

Concurrency Reduction of Untimed Latch Protocols – Theory and Practice

Async 2010

Santosh N. Varanasi, Kenneth S. Stevens, and Graham Birtwistle

University of Utah

University of Sheffield



Supported by a gift from Sun Microsystems

Concurrency and Protocols

Systems are complex. We want to:

1. Better understand concurrency and synchronization
2. Examine composition of concurrent protocols
3. Verify complex systems
4. Validate concurrency reduction assumptions about hardware

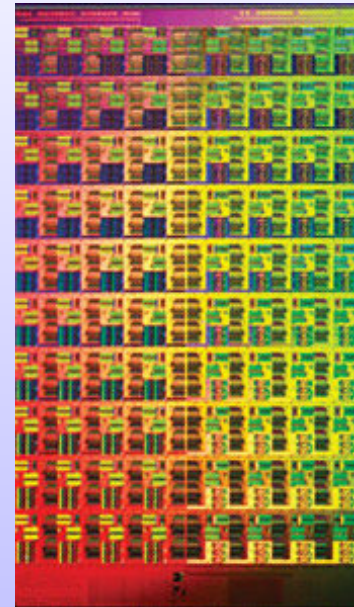


Photo: Intel – 80 core teraflops chip

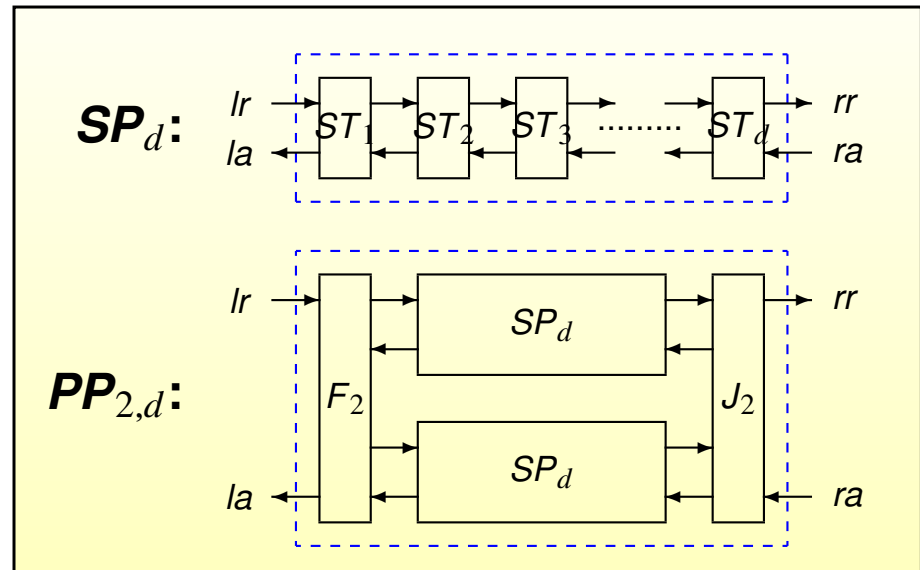
Theory Overview

1. Created formal abstract model

- for linear pipelines
- covers *all designs* in protocol class

2. Concurrency reduction rules

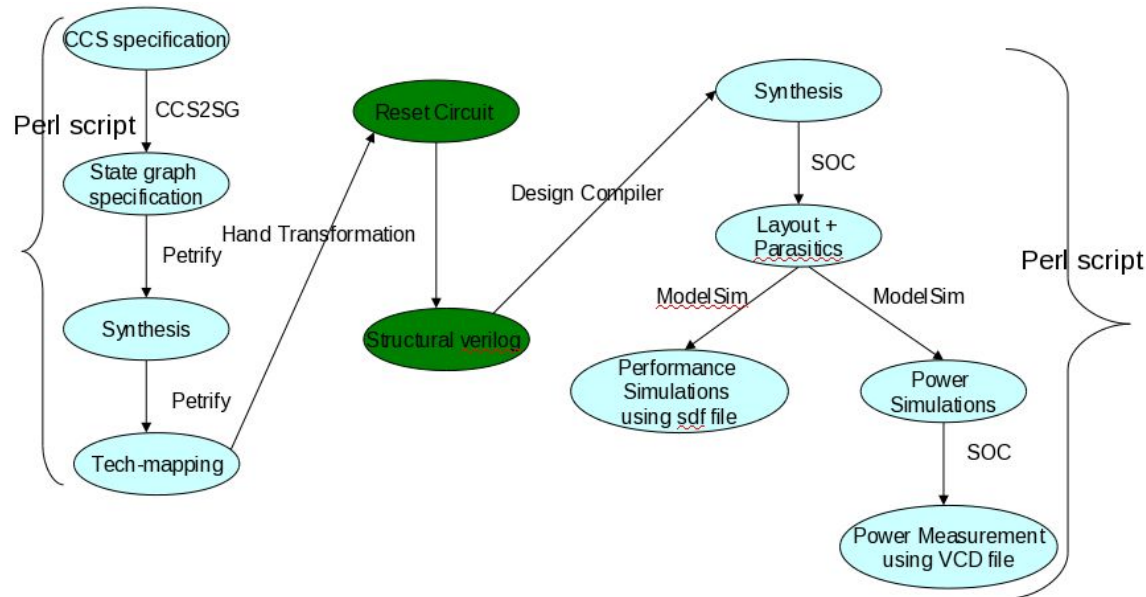
- start with maximum state space
- well defined rules and properties
- 250 / 4900 possible untimed / timed protocols



Practice Overview

Evaluate complete family of 137 untimed protocols from Theory

- Scripted flow evaluating cycle time, latency, area, energy
- Results from physical P&R design in
 - ◆ IBM 65nm 10sf process, Artisan 12T static library



- See how 17 published untimed protocol compare to 114 others
- Evaluation assumptions of concurrency reduction effects

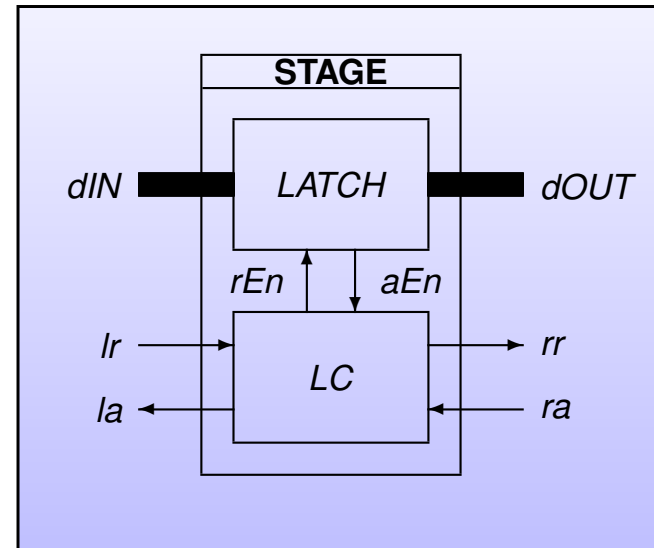
Formal Model for Datapath Abstraction

1. Delay Insensitive Abstraction

- n -bit data path can be abstracted as single bit path

2. Bundled Data abstraction

- a. prove that abstraction is correct
- b. prove that concurrency reduction supports abstraction



We have shown this for most published designs

Formal Model for Datapath Abstraction

Executive summary: *Serializing the latch handshake in process L allows us to prove correctness of abstraction and concurrency reduction approach.*

1. Liveness property holds due to concurrency reduction rules (next foils)
2. Process L ensures **input data integrity** holds
3. Process V ensures **output data validity** holds
4. Process S ensures **output data integrity** holds
5. Protocol timing ensures **input data validity** holds

$$L = lr\uparrow.gS.\overline{rEn}\uparrow.aEn\uparrow.\overline{rEn}\downarrow.aEn\downarrow.pV.\overline{la}\uparrow.lr\downarrow.\overline{la}\downarrow.L$$

$$R = gV.\overline{rr}\uparrow.ra\uparrow.pS.\overline{rr}\downarrow.ra\downarrow.R$$

$$S = \overline{gS}.\overline{pS}.S \qquad V = \overline{pV}.\overline{gV}.V$$

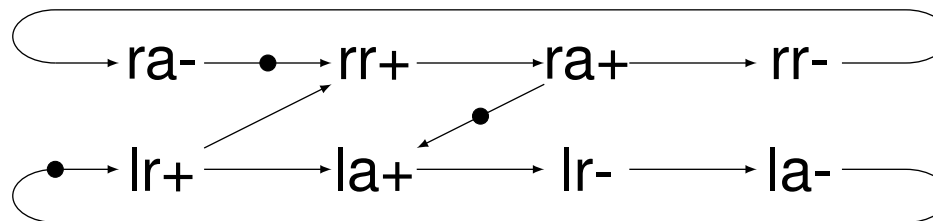
$$LC = (L \mid R \mid S \mid V) \setminus \{gS, pS, gV, pV\}$$

$$LATCH = rEn\uparrow.open.\overline{aEn}\uparrow.rEn\downarrow.closed.\overline{aEn}\downarrow.LATCH$$

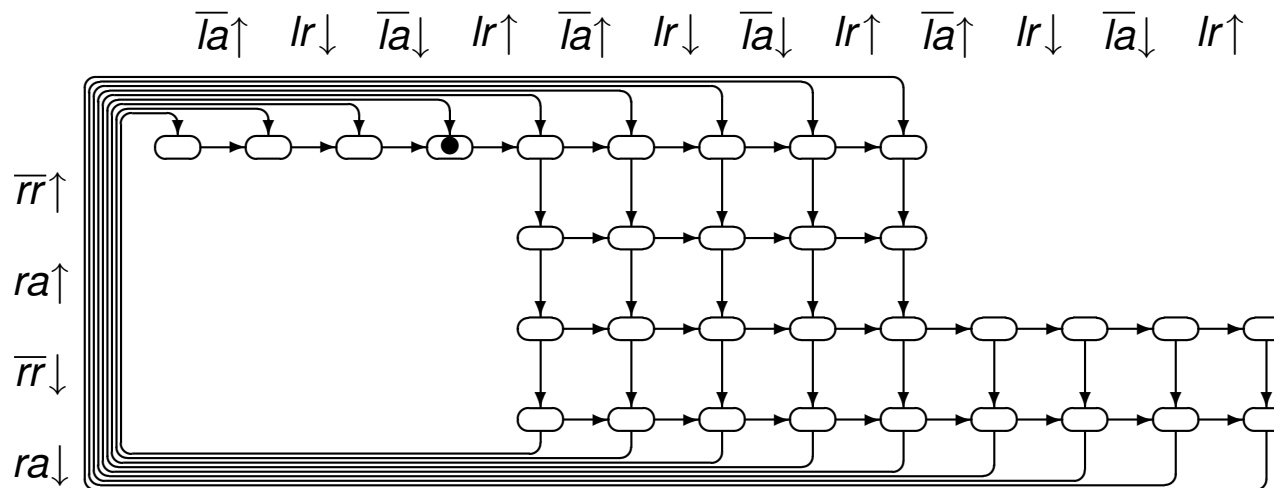
$$max = (LC \mid LATCH) \setminus \{rEn, aEn\} \tag{1}$$

Maximum Concurrency Abstracted Protocol

As a petri net (very much like process on previous foil):

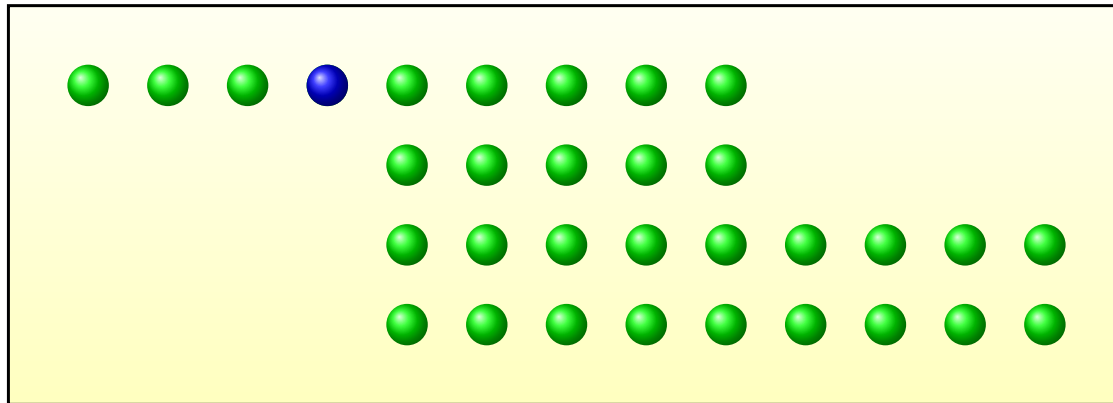


As minimized state graph, in particular configuration called **shape**



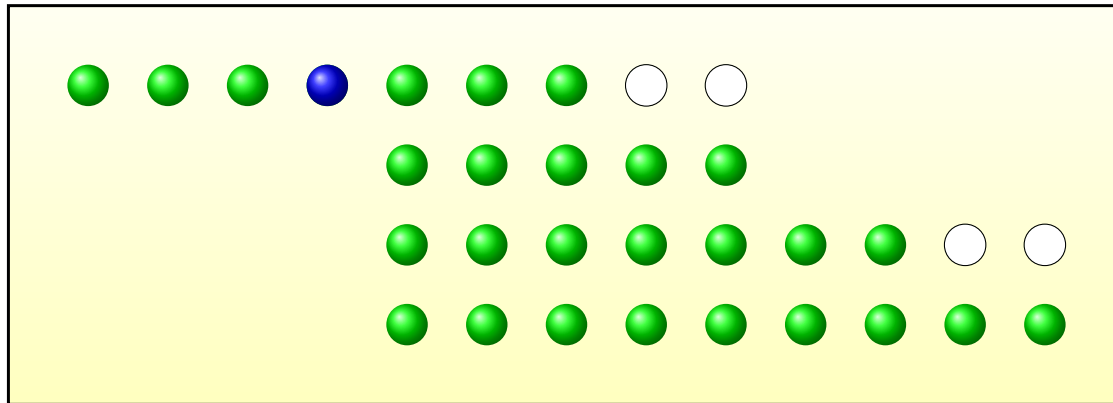
Concurrency Reduction (*Cuts*) from Most Concurrent Design (*Shape*)

- **Shape** is representation of state space in particular configuration
- Arcs not shown for clarity
- Reduce concurrency by removing states from left or right of rows



Concurrency Reduction (*Cuts*) from Most Concurrent Design (*Shape*)

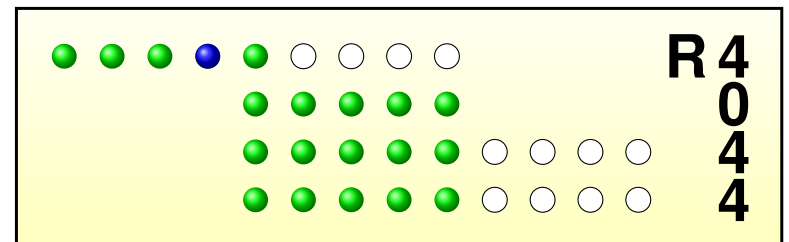
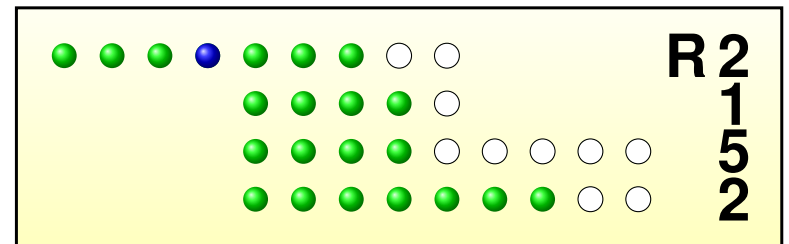
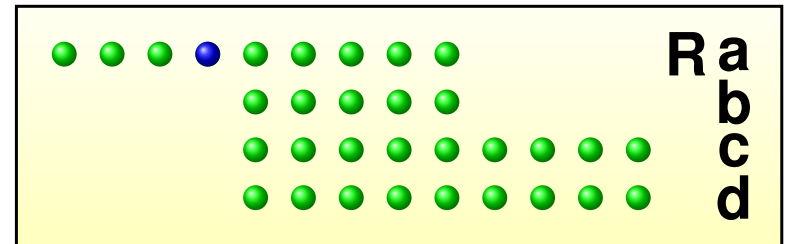
- **Shape** is representation of state space in particular configuration
- Arcs not shown for clarity
- Reduce concurrency by removing states from left or right of rows



Removed (or **cut**) 2 states from right of first and third rows

Concurrency Reduction with Right Cuts

- Remove states from right end of rows.
- Denoted **Rabcd**
- Liveness constraints
 - ◆ limits extend of cuts
(initial state must be reachable)
 - ◆ imply “above” states must be removed



Right Cut Constraints:

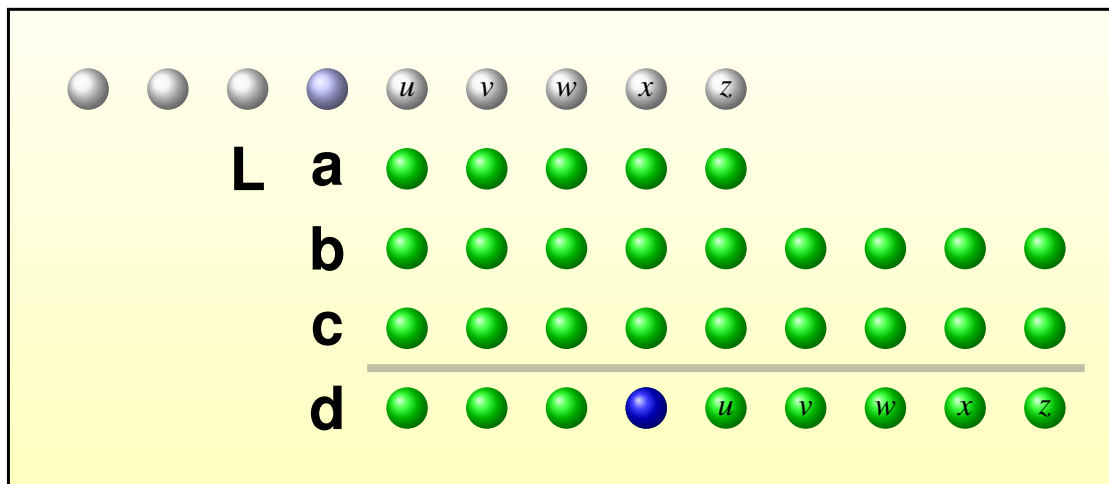
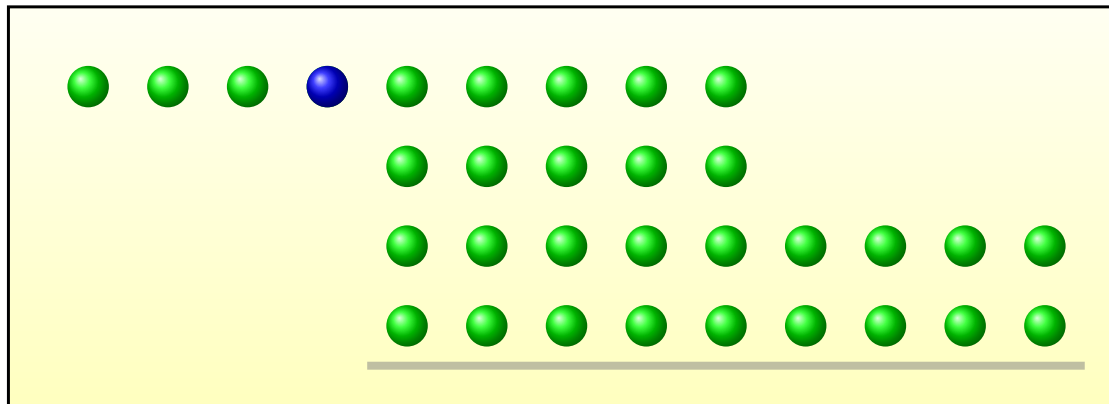
$$0 \leq a, b \leq 4$$

$$0 \leq c, d \leq 8$$

$$a \geq b \wedge b + 4 \geq c \wedge c \geq d \wedge d \geq a$$

Concurrency Reduction with Left Cuts

- Modify shape representation for convenience
 - ◆ top row duplicated below bottom row
 - ◆ cut labels start on second row



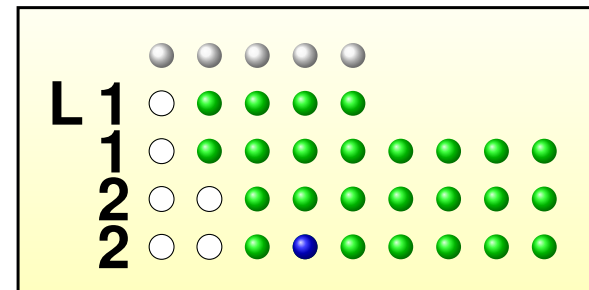
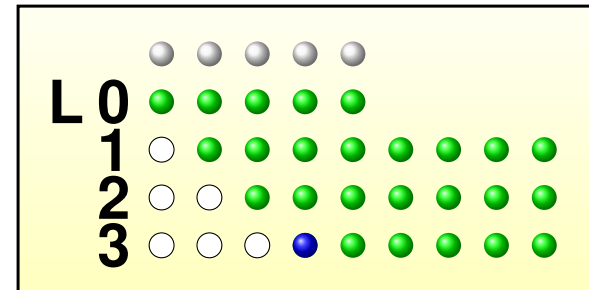
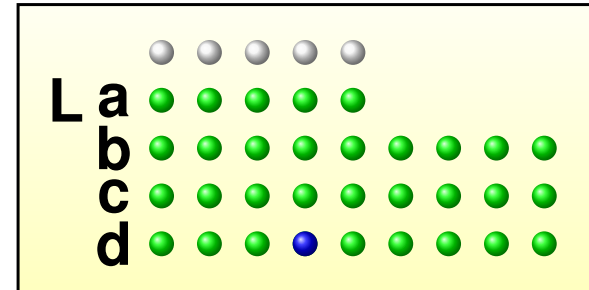
Concurrency Reduction with Left Cuts

- Remove states from left end of rows
- Denoted **Labcd**
- Liveness constraints
 - ◆ limits extend of cuts
 - ◆ imply “below” states must be removed

Left Cut Constraints:

$$0 \leq a, b, c, d \leq 3$$

$$a \leq b \wedge b \leq c \wedge c \leq d$$

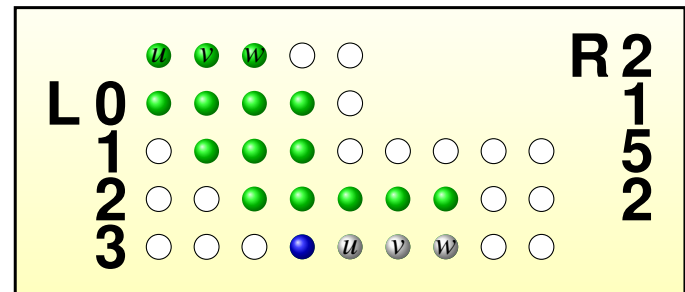
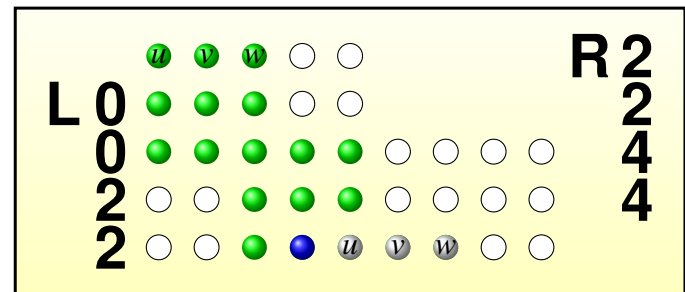
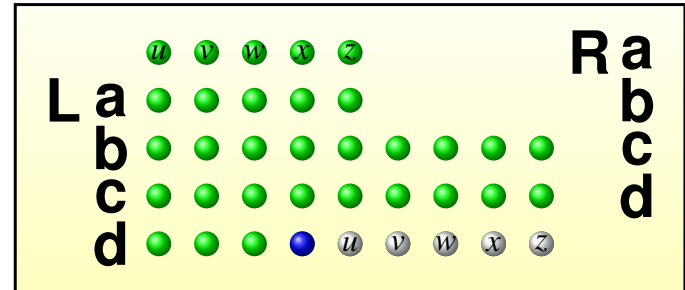


Protocol Family with Left and Right Cuts

- Combine Left \mathcal{L} and Right \mathcal{R} cuts for concurrency reduction on protocol family
- Liveness restricts cuts

Combined Cut Liveness Constraints:

$$\begin{aligned}
 &La + Ra < 5 \wedge Lb + Rb < 5 \wedge \\
 &La + Rb < 5 \wedge \\
 &Lb + Rc < 9 \wedge Lc + Rd < 9 \wedge \\
 &Lc + Rc < 9 \wedge Ld + Rd < 9
 \end{aligned}$$

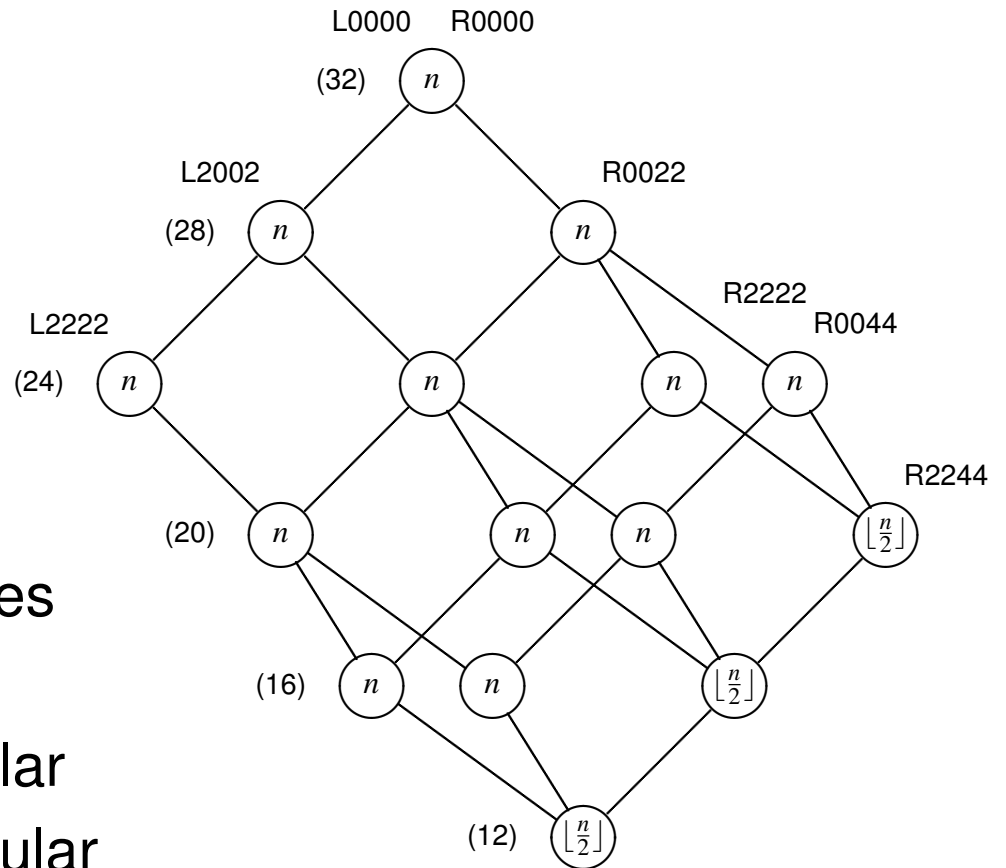


Current Status

1. Abstraction Model

2. Concurrency reduction rules

- complete and coherent
- liveness
- occupancy
- pipeline behavior
 - ◆ family equivalence classes
 - ◆ delay-insensitive
 - ◆ speed-independent regular
 - ◆ speed-independent irregular



Parallel pipeline equivalence classes

3. Practice trailing theory

Practice: Concurrency Reduction on Protocols

Compare all 137 protocols under concurrency reduction.

What is the impact of concurrency reduction on designs?

Can we determine which protocol in the family is best?

What is the best implementation for equivalent protocols when pipelined?

L0000	L0011	L1111	L0022	L1122	L0033	L1133	L2222	L2233	L3333	L o R
	■		■							R0000
■	■									R0020
										R0040
										R0022
			■			■		■	.	R0042
									.	R2022
									.	R2042
■	■	.	■						.	R0044
			■						.	R2044
									.	R4044
							■		.	R2222
			■						.	R2242
									.	R2262
			■						.	R2244
	■	.							.	R2264
	■	.	■						.	R4244
									.	R4264

Coverage of published protocols

Practice: Concurrency Reduction on Protocols

Complex interplay between:

- Concurrency reduction simplifies logic
- Simpler logic assumed to be faster
- Concurrency reduction adds protocol delays
- Hazard removal and technology mapping creates “noise”

Our expectation:

- **Optimal will lie somewhere in middle of concurrency reduction spectrum**

Will this hold globally as well as across protocol equivalent classes?

Circuit Complexity and Concurrency Reduction

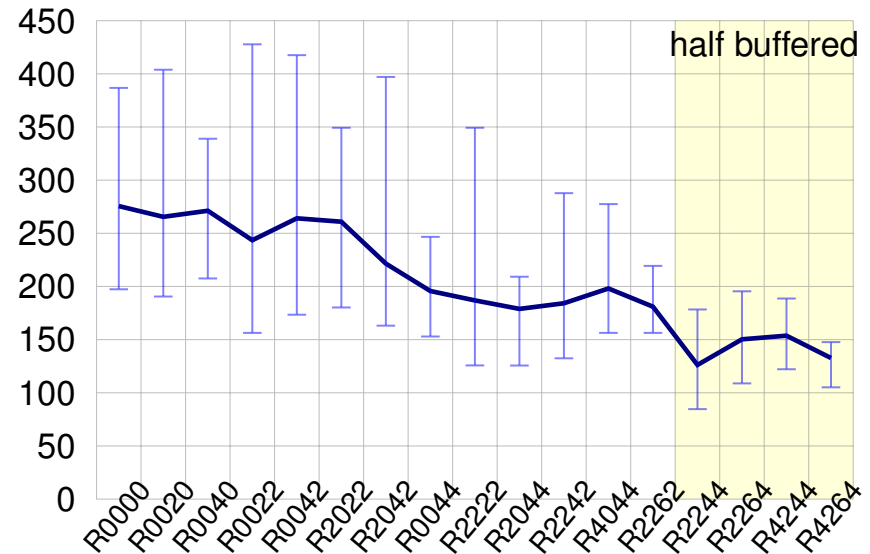
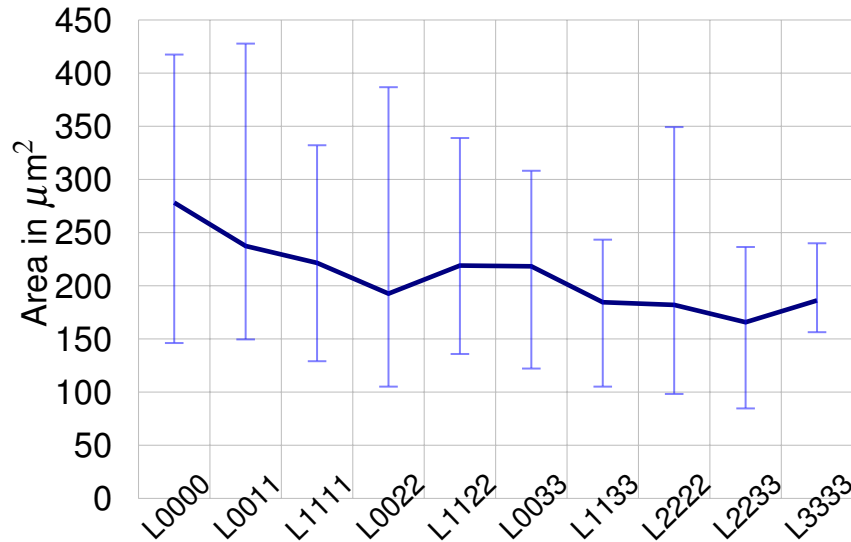
Number of explicit state variables generated by Petrify

L0000	L0011	L1111	L0022	L1122	L0033	L1133	L2222	L2233	L3333	L ◦ R
–	–	2	4	–	–	2	2	2	2	R0000
3	–	2	2	2	2	1	2	1	1	R0020
2	2	2	2	2	2	1	2	1	1	R0040
3	3	2	1	2	2	1	2	1	1	R0022
2	2	2	–	2	2	1	2	1	1	R0042
2	2	2	2	2	2	1	2	1	.	R2022
2	1	1	1	1	1	0	1	0	.	R2042
3	2	2	2	2	2	1	2	1	1	R0044
2	2	1	1	1	1	0	1	0	.	R2044
2	1	.	1	.	1	R4044
2	2	2	2	2	2	1	2	1	.	R2222
2	1	1	2	1	1	0	1	0	.	R2242
2	1	1	1	1	.	.	1	.	.	R2262
2	1	1	1	1	1	0	1	0	.	R2244
2	1	1	1	0	.	.	1	.	.	R2264
2	1	.	1	.	1	R4244
1	0	.	0	R4264

- “–”: no solution found “.”: deadlocking agent
- Complexity reduction assumption: generally holds

Area and Concurrency Reduction

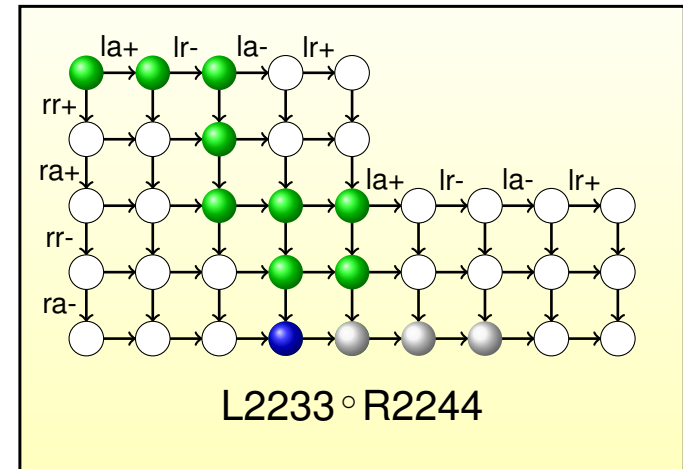
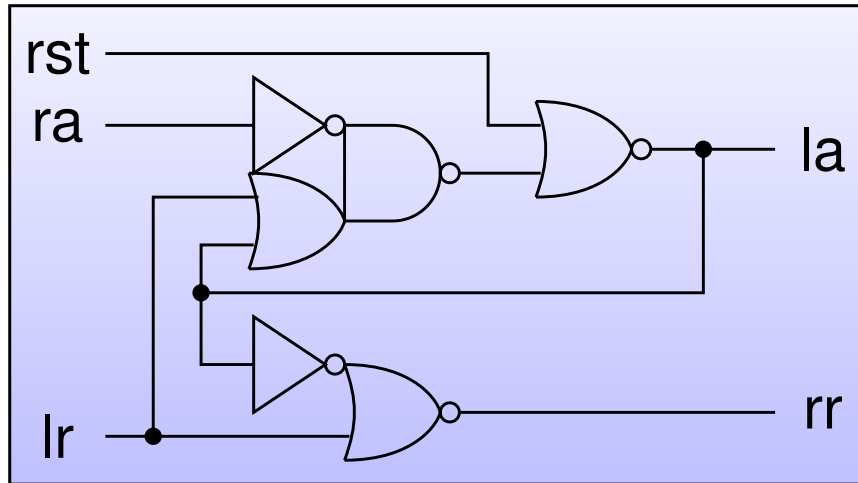
Effect of Concurrency Reduction on Area for Out and In Channels:



Assumption: Concurrency reduction reduces area

- Generally holds on both output (\mathcal{L}) and input (\mathcal{R}) channels
- L2233 cuts and R2244 best on average
- R2044 best cut for fully buffered protocol

Area and Concurrency Reduction

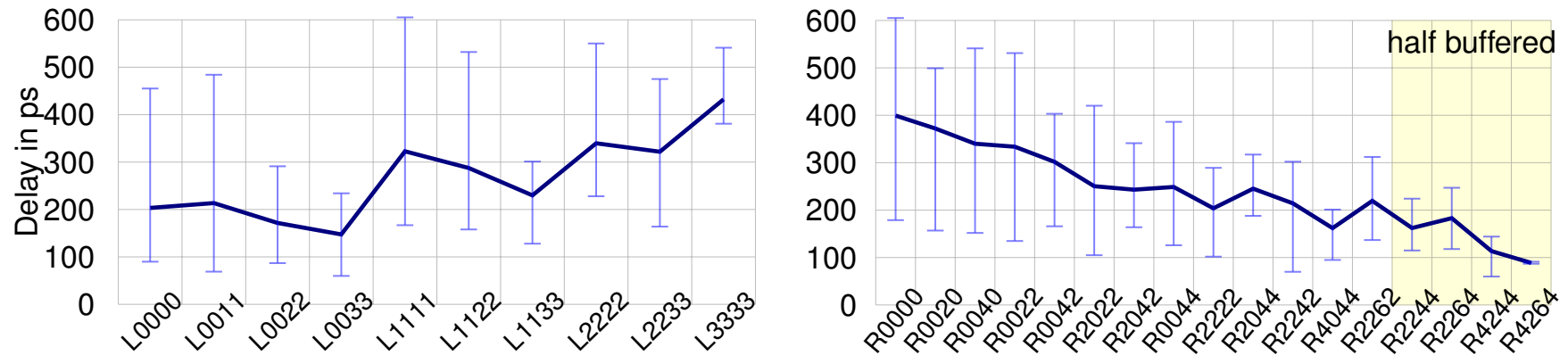


Circuit with smallest area and backward latency

- Half-buffered
- Protocol $L2233 \circ R2244$
- Only three gates

Forward Latency

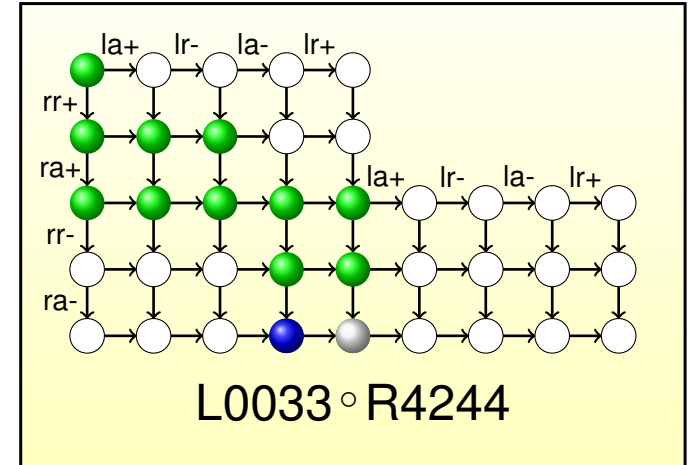
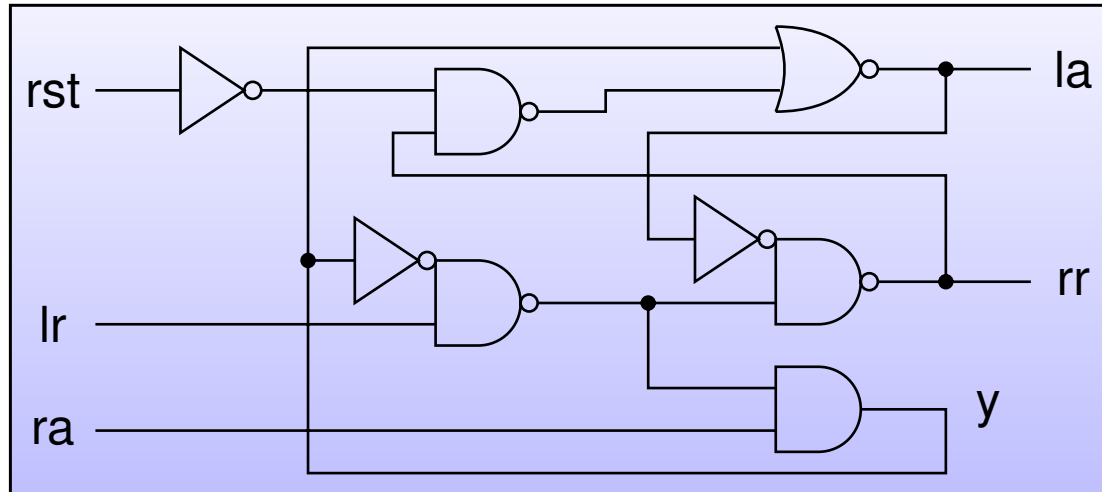
Effect of Concurrency Reduction on Forward Latency



Assumption: Concurrency reduction optimized away from ends

- Generally holds due to competing tradeoffs
 - ◆ concurrency reduction on incoming channel generally reduces latency
 - ◆ outgoing channel \mathcal{L} produces tradeoff
 - generally decreasing
 - outgoing channel cuts delaying $rr \uparrow$ increase latency
 - other reductions have 2nd order effect of reducing latency

Forward Latency



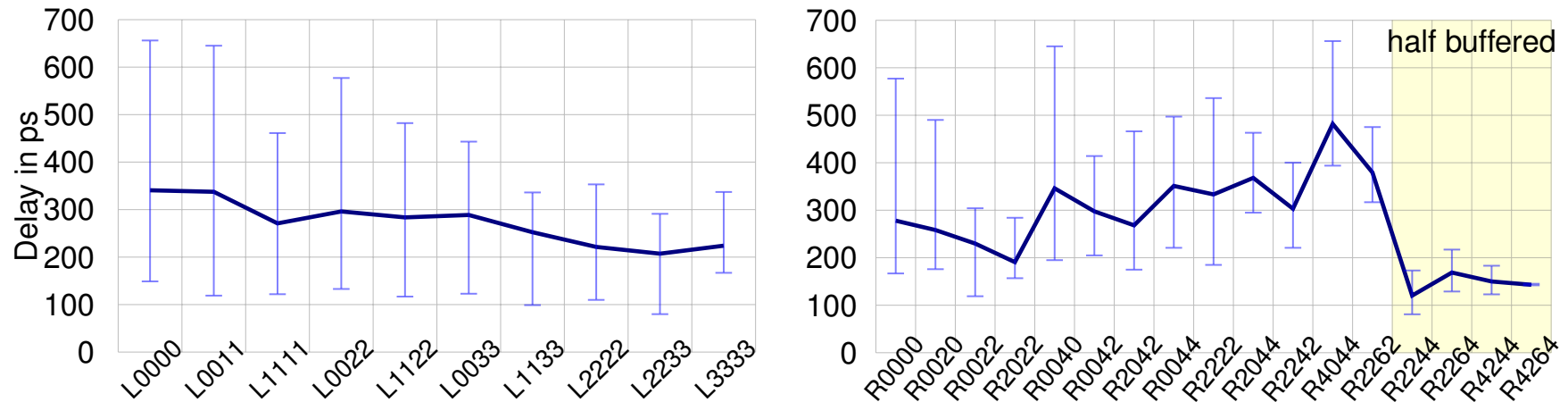
Circuit with smallest forward latency

L0033 generally best cut for forward latency

- Half buffered protocol L0033 ◦ R4244 best
- Full buffered L0033 ◦ R2242 only 10ps slower

Backward Latency

Effect of Concurrency Reduction on Backward Latency

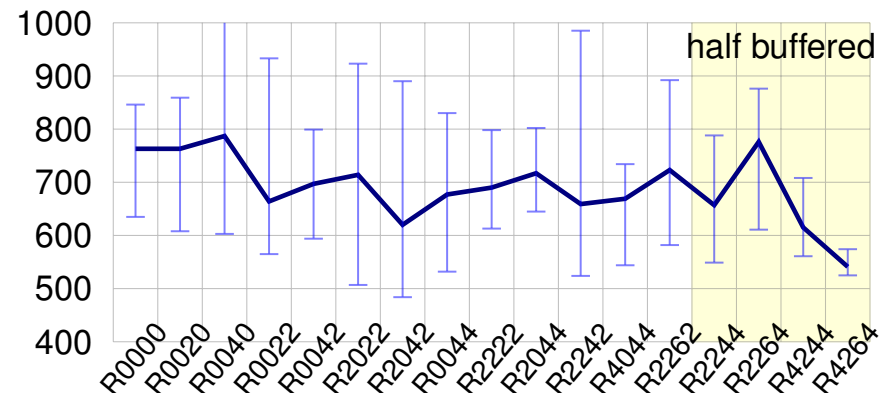
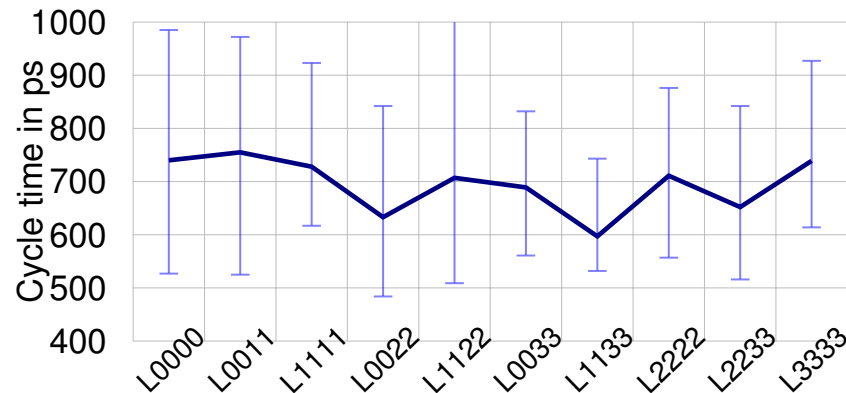


Assumption: Concurrency reduction optimized away from ends

- Generally holds except for half buffered protocols
 - ◆ outgoing channel \mathcal{L} latency generally reduced, though not significantly
 - ◆ incoming channel \mathcal{R} decreases irregularly based on delaying la
 - ◆ incoming half buffered protocols ultra-fast
 - every other stage stalls in different *compatible* state

Cycle Time

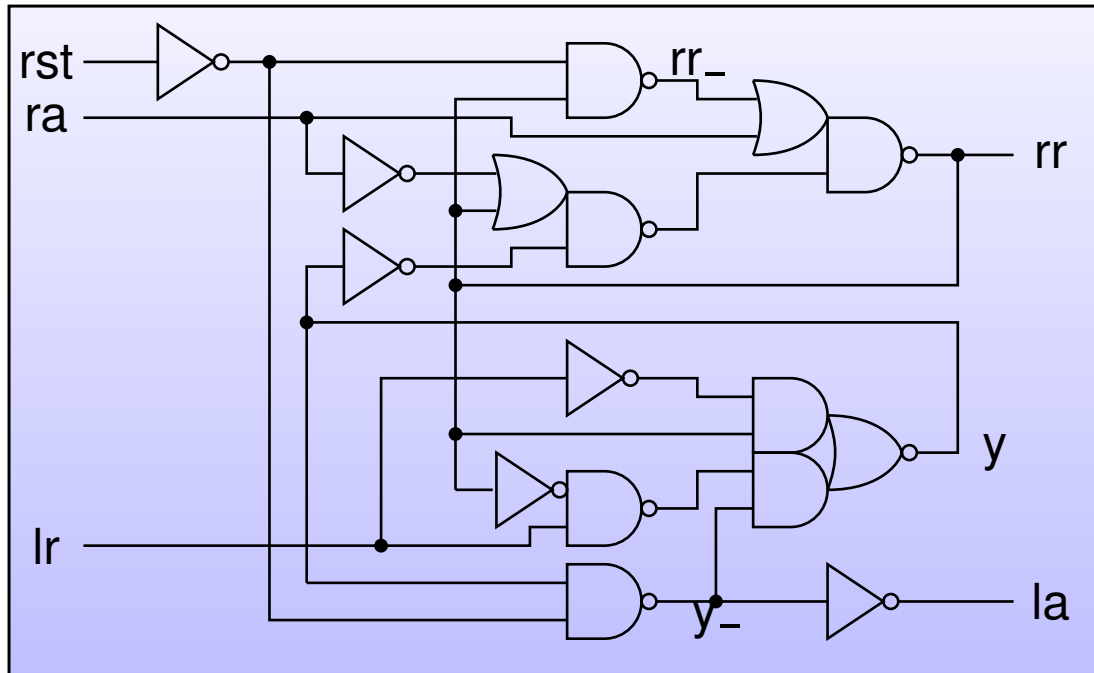
Effect of Concurrency Reduction on Backward Latency



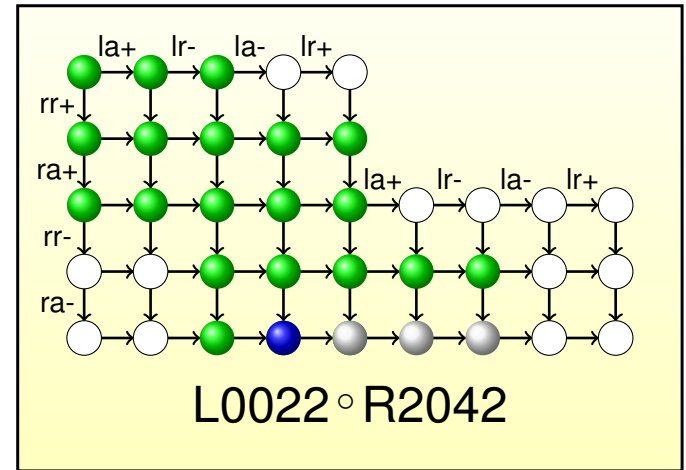
Assumption: Concurrency reduction optimized away from ends

- Generally holds for output channel cuts \mathcal{L}
 - ◆ L1133 generally the best cut
- Generally holds for input channel **except for half buffered protocols**
 - ◆ R2042 best full buffered protocol
 - ◆ **half buffered similar to / affected by the backward latency drop-off**
 - **every other stage stalls in different *compatible* state**

Cycle Time



Circuit with smallest cycle time



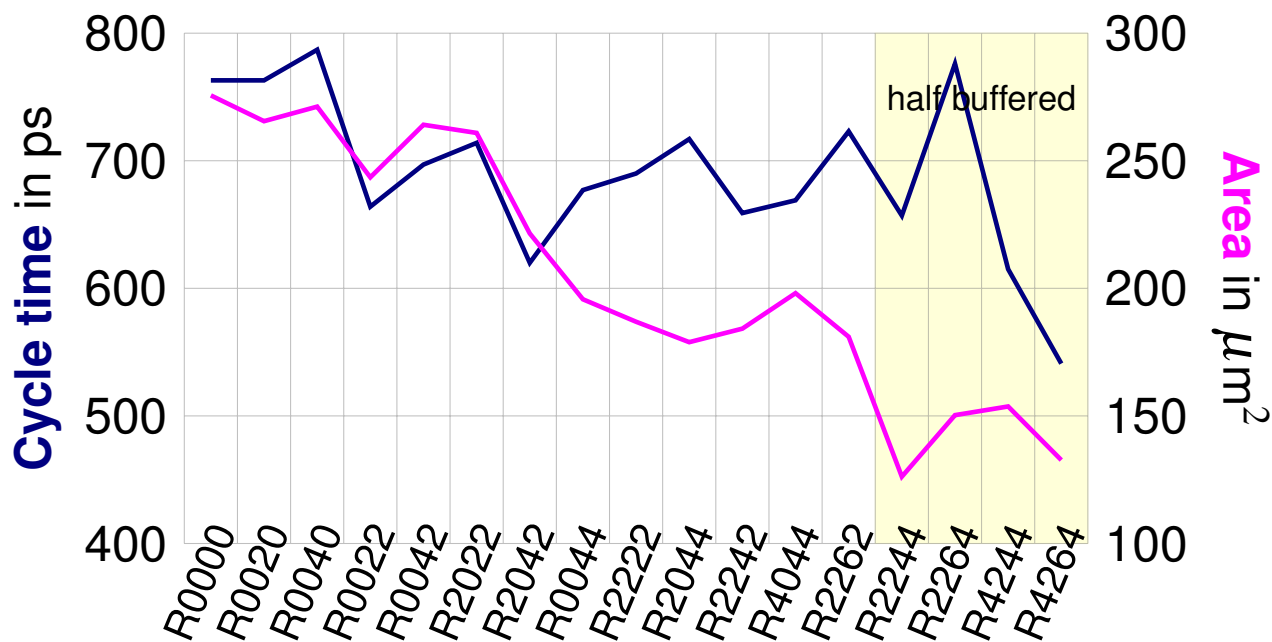
- Full buffered protocol
- Protocol L0022 ◦ R2042

Concurrency: Performance vs Parallelism

Assumption: \exists a tradeoff between performance and parallelism

- Performance increases with concurrency reduction
- Protocol parallelism decreases with concurrency reduction

Evaluate comparing performance (**area**) with parallelism (**cycle time**)



Optimal Cuts vs Published Space

Most of the best cuts for latency (blue) and cycle time (green) have not been explored.

L0000	L0011	L1111	L0022	L1122	L0033	L1133	L2222	L2233	L3333	L o R
	■		■		■	■				R0000
■	■				■	■				R0020
					■	■				R0040
					■	■				R0022
■	■	■	■	■	■	■	■	■	■	R0042
■	■	■	■	■	■	■	■	■	■	R2022
■	■	■	■	■	■	■	■	■	■	R2042
			■		■	■				R0044
■	■		■		■	■				R2044
■	■		■		■	■				R4044
			■		■	■	■			R2222
			■		■	■				R2242
			■		■	■				R2262
■	■	■	■	■	■	■	■	■	■	R2244
	■				■	■				R2264
	■				■	■				R4244
	■		■		■	■				R4264

Conclusions

1. Abstraction approach correct for all protocols in family
2. Regular concurrency reduction rules covering *all* designs
 - limited here to untimed (DI & SI) protocols
3. Rules form complete, coherent, symmetric lattice
4. Complete protocol family was synthesized and characterized
5. Assumptions on effects of concurrency reduction evaluated
 - largely shown true: best design near mid range
 - surprising quality found for half buffered protocols
6. Better designs for certain applications uncovered