Towards PVT-Tolerant Glitch-Free Operation in FPGAs

Safeen Huda and Jason H. Anderson

ECE Department, University of Toronto, Canada

24th ACM/SIGDA International Symposium on FPGAs

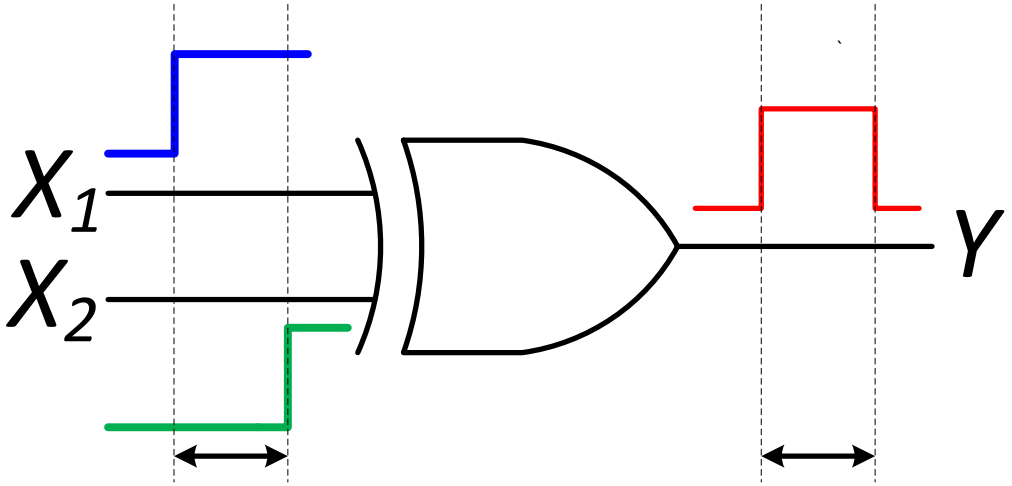
February 22, 2016

Motivation

*Google data centre

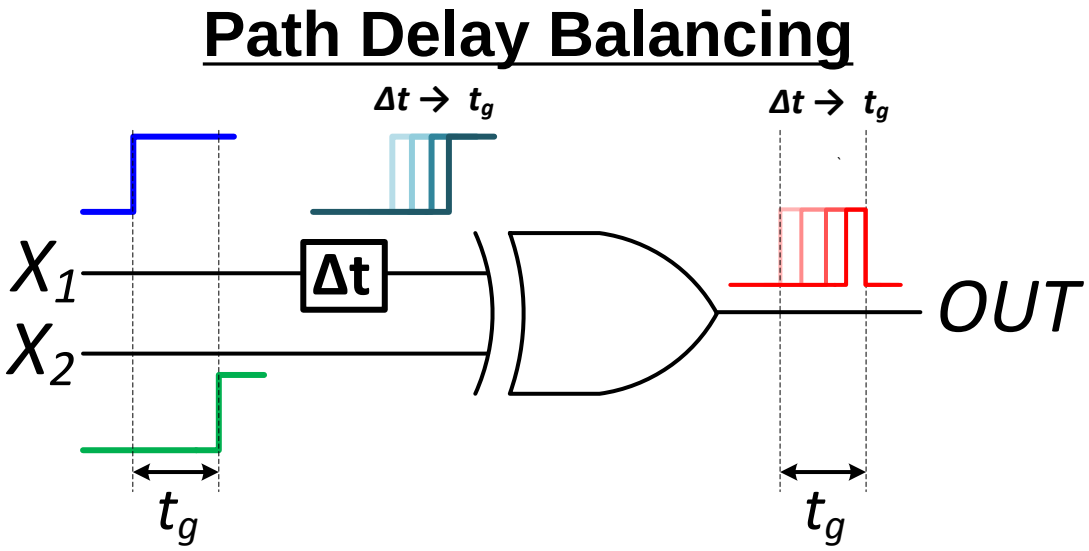
- FPGA power increasingly critical because of new markets
  - Data centers
  - Mobile electronics
- Previous studies have shown glitch power is significant contributor to overall dynamic power consumption:
  - Shum et al. [1] : ~26% of core dynamic power
  - Lamoureaux et al. [2]: ~30% of signal transitions are due to glitches

Reviewing…



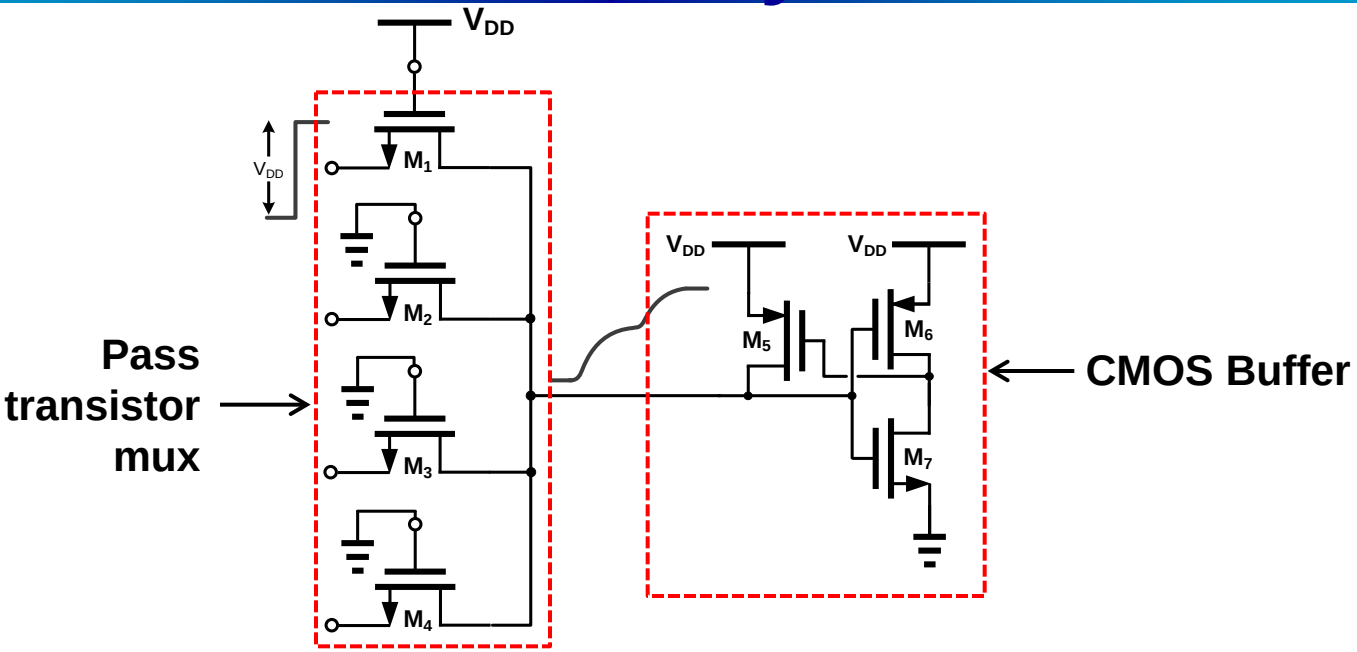
- Unequal arrival times on input pins of combinational logic may result in spurious transitions
- These spurious transitions dissipate power

Previous Approaches



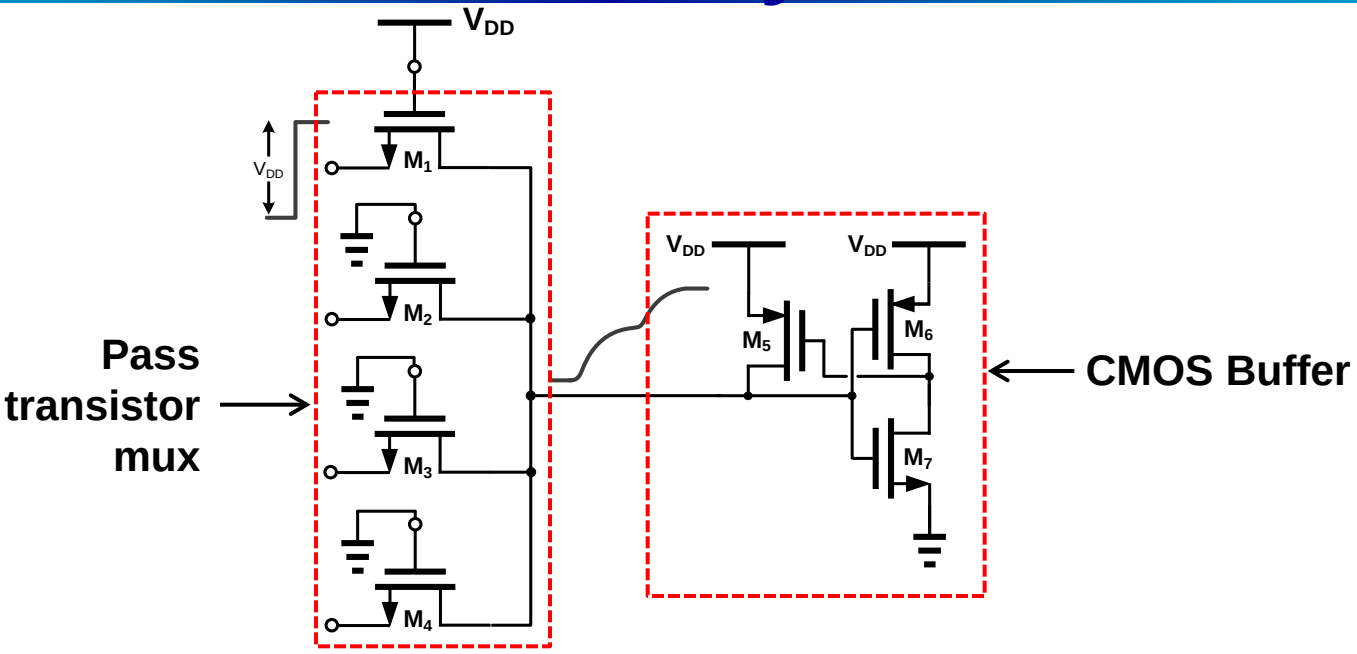
- Add delay to fast-arriving inputs, equalize delay difference
- Delay balance circuitry may have low overhead [2,3]

Delay Predictability of FPGA Circuitry



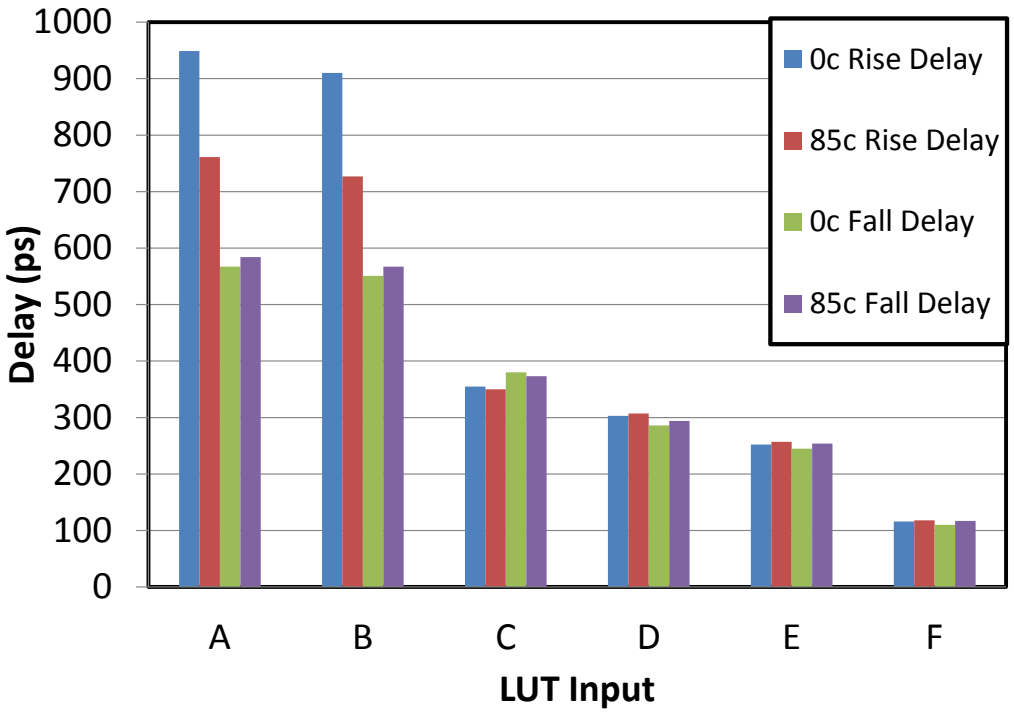
- FPGAs consist of CMOS and pass-gate based circuitry
- CMOS gates
  - Rise/Fall delay inversely proportional to μ, ($V_{DD} - V_T$)
- NMOS pass-gates
  - Fall delay similar to CMOS
  - Rise delay has ***inverse quadratic relationship*** with (Vdd – V_T)

Delay Predictability of FPGA Circuitry



- Two styles of circuitry have different sensitivities to temperature
- Rise/Fall delay balance hard to ensure with NMOS pass-gates

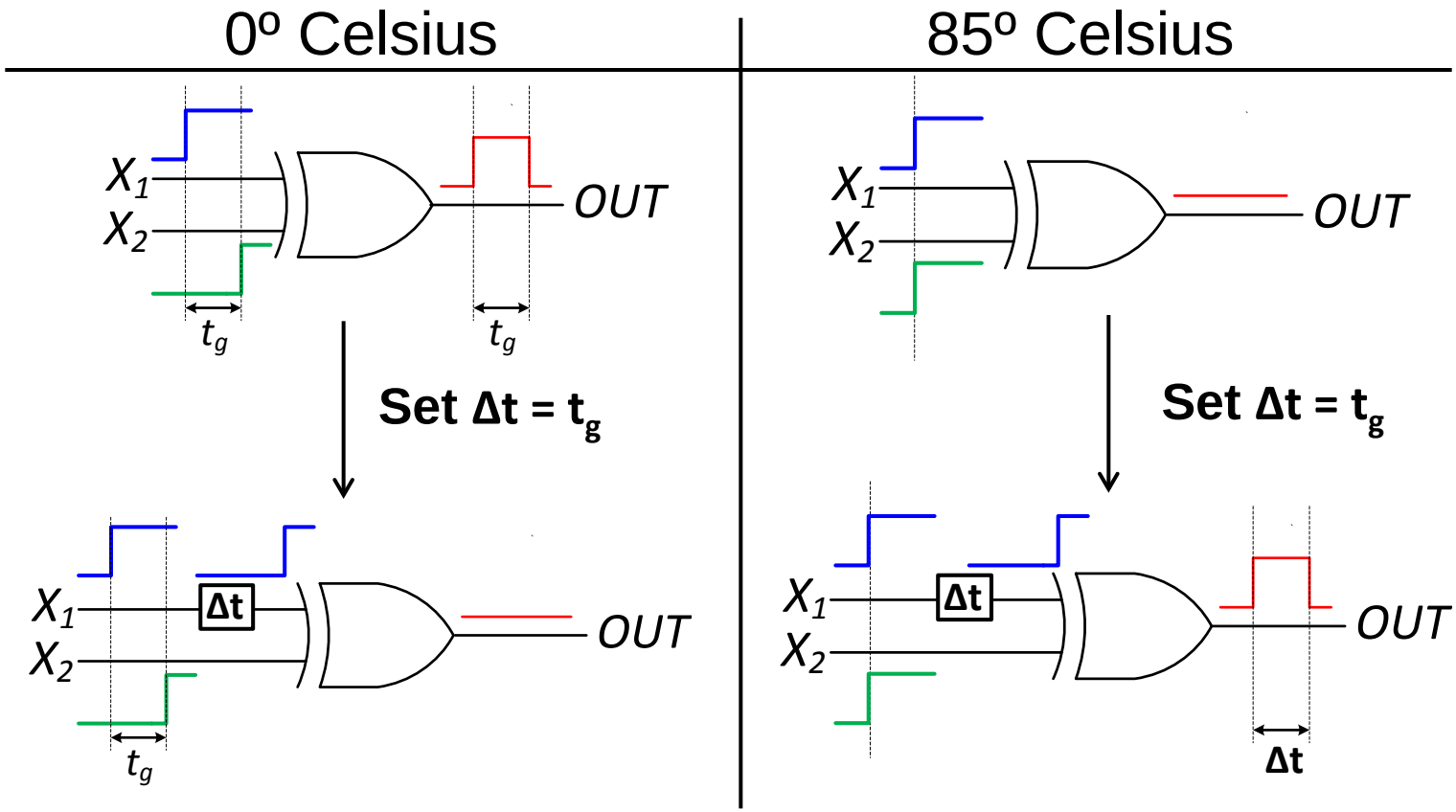
LUT Delay Characteristics



- Stratix-III LUT delays for each input for different temp.
- Inputs A & B have negative temperature dependence, large rise/fall delay imbalance
- Inputs C – F have different delay characteristics

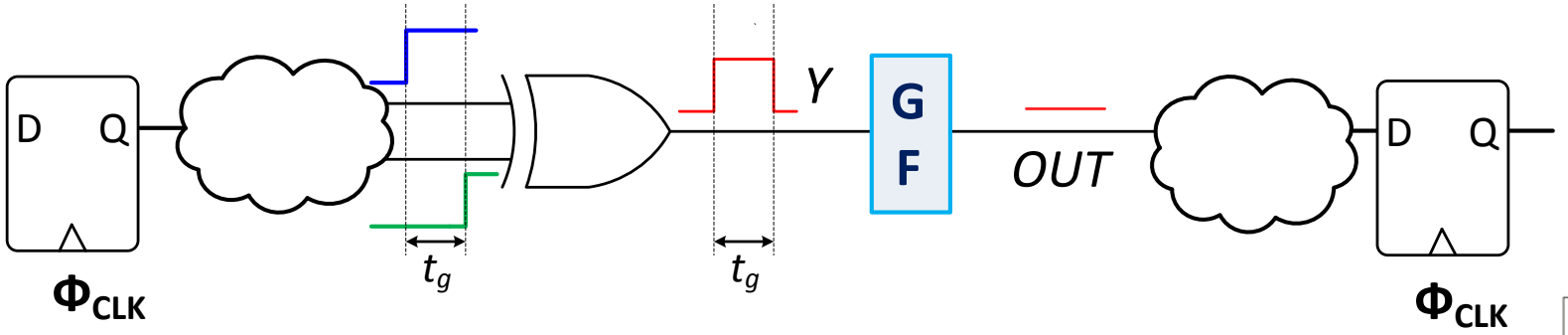
Limitation of Path Delay Balancing

- Difficult to balance input delays over all logic states and operating conditions!

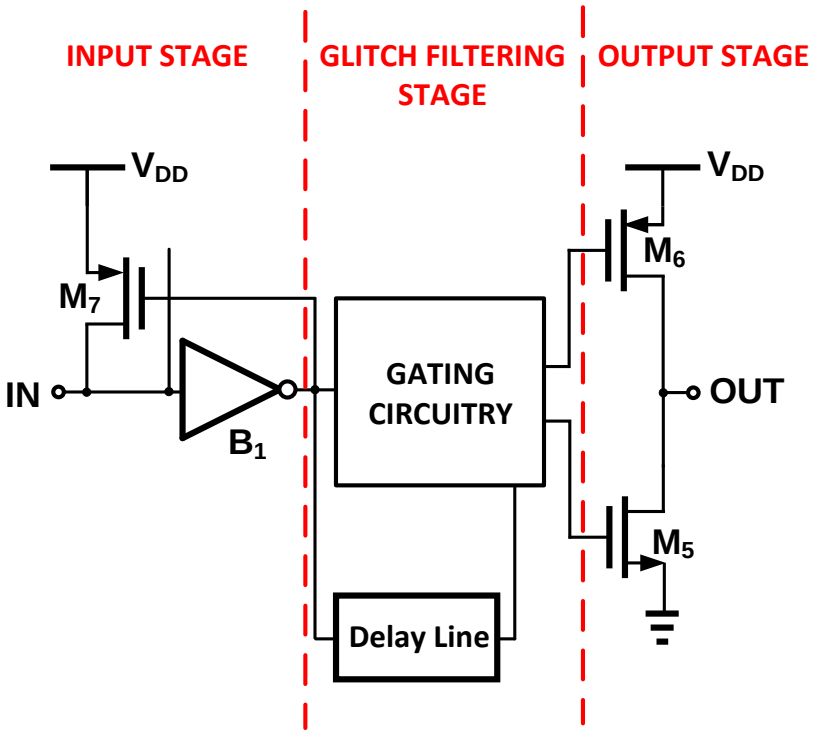


Proposed Glitch Reduction Technique

- Want:
  - Reliable glitch reduction over all PVT corners
  - Glitch reduction technique applicable over a wide range of circuits
  - Low cost
- This work:
  - Replace glitch filtering FF/latch with asynchronous element
  - Eliminates cost and granularity problems of synchronous technique

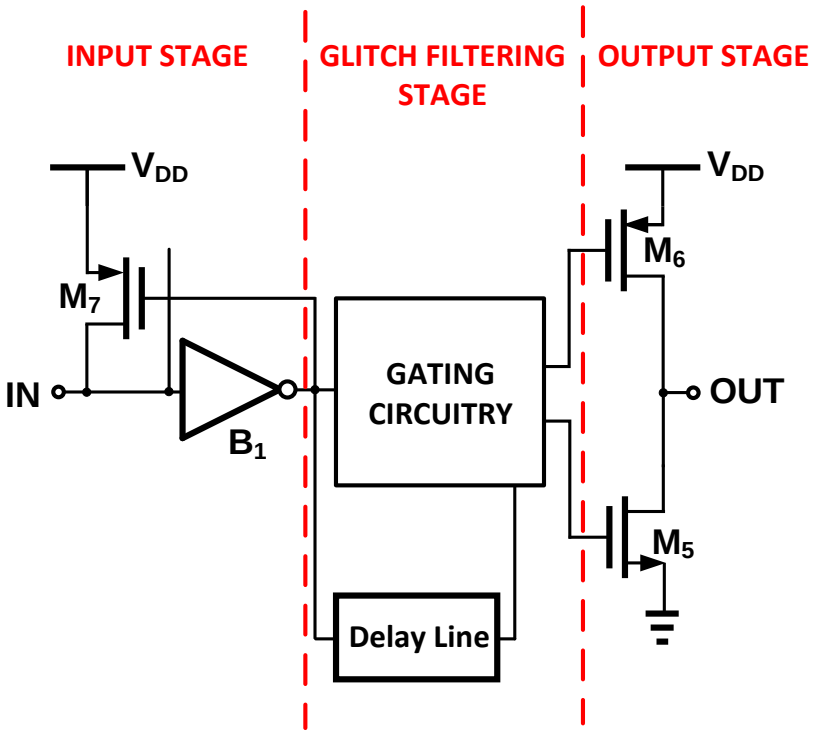


Asynchronous Glitch Filtering Circuitry



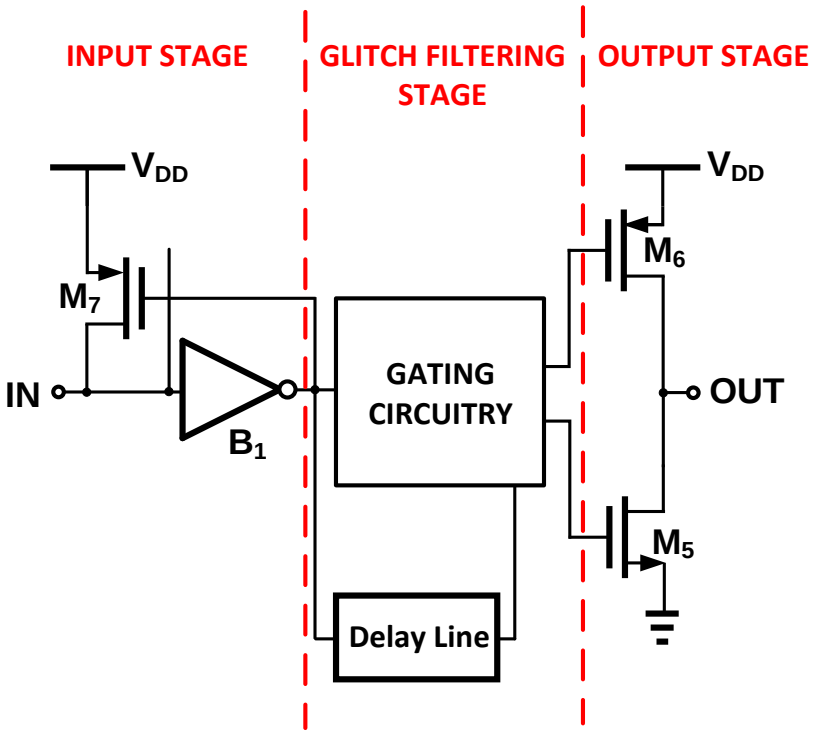
- Output stage “disconnected” from input immediately following transition at input stage
- Transition must propagate through delay-line before being “sampled” by output

Asynchronous Glitch Filtering Circuitry



- If input is a pulse with width $t_{pw} < t_d$ (delay line delay), will not appear at output
- *Programmable* delay-line allows glitches of varying widths to be filtered

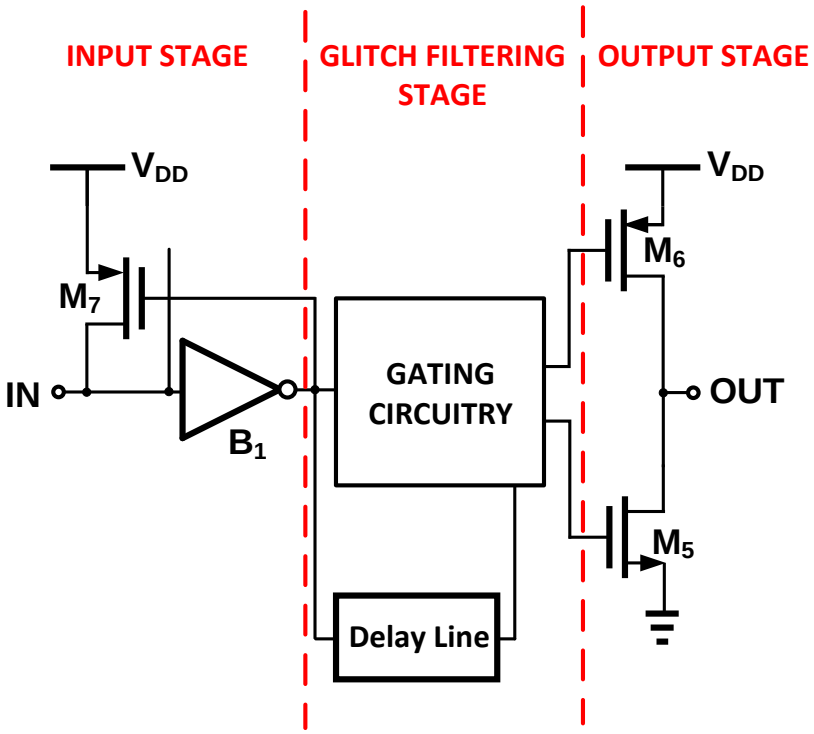
Asynchronous Glitch Filtering Circuitry



- If input is a pulse with width $t_{pw} < t_d$ (delay line delay), will not appear at output
- *Programmable* delay-line allows glitches of varying widths to be filtered

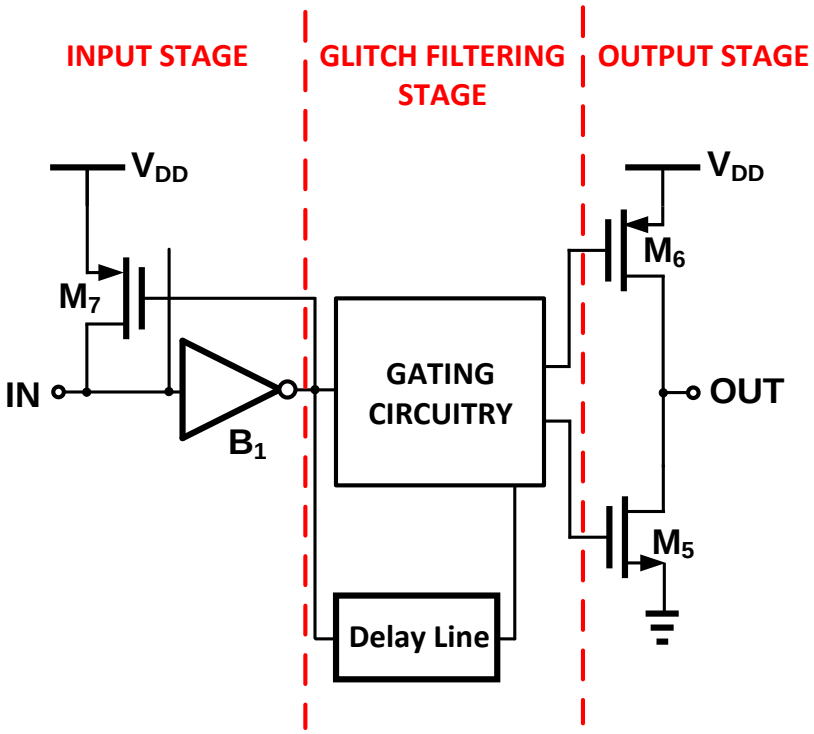
Implies better tolerance to PVT

Asynchronous Glitch Filtering Circuitry



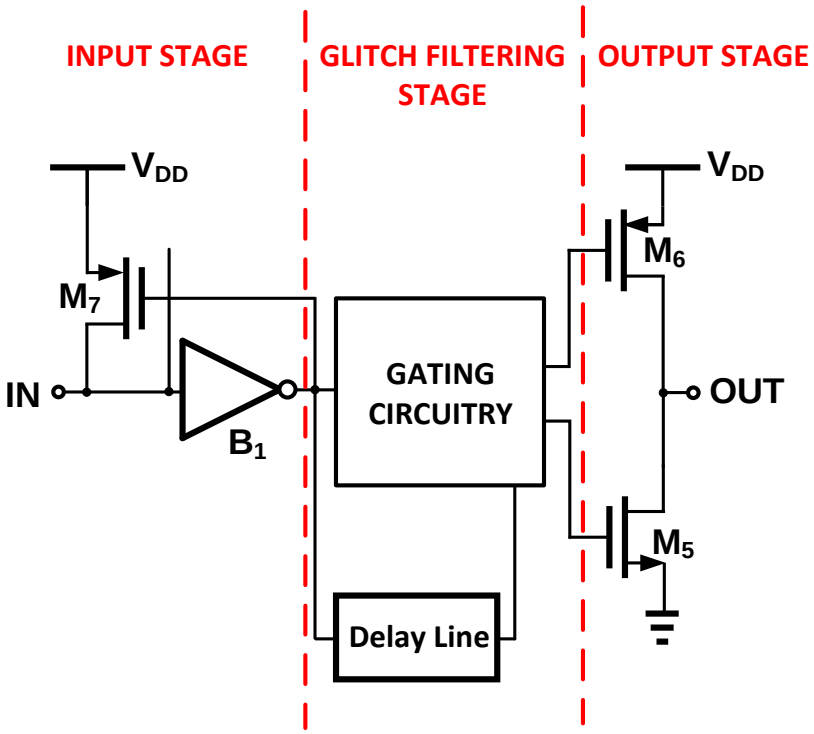
- Cost of glitch filtering:
  - When *used,* delay penalty t_D (hundreds of *p*s to *n*s range)
- Cannot apply to *all* paths
  - Only use to slack fill

Asynchronous Glitch Filtering Circuitry



- Cost of glitch filtering:
  - When *unused*, delay penalty ~30*p*s (can be much smaller with improved topology)

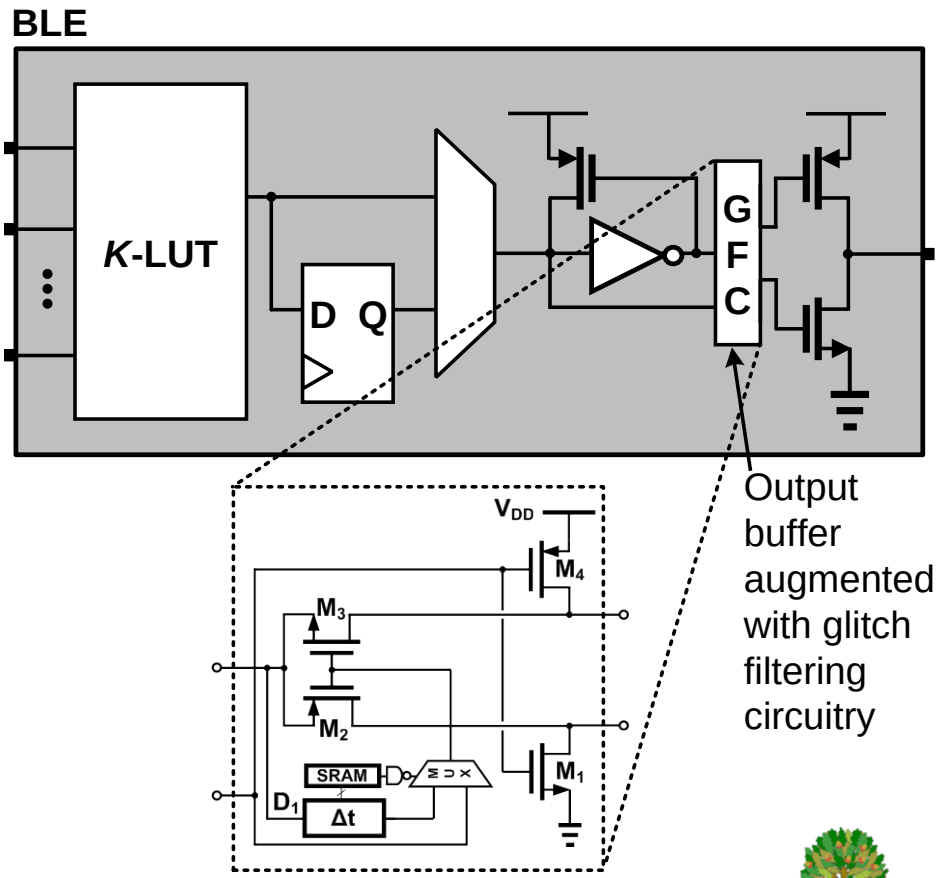
Asynchronous Glitch Filtering Circuitry



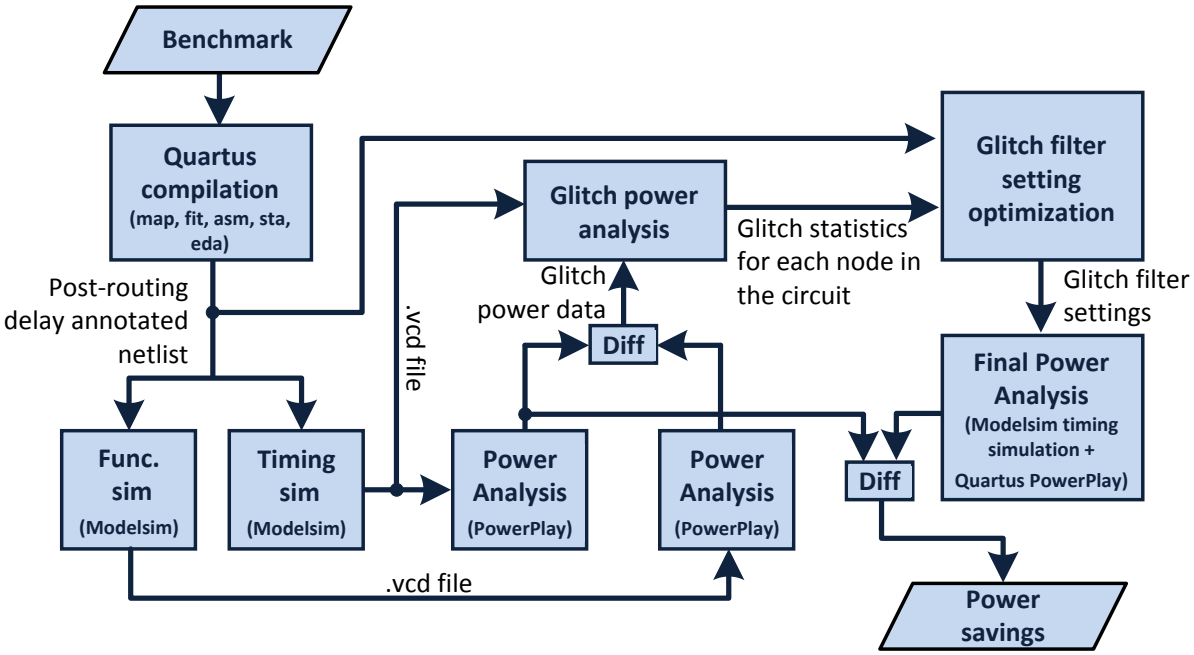
- Cost of glitch filtering:
  - Area: small (will be shown later), dominated by delay-line range and granularity

Proposed Architecture

- Augment BLE output buffer with asynchronous glitch filtering circuitry
- GFC's programmable delay line allows glitch suppression/slack trade-off to be made at each BLE output
- GFC's delay line setting determined by CAD flow

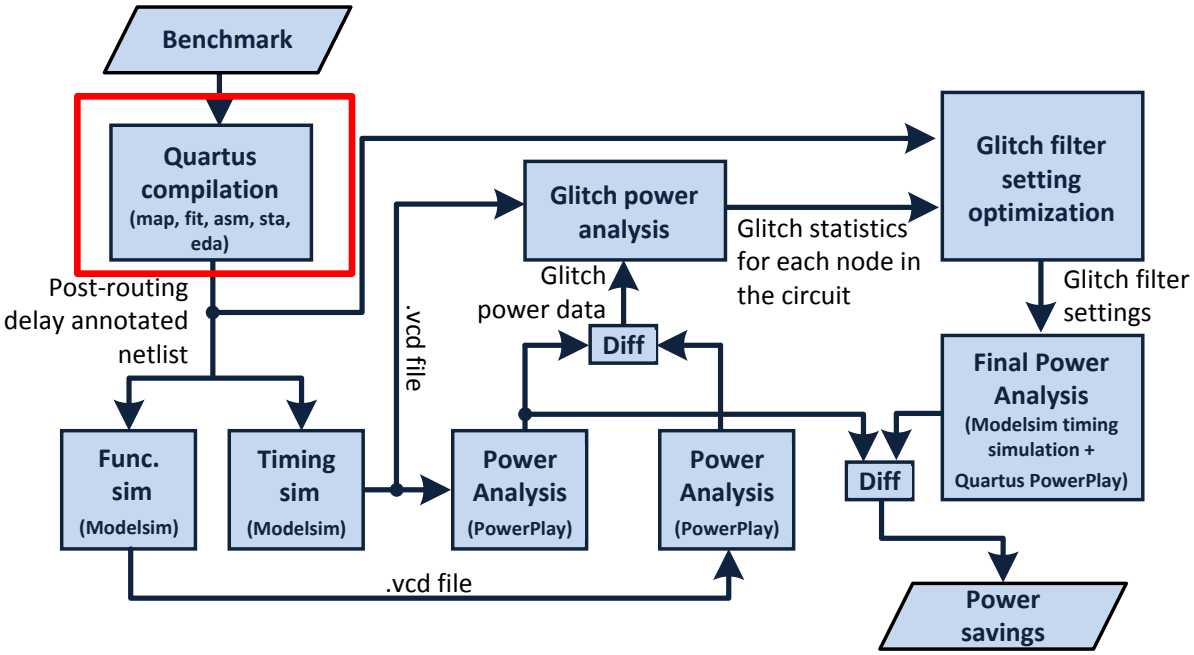


CAD Flow



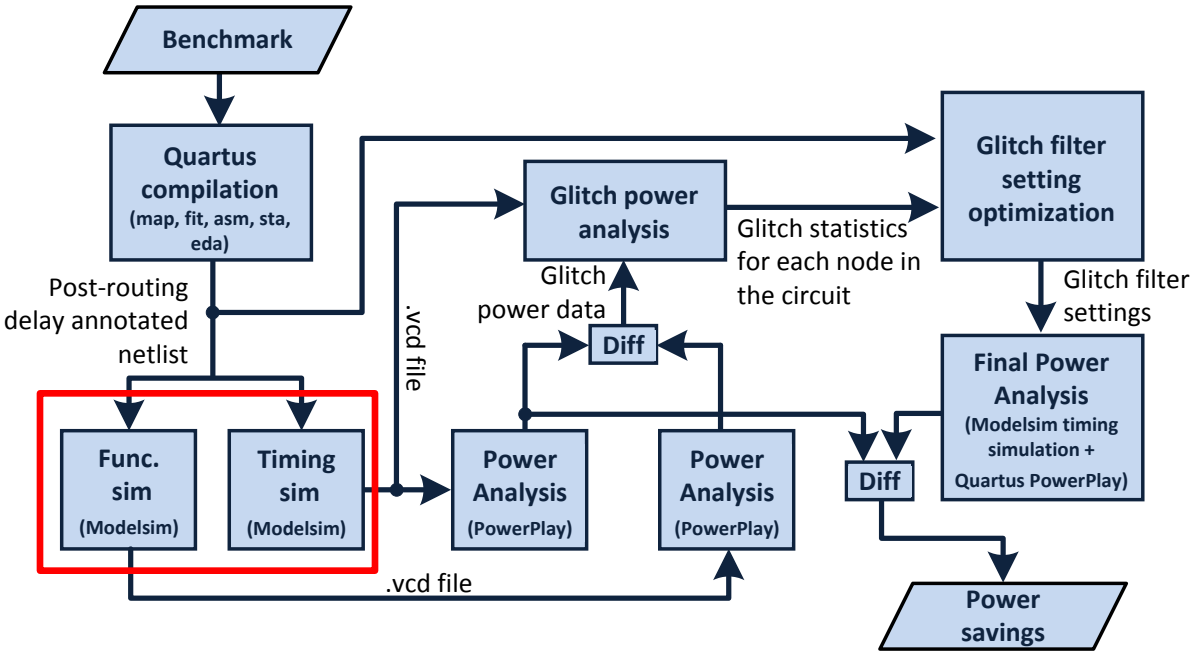
- Uses Quartus for compilation + power analysis
- Modelsim for timing and functional analysis
- Custom routines for extraction of glitch power statistics + optimization

CAD Flow



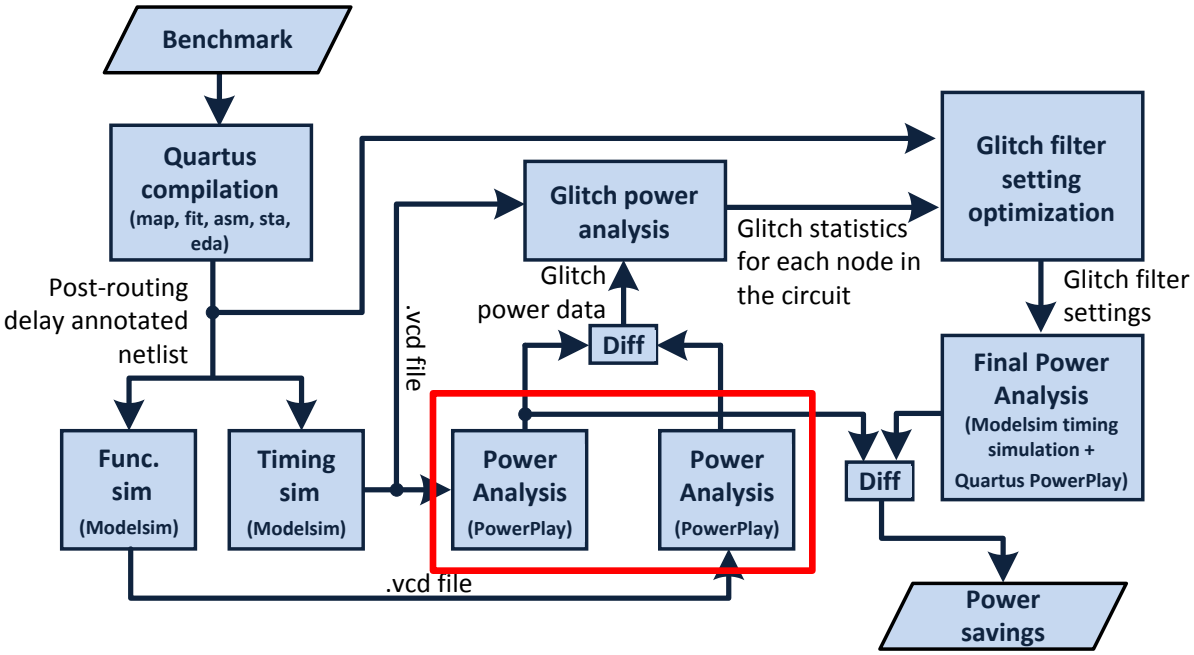
- Compile design with Quartus, output delay annotated netlist

CAD Flow



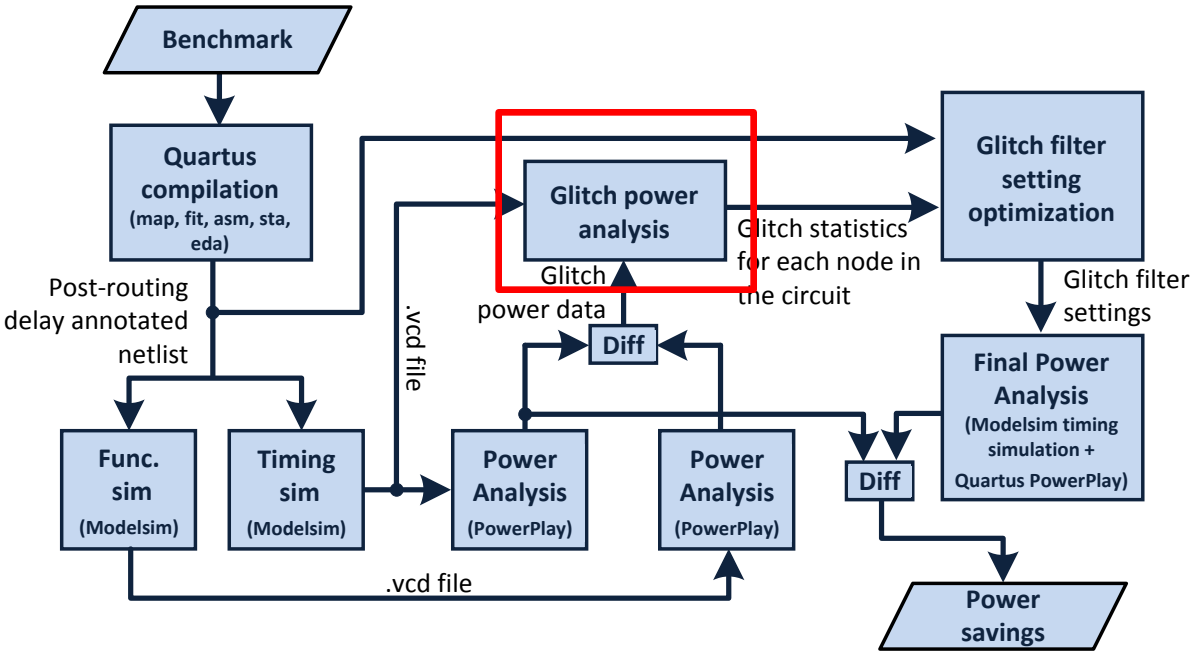
- Run Timing and Functional simulations with Modelsim

CAD Flow



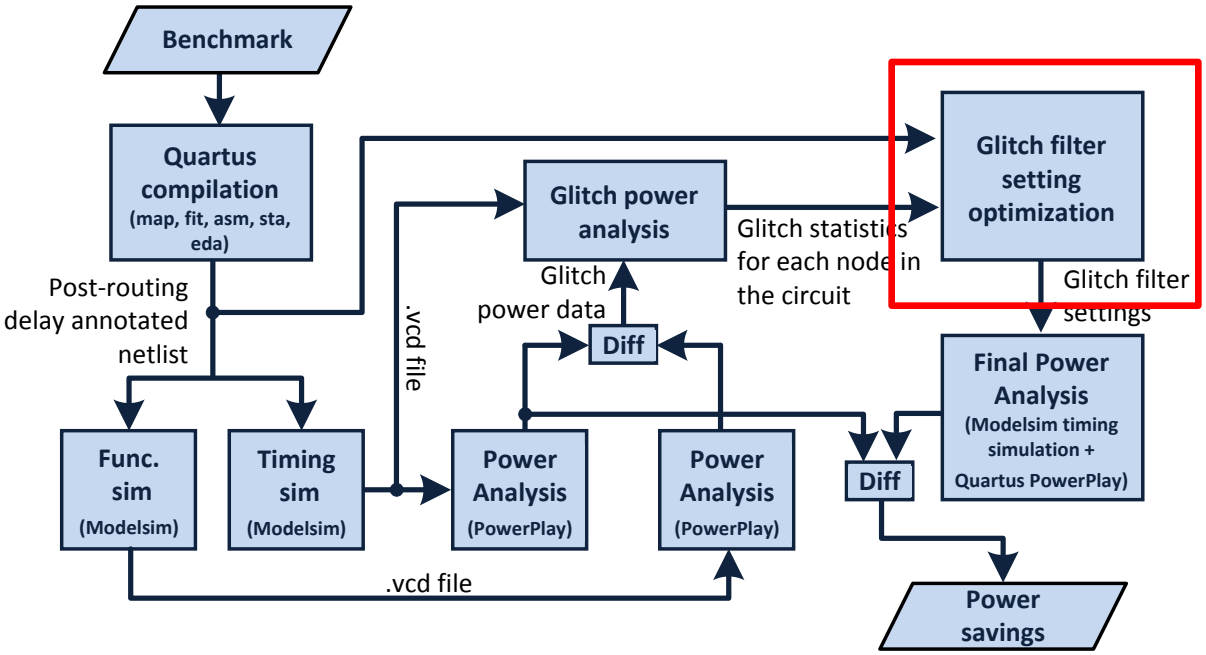
- Use Quartus to analyse power based on timing and func. Sims
- Difference between two sims corresponds to glitch power

CAD Flow



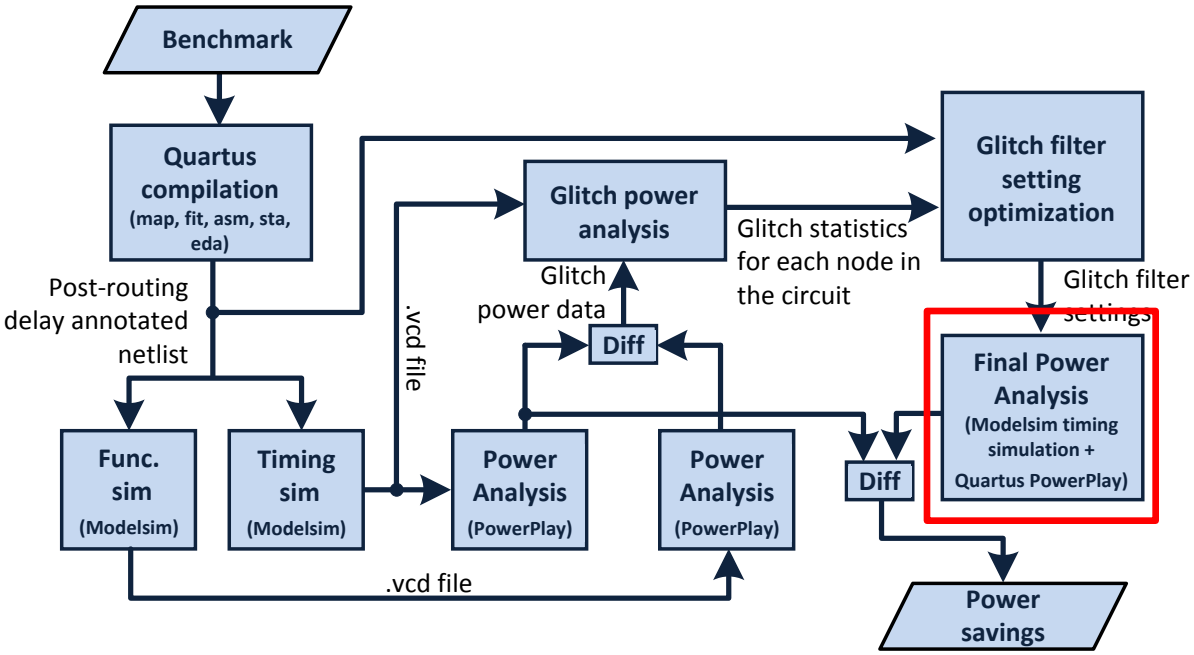
- Collect glitch statistics for each node in the circuit
  - # glitches for each pulse width
  - Glitch power dissipated at each node

CAD Flow



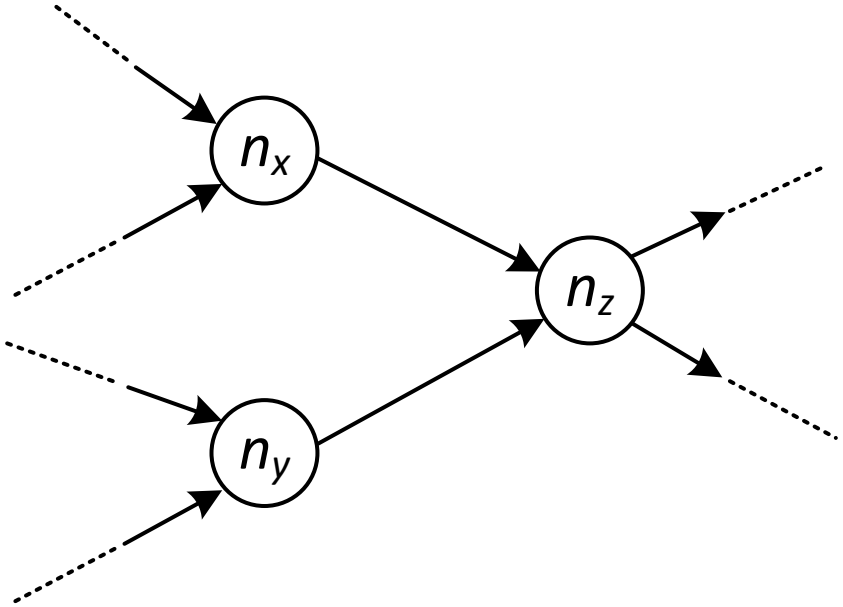
- Optimize power given glitch statistics and timing graph

CAD Flow



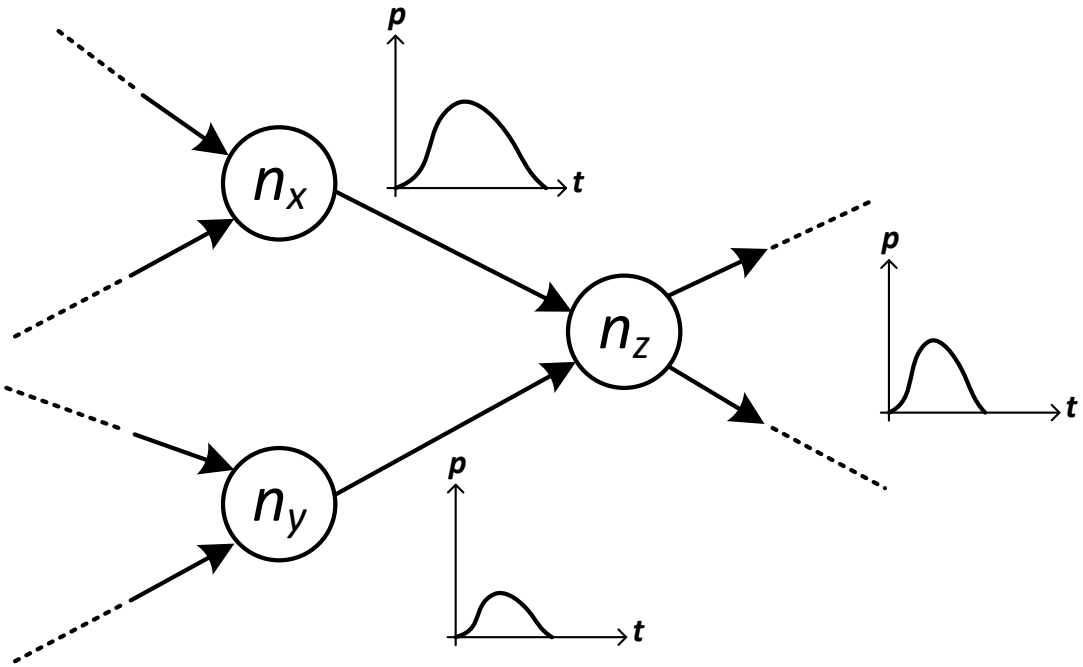
- Append *behavioural model* of GF to netlist
- Run timing sim. with glitch filter settings
- Run power analysis with new netlist and compare w/ prev.

Glitch Filter Setting Opt.



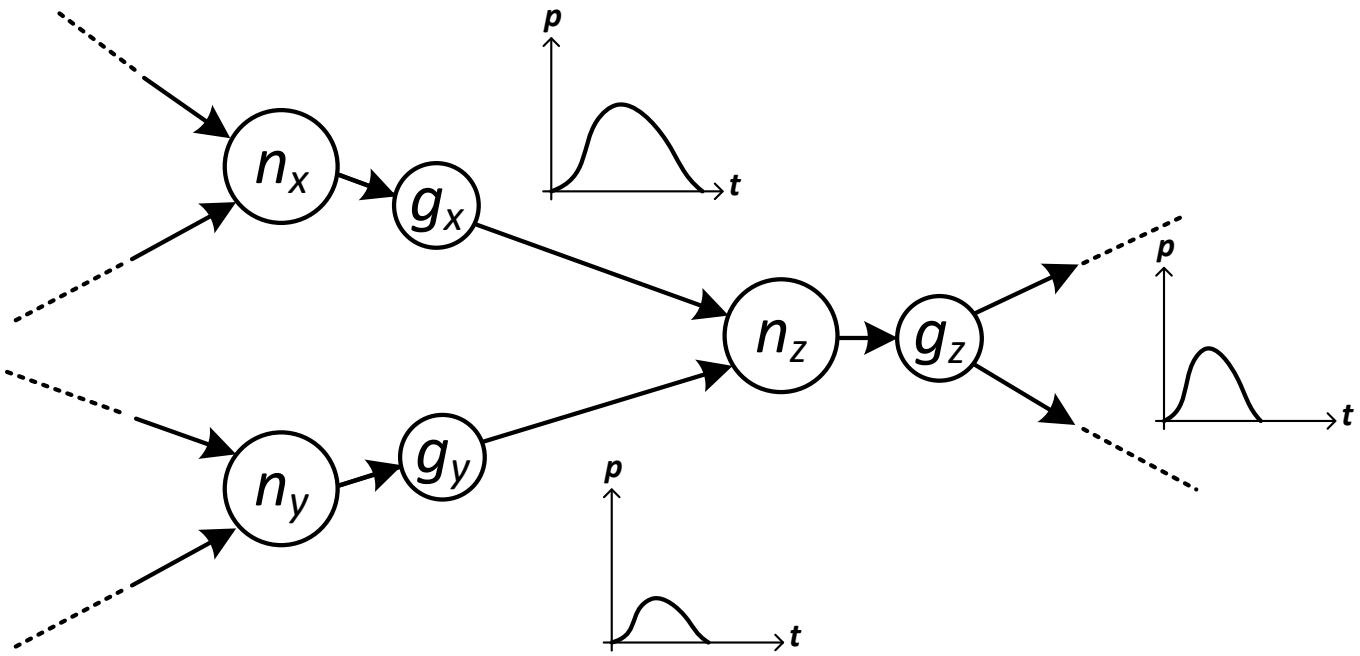
- Given a timing graph for a circuit with nodes n_i representing comb. gates, FFs, PIs, POs, etc.
- Edges annotated with delays

Glitch Filter Setting Opt.



- Annotate edges with *glitch power density functions* $GP_i(t)$
- Can be obtained through a combination of functional, timing, and power analyses

Glitch Filter Setting Opt.

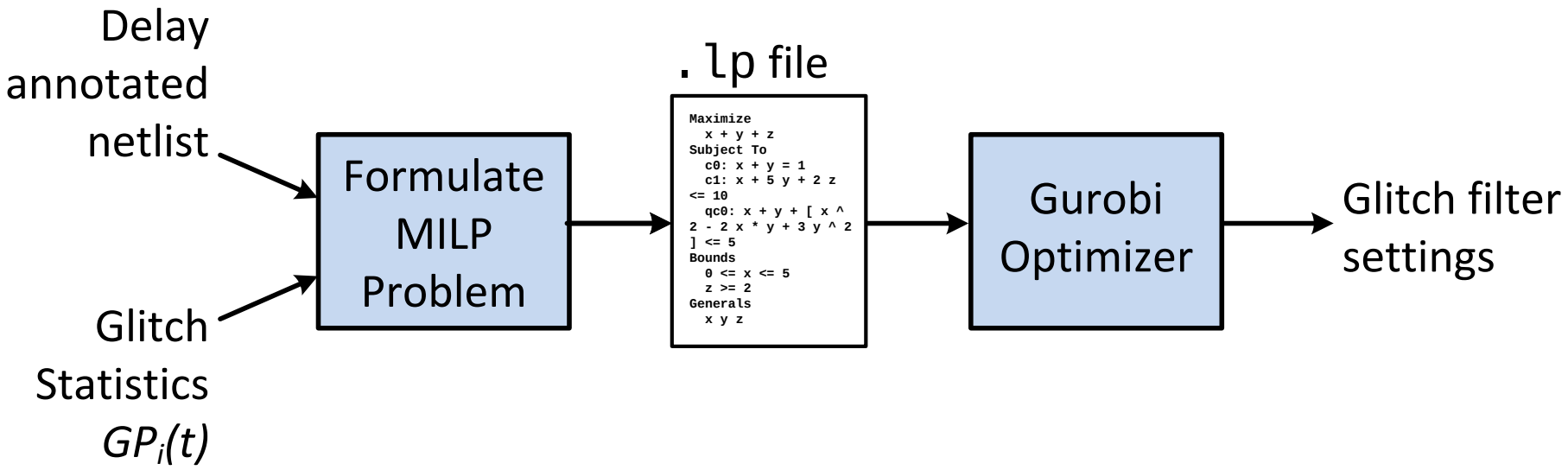


- Append nodes g_i to outputs of nodes n_i
  - These represent glitch filters
- Delay from n_i to g_i corresponds to glitch filter setting at i
  - We shall call this delay d_i

Glitch Filter Setting Opt.

- For every choice of d_i we get a corresponding glitch power **decrease** at node *i*
  - All glitches with pulse width less than d_i are suppressed
- Cannot arbitrarily set d_i
  - Must obey timing constraints
- Certain settings of d_i may have adverse effects
  - Can potentially create *new* glitches downstream
  - Optimization tool must be aware of this and avoid where needed

Glitch Filter Setting Opt.



- Optimization problem can be described as a set of linear equations with linear constraints
- Cast problem as *Mixed Integer Linear Program* (MILP)
  - Solve using Gurobi optimizer tool

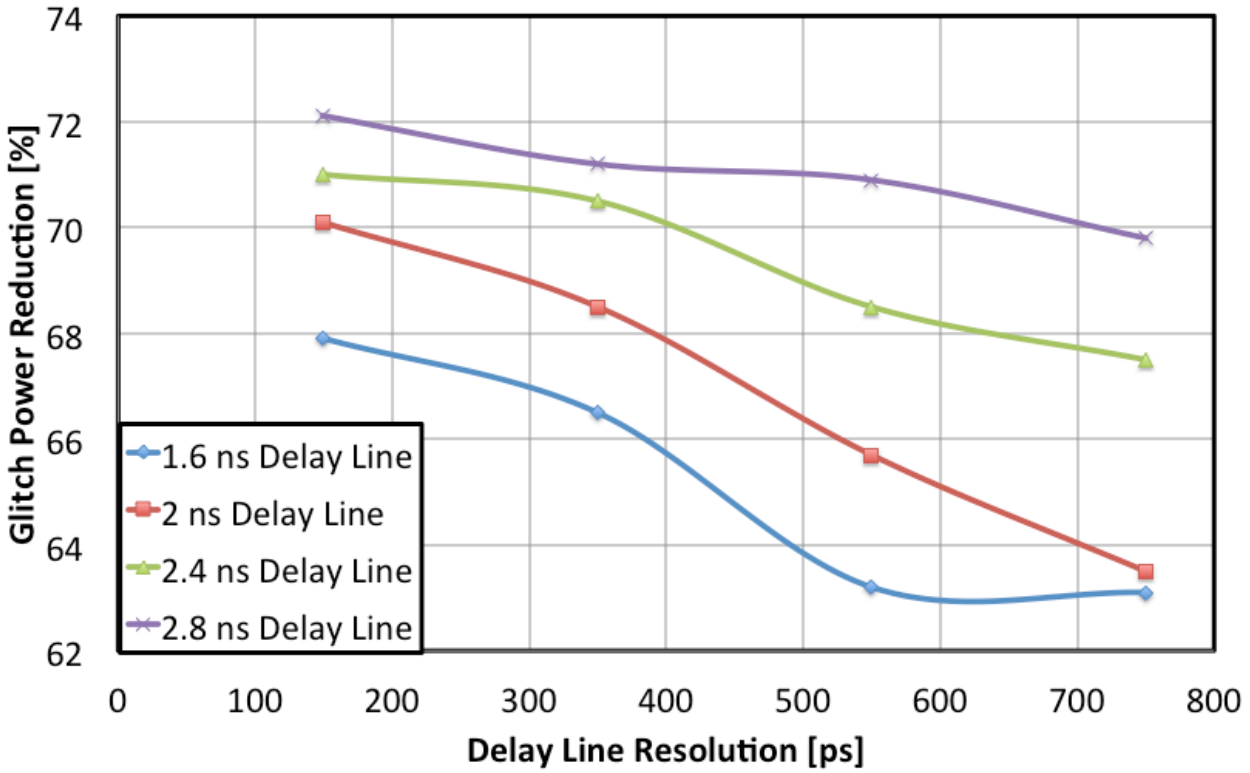
Experimental Study

- Used MCNC benchmarks to assess power reduction possibilities
  - Easy to use random vectors to perform power analysis
  - Supplied 10000 random vectors for timing/func. simulation
  - As previously reported, glitches account for ~26% of core dynamic power [1]
- Used benchmark set from UMass [4]
  - Communications/signal processing benchmark set of 6 designs
  - Glitch power was insignificant for one design (ava)
  - For other designs, ranged from 5-22% of core dynamic power
- All delay/power/functional characterization of circuits done using STMicroelectronics 65*n*m models

Experimental Study

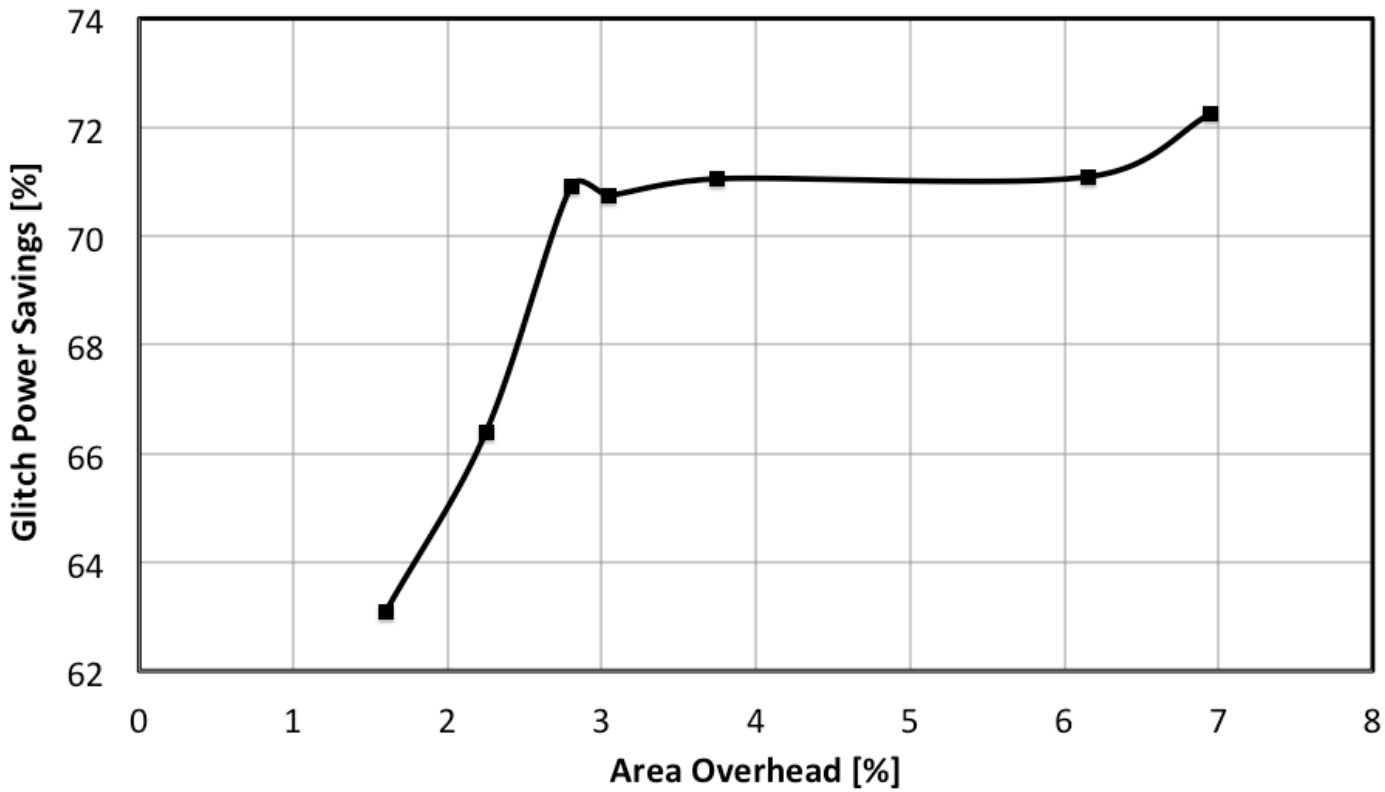
- Want to understand best design parameters for GF
- Delay setting of each GF constrained by physical limits
- *Ideal* GF delay line has:
  - Infinite range → delay can be as large as needed
  - Infinitesimal resolution → GF delay can be set to *any* value
- *Real* GF delay line has:
  - Finite range → maximum delay is *bounded*
  - Finite resolution → GF delay values *quantized* (100*p*s, 200*p*s,…)
- Finite res. and range limit ability to optimize glitch power
- Increased range and/or decreased res. → increased area
  - GF area dominated by delay line

Delay Line Parameters



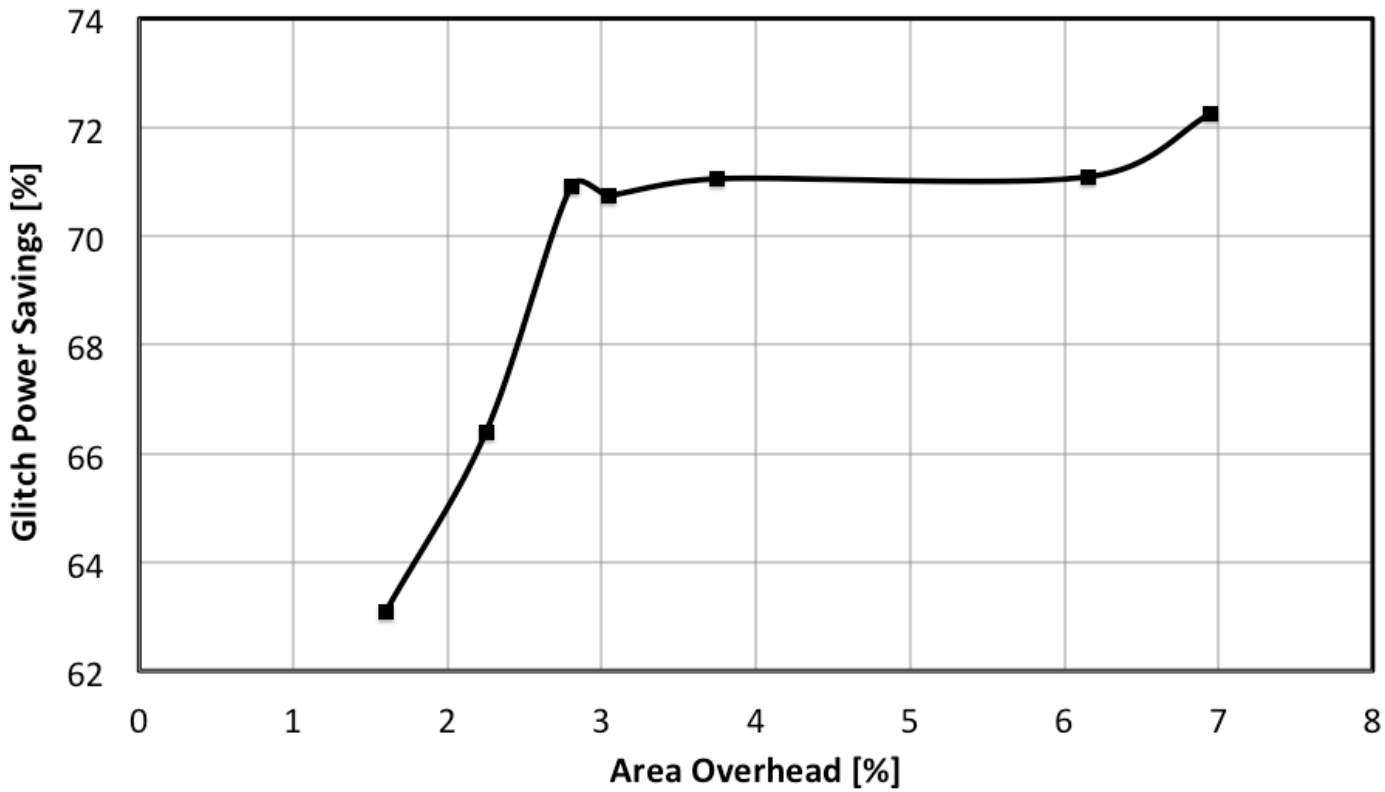
- Glitch power reduction ranges from ~62% to ~72% as delay line granularity and range are varied
- Ideal delay line allows ~75% glitch power to be reduced
  - Very close to ideal with *real* parameters

Power vs Area Overhead



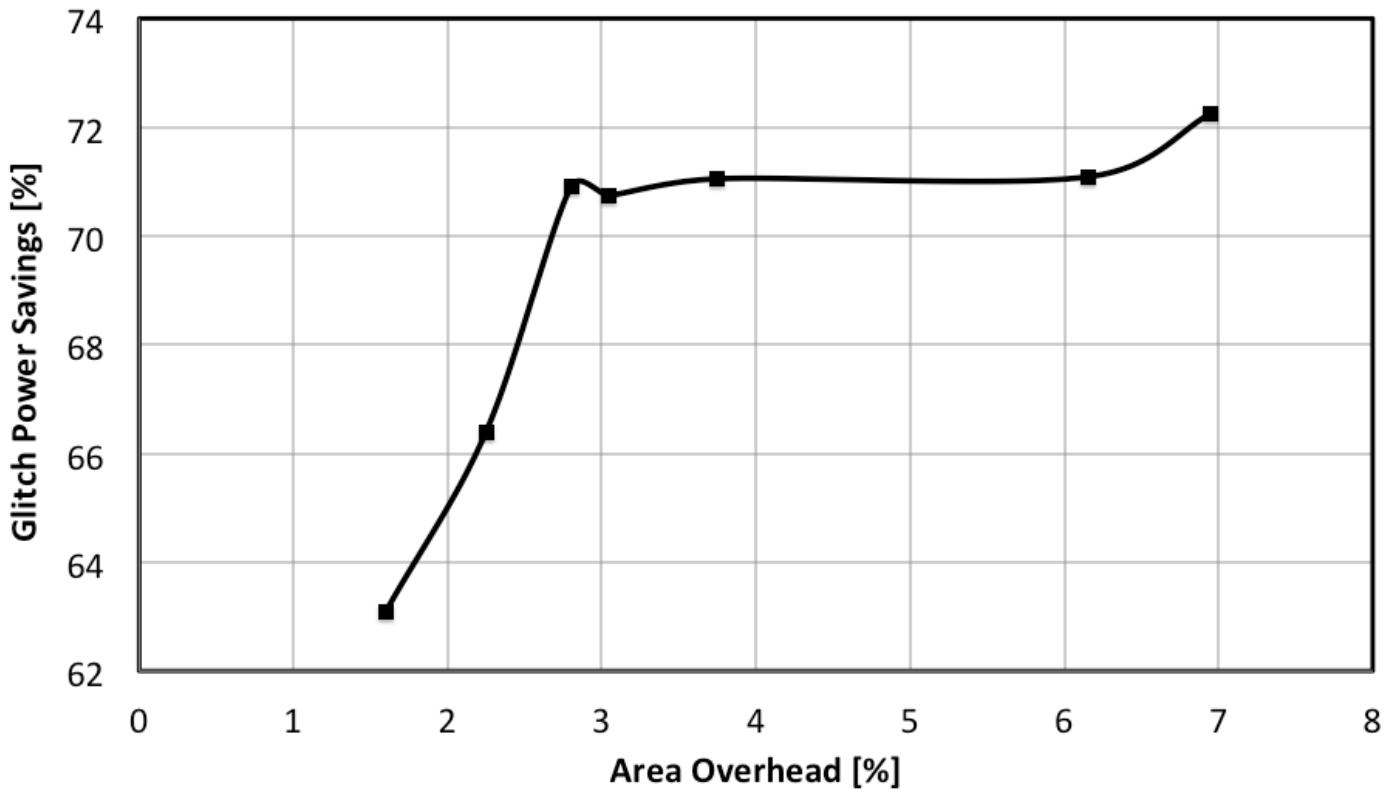
- Used [5] to estimate LUT area and estimate GF area
- Assume LUT:Routing = 25:75

Power vs Area Overhead



- ~2% area overhead allows ~62% glitch power reduction
  - Corresponds to ~12% decrease in core dynamic power
- Up to ~72% glitch power reduction at ~7% area cost
  - ~14% core dynamic power reduction

Power vs Area Overhead



- For designs studied in this work, delay penalty ~1%
  - 30*p*s LUT delay increase may not correspond to 1% critical path hit for other designs
  - Can be reduced with improved GF topology

Conclusion

- Proposed a glitch reduction technique which overcomes some of the shortcomings of previous techniques
- Able to reduce glitch power by ~60-70% at an area overhead of ~2-3%, c.p. degradation ~1%
  - Corresponds to total core dynamic power reduction of ~12-14%
- **Future Work:**
- Improve CAD flow
  - Inherently pessimistic assumptions/approach
  - A model to relate arrival time to glitch power could improve decision making significantly
  - Investigate heuristics to replace MILP formulation
- Assess multi PVT corner optimization of glitch power

References

[1] Shum, W. and Anderson, J.H. "FPGA glitch power analysis and reduction." IEEE ISLPED 2011

[2] Lamoureux, J. et al. "GlitchLess: Dynamic power minimization in FPGAs through edge alignment and glitch filtering." IEEE TVLSI no. 16.11 (2008): 1521-1534.

[3] Dinh, Q. et al. "A routing approach to reduce glitches in low power FPGAs." IEEE TCAD no 29.2 (2010): 235-240.

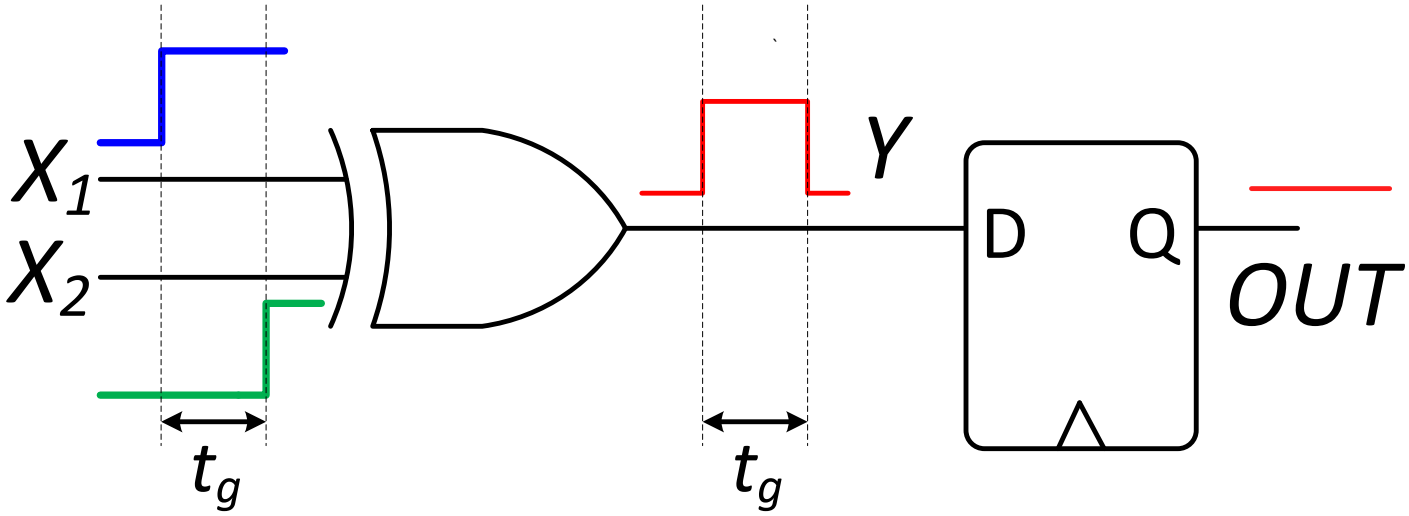
[4] www.ecs.umass.edu/ece/tessier/rcg/benchmarks/

[5] Chiasson, C. and Betz, V. "COFFE: Fully-Automated Transistor Sizing for FPGAs," IEEE FPT 2013

Backup Slides

Previous Approaches

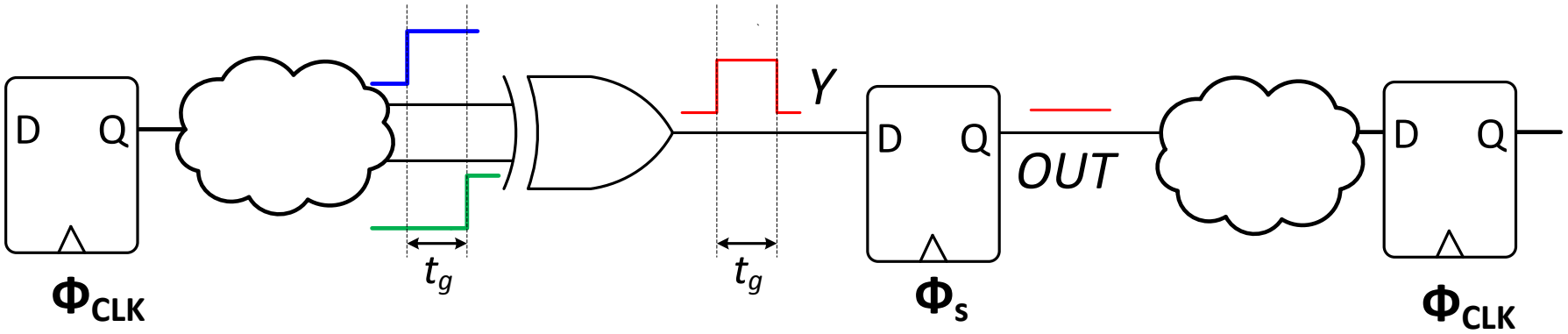
Pipelining



- Synthesis techniques to heavily pipeline [3]
  - Effective in removing glitches – blocked from propagating to interconnect
  - Potentially not suitable/applicable to all designs

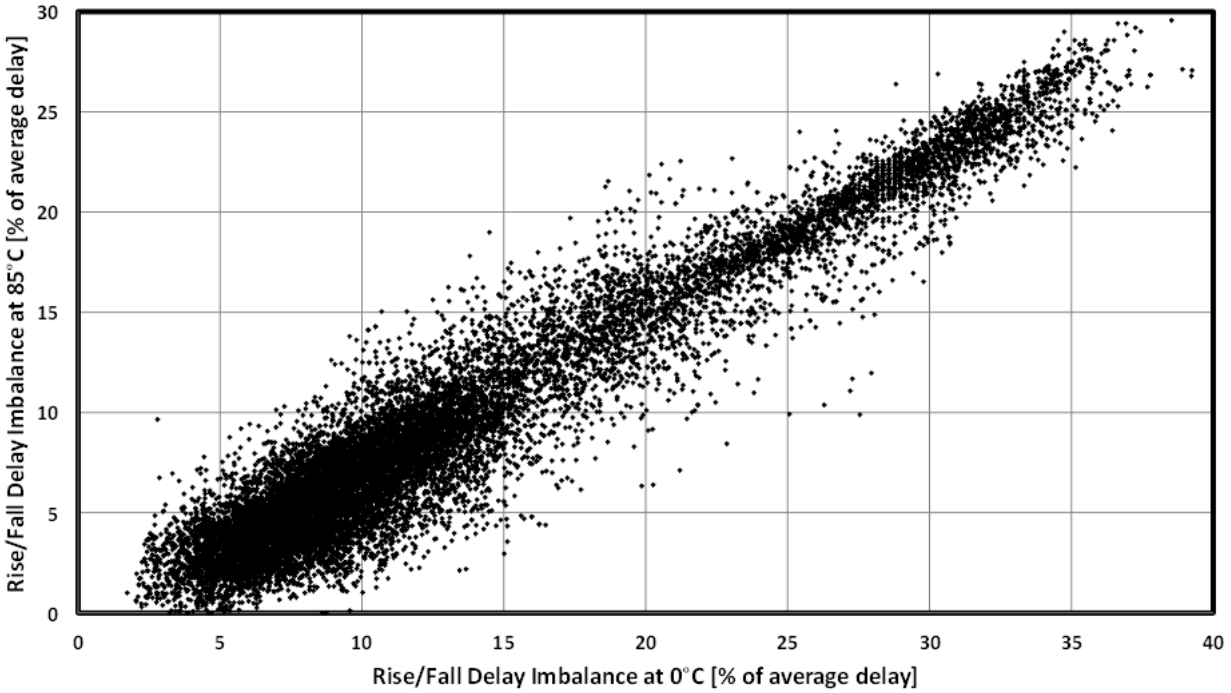
Previous Approaches

Synchronous Glitch Filtering



- Add additional FF or latch to sample intermediate points in datapath using a clock with different phase ϕ_s [4]
  - Does not affect behaviour of original design
  - Has limited scope to filter glitches, may not be applicable to many parts of complex design
  - Supplying ϕ_s may be costly/not feasible

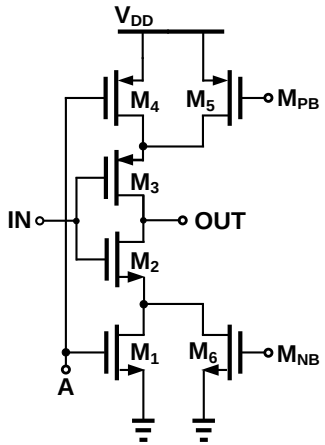
Routing Delay Characteristics



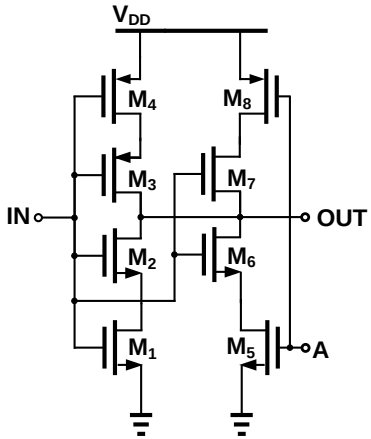
- Stratix-III routing delay characteristics extracted from a benchmark
- Shows routing state & temperature dependent delay highly unpredictable

Some Implementation Details

CS Delay Cell

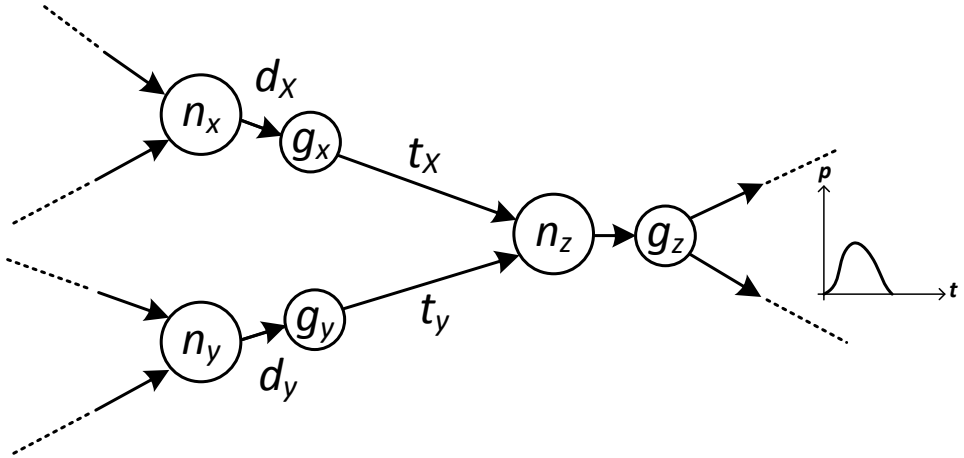


Conv. Delay Cell



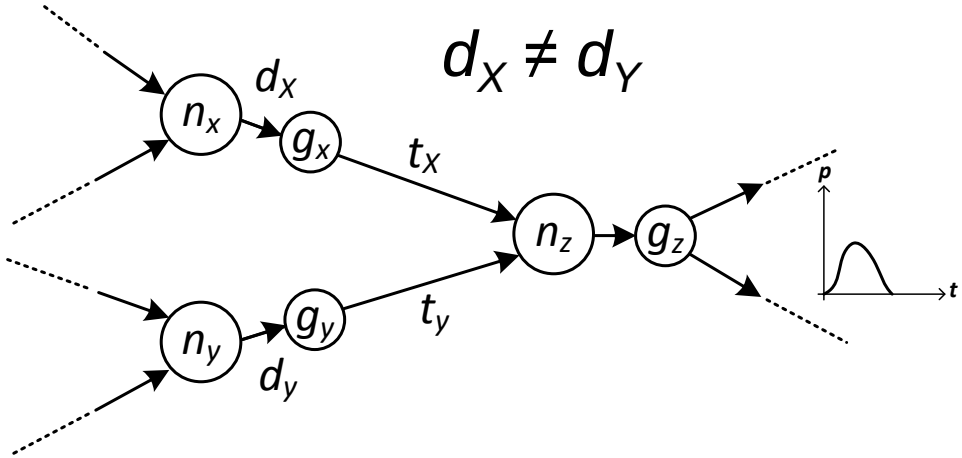
- Two possible delay cell implementations considered in this paper
- Current Starved (CS) potentially has lower area overhead, at a cost of increased PVT sensitivity
- Conventional less sensitive to PVT, greater area

Additional Constraint



- Obtaining $GP_v(t)$ is time consuming
- Would like to minimize number of times we compute these
  - Ideally compute only once!
- Glitch profile function of relative arrival times of signals at inputs to combinational nodes

Additional Constraint

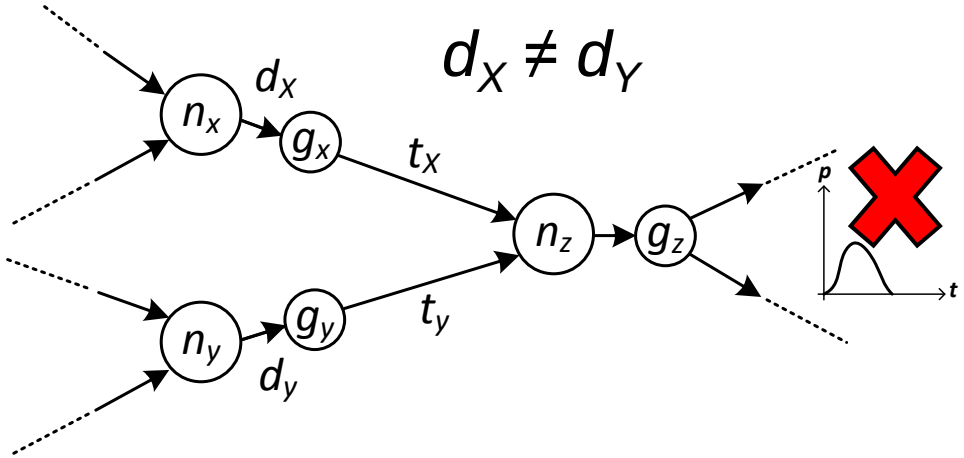


- What happens if $d_X \neq d_Y$?
- Relative arrival time at Z = $(d_x + t_x) - (d_y + t_y)$

  $= (t_x - t_y) - (d_x - d_y)$

  $\neq (t_x - t_y)$
- $GP_Z(t)$: function of relative arrival times at z

Additional Constraint



- What happens if $d_X \neq d_Y$?
- Relative arrival time at Z = $(d_x + t_x) - (d_y + t_y)$

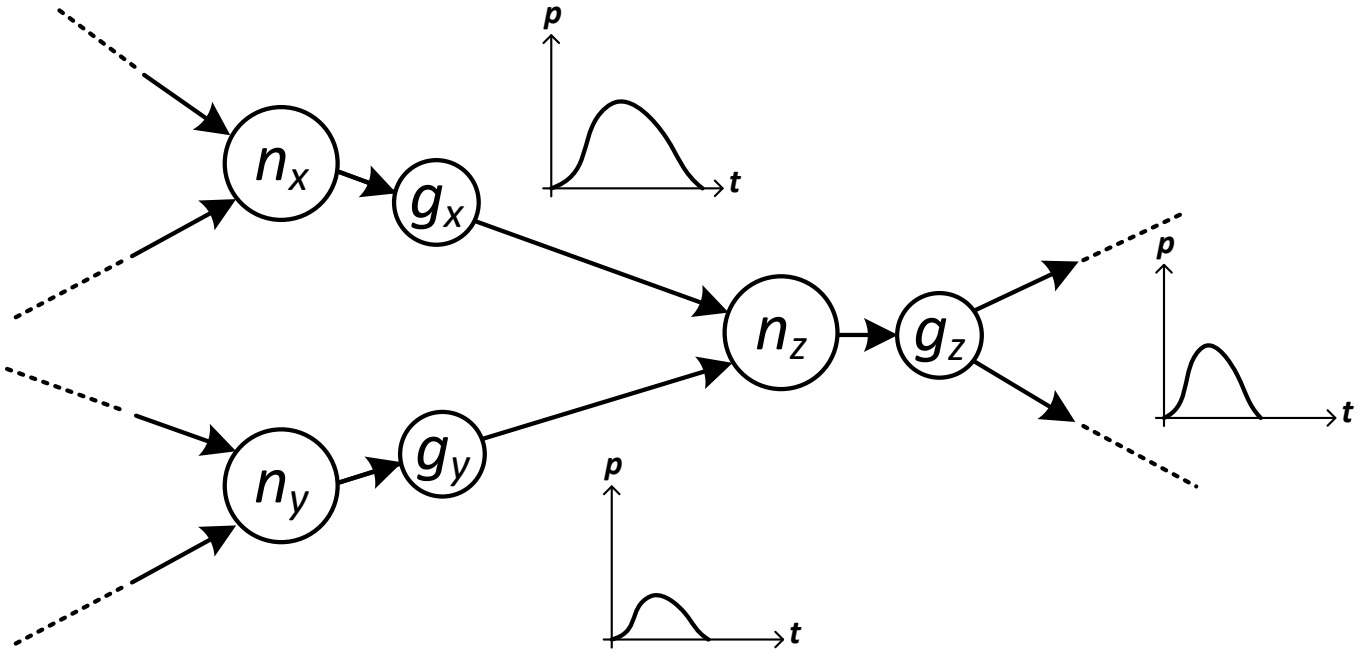
  $= (t_x - t_y) - (d_x - d_y)$

  $\neq (t_x - t_y)$
- $GP_Z(t)$: function of relative arrival times at z
- Setting $d_X \neq d_Y$ makes $GP_z(t)$ stale, need to recompute!

Our Approach

- Only compute $GP_v(t)$ once
- If relative arrival times at *v* are altered, $GP_v(t)$ discarded
  - From optimizer's point of view, *v*'s power cannot be reduced
- Ensure relative arrival times of *consequential* nodes is preserved
  - Nodes whose power we want to reduce!
  - Nodes whose worst case glitch power is deemed to be insignificant are free to have input relative arrival times altered

Optimization Problem (revisited)



- Objective:
  - *Minimize P_{total} subject to timing constraints and soft arrival time equalization constraints*