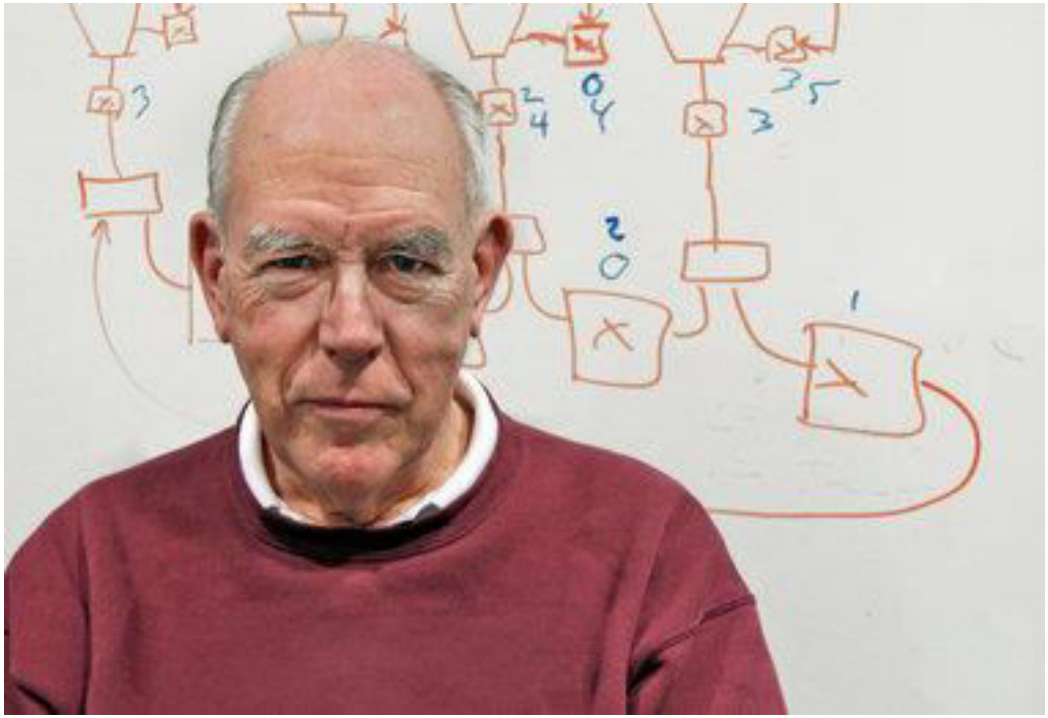
A Random Walk from Async to Sync

Paul Cunningham & Steev Wilcox



Thank You Ivan

In the Beginning…

azuro

March 2002

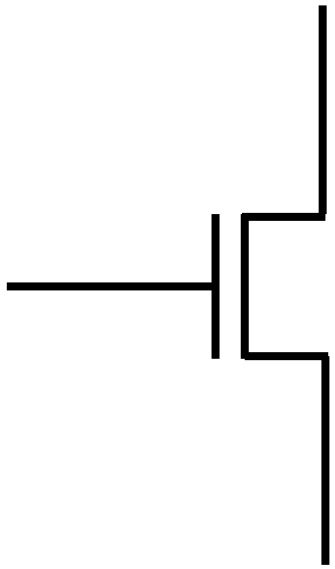
Azuro Day 1

- Some money in the bank from Angel Investors
- 2 employees
- Small Office rented from Cambridge University Computer Lab
- Vision to automatically compile mainstream synchronous designs into low power asynchronous equivalents

…So why *is* Asynchronous design low power!?

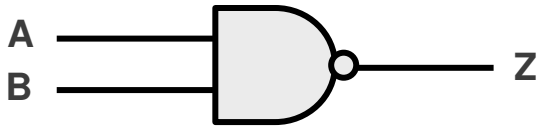
Azuro Day 100

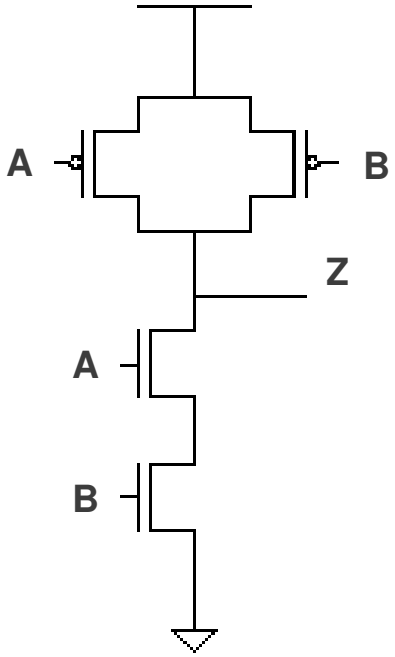
…after several meetings with chip companies, VCs and lots of whiteboard time



Level Sensitive Device

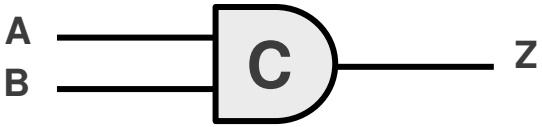
LEVEL SENSITIVE AND GATE

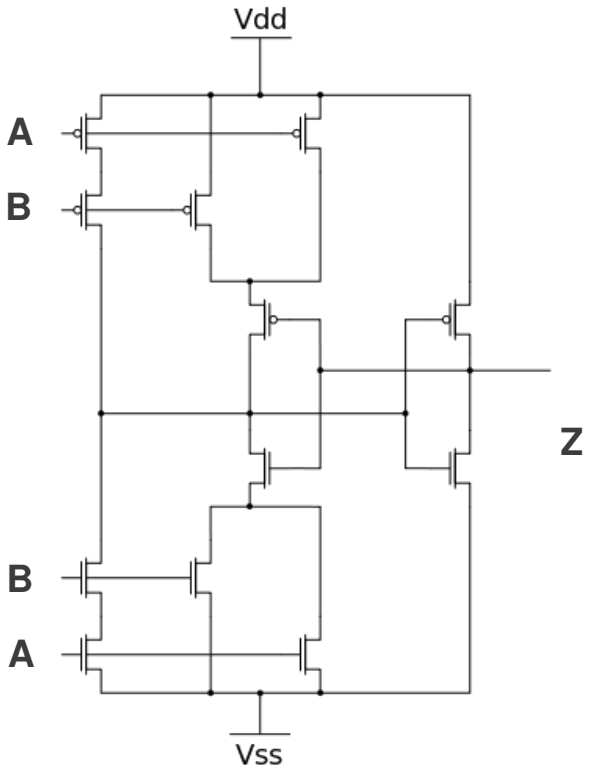




4 transistors

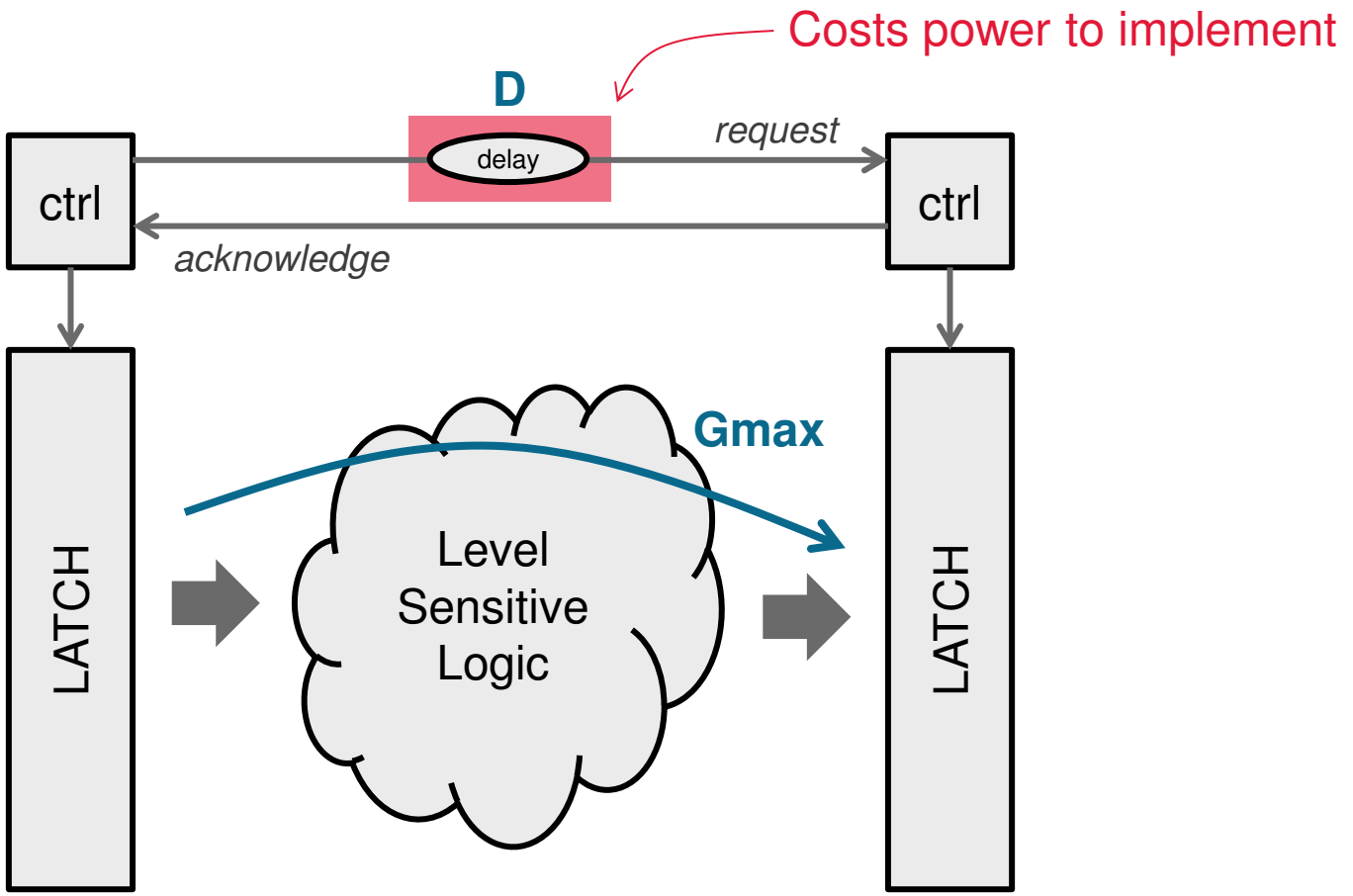
EDGE SENSITIVE AND GATE





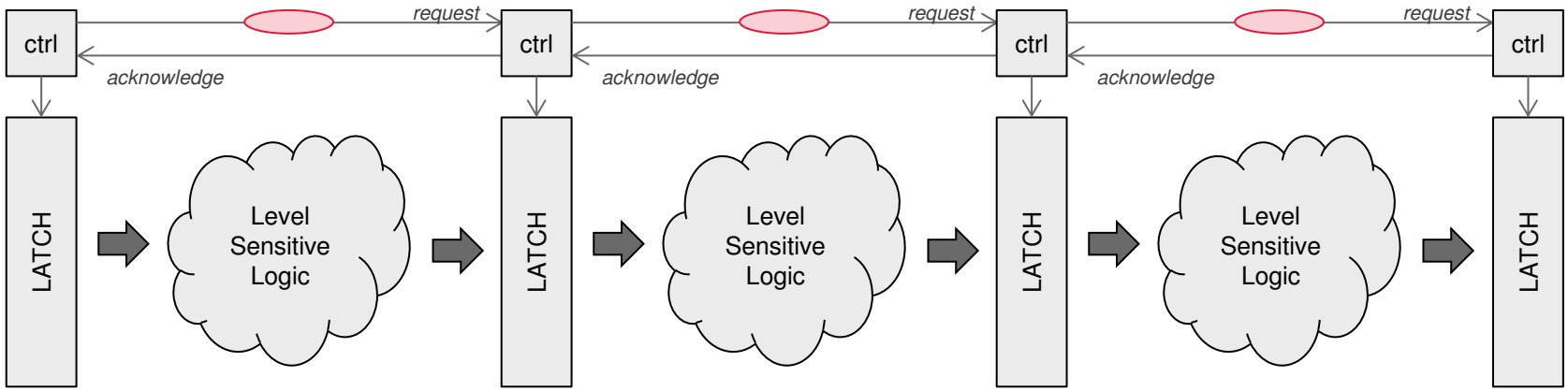
3X

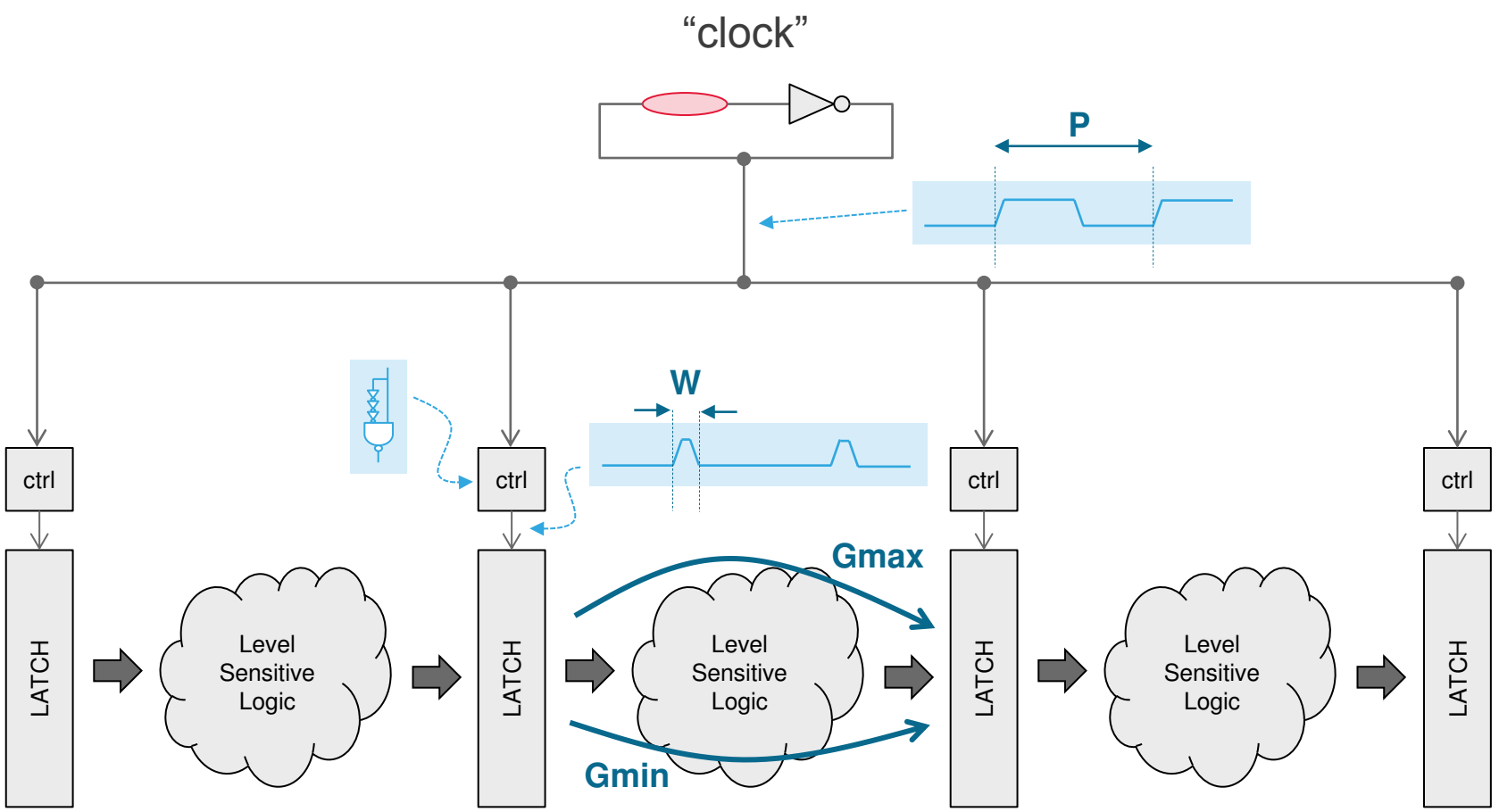
12 transistors



Gmax < D

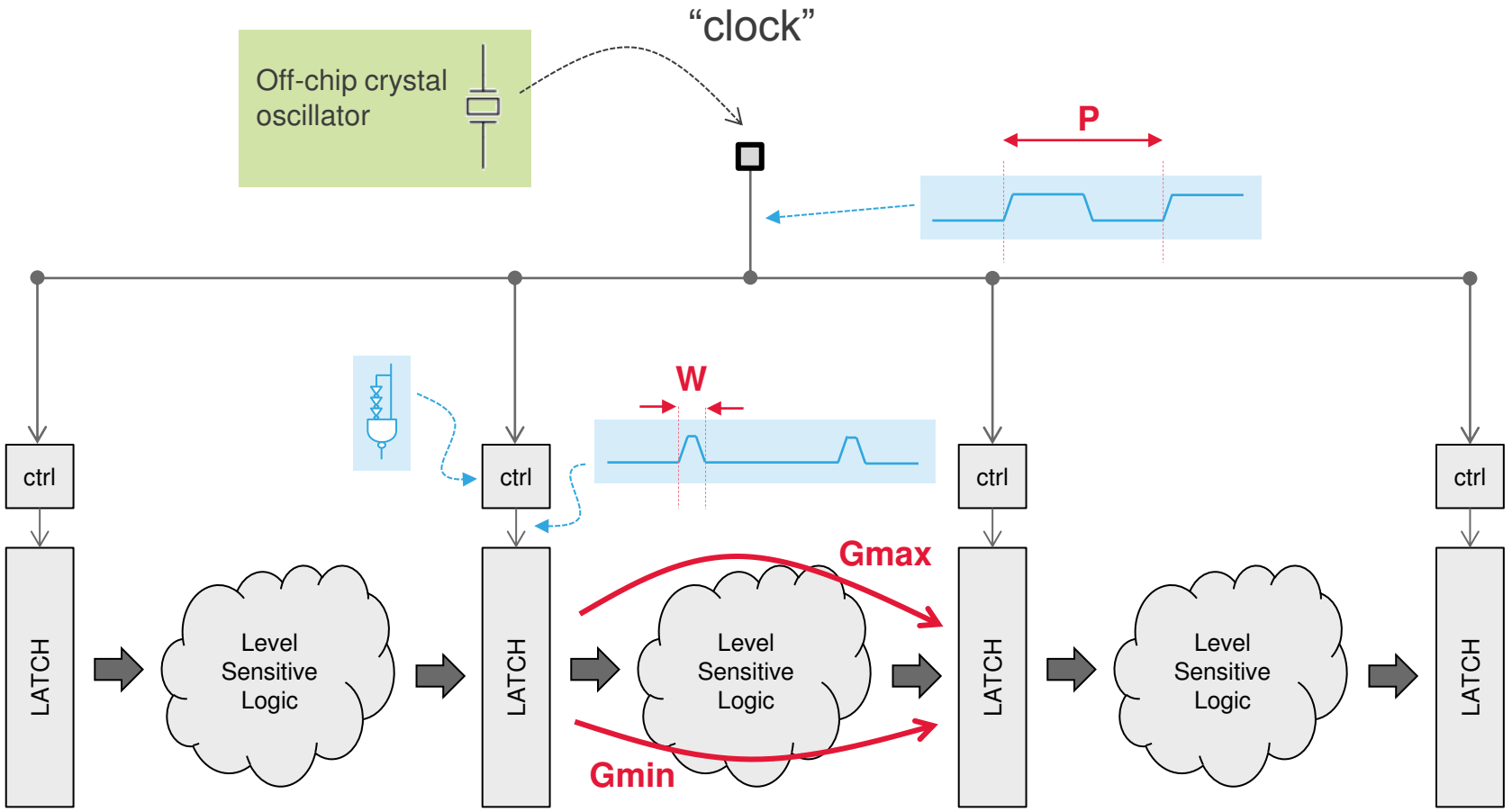
Hard to “sign-off” across different process corner, voltage, temperature


ctrl
request
ctrl
request
ctrl
request
ctrl
acknowledge
acknowledge
acknowledge
LATCH
Level
Sensitive
Logic
LATCH
Level
Sensitive
Logic
LATCH
Level
Sensitive
Logic
LATCH



Hold Check: Gmin > W

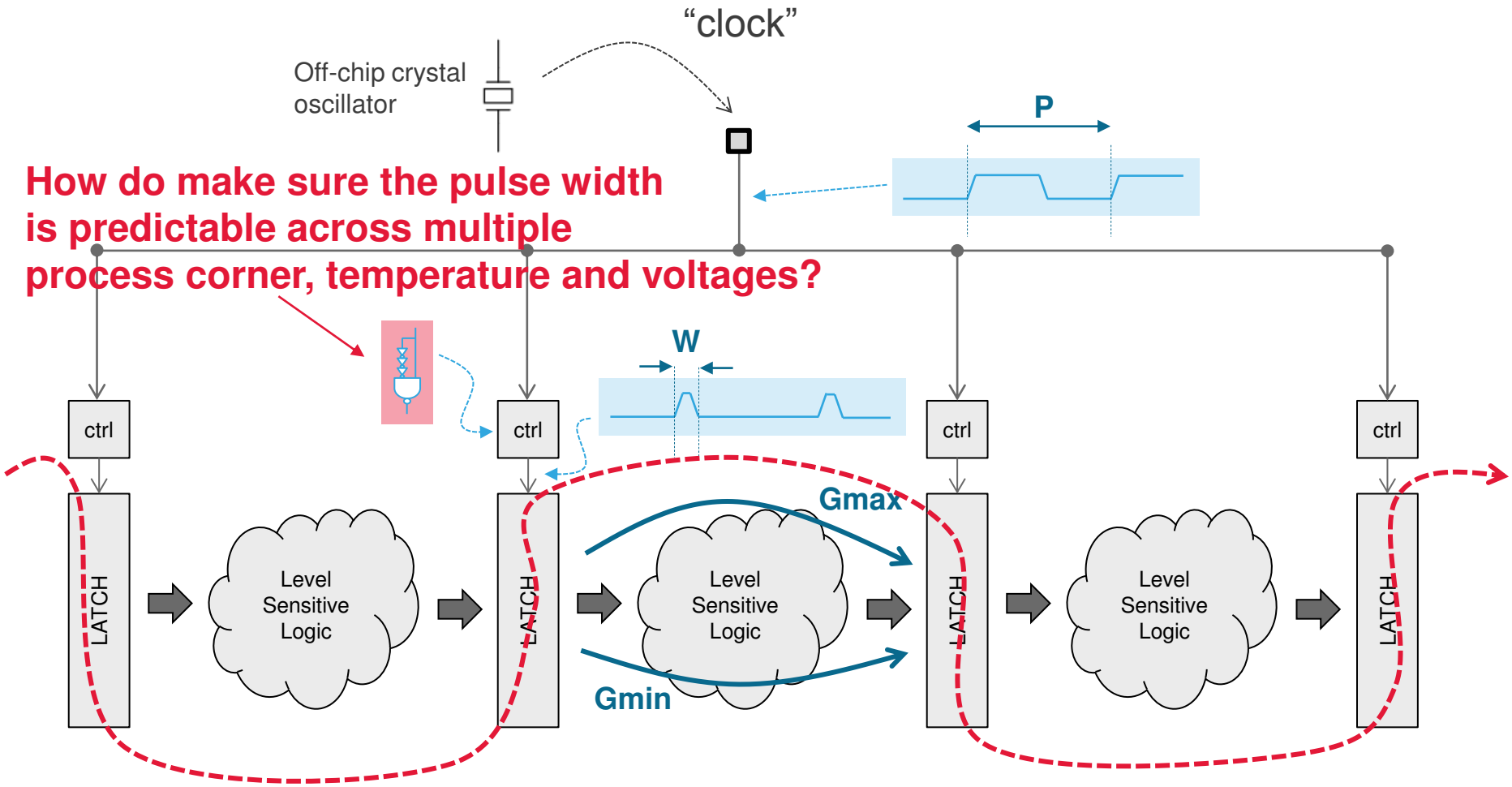
Setup Check: Gmax < P



Hold Check: Gmin > W

Setup Check: Gmax < P

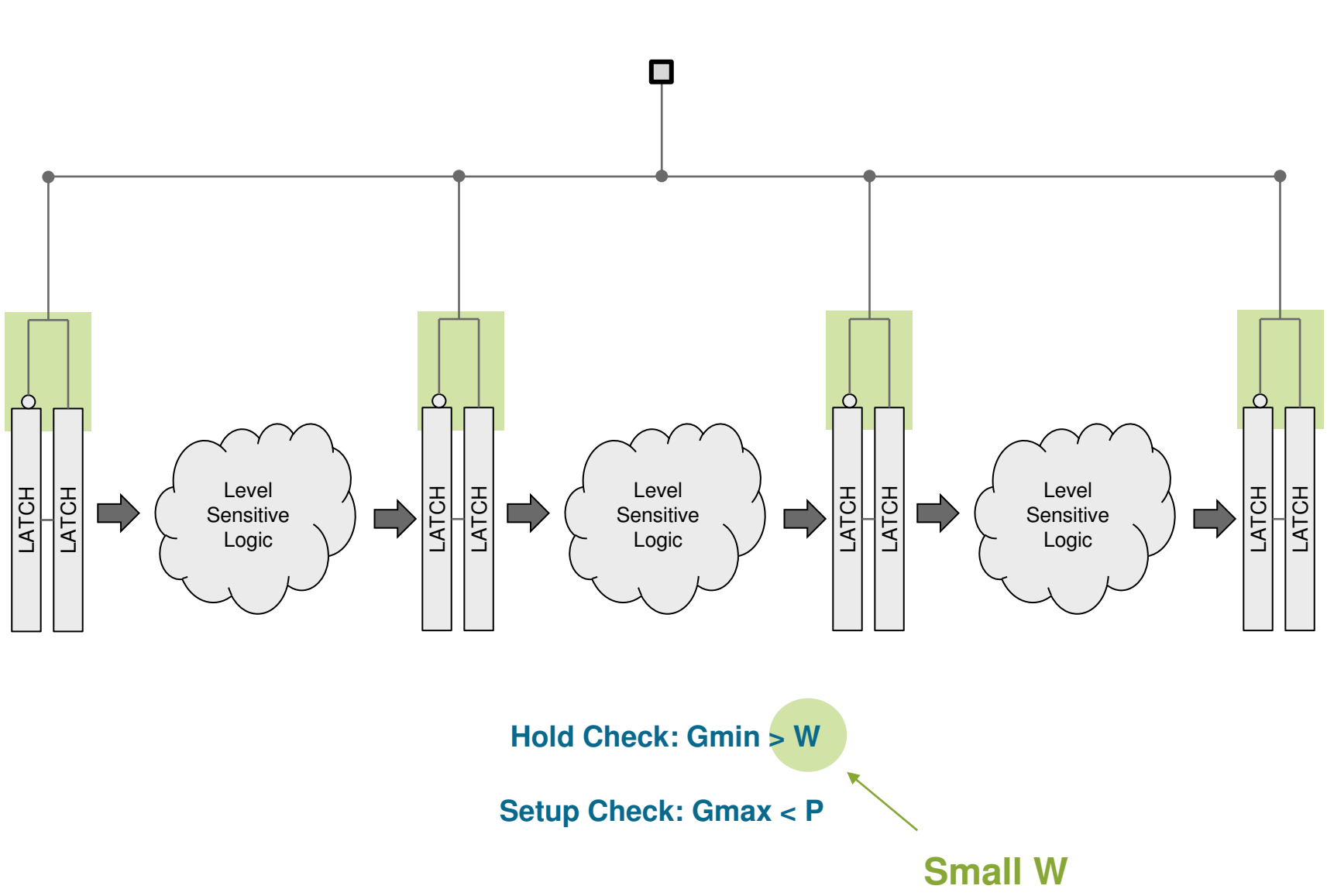
Pulse-flopped Synchronous Design

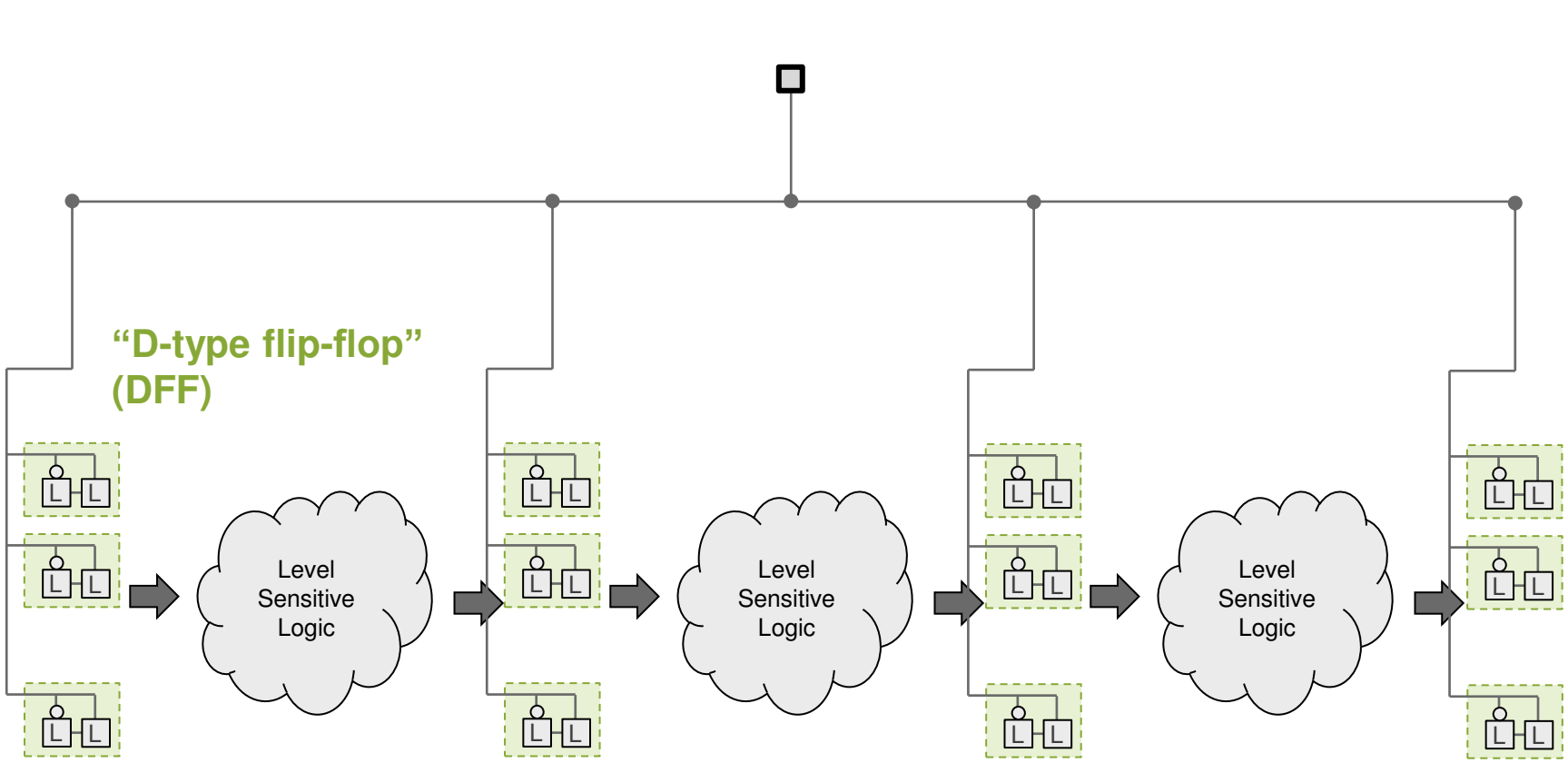


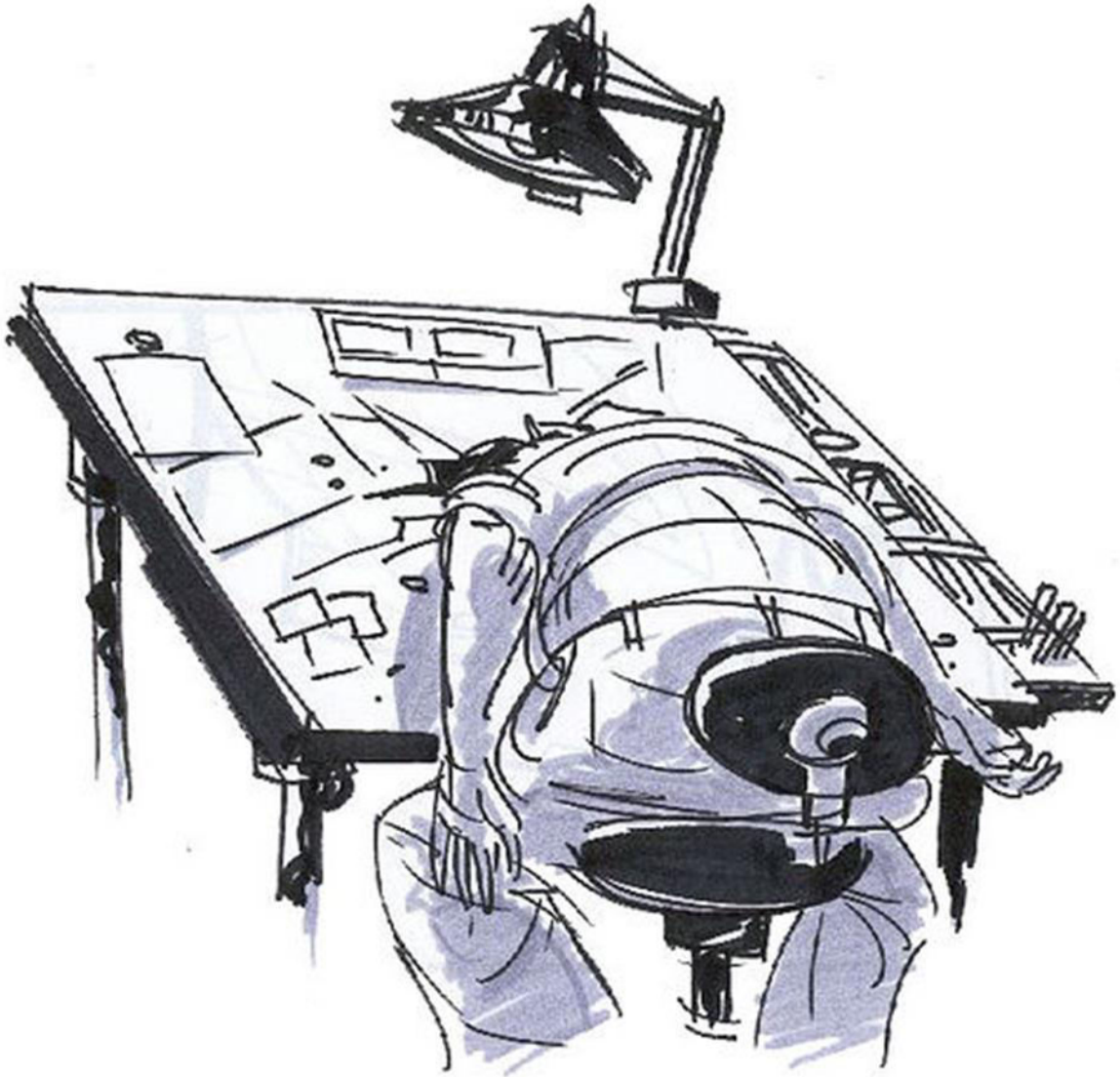
How do you shift values in and out during chip test without violating the hold check?

Hold Check: Gmin > W

Setup Check: Gmax < P


“clock”
LATCH
LATCH
Level Sensitive Logic
LATCH
LATCH
Level Sensitive Logic
LATCH
LATCH
Level Sensitive Logic
LATCH
LATCH
Hold Check: Gmin > W
Setup Check: Gmax < P
Small W


"clock"
"D-type flip-flop"
(DFF)
Level
Sensitive
Logic
Level
Sensitive
Logic
Level
Sensitive
Logic
Hold Check: Gmin > W
Setup Check: Gmax < P
Small local circuit which can be
carefully crafted and characterized
across multiple process corners, voltages, temperatures
Very small and *predictable* W

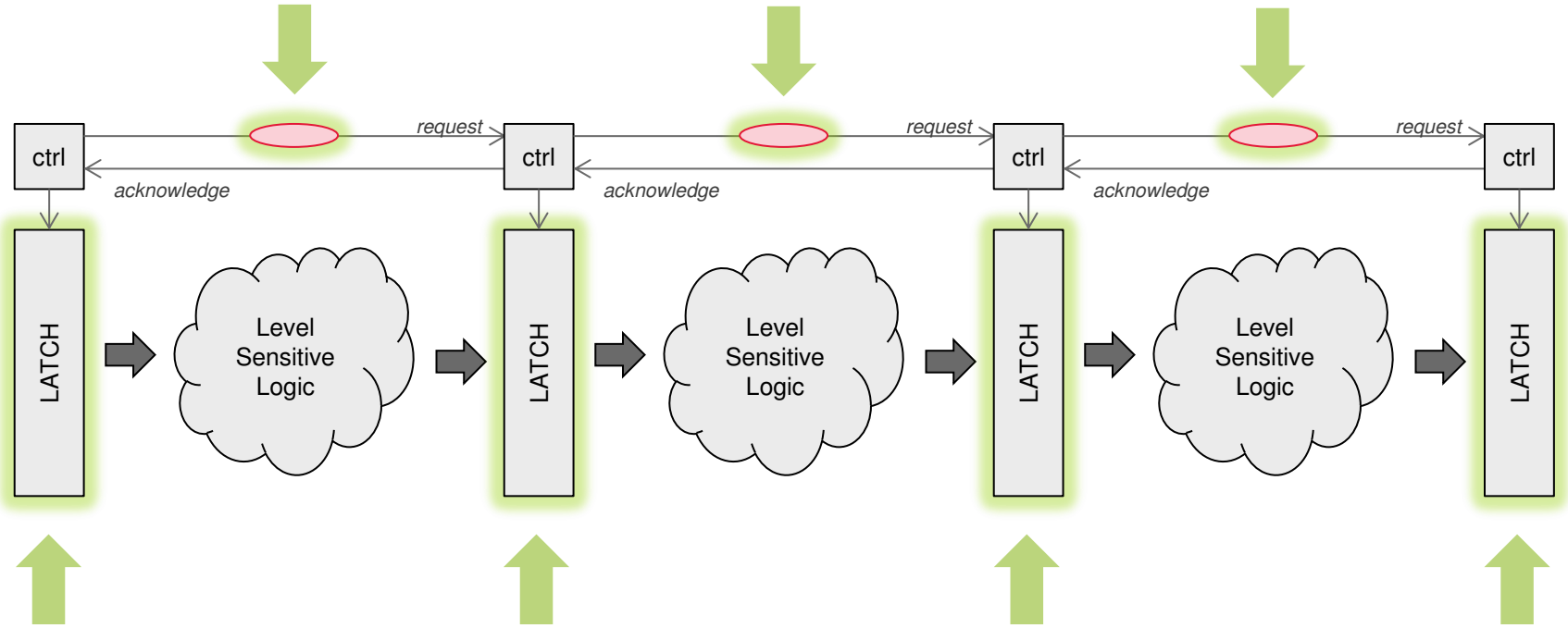


Azuro Day 1

- Some money in the bank from Angel Investors
- 2 employees
- Small Office rented from Cambridge University Computer Lab
- ~~Vision to automatically compile mainstream synchronous designs into low power asynchronous equivalents~~

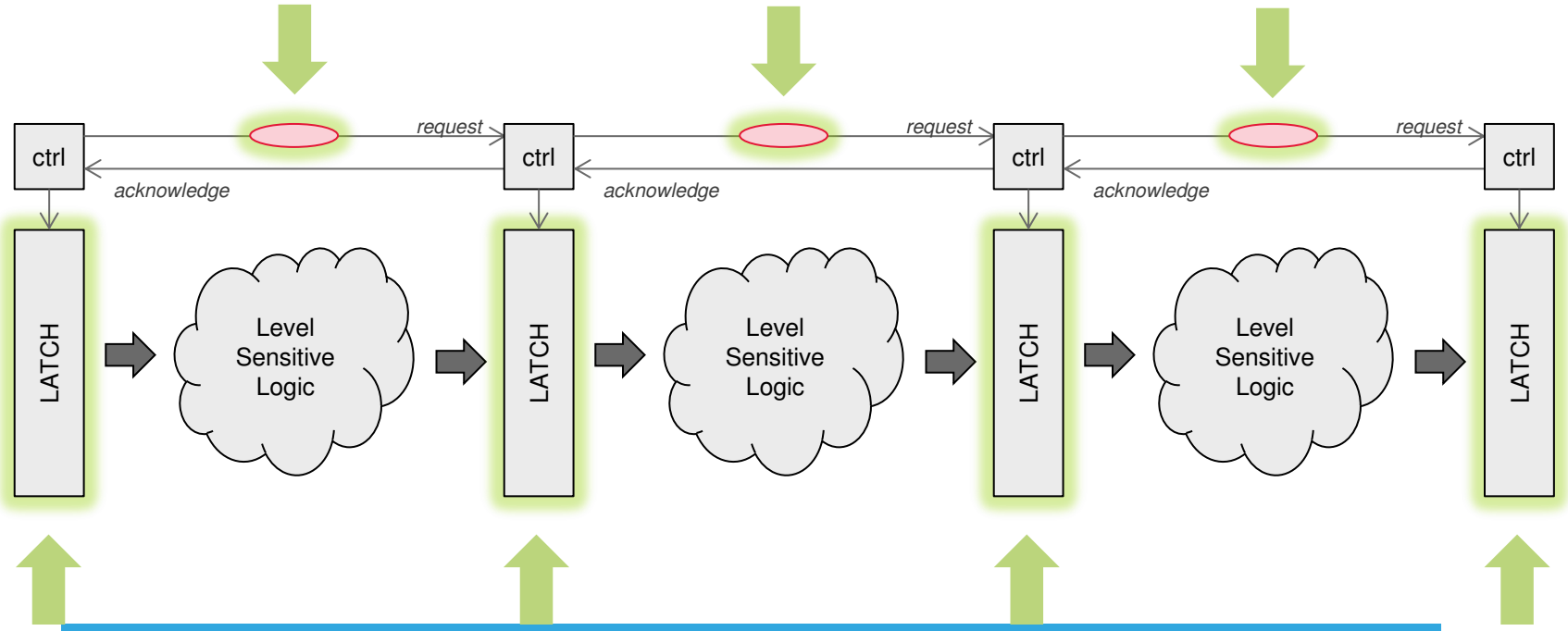
Vision to bring the low power *benefits* of asynchronous design to the mainstream synchronous community

These delays don't all have to be the same



No data no latching
(consume power only when needed)

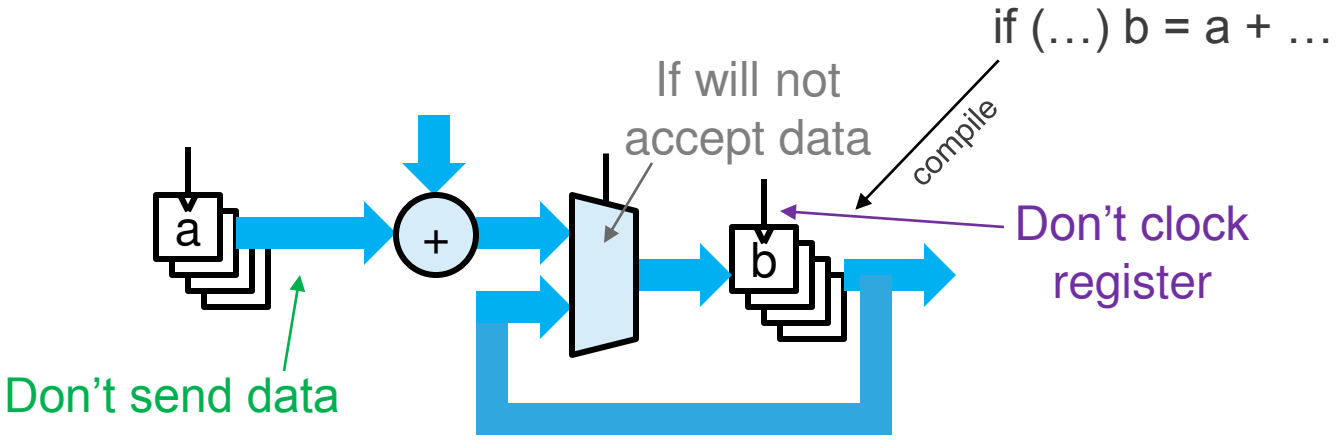
These delays don't all have to be the same



No data no latching
(consume power only when needed)

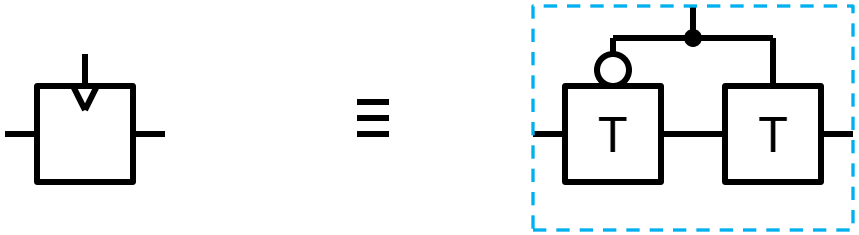
Power Saving Ideas

- Power in the clock
  - Is wasted if the register did not capture new data
  - → Clock gating
- Power in the datapath
  - Is wasted if the result of computation is not captured
  - → Guarding

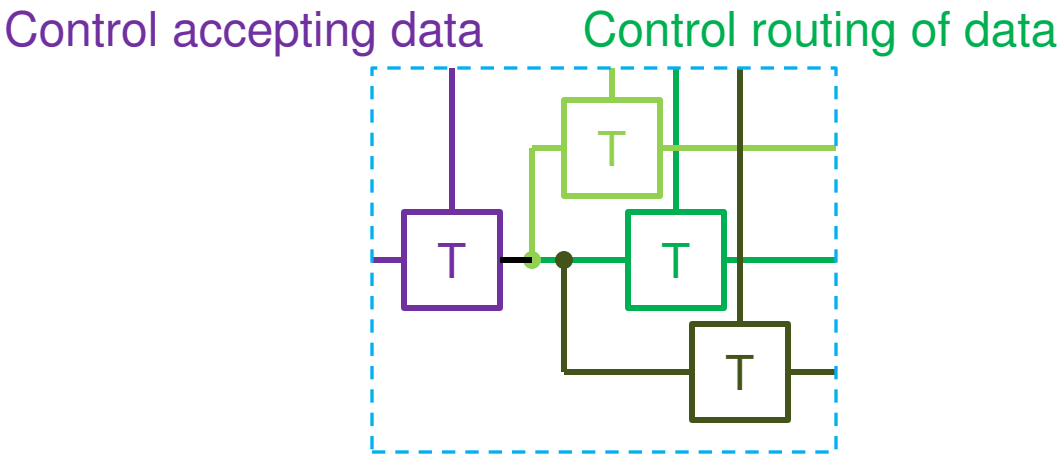


First “Real” Idea: Guard Flops

- Saw that flop can be decomposed



- So create “Guard Flop”

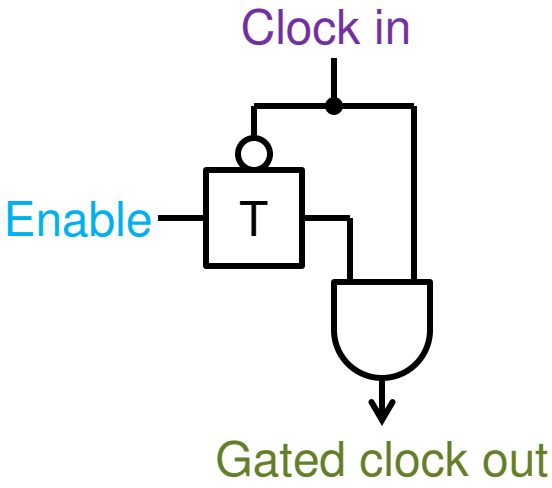


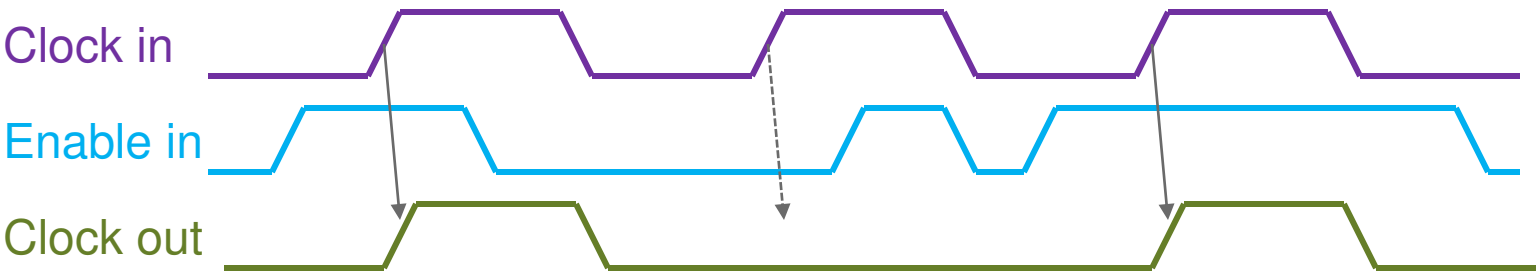
First “Real” Idea: Guard Flops

- Gained two US patents
- Reception was that it was still too disruptive
  - Needed to create new standard cells
  - New Methodologies …
  - Still too Async?
- Needed to come closer to the mainstream

Focus on Clock Gating

- Clock gates are like AND gates with one rising-edge sensitive input





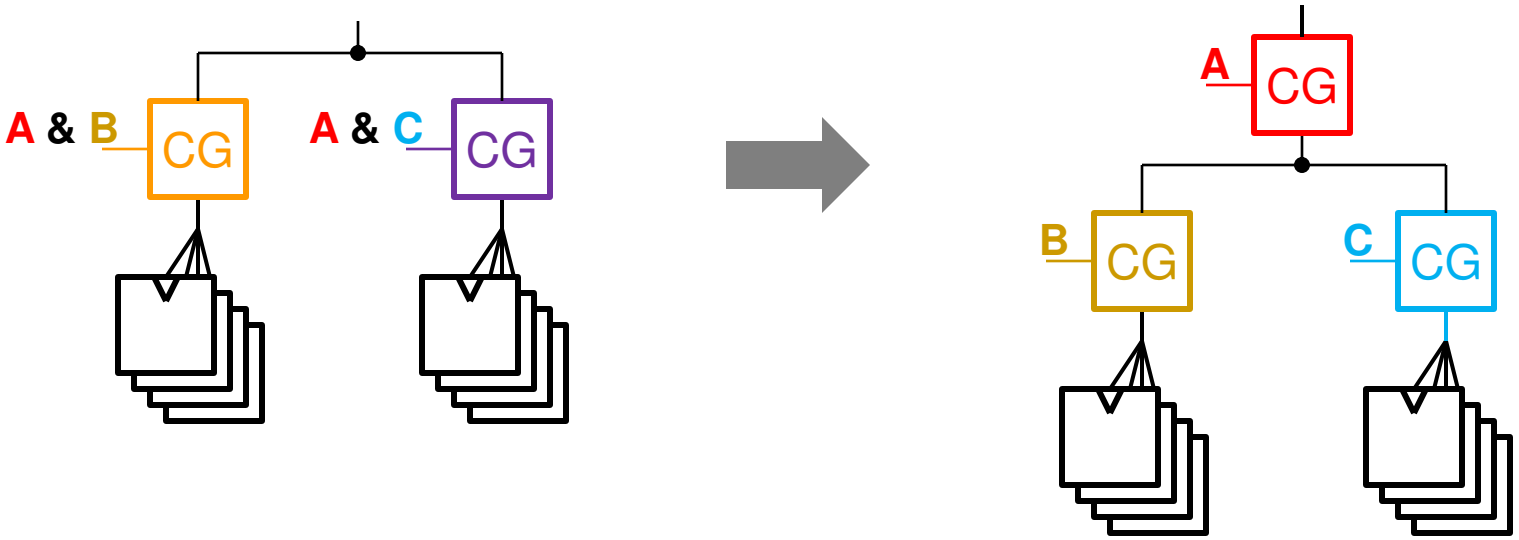
Focus on Clock Gating

- Clock Gating in 2002 was primitive
  - Instantiated directly from RTL expressions only

```
if (a)
  b <= c;
```

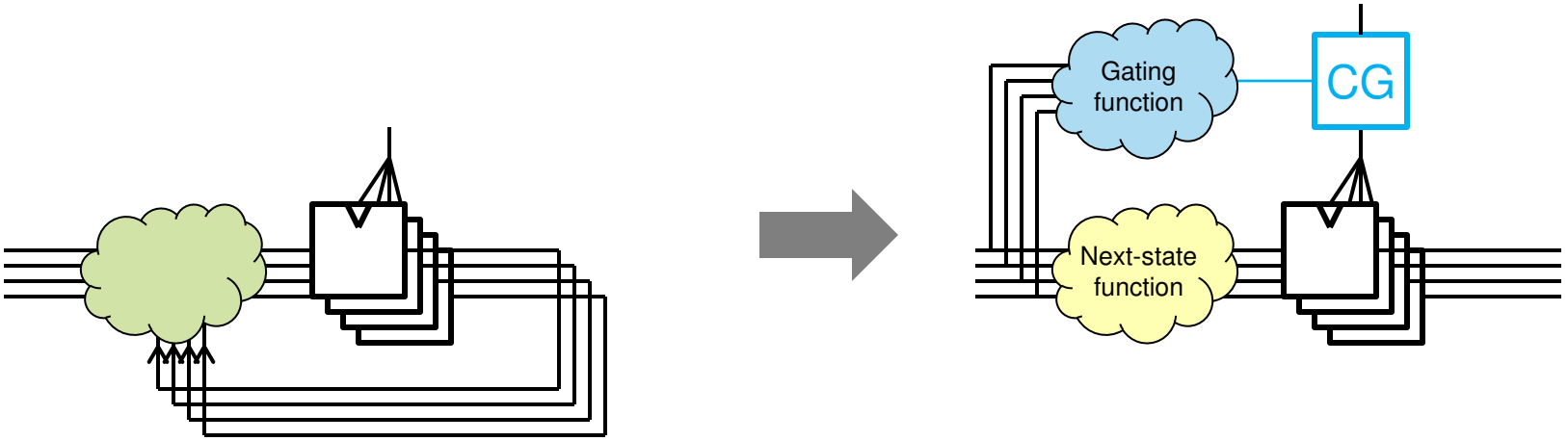
- Three ways to improve:
  - Find and exploit hierarchy
  - Find new register-level expressions not in netlist
  - Find sub-register-level expressions

Hierarchical Gating



- Finding expressions requires BDD-based netlist traversal
  - Rarely as simple as observing an AND gate in the netlist!
- Multiple levels of hierarchy usually available
- Heuristics to limit levels to 2 or 3 to avoid clock impact

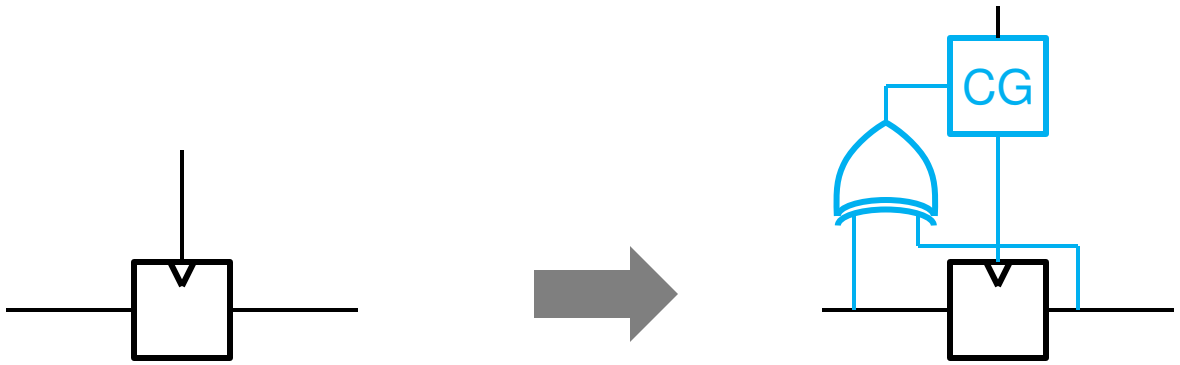
Symbolic Gating



- Use BDDs to decompose existing D-side function
  - Gating function for whole register
  - Next-state function assuming gating function pulled out
- May need to avoid decomposition
  - If timing is tight, might need to pull out CG late in the day, and put back in old (green) function

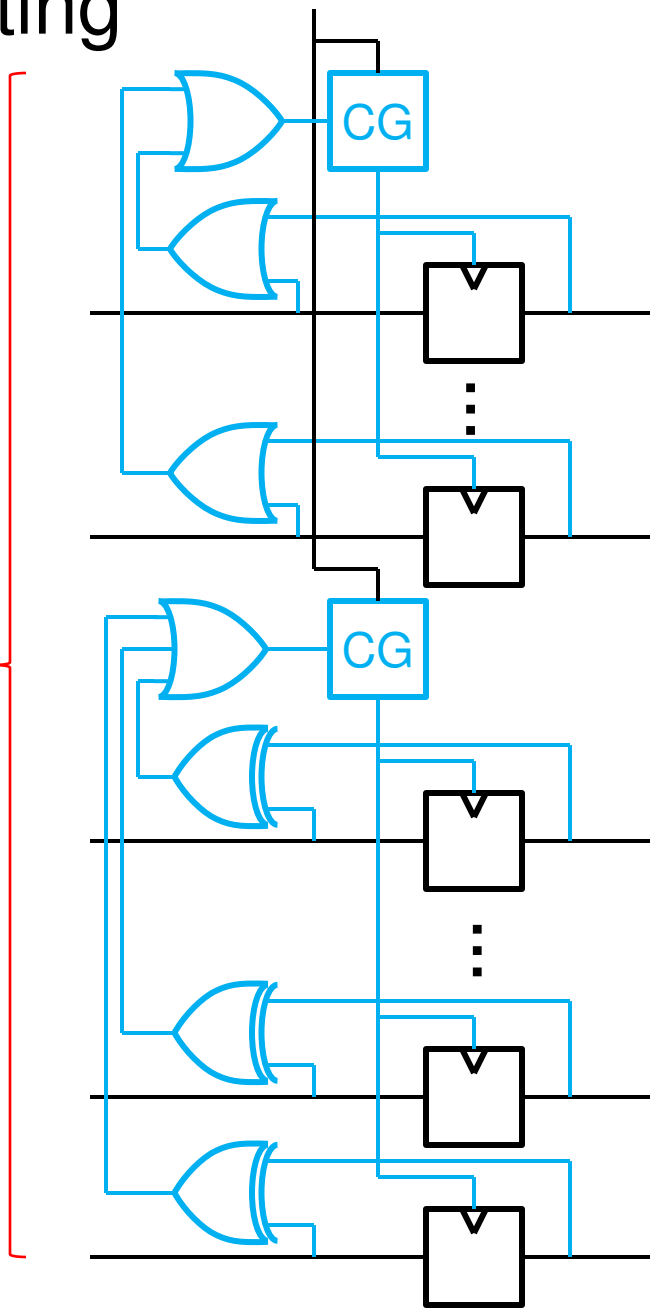
Structural Gating

- Generalization of previously-published work:



- On it's own, this is useless
  - Clock cap of CG as much or larger than flop!
- But can be combined with NAND, OR, and across subsets of registers

Structural Gating

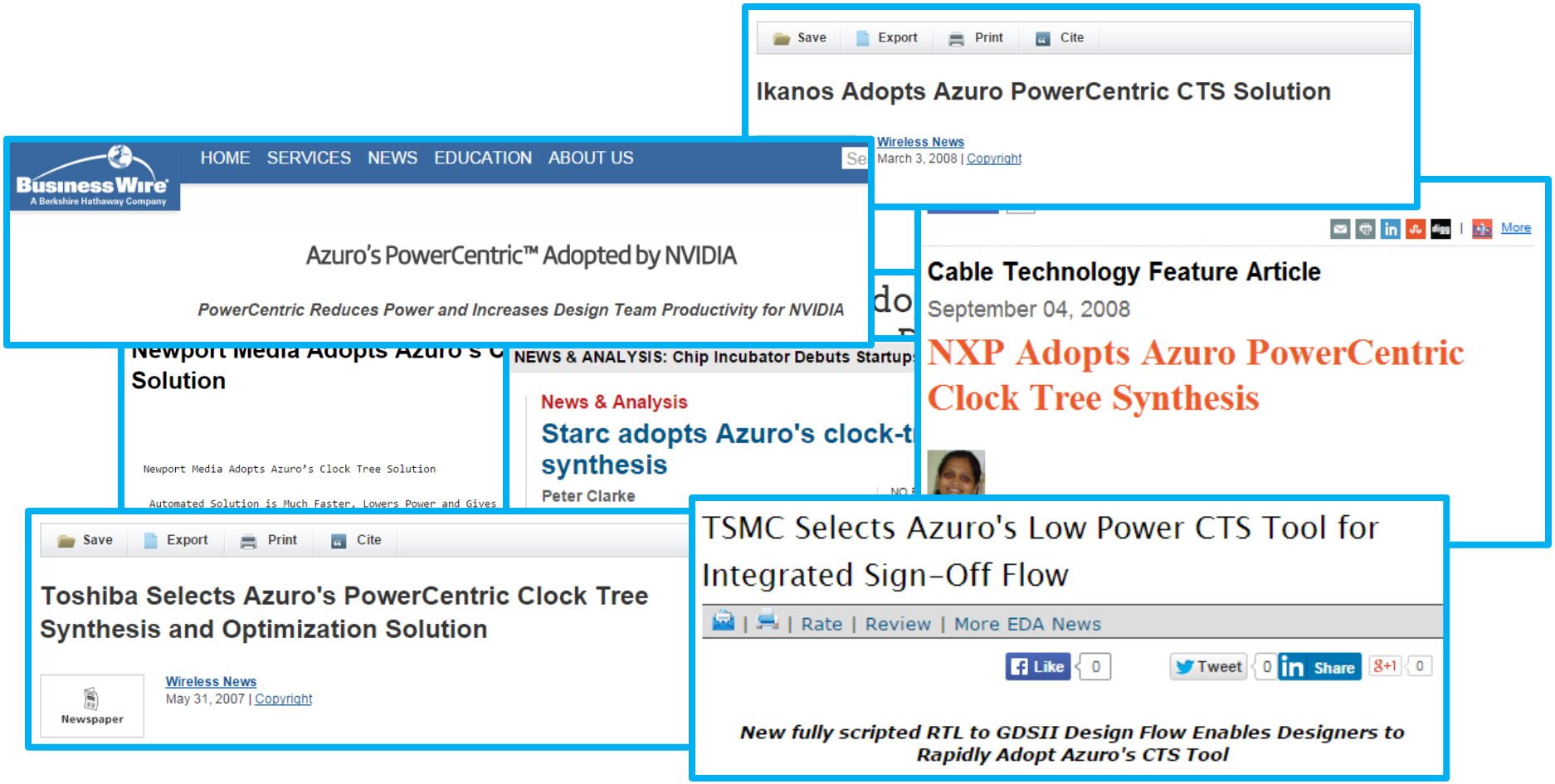


Register split into number of clusters, depending on simulation information (SAIF, VCD)

Some parts of register may stay at logic 0 more than logic 1
➔ Use OR gate for comparison

Key is to make the whole thing **activity-driven**

First Product: PowerCentric



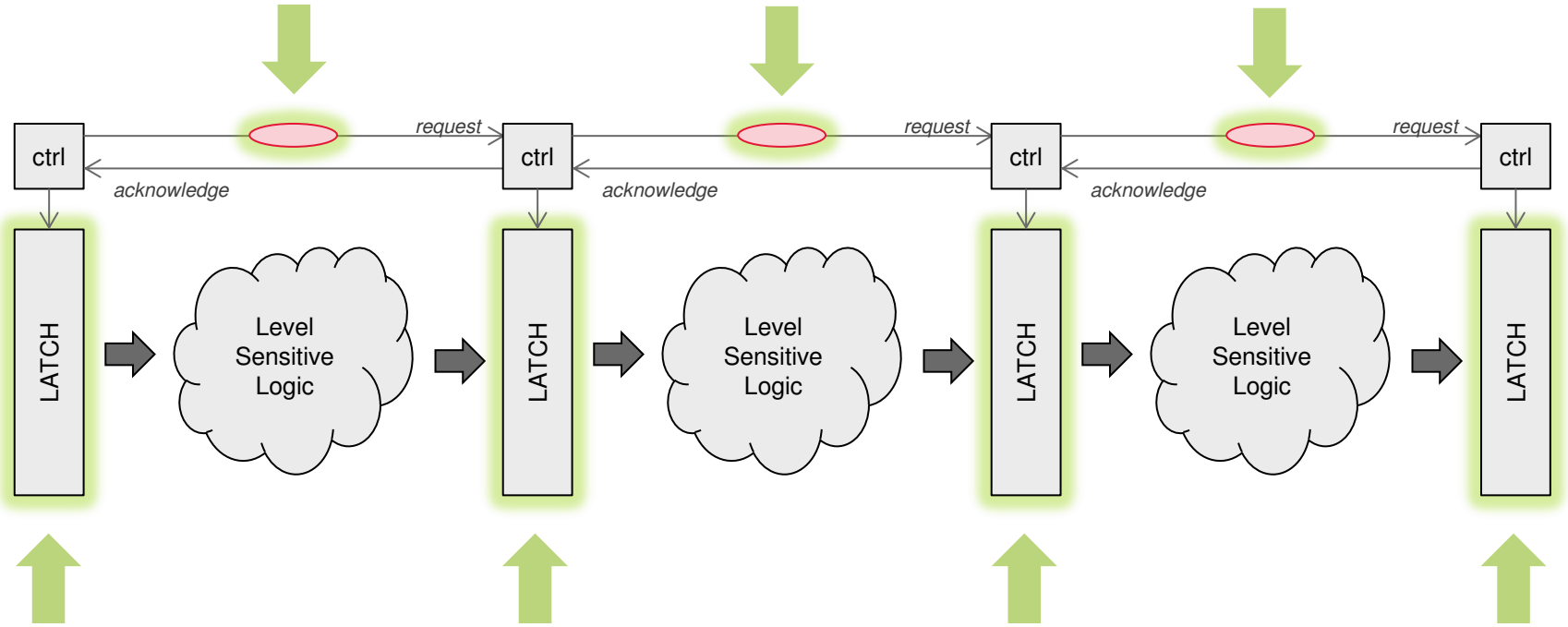
VICTORY!

First Product: PowerCentric

First Product: PowerCentric

Launch abort! Hunker down for phase 2

These delays don't all have to be the same



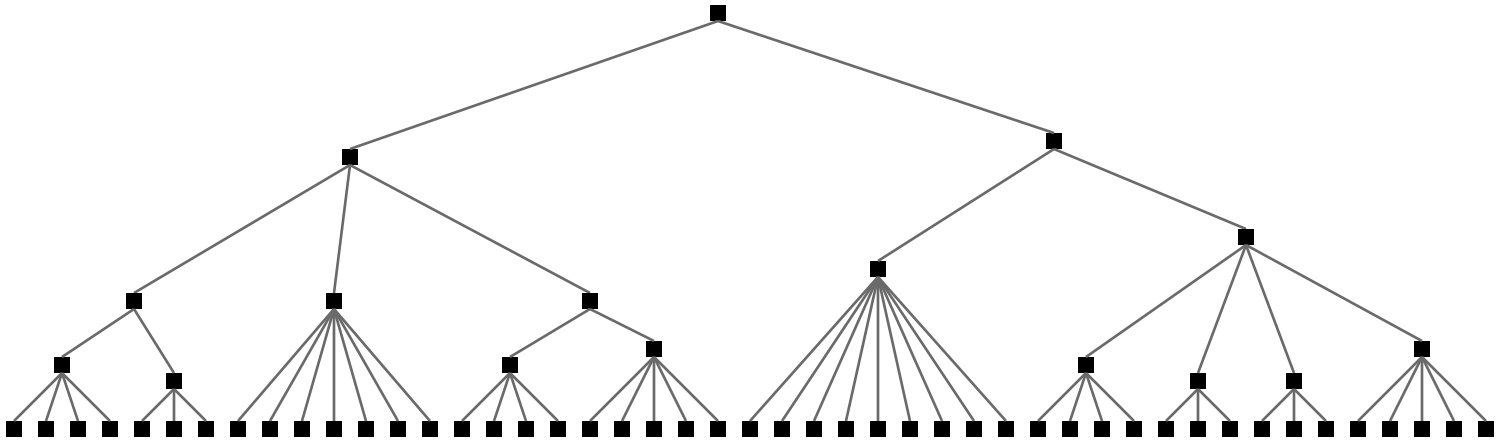
No data no latching
(consume power only when needed)

2004, In Azuro R&D

- Could now gate clock far more efficiently than any competitor
- However, we couldn’t estimate effect on the clock tree
- Needed to generate our own clock tree synthesis algorithm
  - Naturally, we used our Async experience …

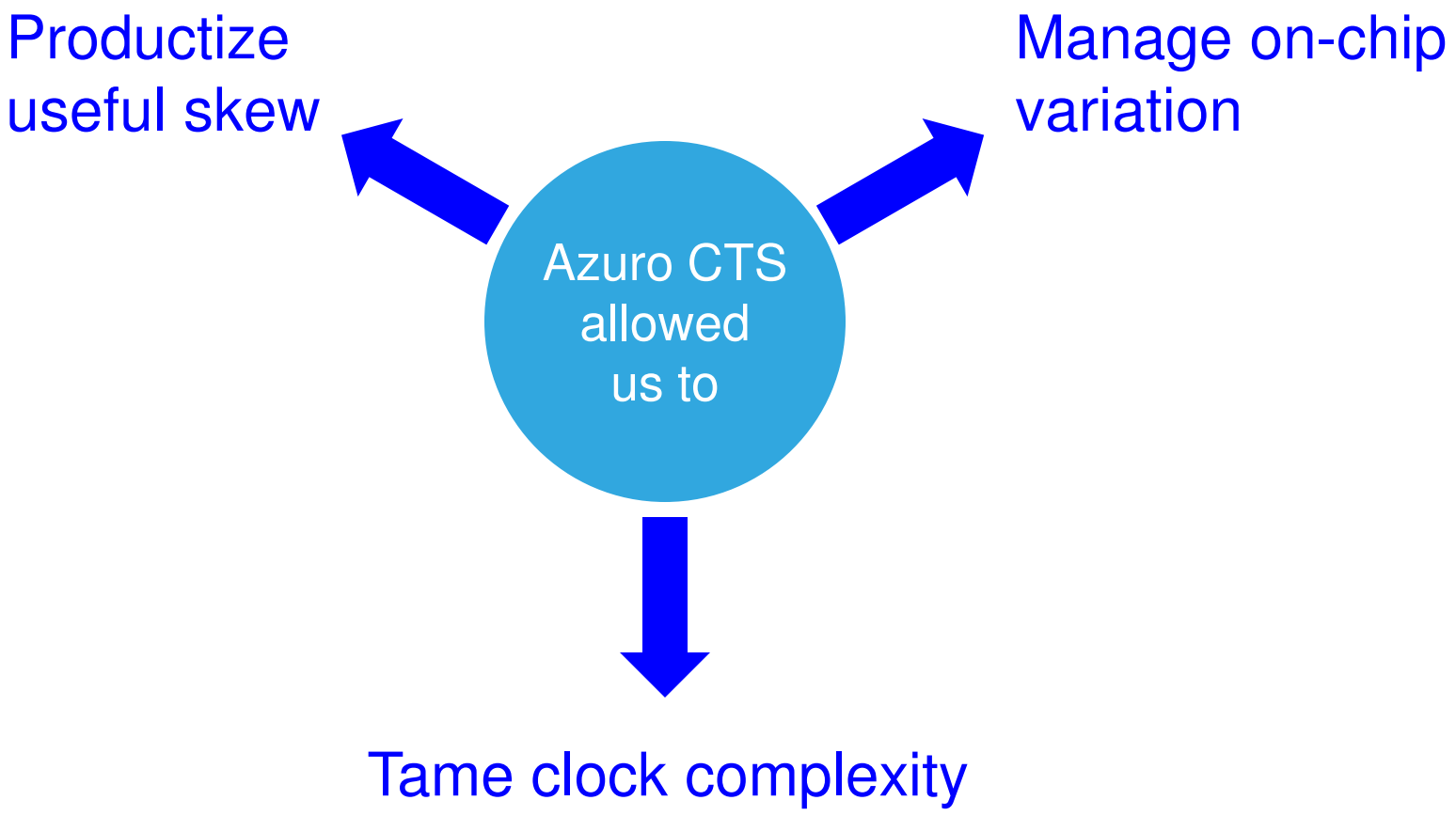
Azuro CTS

- Bottom-up clustering approach
  - Cluster lower drivers together first
  - More flexible, less “synchronous” in a way

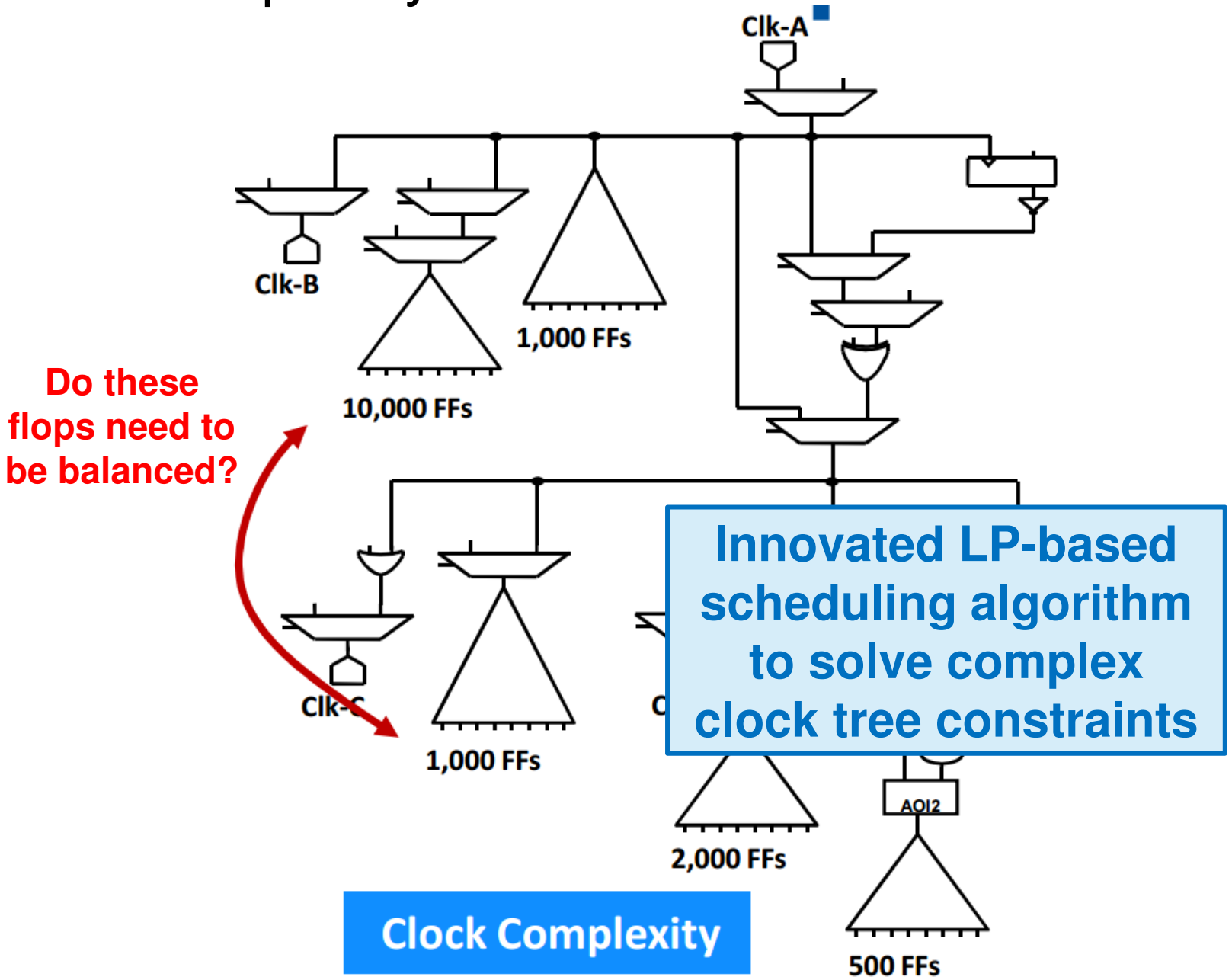


- Ideally suited to deep clock gating
  - Also ~10% lower power
  - But has other benefits

Azuro CTS

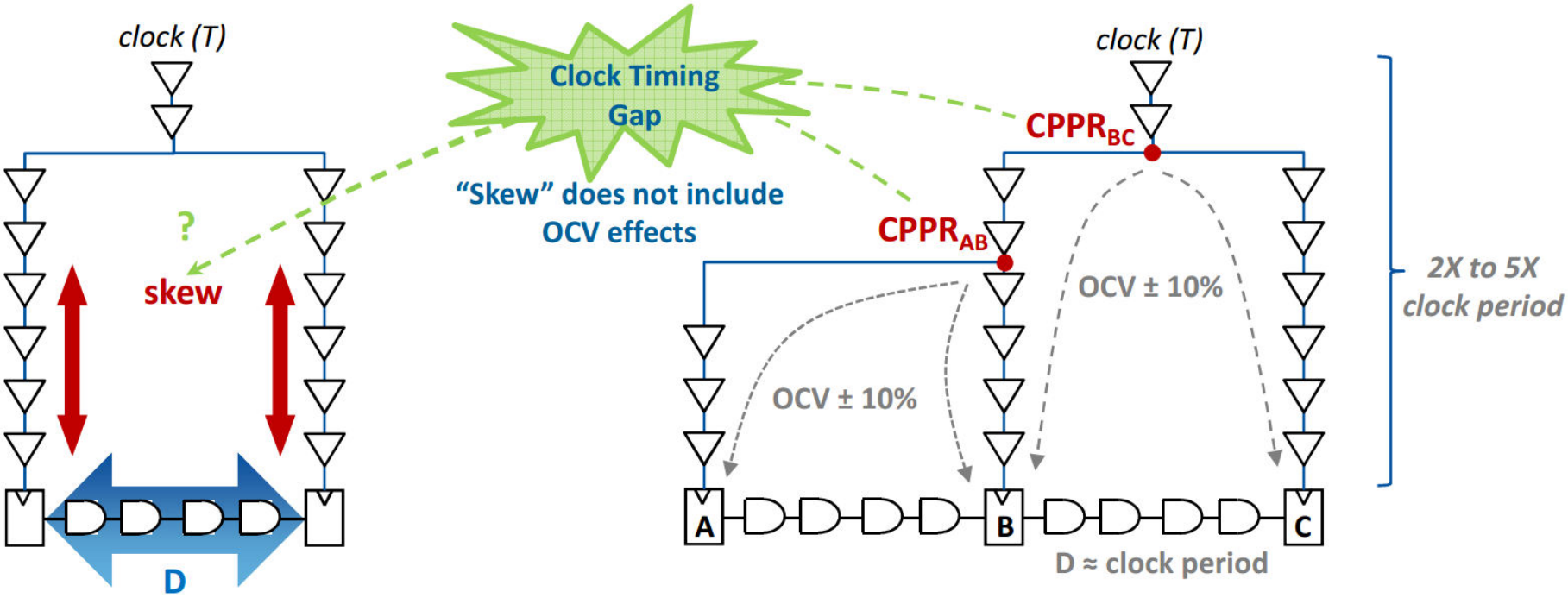

Productize
useful skew
Manage on-chip
variation
Azuro CTS
allowed
us to
Tame clock complexity

Clock Complexity



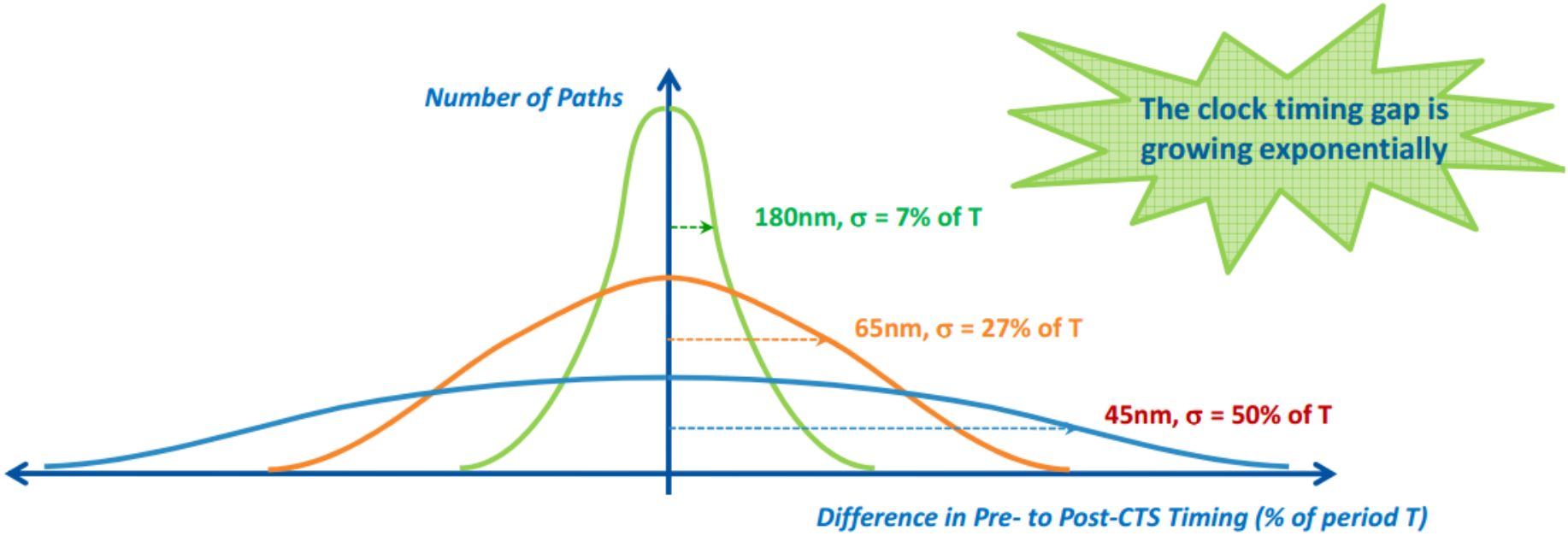
On-Chip Variation

OCV derates hidden until CTS



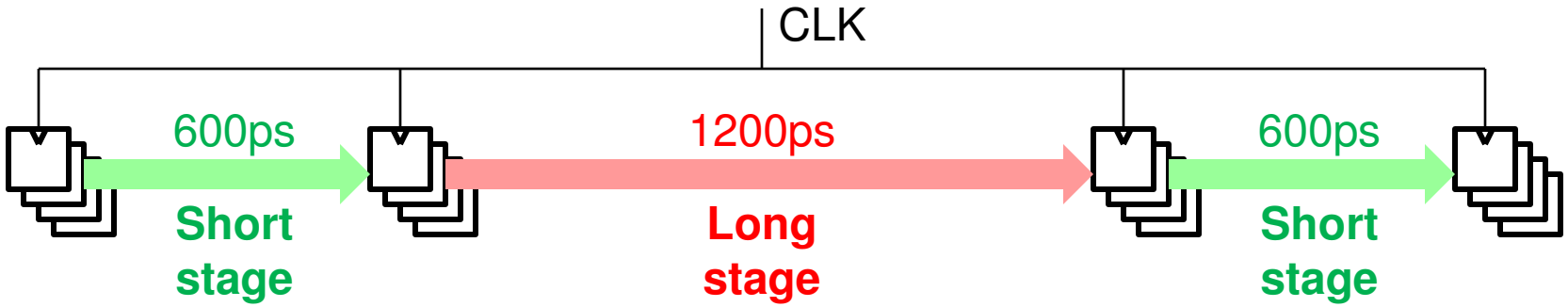
OCV + Clock Complexity = Clock Timing Gap

Propagated clocks timing and ideal clocks timing are diverging



Useful Skew

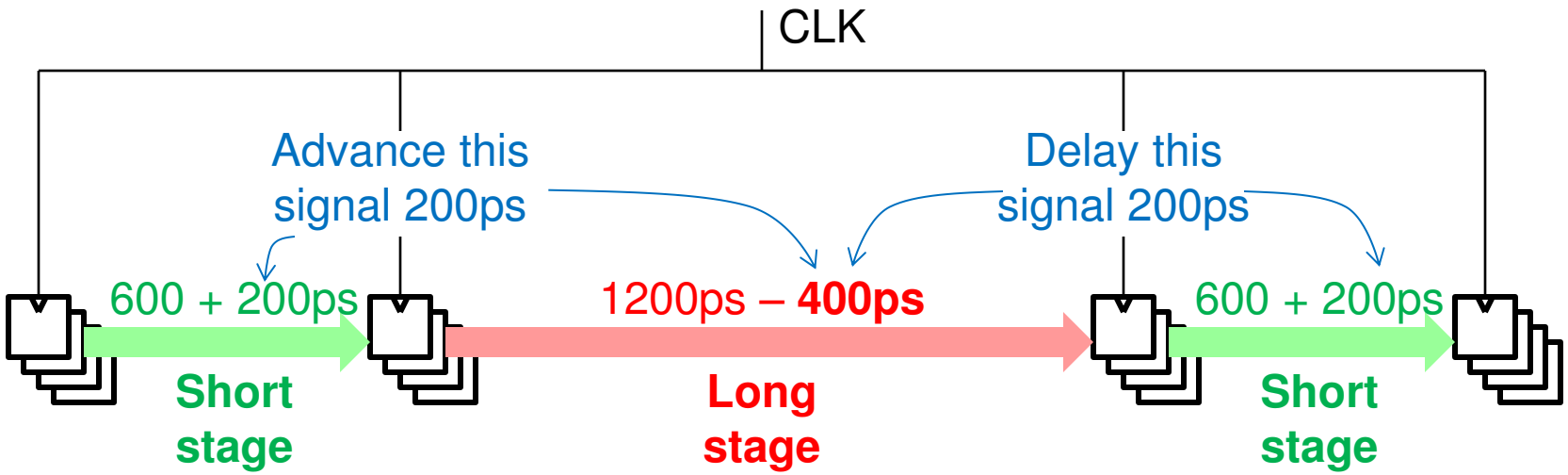
- Why does the clock need to arrive at every point at the same time?



- Cycle time: 1200ps

- Think Async: What would we do?

Useful Skew

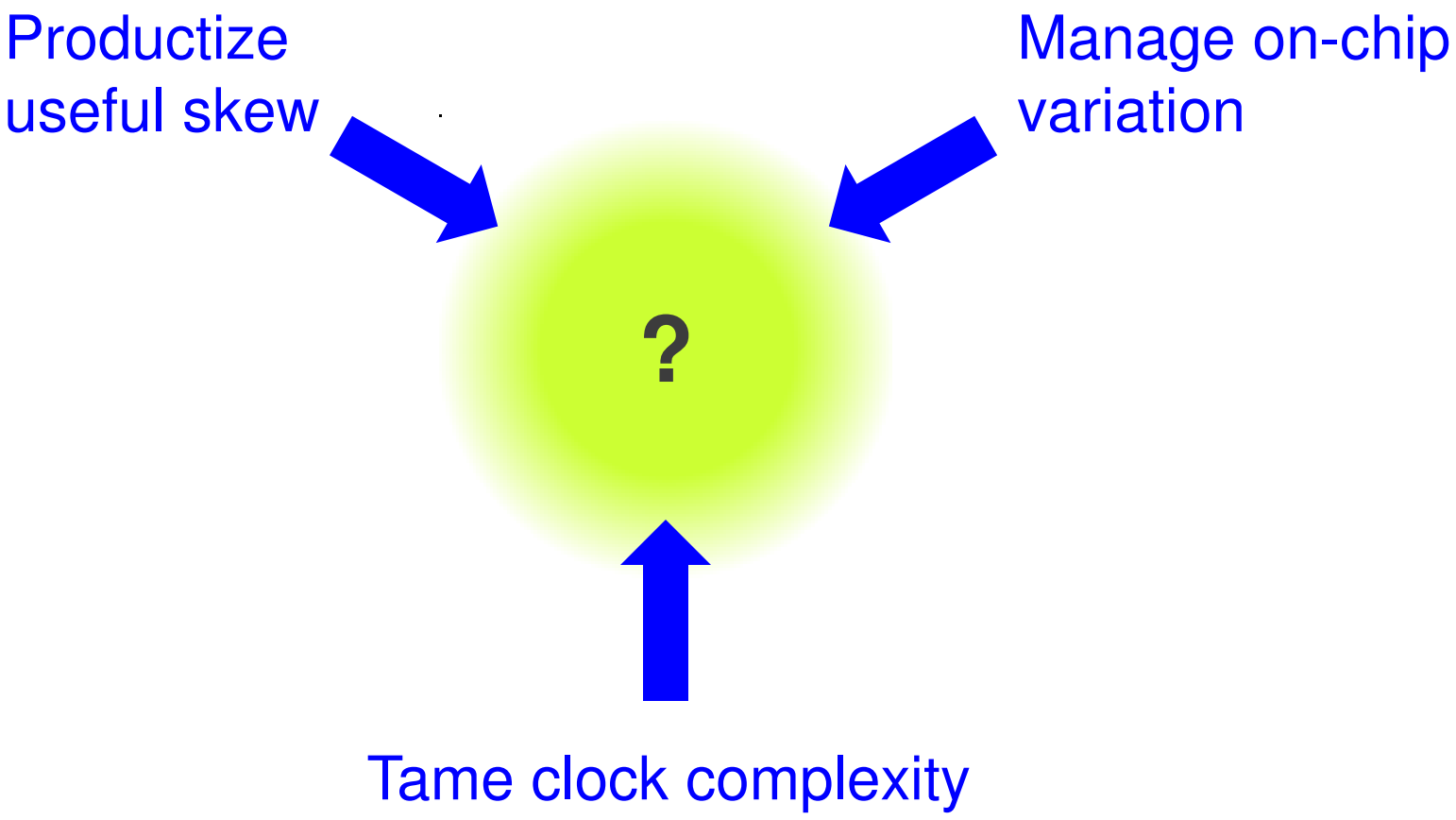


- Cycle time now: 800ps
  - 50% frequency improvement!

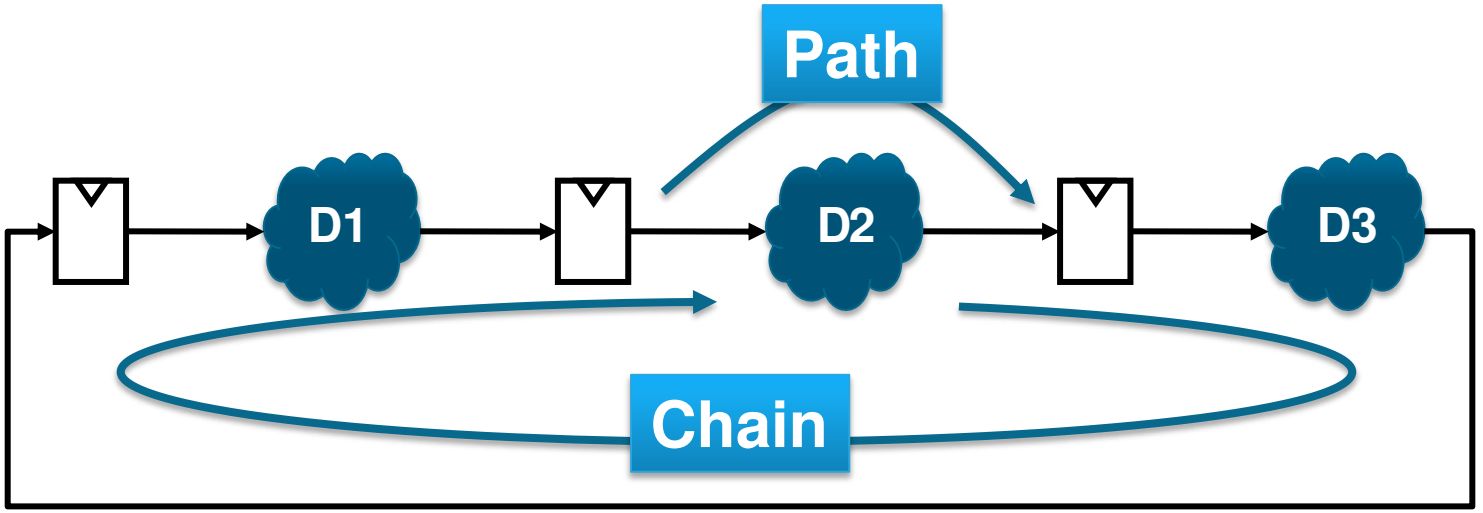
Useful Skew

- Has been tried before
  - EDA vendors had useful skew engines
  - Numerous academic papers on this
- Focus was either too lowbrow or too highbrow
  - Tradtional approach was adding individual buffers. No ability to decrease delay. Buffers cost power.
  - Academic approach was solving chip-wide LP-style problems. Too hard for “real” chips (1M+ instances).
  - No consideration of complex clocks
  - No consideration of on-chip variation

Azuro CTS

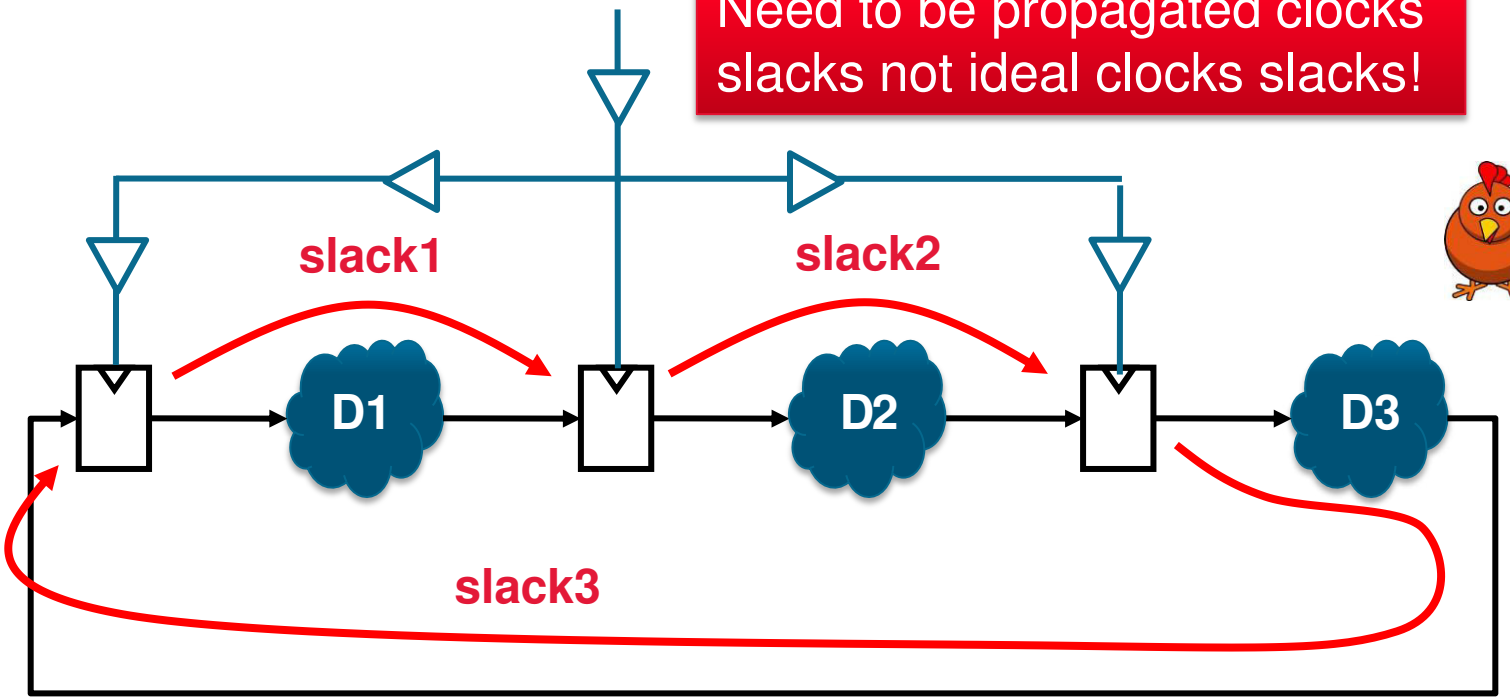

Productize
useful skew
Manage on-chip
variation
?
Tame clock complexity

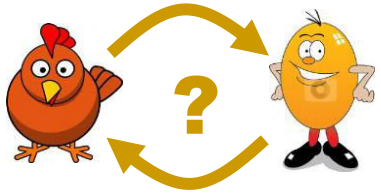
Traditional Design → **Worst Path Closure**
Speed limited by **max D**



Clock Concurrent Design → **Worst Chain Closure**
Speed limited by **avg D**

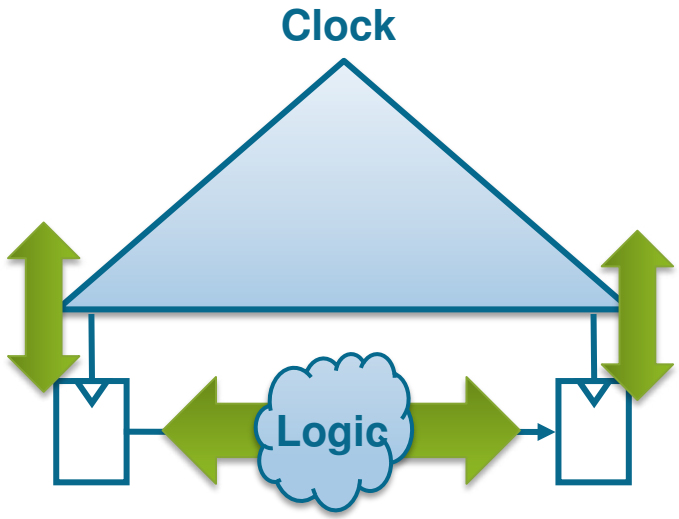
Need to be propagated clocks slacks not ideal clocks slacks!





slack1 = slack2 = slack3

Clock Concurrent Optimization (CCOpt)



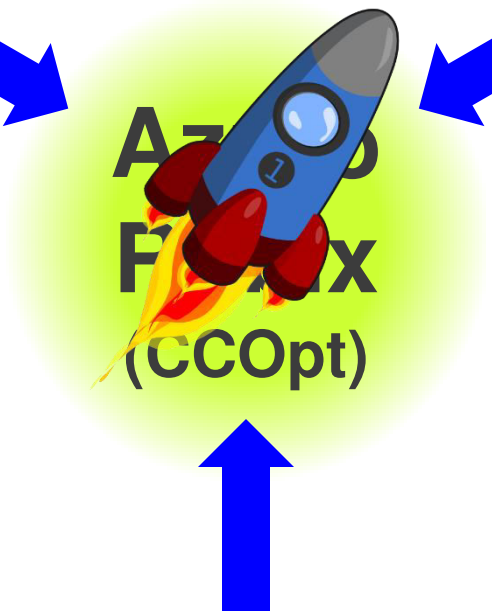
Skew clocks and optimize logic concurrently

Always using propagated clocks slacks

Azuro CTS

Productize
useful skew

Manage on-chip
variation



Tame clock complexity

Cadence Acquires Azuro

Disruptive Technology Strengthens Cadence Digital Implementation Flow, Brings QoS in Power, Performance and Area to Next-gen SoC Designs

SAN JOSE, Calif. , 12 Jul 2011

Cadence Design Systems, Inc. (NASDAQ: CDNS), a leader in global electronic design innovation, today announced it has acquired Azuro, Inc., a company that has pioneered a paradigm shift in the digital implementation and optimization of next-generation SoCs. Azuro offers unique clock concurrent optimization technology, also known as ccopt, which delivers superior capabilities for designers faced with increasing performance, power and area challenges. Specifically, ccopt technology has delivered significant quality of silicon (QoS) on high-speed processor designs in the areas of:

- Power (clock tree power reduction up to 30 percent and total power improvements of up to 10 percent),
- Performance (improvements of up to 100 MHz for a GHz design), and
- Area (clock tree area reduction up to 30 percent)

Thank You