Optimization of Robust Asynchronous Threshold Networks Using Local Relaxation Techniques

Cheoljoo Jeong*   Steven M. Nowick

Computer Science Department
Columbia University

*[now at Cadence Design Systems]

Outline

Recent Challenges in Microelectronics Design

- Reliability challenge
  - Variability issues in deep submicron technology
    - process, temperature, voltage
    - noise, crosstalk
  - Dynamic voltage scaling
- Communication challenge
  - Increasing disparity between gate and wire delay
- Productivity challenge
  - Increasing system complexity + heterogeneity
  - Shrinking time to market, timing closure issues
  - Even when IP blocks are used, interface timing verification is difficult

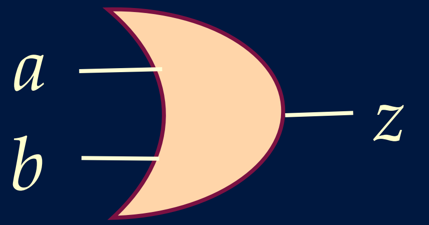
Benefits and Challenges of Asynchronous Circuits

- Potential benefits:
  - Mitigates timing closure problem
  - Low power consumption
  - Low electromagnetic interference (EMI)
  - Modularity, “plug-and-play” composition
  - Accommodates timing variability

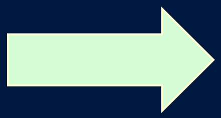
- Challenges:
  - Robust design is required: hazard-freedom
  - Area overhead (sometimes)
  - Lack of CAD tools
  - Lack of systematic optimization techniques

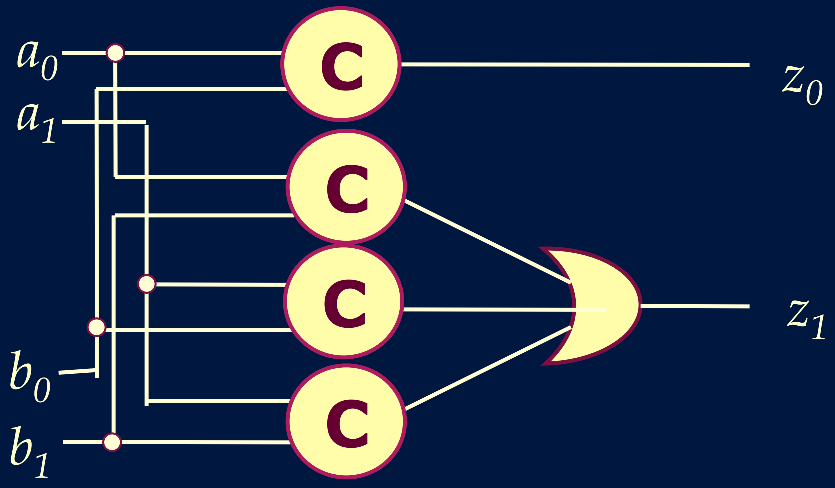
Asynchronous Threshold Networks

- Asynchronous threshold networks
  - One of the most robust asynchronous circuit styles
  - Based on *delay-insensitive encoding*
    - Communication: robust to arbitrary delays
    - Logic block design: imposes very weak timing constraints (1-sided)
- Simple example: OR2



Boolean OR2 gate





Async dual-rail threshold network for OR2

Asynchronous Threshold Networks in Emerging Technologies

- Ultra-Low Voltage (ULV): extreme variability
  - 8051 microcontroller - extreme PVT variability at subthreshold voltages
  - K.-L. Chang et al., "Synchronous-Logic and Asynchronous-Logic 8051 Microcontroller Cores for Realizing the Internet of Things: a Comparative Study on Dynamic Voltage Scaling and Variation Effects," IEEE J. Emerg. Sel. Topics Circuits Systems., vol. 3:1, 2013, pp. 23-34.
- Space Applications: extreme environments
  - 8-bit data transfer system for space flights
  - Fully operational over 400^0 C temperature range (-175^0 to $+225^0$ C)
  - P. Shepherd et al., "A Robust, Wide-Temperature Data Transmission System for Space Environments," Proc. IEEE Aerospace Conf. (AERO), 2013, pp. 1812-1819.
- Nano-Magnetic Logic Circuits:
  - M. Vacca, M. Graziano and M. Zamboni, "Asynchronous Solutions for Nano-Magnetic Logic Circuits," ACM Journal on Emerging Technologies in Computing Systems (JETC), vo. 7:4, 15:1-15:18 (2011).
- Quantum-Dot Cellular Automata (QCA):
  - M. Choi et al., "Efficient and Robust Delay-Insensitive QCA (Quantum-Dot Cellular Automata) Design," Proc. IEEE Int. Symp. Defect and Fault-Tolerance in VLSY Systems (DFT), 2006, pp. 80-88.

Challenges and Overall Research Goals

- Challenges in asynchronous threshold network synthesis
  - Large area and latency overheads
  - Few existing optimization techniques
  - Even less support for CAD tools

- Overall Research Agenda:
  - Develop systematic optimization techniques and CAD tools for highly-robust asynchronous threshold networks
  - Support design-space exploration: automated scripts, target different cost functions
  - Current optimization targets: area + delay + delay-area tradeoffs
  - Future extensions: power (straightforward)

Overall Research Goals

Two automated optimization techniques proposed

1. Relaxation algorithms: multi-level optimization
   - Existing synthesis approaches are conservative = over-designed
   - Approach: selective use of eager-evaluation logic
     - without affecting overall circuit's timing robustness
   - Can apply at two granularities:
     - gate-level [Jeong/Nowick ASPDAC-07, Zhou/Sokolov/Yakovlev ICCAD-06]
     - block-level [Jeong/Nowick Async-08]

C. Jeong and S.M. Nowick, "Optimization of Robust Asynchronous Circuits by Local Input Completeness Relaxation," Proc. Of IEEE Asia and South Pacific Design Automation Conference (ASPDAC), 2007.

C. Jeong and S.M. Nowick, "Block-Level Relaxation for Timing-Robust Asynchronous Circuits Based on Eager Evaluation," Proc. Of IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC), 2008.

Overall Research Goals (cont.)

2. Technology mapping algorithms

- First general and systematic technology mapping for robust asynchronous threshold networks
- Evaluated on substantial benchmarks:
  - > 10,000 gates, > 1000 inputs/outputs
- Use fully-characterized industrial cell library (Theseus Logic):
  - slew rate, loading, distinct i-to-o paths/rise vs. fall transitions
- Significant average improvements:
  - Delay: 31.6%, Area: 9.5% (runtime: 6.2 sec)

C. Jeong and S.M. Nowick, "Technology Mapping and Cell Merger for Asynchronous Threshold Networks," IEEE Trans. on Computer-Aided Design (TCAD), vol. 27:4, pp. 659-672 (April 2008).

"ATN_OPT" CAD Package: implements both steps
downloadable (for Linux) + tutorials/benchmarks
URL: http://www.cs.columbia.edu/~nowick/asynctools

Basic Synthesis Flow
(Theseus Logic/Camgian Networks)

Single-rail Boolean network ⟵⋯⋯⋯ *Considered as abstract multi-valued circuit*

↓ simple dual-rail expansion (delay-insensitive encoding)

Dual-rail async threshold network ⟵⋯⋯⋯ *Instantiated Boolean circuit (robust, unoptimized)*

New Optimized Synthesis Flow

Single-rail Boolean network

$\Downarrow$ Relaxation
(i.e. relaxed dual-rail expansion)

"Relaxed" dual-rail async threshold network $\longleftarrow\cdots\cdots$ *optimized*

$\Downarrow$ Technology mapping

Optimally-mapped dual-rail async threshold network $\longleftarrow\cdots\cdots$ *optimized*

New Optimized Synthesis Flow

Focus of this talk

Single-rail Boolean network

Relaxation
(i.e. relaxed dual-rail expansion)

"Relaxed" dual-rail async threshold network ←·········· *optimized*

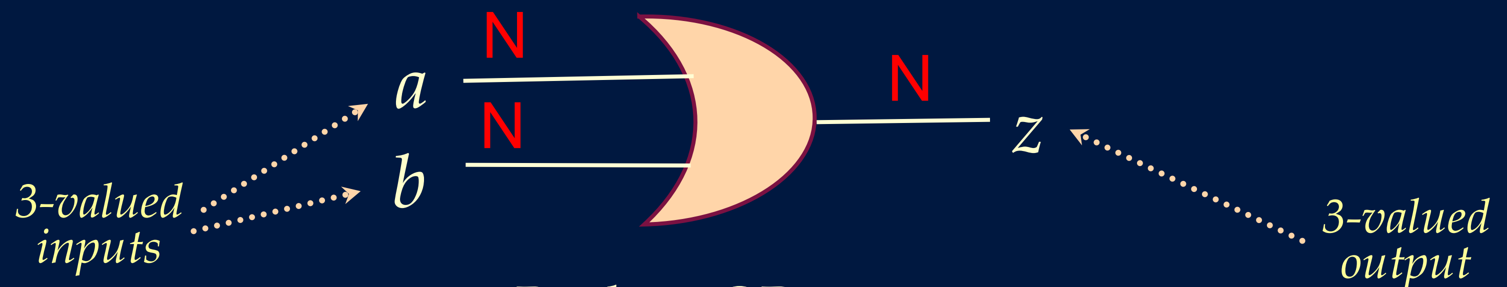
Technology mapping

Optimally-mapped dual-rail async threshold network ←·········· *optimized*

Outline

Single-Rail Boolean Networks

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

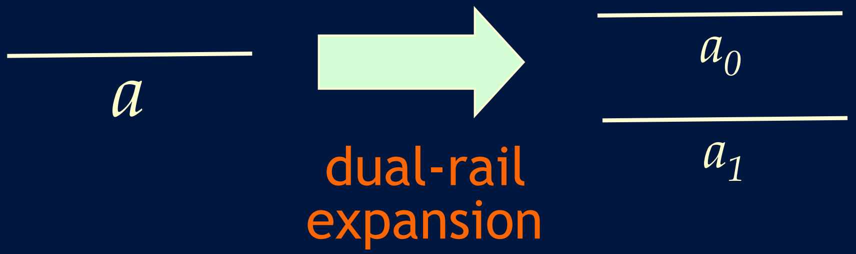


Boolean OR gate

  - 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
  - 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

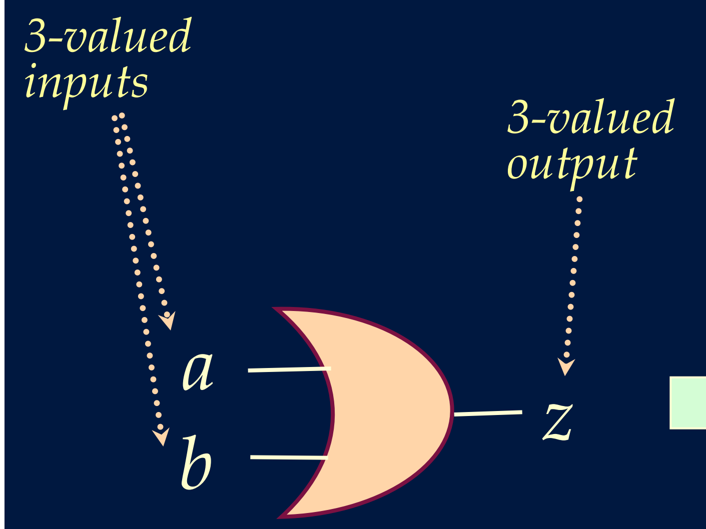
invalid

Encoding table

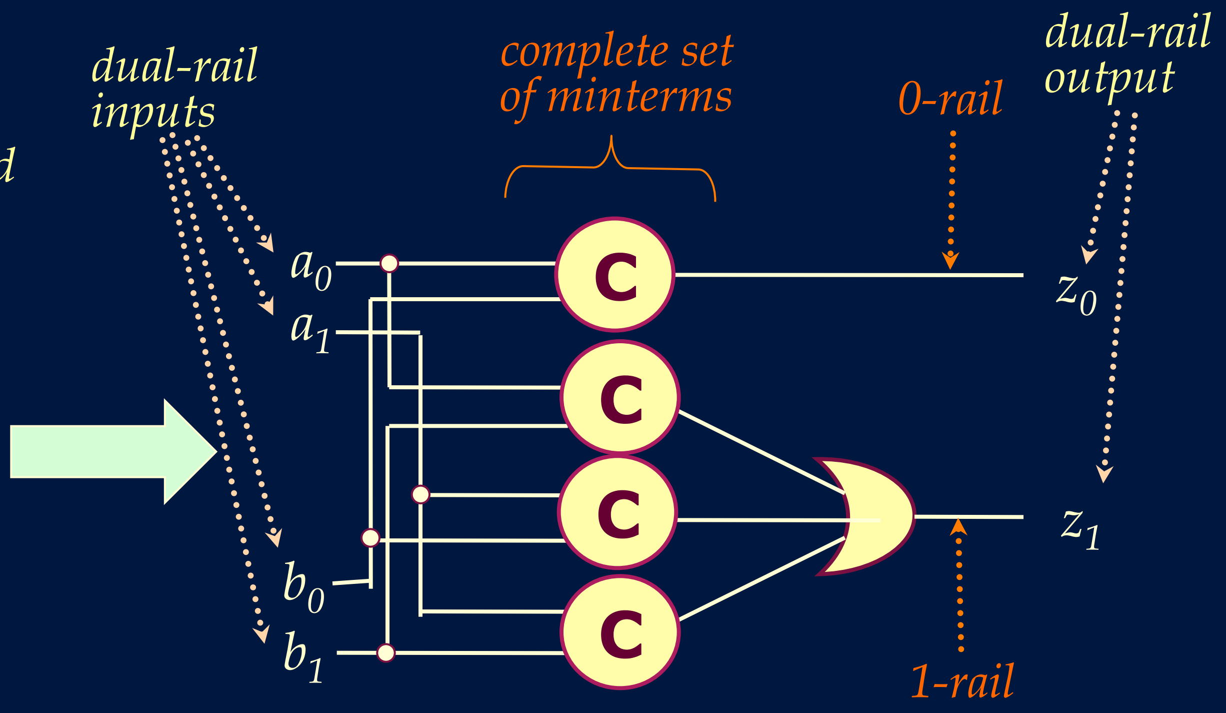
- Motivation: robust data communication

Dual-Rail Asynchronous Circuits

- DIMS-Style Dual-Rail Expansion:
  - "delay-insensitive minterm synthesis" style
  - Single Boolean gate: expanded into 2-level network



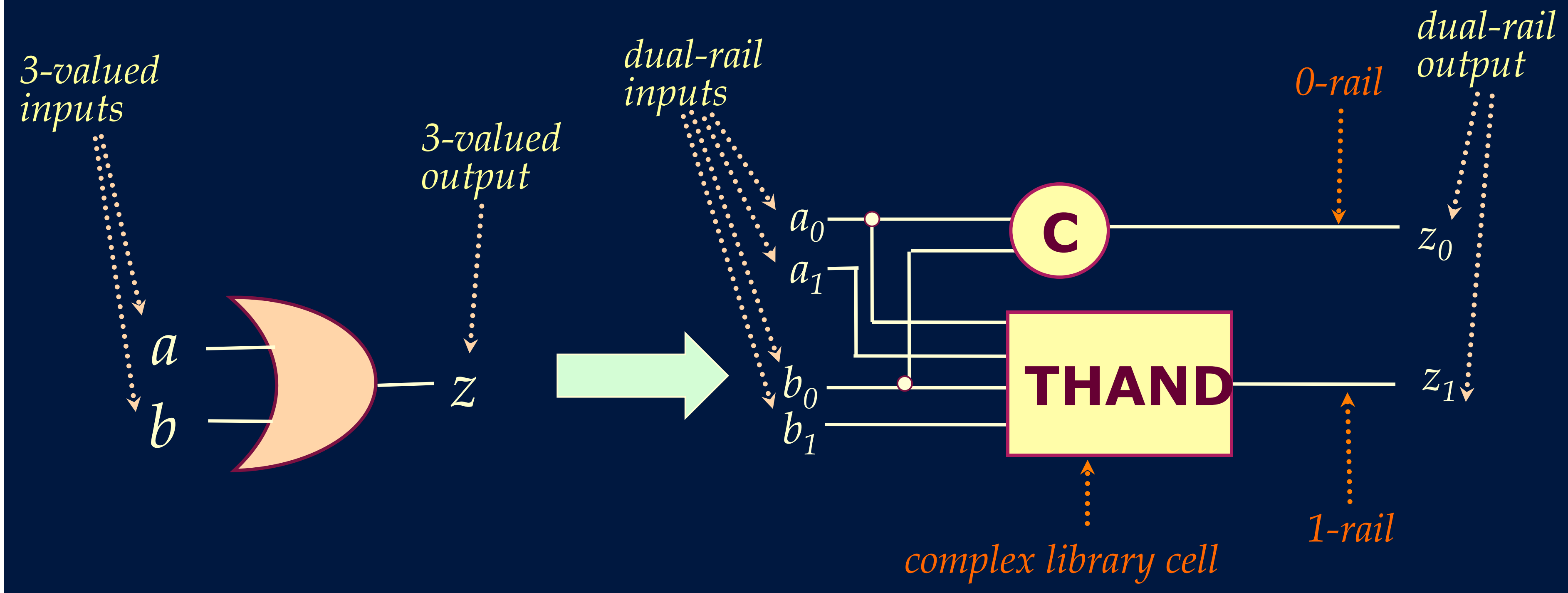
Boolean OR gate



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)

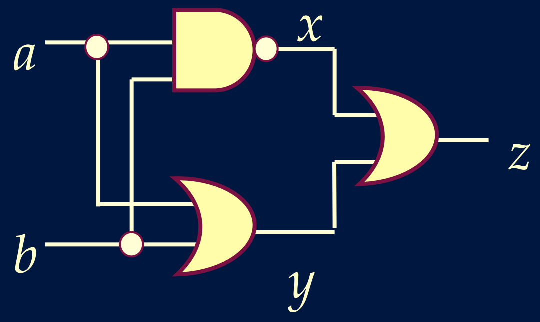


Boolean OR gate

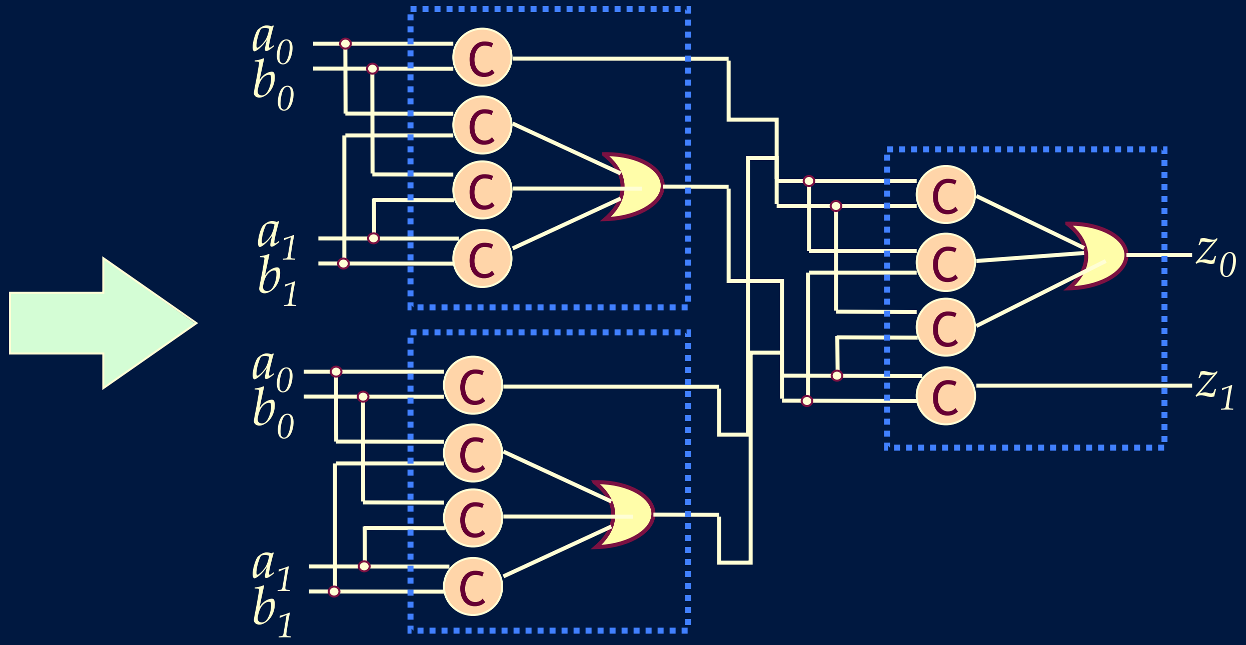
NCL-style dual-rail OR circuit

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
- End point: unoptimized dual-rail asynchronous threshold network
- Result: timing-robust asynchronous netlist



Boolean logic network



Dual-rail asynchronous threshold network

Hazard Issues

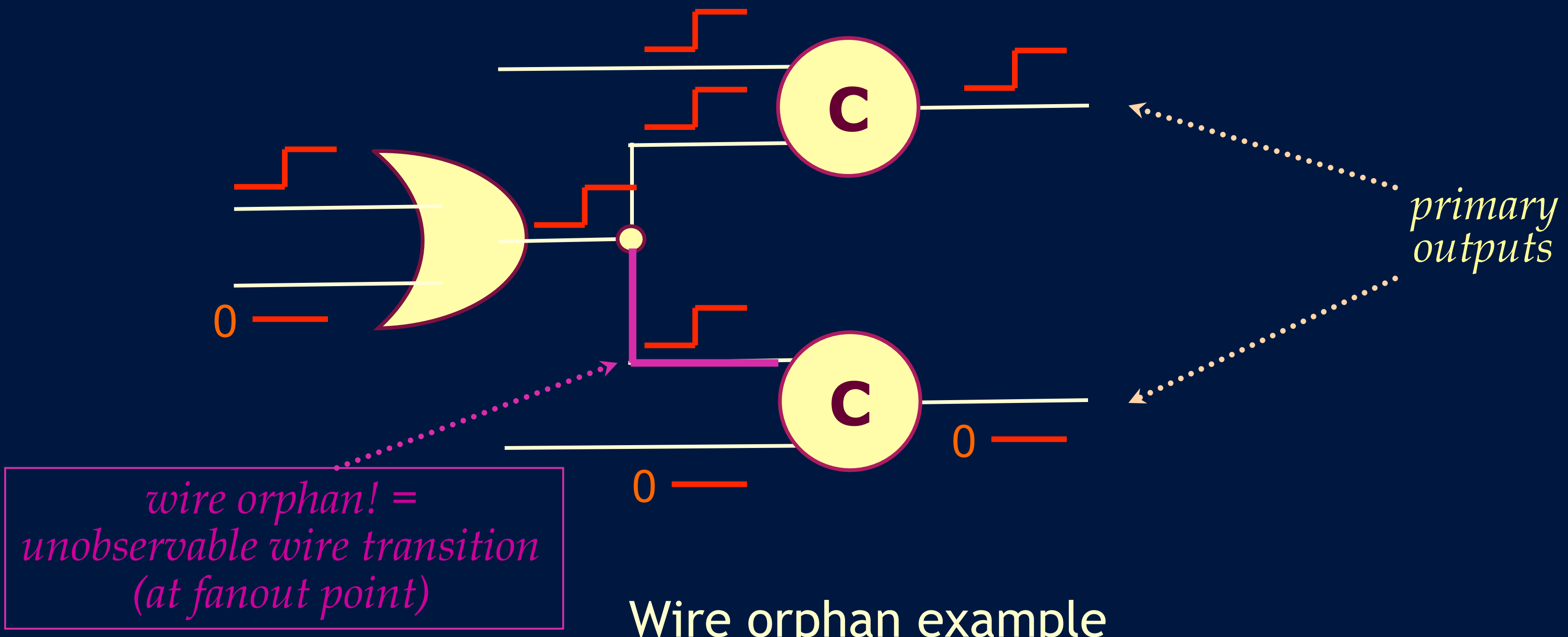
- Ideal Goal = Delay-Insensitivity (delay model)
  - Allows arbitrary gate and wire delay
    - circuit operates correctly under all conditions
  - Most robust design style
    - when circuit produces new output, all gates stable

      = "timing robustness"

- "Orphans" = hazards to delay-insensitivity
  - "unobservable" *signal transition sequences*
  - *Wire orphans*: unobservable wires at fanout
  - *Gate orphans*: unobservable paths at fanout

Hazard Issues

- Wire orphan example:

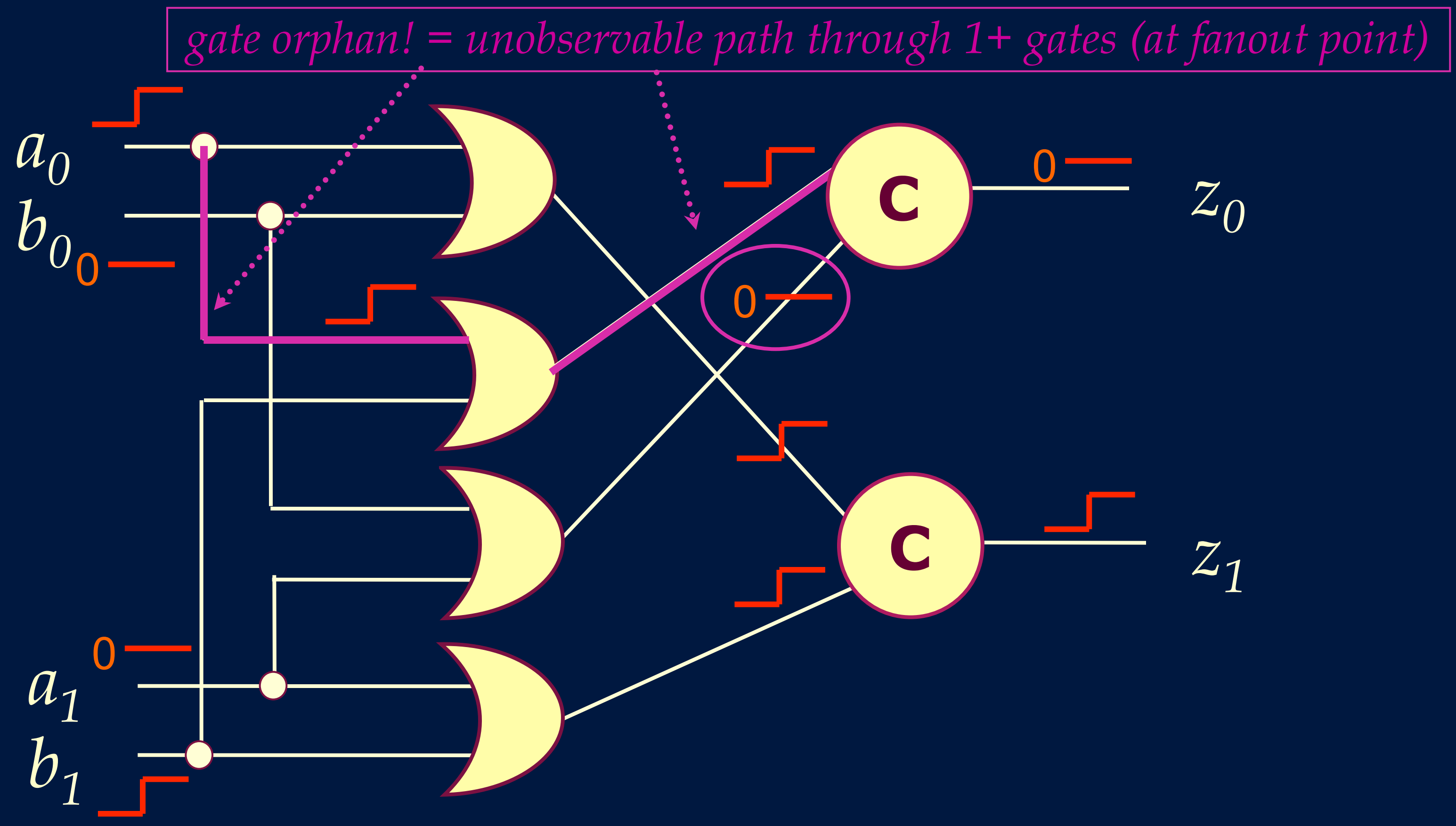

C
C
0
0
0
primary outputs
wire orphan! = unobservable wire transition (at fanout point)

Wire orphan example

If unobservable wire too slow, will interfere with next data item (glitch)

Hazard Issues

- Gate orphan example:



gate orphan! = unobservable path through 1+ gates (at fanout point)
a_0
b_0
0
a_1
b_1
0
0
0
C
C
z_0
z_1

Gate orphan example

If unobservable path too slow, will interfere with next data item (glitch)

Hazard Issues: Summary

- Wire orphans: typically not a problem in practice
  - unobserved signal transition on wire (at fanout point)
  - Solution: handle during physical synthesis (e.g. Theseus Logic)
    - enforce simple 1-sided timing constraint:
      - similar to "quasi-delay-insensitivity" (QDI)
- Gate orphans: difficult to handle
  - unobserved signal transition on path (at fanout point)
  - can result in unexpected glitches: if delays too long
  - harder to overcome with physical design tools

invariant of the proposed optimization algorithms:
ensure no gate orphans introduced

Outline

1. Introduction
2. Background: Asynchronous Threshold Networks
3. Gate-Level Relaxation
4. Block-Level Relaxation
5. Experimental Results
6. Conclusions and Future Work

Overview of Relaxation

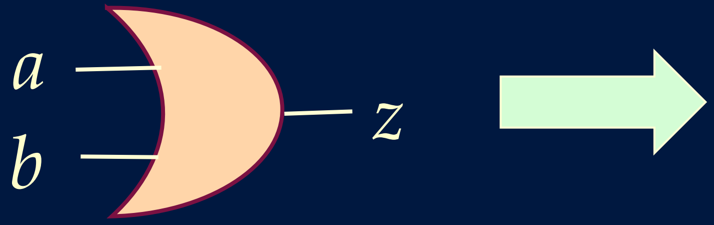
- Relaxation: Multi-level optimization
  - Allows more efficient dual-rail expansion using eager-evaluating logic
  - Idea: *selectively replace* some gates by eager blocks
    - either at *gate-level* or *block-level*
  - Advantage: if carefully performed, *no* loss of overall circuit robustness

- Proposed flow

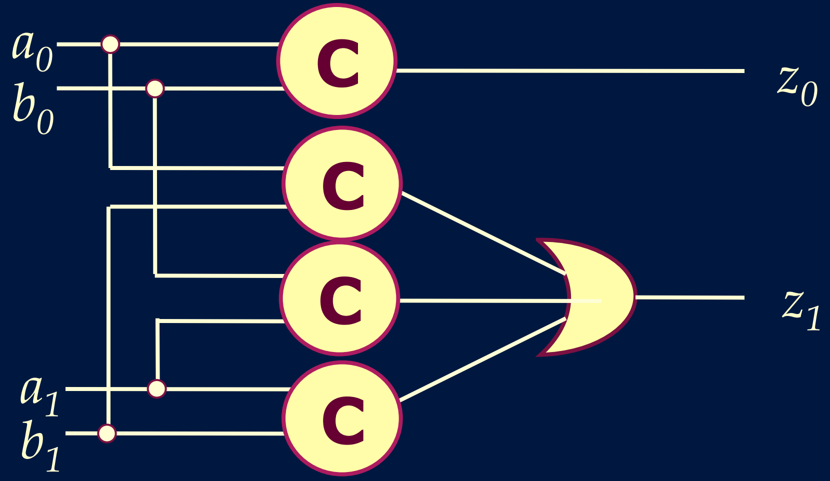
$$\begin{tikzcd}
\text{Single-rail Boolean network} \arrow[d, "\text{Relaxation}"] & \\
\text{Relaxed dual-rail async threshold network} & \textit{optimized} \arrow[l, dotted]
\end{tikzcd}$$

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



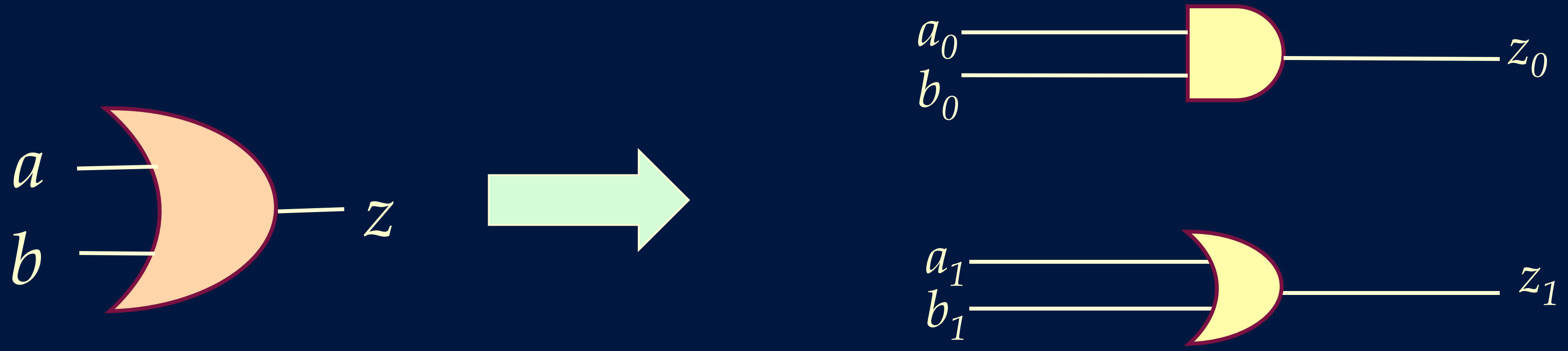
Input-complete dual-rail OR network

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

Enforcing input completeness for every gate is the traditional synthesis approach to avoid hazards (i.e. gate orphans).

Input Incompleteness

- A dual-rail implementation of a Boolean gate is input-incomplete w.r.t. its input signals ("eager-evaluating"), if the output can change *before* all inputs arrive.

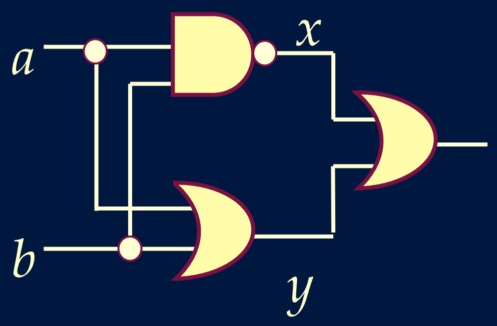


Boolean OR gate

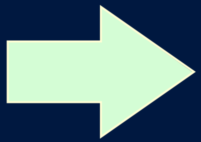
Input-incomplete dual-rail OR network

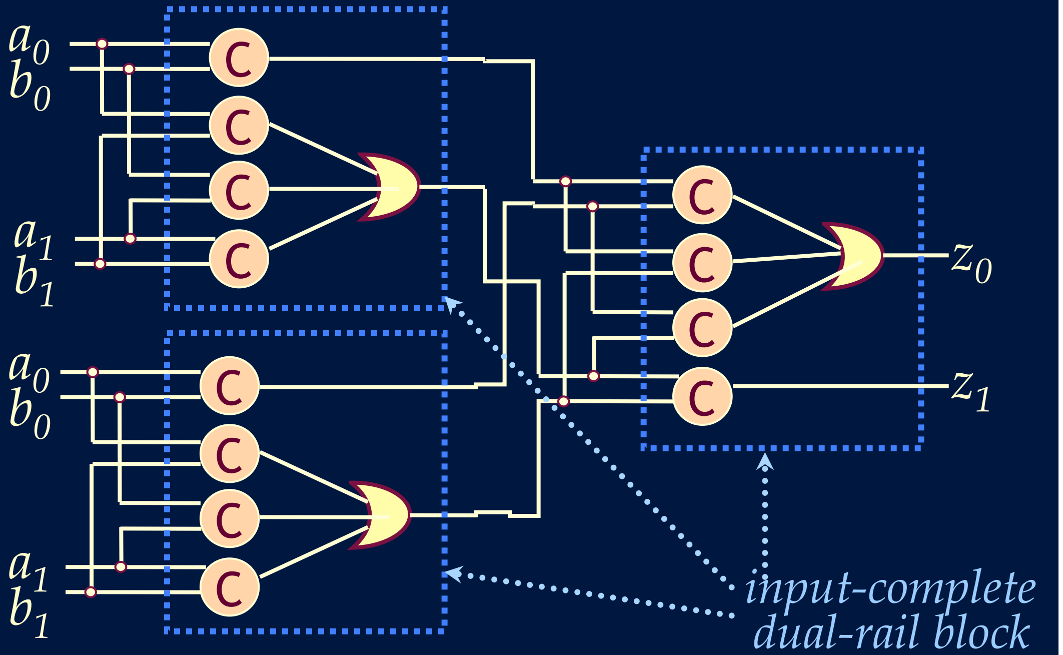
Gate-Level Relaxation Example #1

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



Boolean network

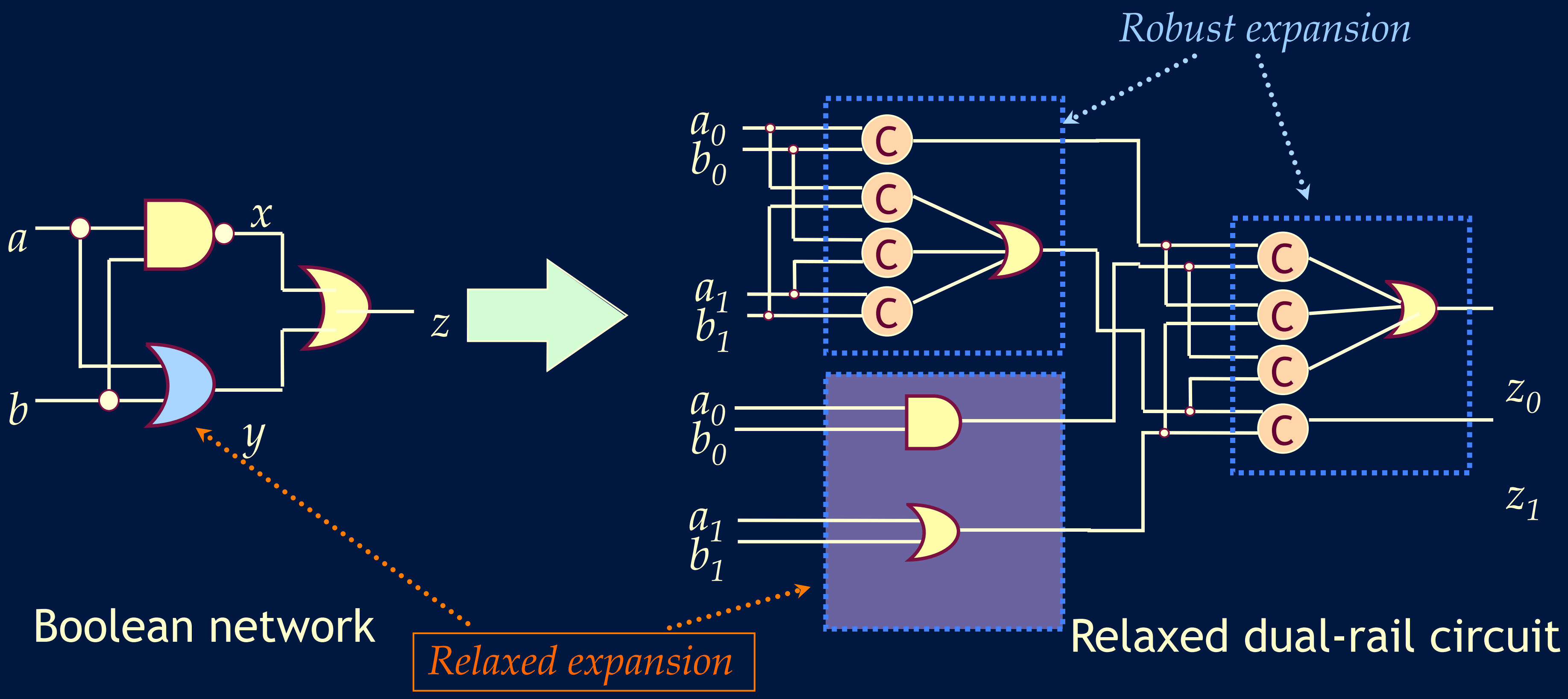




Dual-rail circuit with full expansion (no relaxation)

Gate-Level Relaxation Example #1 (cont.)

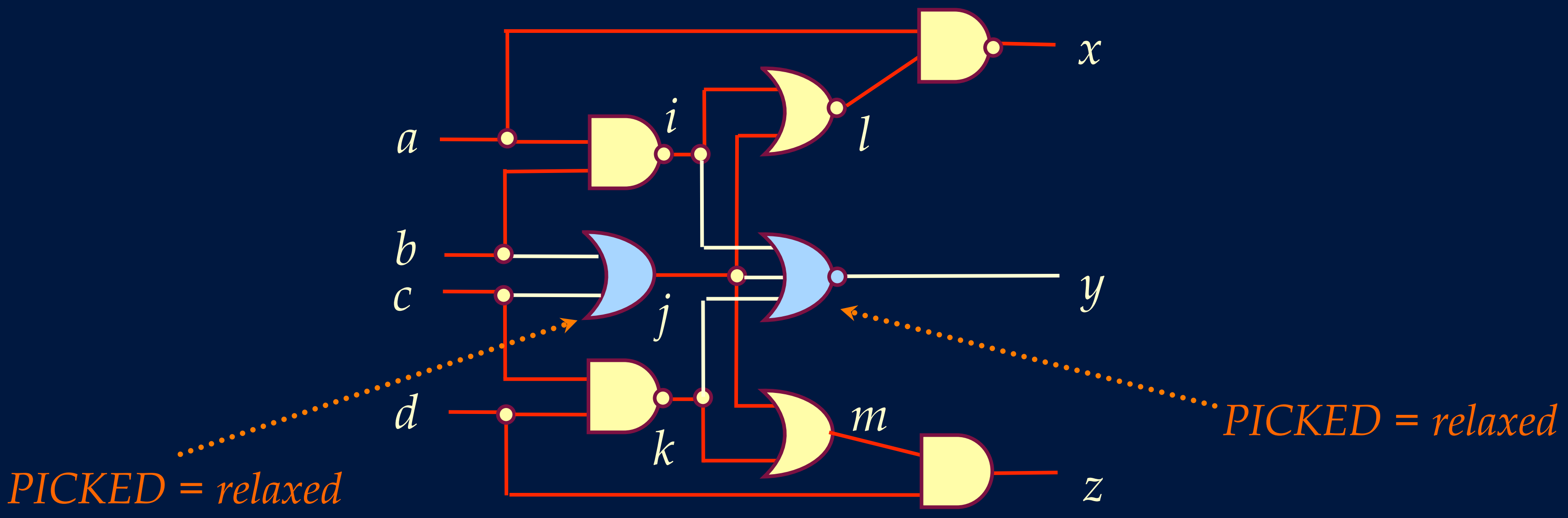
- Not every Boolean gate needs to be expanded into input-complete block.



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

Gate-Level Relaxation Example #2

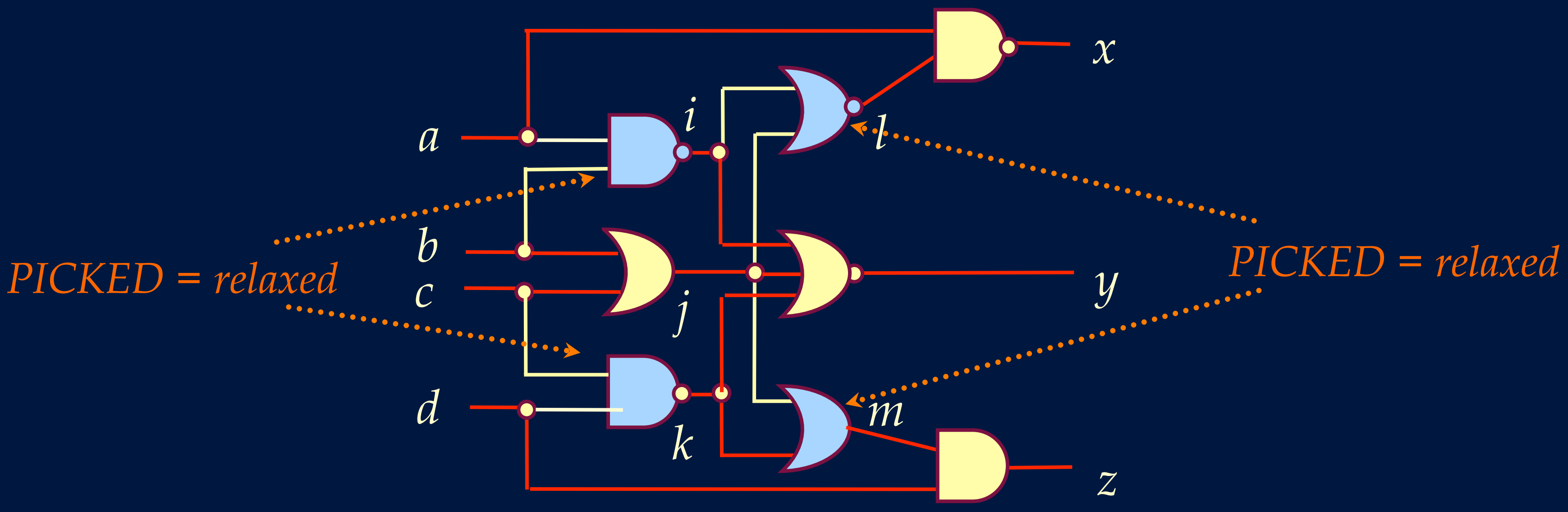
- Different choices may exist in relaxation.



Relaxation of Boolean network with *two* relaxed gates

Gate-Level Relaxation Example #2 (cont.)

- Different choices may exist in relaxation.



Relaxation of Boolean network with *__four__* relaxed gates

Gate-Level Relaxation: Summary

- Conservative approach:
  - Every path from a gate to a primary output must contain only robust (input-complete) gates

- Optimized approach: [Nowick/Jeong ASPDAC-07, Zhou/Sokolov/Yakovlev ICCAD-06]
  - At least one path from each gate to some primary output must contain only robust (i.e. input-complete) gates (Theorem)
  - … all other gates can be safely 'relaxed' (I.e. input-incomplete)

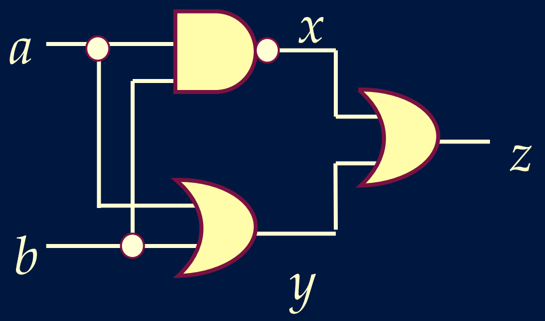
Resulting implementation has no loss of timing robustness (remains "gate-orphan-free")

Which Gates Can Safely Be Relaxed?

- Localized theorem: gate relaxation [Jeong/Nowick ASPDAC-07]

  A 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 input-complete (I.e. not relaxed).

- Example:



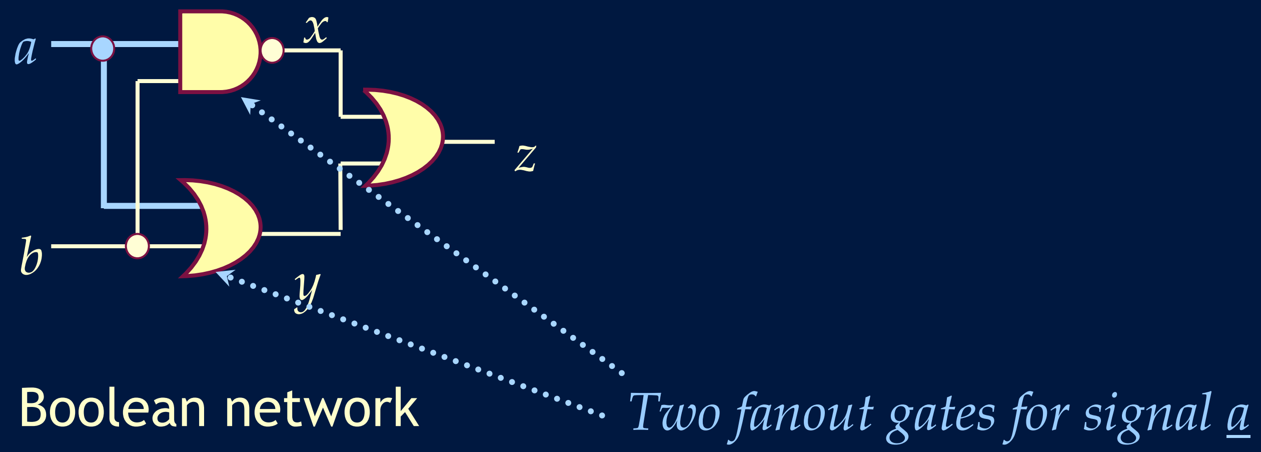
Boolean network

Which Gates Can Safely Be Relaxed?

- Localized theorem: gate relaxation [Jeong/Nowick ASPDAC-07]

  A 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 input-complete (i.e. not relaxed).

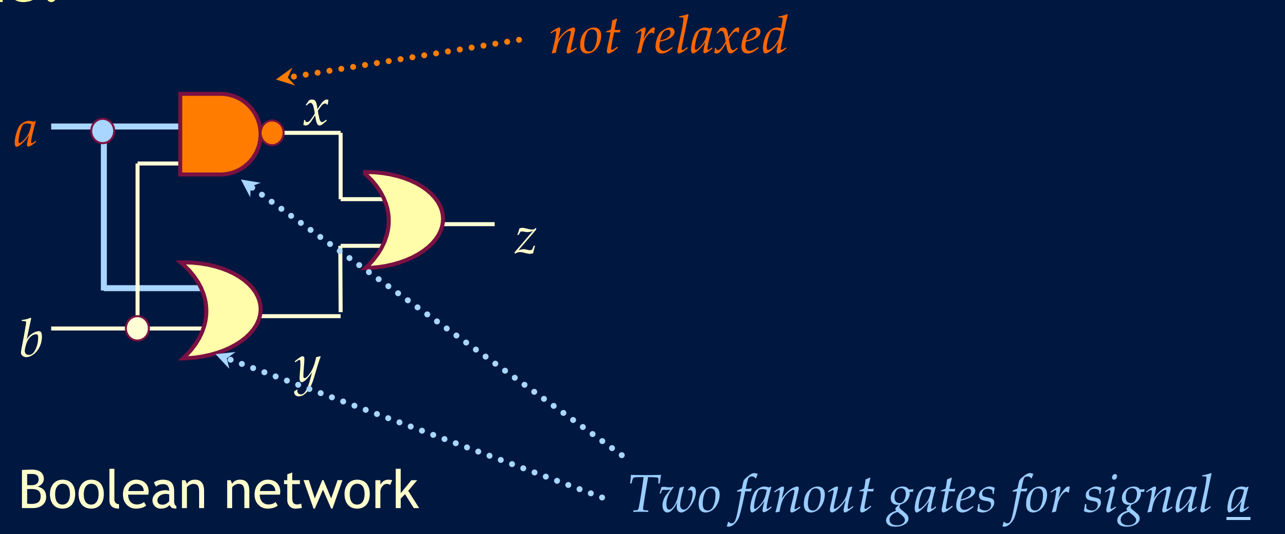
- Example:



Boolean network

Which Gates Can Safely Be Relaxed?

- Localized theorem: [Jeong/Nowick ASPDAC-07]
  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 input complete (I.e. not relaxed).
- Example:



Boolean network

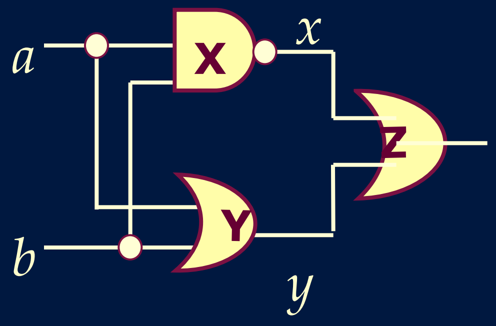
Only one of two fanout gates must be input-complete.

Problem Definition

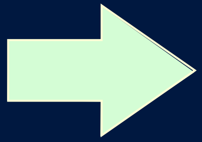
- The Input Completeness Relaxation Problem
  - Input: single-rail Boolean logic network
  - Output: relaxed dual-rail asynchronous circuit, which is still timing-robust
- Overview of the Proposed Algorithm
  - Relaxes overly-restrictive style of existing approaches
    - Performs selective relaxation 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
  - Step 3: generate dual-rail threshold network



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 <u>not relaxing</u> 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: <u>more relaxation of critical path gates</u>

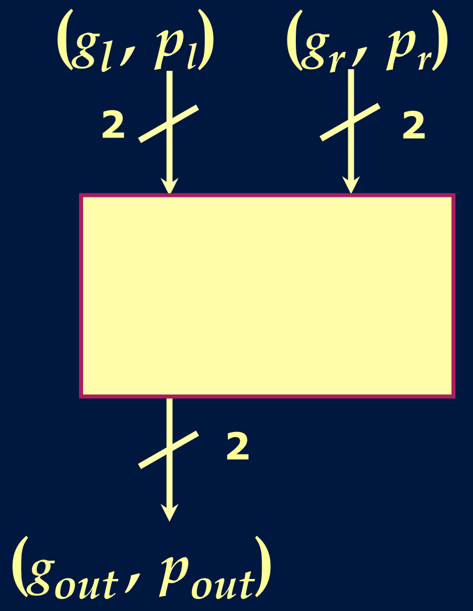
Outline

1. Introduction
2. Background: Asynchronous Threshold Networks
3. Gate-Level Relaxation
4. Block-Level Relaxation
5. Experimental Results
6. Conclusions and Future Work

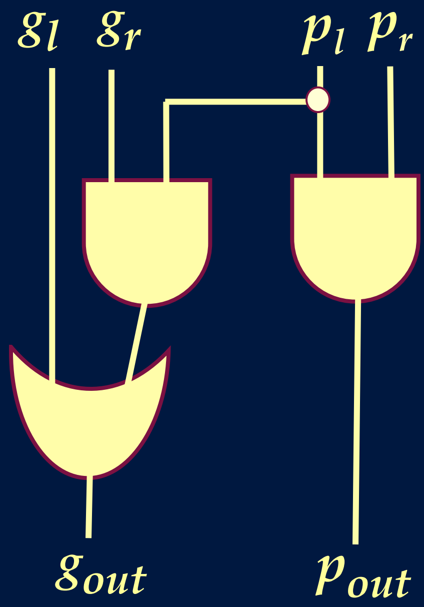
Block-Level Relaxation

- Block-level vs. Gate-level circuits

Block-level circuit	Gate-level circuit
Consists of large granularity blocks	Consists of simple gates
Blocks have *multiple* outputs	Gates have *single* output



P/G block in prefix adders



Gate-level implementation of P/G block

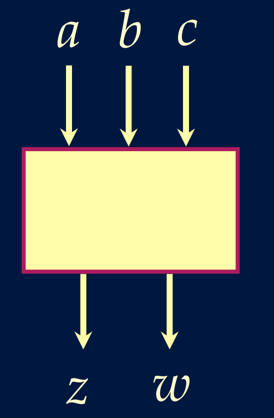
Why Relaxation at Block-Level?

- Like gate-level relaxation: blocks are either
  - input complete: wait for all inputs to arrive
  - relaxed: eager, do not wait for all inputs to arrive
- New idea: 3rd possibility
  - "partially-eager":
    - input complete: each input vector acknowledged on *some output*
    - partially-eager: allows some outputs to fire early

Block-Level Relaxation Example

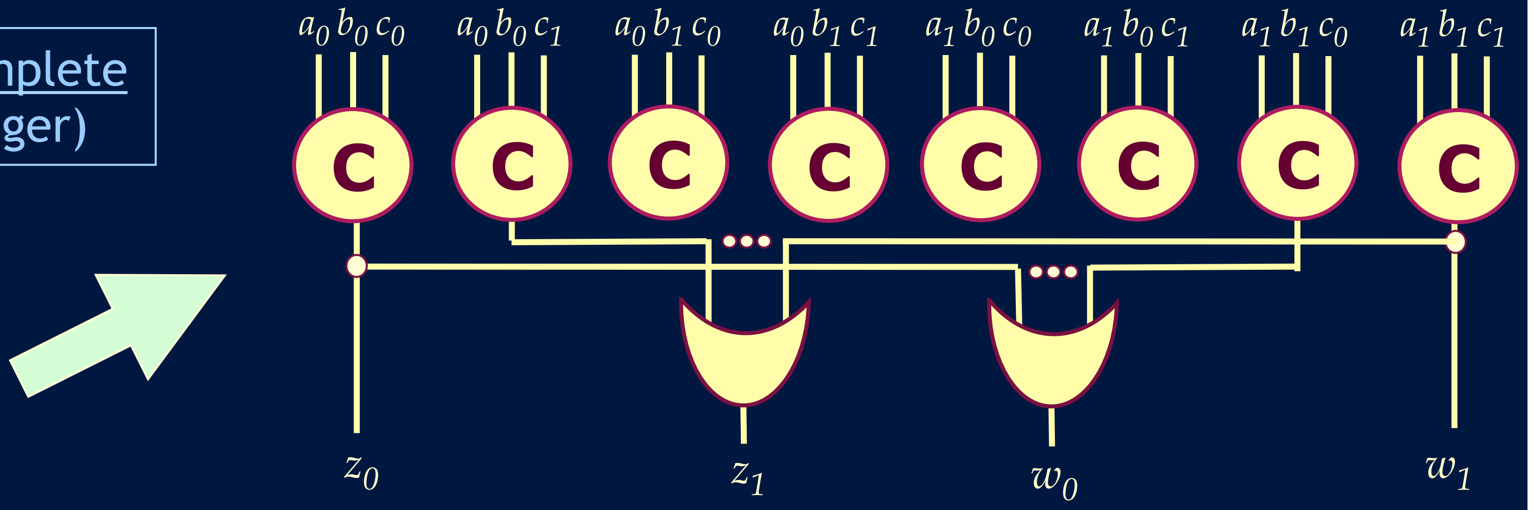
- Basic approach = direct extension of gate-level relaxation
  - No output in robust block fires before all inputs arrive

Input-complete (non-eager)



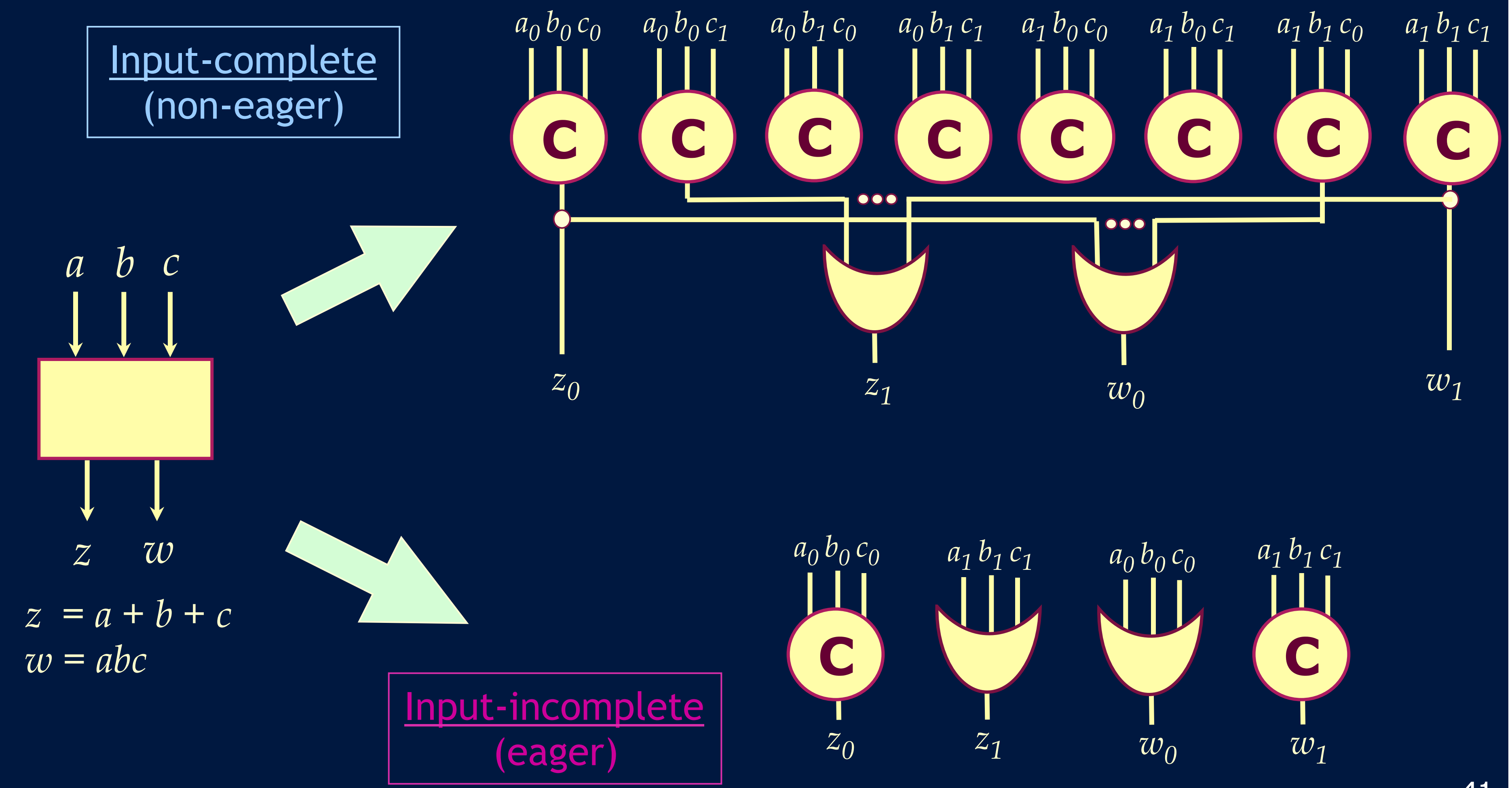
$z = a + b + c$
$w = abc$

Block example



Block-Level Relaxation Example

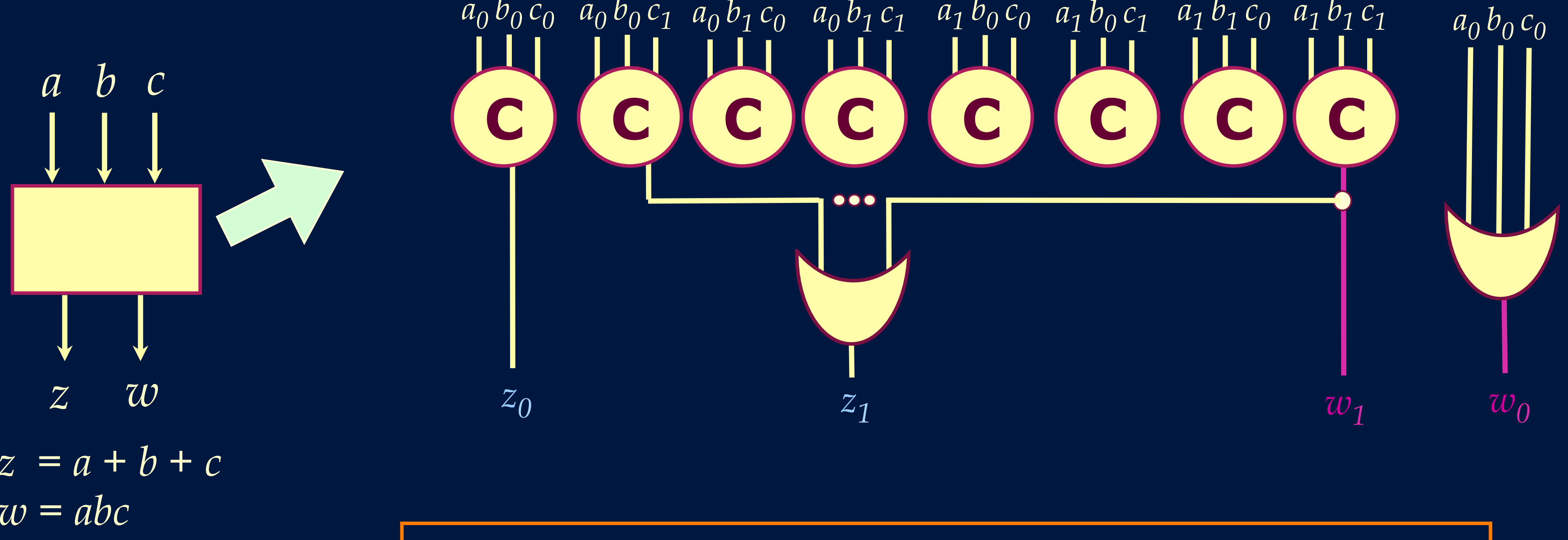
- Basic approach = direct extension of gate-level relaxation
  - No output in robust block fires before all inputs arrive



Block-Level Relaxation Example

- New Option #1: "Biased Approach"
  - In biased implementation of blocks, only one output is implemented in a robust way; other outputs are eager-evaluating

Input-complete block (and partially eager!)



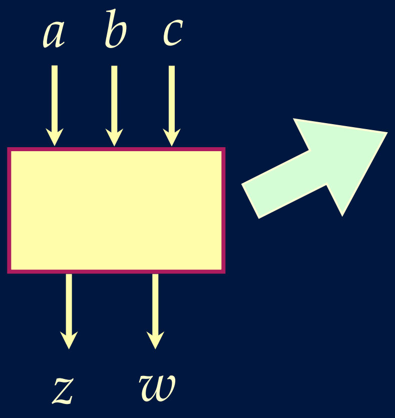
Block example

Output z: waits for all inputs ("non-eager")
Output w: early evaluating ("eager")

Block-Level Relaxation Example

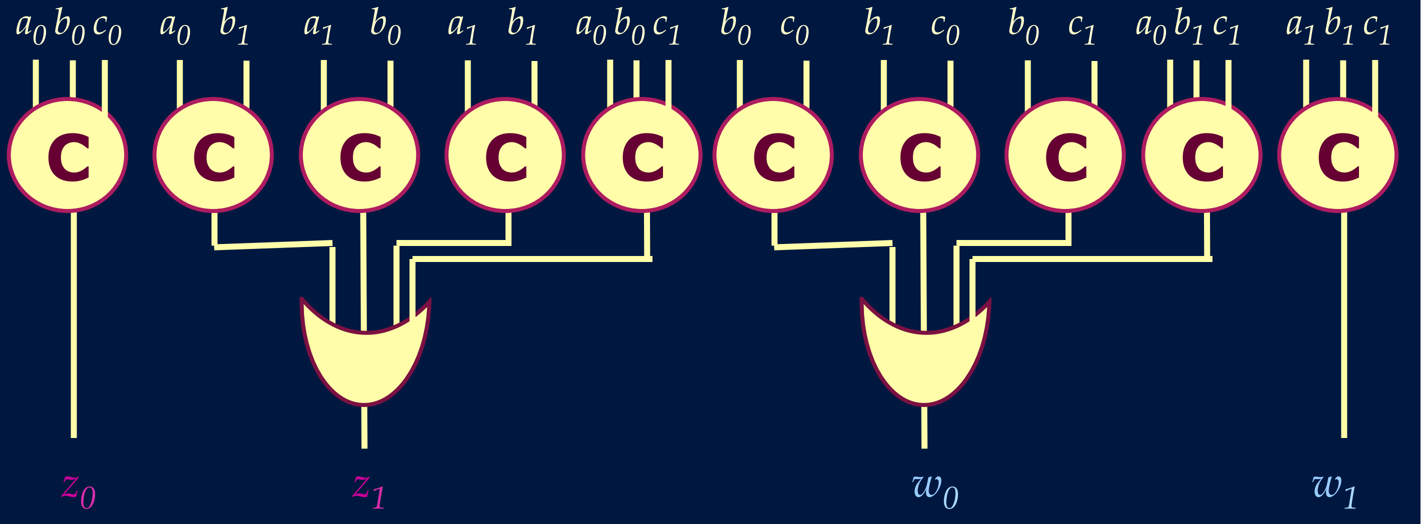
- New Option #2: "Distributive Approach"
  - outputs jointly share responsibility to detect arrival of all input vectors
  - each block output: also partially "eager"!

Input-complete block
(and partially eager!)



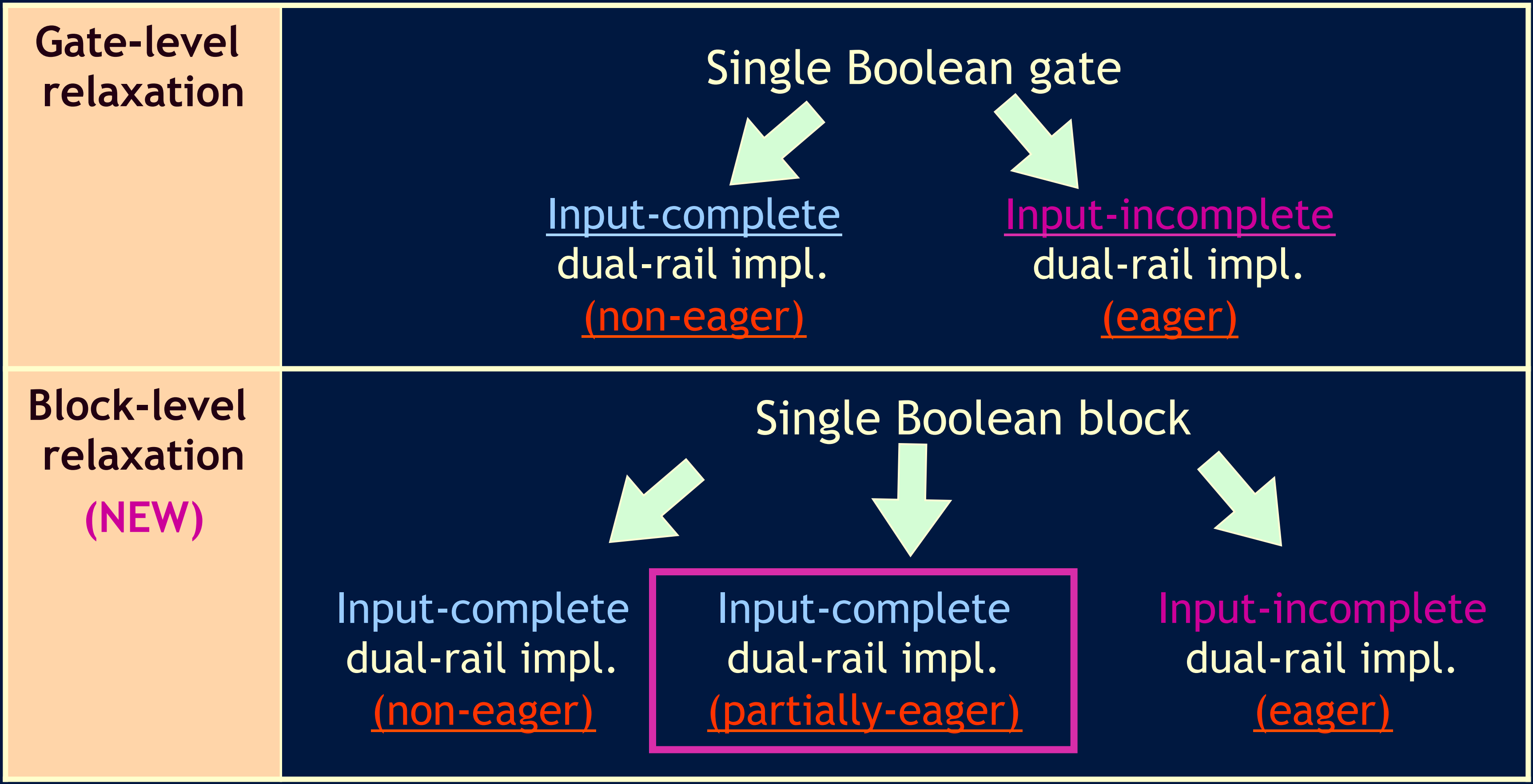
$z = a + b + c$
$w = abc$

Block example



Output z: waits for inputs a/b (otherwise eager)
Output w: waits for inputs b/c (otherwise eager)

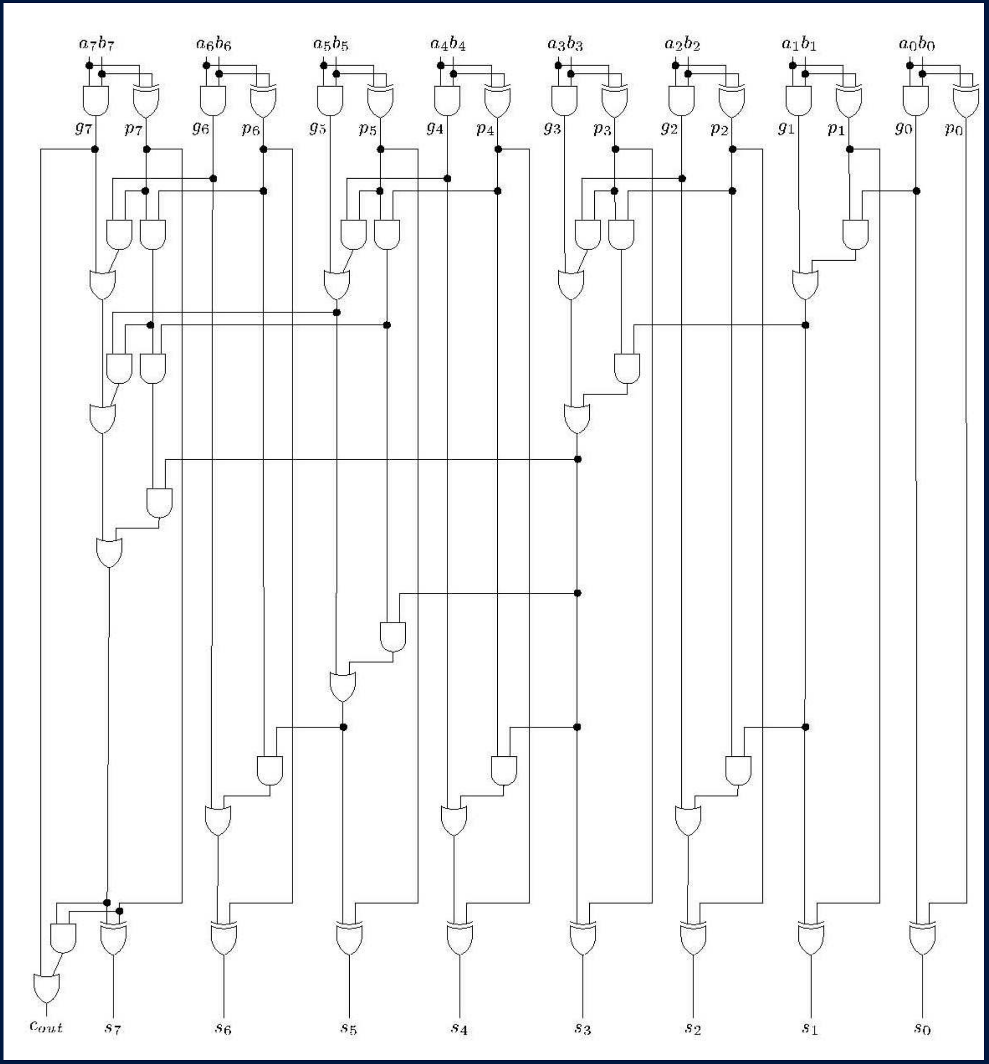
Summary: Why Relaxation at Block-Level?


Gate-level relaxation
Single Boolean gate
Input-complete dual-rail impl. (non-eager)
Input-incomplete dual-rail impl. (eager)
Block-level relaxation (NEW)
Single Boolean block
Input-complete dual-rail impl. (non-eager)
Input-complete dual-rail impl. (partially-eager)
Input-incomplete dual-rail impl. (eager)

More optimization opportunities + larger design space

Block- vs Gate-Level Relaxation Example

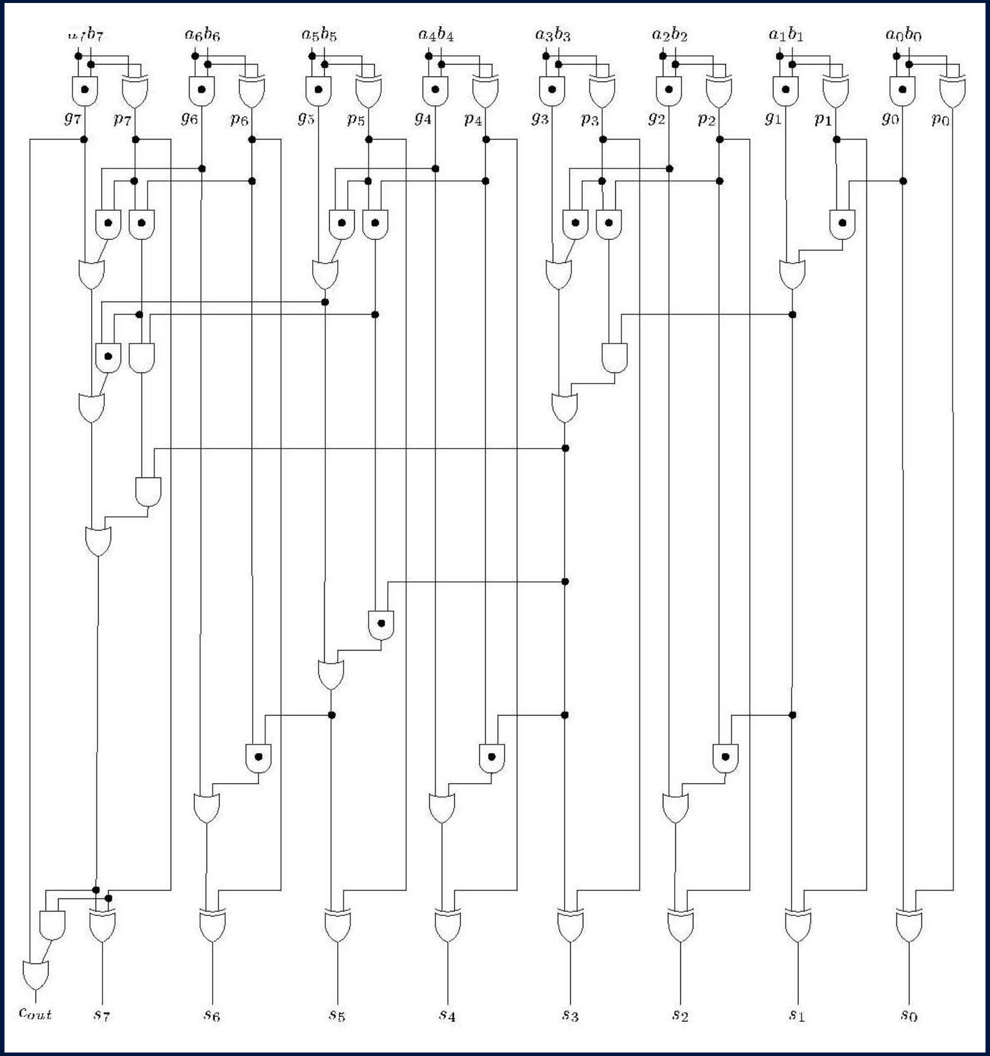
- Gate-level relaxation example



Gate-level 8-bit Brent-Kung adder circuit (Initial Boolean network)

Block- vs Gate-Level Relaxation Example

- Gate-level relaxation example

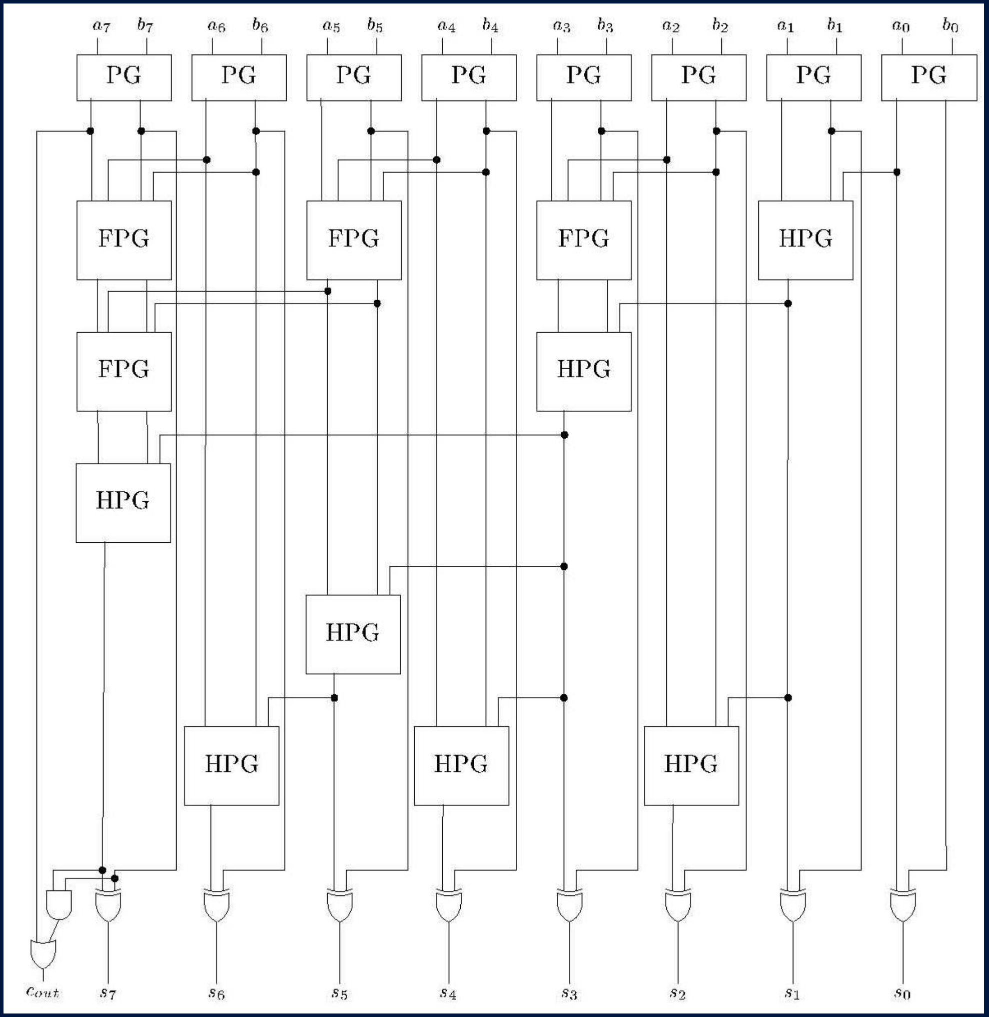


a_7b_7
a_6b_6
a_5b_5
a_4b_4
a_3b_3
a_2b_2
a_1b_1
a_0b_0
g_7 p_7 g_6 p_6 g_5 p_5 g_4 p_4 g_3 p_3 g_2 p_2 g_1 p_1 g_0 p_0
c_{out}
s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Gate-level 8-bit Brent-Kung adder circuit w/ relaxed gates marked

Block- vs Gate-Level Relaxation Example

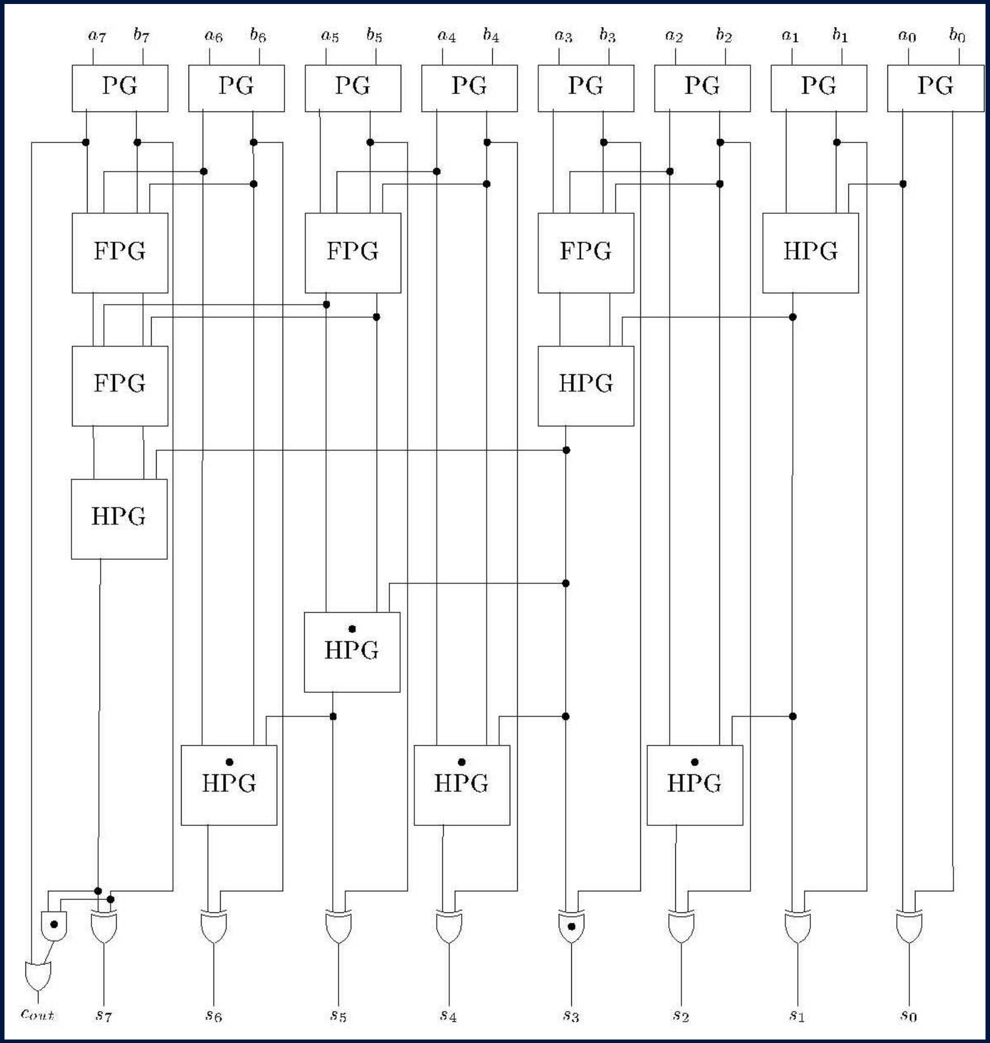
- Block-level relaxation example


a_7 b_7 a_6 b_6 a_5 b_5 a_4 b_4 a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0
PG
FPG
HPG
C_{out} s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Block-level 8-bit Brent-Kung adder circuit (Initial Boolean network)

Block- vs Gate-Level Relaxation Example

- Block-level relaxation example


a_7 b_7 a_6 b_6 a_5 b_5 a_4 b_4 a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0
PG
FPG
HPG
C_{out} s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Block-level 8-bit Brent-Kung adder circuit w/ relaxed blocks marked

Outline

1. Introduction
2. Background: Asynchronous Threshold Networks
3. Gate-Level Relaxation
4. Block-Level Relaxation
5. Experimental Results
6. Conclusions and Future Work

Experimental Results: Gate-Level Relaxation

- Results for DIMS-style asynchronous circuits

Original Boolean netwo rk		Unoptimized DIMS circuit			Optimization Run		
					# Relaxed n odes min.	Area min.	Delay opt.
name	#i/#o/#g	# full blo cks	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: Gate-Level Relaxation

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

Original Boolean netwo rk		NCL circuit			Optimization Run		
					# Relaxed n odes min.	Area min.	Delay opt.
name	#i/#o/#g	# full blo cks	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: Gate-Level Relaxation

- 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

Experimental Results: Block-Level Relaxation

Experiment #2: Gate-level vs. Block-level relaxation

- Evaluation on several arithmetic circuits:
  - Brent-Kung/Kogge-Stone adders, combinational multipliers
- Block-relaxation had 8.8% better delay with 10.8% worse area (avg.), compared to gate-level relaxation

Original Boolean network		Relaxed gate-level dual-rail circuit		Relaxed block-level dual-rail circuit	
name	#i/#o/#g	area	critical delay	area	critical delay
8-b Brent-Kung	32/18/49	4688.6	7.48	6094.1	6.64
16-b Brent-Kung	4/34/110	10396.8	10.69	13587.8	9.65
8-b Kogge-Stone	32/18/67	6341.8	5.57	9624.9	5.84
16-b Kogge-Stone	64/34/179	16571.5	6.99	22596.4	7.57
8-b unopt. mult	32/16/323	28828.4	25.69	24998.4	23.52
16-b unopt. mult	64/32/1411	125915.0	55.87	108728.0	52.29
8-b opt. mult	32/16/320	28523.1	20.98	24745.0	15.44
16-b opt. mult	64/32/1408	125610.0	46.70	108474.0	32.97
Average percentage				110.8%	91.2%

Experimental Results (cont.)

Experiment #2: Gate-level vs. Block-level relaxation

- Block-relaxation had 8.8% better delay with 10.8% worse area (avg.), compared to gate-level relaxation
- For 16-bit multiplier, 29.5% delay improvement

Original Boolean network		Relaxed gate-level dual-rail circuit		Relaxed block-level dual-rail circuit	
name	#i/#o/#g	area	critical delay	area	critical delay
8-b Brent-Kung	32/18/49	4688.6	7.48	6094.1	6.64
16-b Brent-Kung	4/34/110	10396.8	10.69	13587.8	9.65
8-b Kogge-Stone	32/18/67	6341.8	5.57	9624.9	5.84
16-b Kogge-Stone	64/34/179	16571.5	6.99	22596.4	7.57
8-b unopt. mult	32/16/323	28828.4	25.69	24998.4	23.52
16-b unopt. mult	64/32/1411	125915.0	55.87	108728.0	52.29
8-b opt. mult	32/16/320	28523.1	20.98	24745.0	15.44
16-b opt. mult	64/32/1408	125610.0	$\boxed{46.70}$	108474.0	$\boxed{32.97}$
Average percentage				110.8%	91.2%

Experimental Results (cont.)

Experiment #2: Gate-level vs. block-level relaxation

- Block-relaxation had 8.8% better delay with 10.8% worse area (avg.), compared to gate-level relaxation
- For 16-bit multiplier, 29.5% delay improvement
- For multipliers, 14.5% smaller area, on average

Original Boolean network		Relaxed gate-level dual-rail circuit		Relaxed block-level dual-rail circuit	
name	#i/#o/#g	area	critical delay	area	critical delay
8-b Brent-Kung	32/18/49	4688.6	7.48	6094.1	6.64
16-b Brent-Kung	4/34/110	10396.8	10.69	13587.8	9.65
8-b Kogge-Stone	32/18/67	6341.8	5.57	9624.9	5.84
16-b Kogge-Stone	64/34/179	16571.5	6.99	22596.4	7.57
8-b unopt. mult	32/16/323	28828.4	25.69	24998.4	23.52
16-b unopt. mult	64/32/1411	125915.0	55.87	108728.0	52.29
8-b opt. mult	32/16/320	28523.1	20.98	24745.0	15.44
16-b opt. mult	64/32/1408	125610.0	46.70	108474.0	32.97
Average percentage				110.8%	91.2%

Conclusion

- Local Relaxation Technique:
  - Optimization technique for robust asynchronous threshold circuits
  - Relaxes overly-restrictive style: selective use of "eager evaluation"
  - Can target three different cost functions:
    - # relaxed nodes, area, critical path delay
  - CAD tool developed/released: "ATN-OPT"
  - Gate-level relaxation: exhibits significant improvements
    - 48.2% of gates relaxed (avg.)
    - 25.9% area improvement (vs. NCL custom mapping)
    - 17.7% delay improvement (vs. NCL custom mapping)
  - Block-level relaxation:
    - 8.8% additional delay improvement (best: 29.5%)
    - 10.8% additional area overhead (best: 14.5% reduction)

⇨ No change to overall timing-robustness of circuits