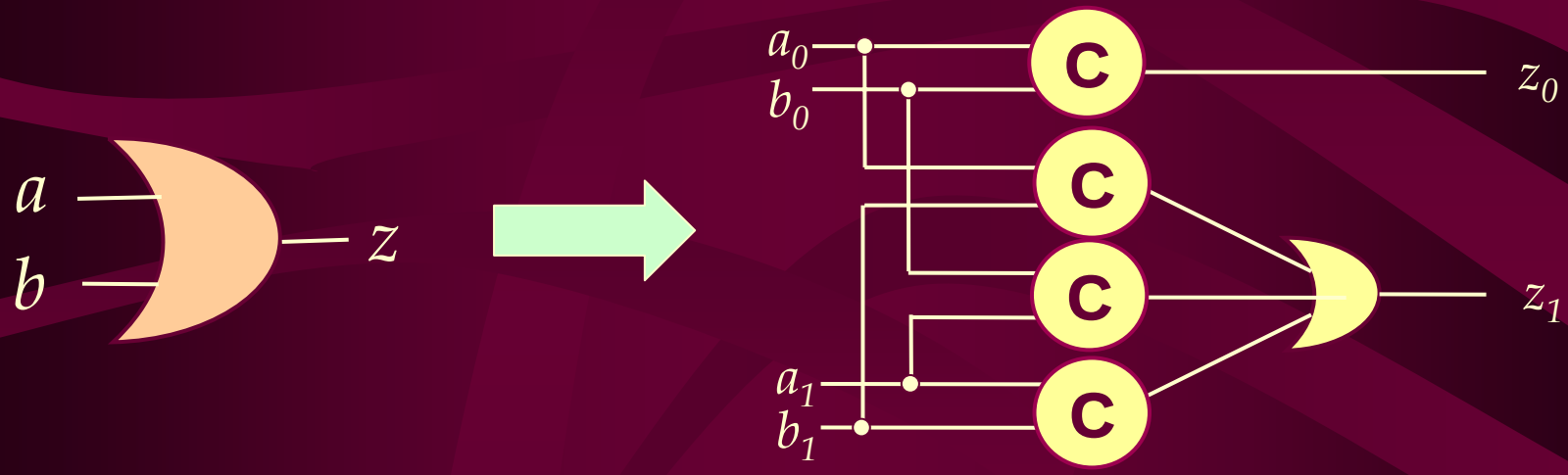
Gate orphan example

Timing-Robustness of Dual-Rail Circuits

- Wire Orphans: not a problem in practice
  - ⇒ Eliminated by enforcing physical timing constraints
  - Option #1: Isochronic fork timing requirement
    - At each fanout point: make every fanout delay nearly identical
  - Option #2: Loop timing requirement
    - At each fanout point: make every fanout delay less than time-to-reset
- Gate Orphans: challenge to robust circuit design
  - ⇒ Must avoid introducing gate orphans during synthesis
    - our goal

Input Completeness

- A dual-rail implementation of a Boolean gate is input-complete w.r.t. its input signals if an output changes *only after* all the inputs arrive.



Boolean OR gate

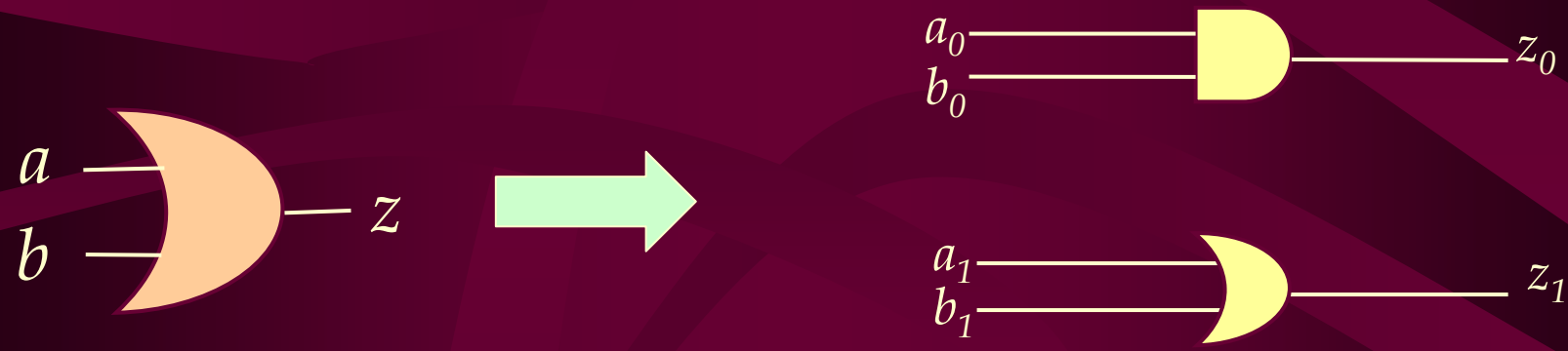
Input-complete dual-rail OR network

(input complete w.r.t. input signals a and b)

Enforcing input completeness for every gate is the traditional synthesis approach to avoid gate orphans.

Input Completeness (cont.)

- In *input-incomplete* (i.e. *"eager-evaluating"*) blocks, output can fire before all inputs arrive.



Boolean OR gate

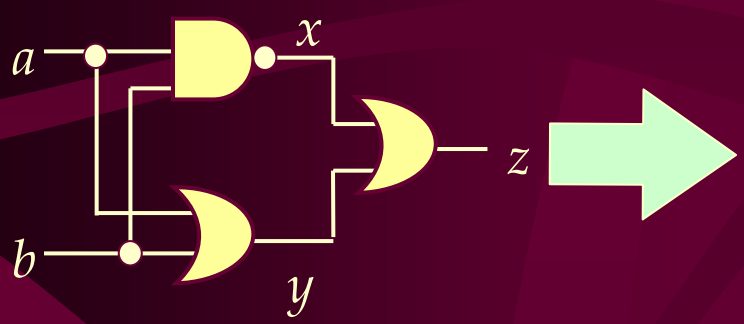
Input-incomplete dual-rail OR network

Outline

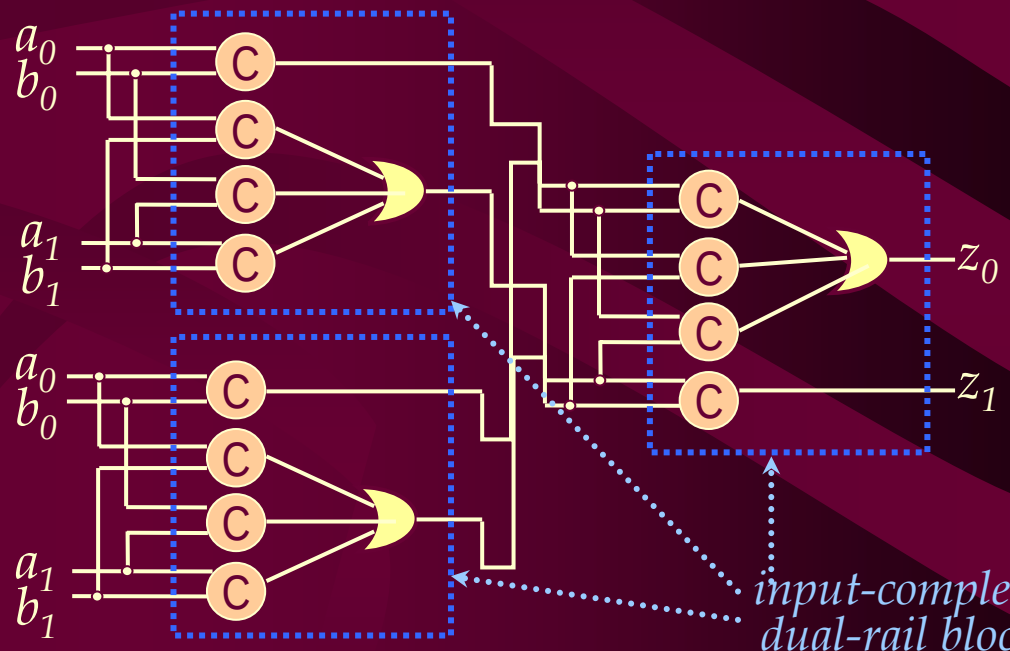
1. Introduction
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

Motivational Example #1

- Existing approach to dual-rail expansion is too restrictive.
  - Every Boolean gate is fully-expanded into an *input-complete* block.



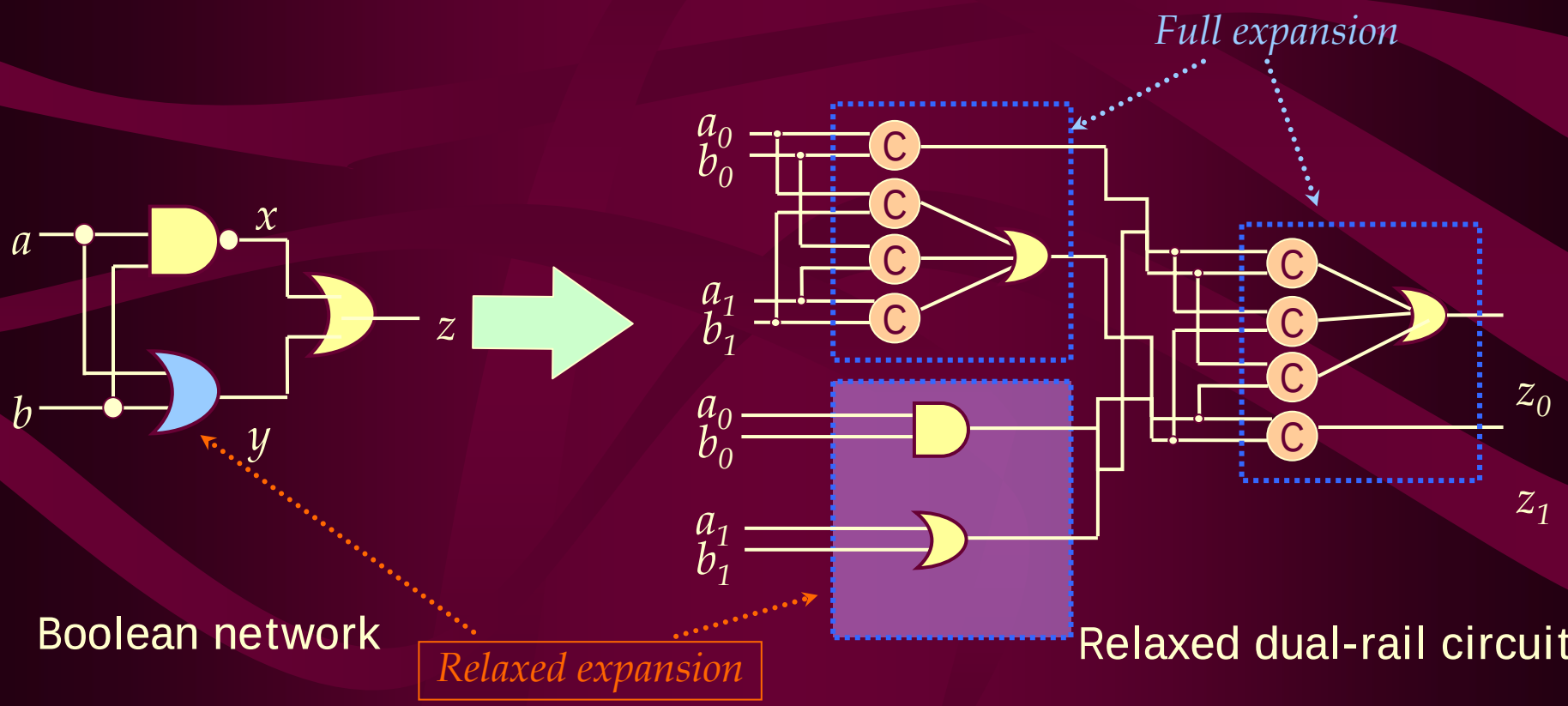
Boolean network



Dual-rail circuit with full expansion (no relaxation)

Motivational Example #1 (cont.)

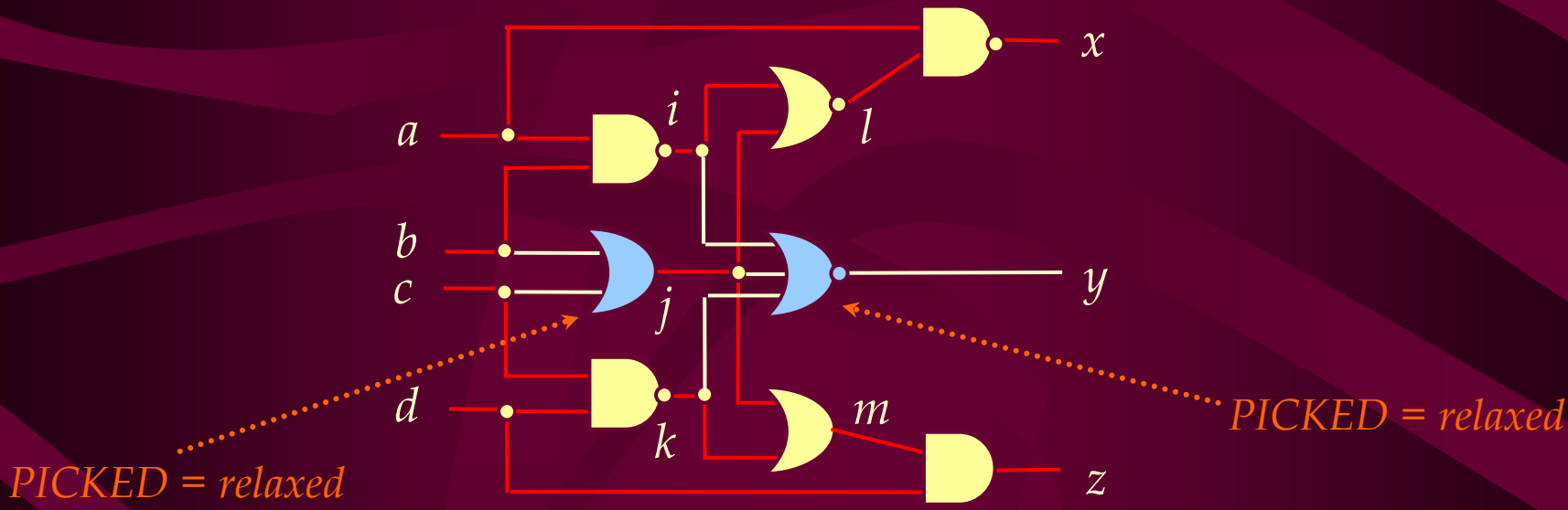
- Not every Boolean gate needs to be expanded into input-complete block.



Optimized dual-rail circuit is still robust (gate-orhpan-free)

Motivational Example #2

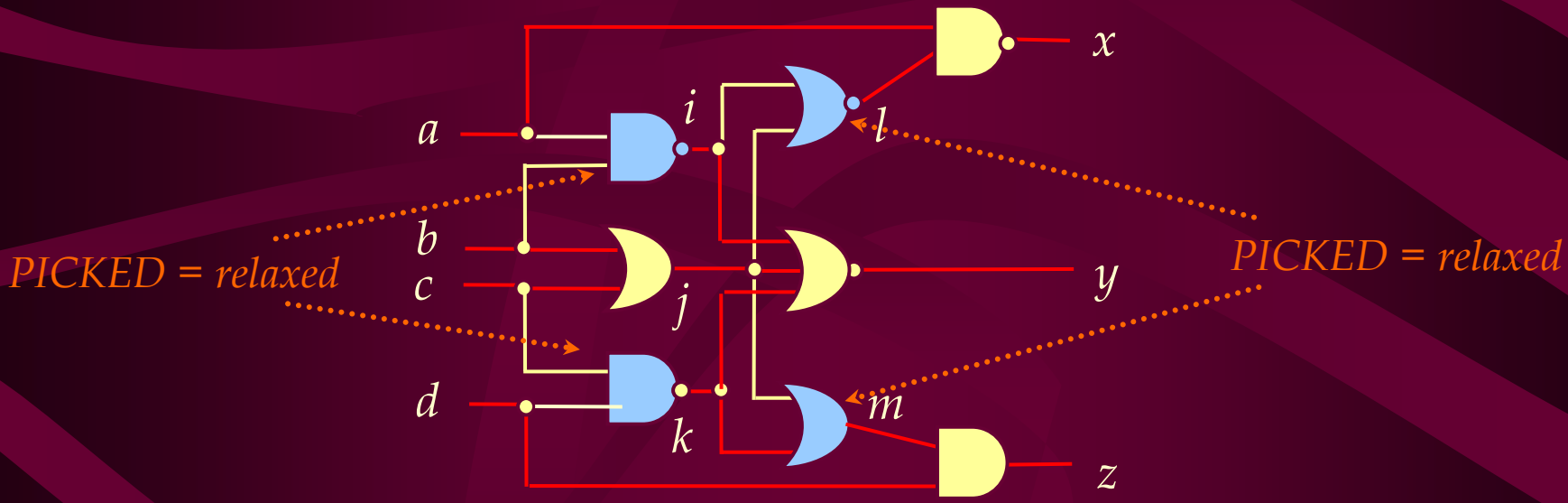
- Different choices may exist in relaxation.



Relaxation of Boolean network with *two* relaxed gates

Motivational Example #2 (cont.)

- Different choices may exist in relaxation.



Relaxation of Boolean network with *four* relaxed gates

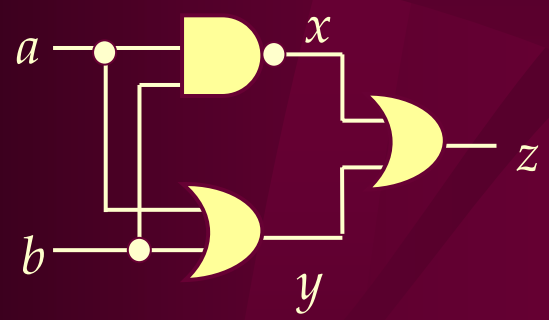
Outline

1. **Introduction**
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

Input Completeness Relaxation

- Theorem: *Dual-rail implementation of a Boolean network is timing-robust (i.e. gate-orphan-free) if and only if, for each signal, at least one of its fanout gates is not relaxed (i.e. ensures input completeness).*

- Example:

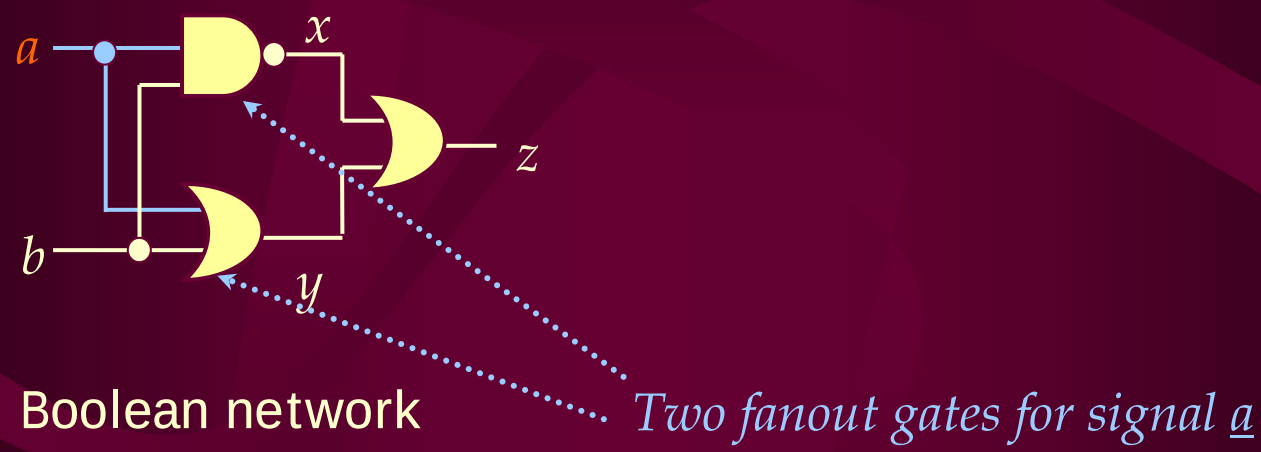


Boolean network

Input Completeness Relaxation

- Theorem: *Dual-rail implementation of a Boolean network is timing-robust (i.e. gate-orphan-free) if and only if, for each signal, at least one of its fanout gates is not relaxed (i.e. ensures input completeness).*

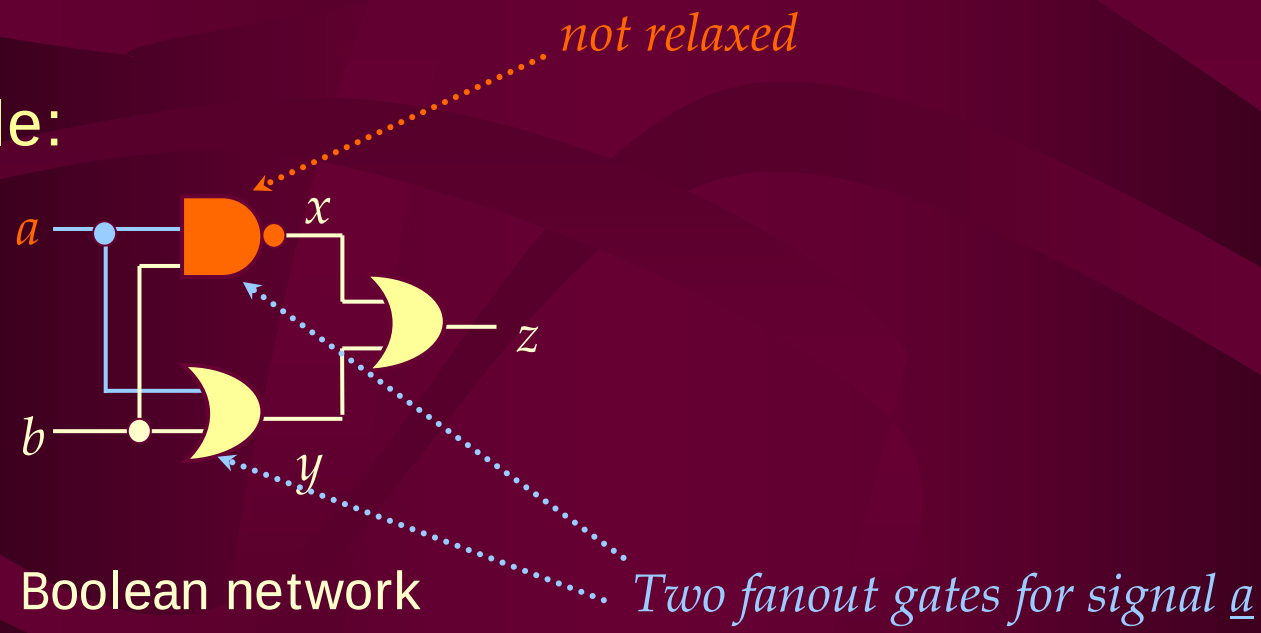
- Example:



Input Completeness Relaxation

- Theorem: *Dual-rail implementation of a Boolean network is timing-robust (i.e. gate-orphan-free) if and only if, for each signal, at least one of its fanout gates is not relaxed (i.e. ensures input completeness).*

- Example:



Only one of two fanout gates needs to be input-complete.

Outline

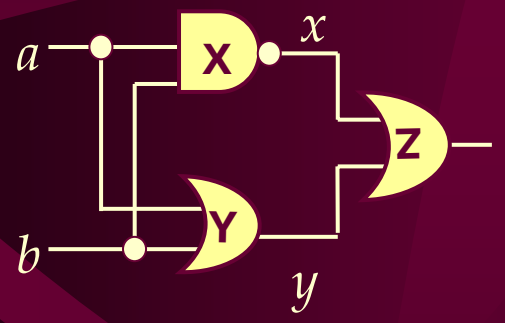
1. Introduction
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

Problem Definition

- The Input Completeness Relaxation Problem
  - Input: single-rail Boolean logic network
  - Output: relaxed dual-rail asynchronous circuit, which is still timing-robust
- Overview of the Proposed Algorithm
  - Relaxes overly-restrictive style of existing approaches
    - Performs selective relaxation of individual nodes
  - Targets three cost functions:
    - Number of relaxed-gates
    - Area after dual-rail expansion
    - Critical path delay
  - Based on *unate covering framework*:
    - Each gate output must be covered by at least one fanout gate,

Relaxation Algorithm

- Algorithm Sketch
  - Step 1: setup covering table
    - For each pair <*u*, *v*>, signal *u* fed into gate *v*:
      - Add *u* as a covered element (row)
      - Add *v* as a covering element (column)
  - Step 2: solve unate covering problem



Boolean network

gates

signals

	X	Y	Z
a	1	1	0
b	1	1	0
x	0	0	1
y	0	0	1

Covering table

Targeting Different Cost Functions

- Maximization of Number of Relaxed Gates
  - Weight of a gate = 1
- Minimization of Area of Dual-Rail Circuit
  - Weight of a gate = area penalty for not relaxing the gate
- Criticial Path Delay Optimization in Dual-rail Circuit
  - Find a critical path in non-relaxed dual-rail circuit
  - Assign higher weights to critical gates
  - Assign lower weights to non-critical gates
  - GOAL: more relaxation of critical path gates

Outline

1. Introduction
2. Background: Hazard Issues
3. Motivational Examples
4. Theorem on Input Completeness
5. Relaxation Algorithm
6. Experimental Results
7. Concluding Remarks

Experimental Results

- Results for DIMS-style asynchronous circuits

Original Boolean network		Unoptimized DIMS circuit			Optimization Run		
					# Relaxed nodes min.	Area min.	Delay opt.
name	#i/#o/#g	# full blocks	area	delay	# full blocks	area	delay
C1908	33/25/462	343	94532	30.0	180	58618	25.9
C3540	50/22/1147	911	281918	46.0	476	189612	38.7
C5315	178/123/1659	1259	335801	32.7	727	235391	28.5
C6288	32/32/3201	2385	567010	133.6	1246	361478	106.1
C7552	207/108/2155	1677	427101	44.8	1042	305203	43.4
dalu	75/16/756	633	201912	20.0	346	144288	14.8
des	256/245/2762	2329	712145	23.2	1157	462165	19.5
K2	45/43/684	597	222326	18.9	289	131498	14.0
t481	16/1/510	476	154466	20.8	211	99514	17.5
vda	17/39/383	309	121947	17.7	137	69231	15.7
Average percentage					51.8%	65.1%	83.9%

(selected MCNC combinational benchmarks)

Experimental Results

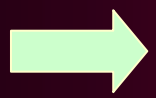
- Results for NCL asynchronous circuits
  - *(style used at Theseus Logic)*

Original Boolean network		NCL circuit			Optimization Run		
					# Relaxed nodes min.	Area min.	Delay opt.
name	#i/#o/#g	# full blocks	area	Delay	# full blocks	area	delay
C1908	33/25/462	343	55940	33.3	180	37917	28.3
C3540	50/22/1147	911	189970	51.0	476	147575	42.8
C5315	178/123/1659	1259	189370	36.4	727	154238	31.0
C6288	32/32/3201	2385	264750	151.1	1246	203490	123.0
C7552	207/108/2155	1677	224790	48.8	1042	180362	46.9
dalu	75/16/756	633	140190	21.7	346	113949	15.5
des	256/245/2762	2329	364812	24.8	1157	358692	20.9
K2	45/43/684	597	175590	20.2	289	108765	14.8
t481	16/1/510	476	109000	22.1	211	84655	17.7
vda	17/39/383	309	100230	19.0	137	60214	15.7
Average percentage					51.8%	74.1%	82.3%

(selected MCNC combinational benchmarks)

Experimental Results

- Minimizing Number of Relaxed Nodes:
  - DIMS circuits: 48.2% relaxed
  - NCL circuits: 48.2% relaxed
- Area minimization:
  - DIMS circuits: 34.9% improvement
  - NCL circuits: 25.9% improvement
- Critical Path Delay optimization:
  - DIMS circuits: 16.1% improvement
  - NCL circuits: 17.7% improvement



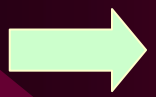
No change to overall timing-robustness of circuits

Related Work

- Two approaches to relaxation
  - Relax every gate but add local completion detectors
  - Relax selectively with no additional circuitry
- Approach #1: Relax every gate
  - David et al. *An Efficient Implementation of Boolean Circuits as Self-Timed Circuits*, IEEE Trans. Computers 1992
  - (+) More relaxation opportunities
  - (-) Overhead of additional circuitry to ensure robustness
- Second approach: Selective relaxation
  - Smith et al. *Optimization of Null Convention Self-Timed Circuits*, Integration 2004
  - (+) No overhead of additional circuitry
  - (-) No general relaxation algorithms
  - Zhou et al. *Cost-aware Synthesis of Asynchronous Circuits Based on Partial Acknowledgement*, ICCAD 2006
    - Related approach (developed in parallel independently)

Conclusion

- Input Completeness Relaxation Algorithm
  - Optimization technique for robust "asynchronous" circuits
  - Relaxes overly-restrictive style of existing approaches
  - Can target three different cost functions:
    - # Relaxed nodes, area, critical path delay
  - CAD tool developed
  - Significant improvement:
    - Average relaxation of 48.2% of gates
    - Average area improvement of up to 34.9%
    - Average delay improvement of up to 17.7%



No change to overall timing-robustness of circuits

- Future Work
  - More sophisticated delay optimization scheme
  - More fine-grained relaxation schemes: e.g. independent consideration of set/reset phases