

Long wires and asynchronous control

R. Ho, J. Gainsley, R. Drost
Sun Microsystems Laboratories

SML2004-0323

Funded by
DARPA contract
NBCH30390002

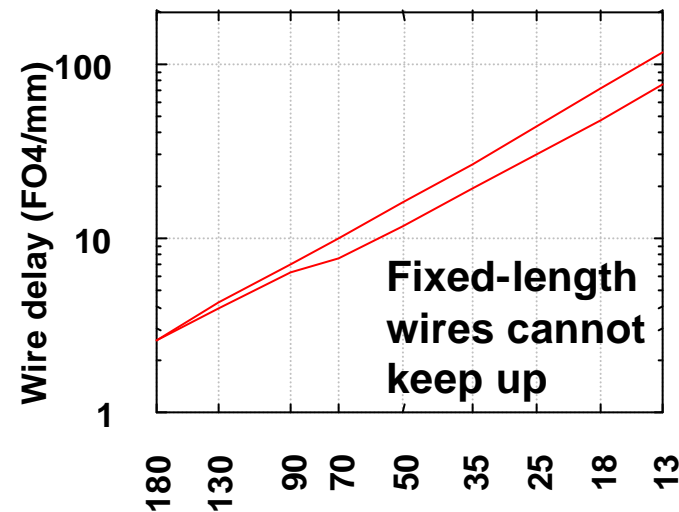
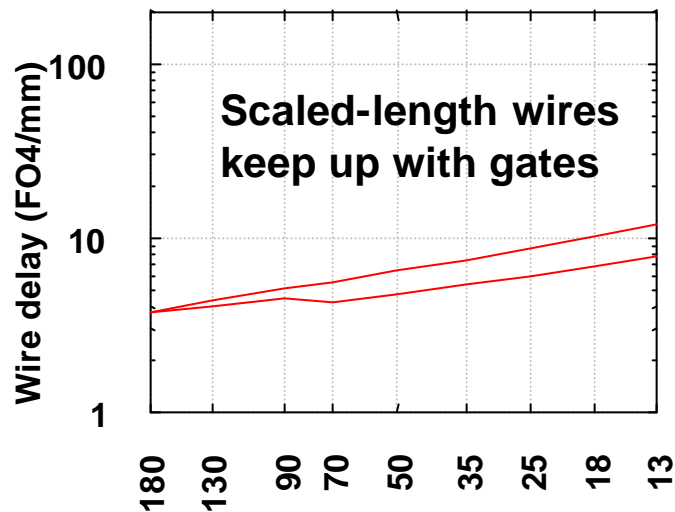
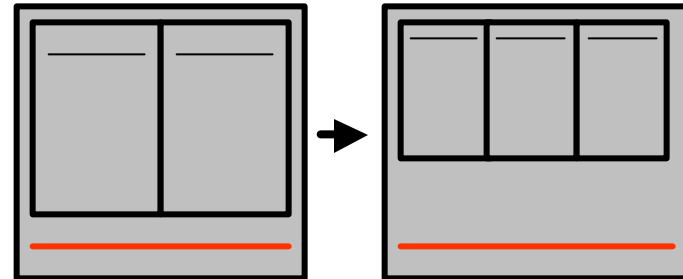
Public Information

 *Sun*
microsystems
We make the net work.

How do on-chip wires scale?

Are they really as bad as “they” say?

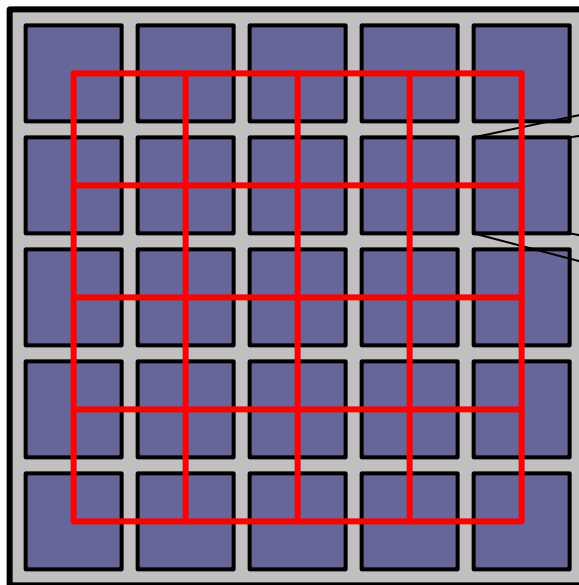
- There are really two kinds of on-chip wires
 - Span a block of constant complexity
 - Scaled-length wires
 - Span a fixed distance
 - Constant-length wires



What this means for designers

Build modular machines

- **Build what the VLSI constraints (wires) demand**



Computation block

- Lots of xstrs
- Local memory
- Local communication
- Locally synchronous?

Global network

- Explicit (expensive) communication
- Lots of long wires
- Globally asynchronous?

- **Global network ties all the blocks together**
 - **How can we get high bandwidth and low latency?**

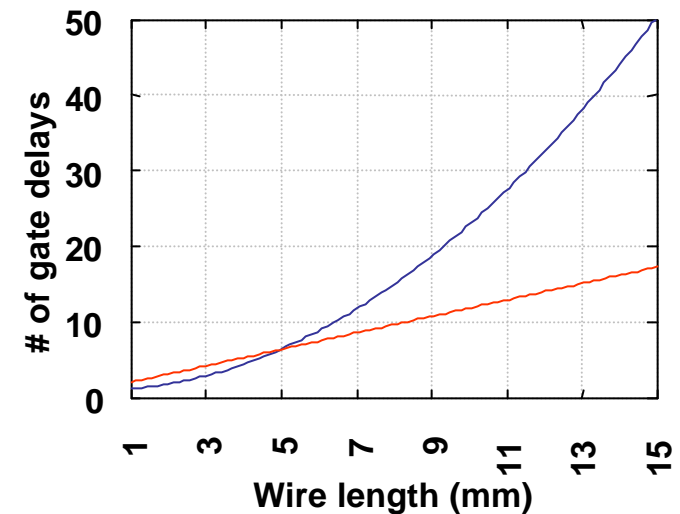
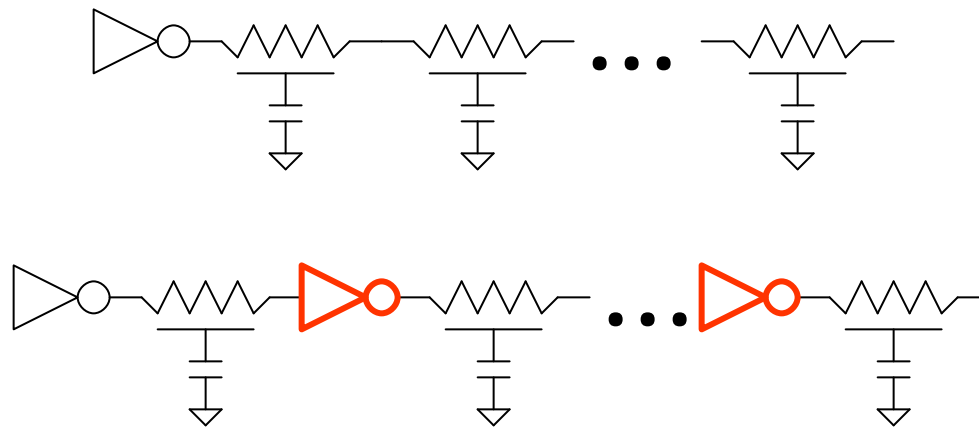
Outline

- **Speeding up global wires**
 - **Asynchronous control improves performance**
- **Optimizing wire latency**
 - **Well-known circuit models lead to analysis**
- **Optimizing wire bandwidth**
 - **Dual-path control reduces transactional penalty**
- **What about power?**
- **Conclusion**

Speeding up global wires

Flow-through repeaters

- **Flow-through repeaters help latency (for power)**

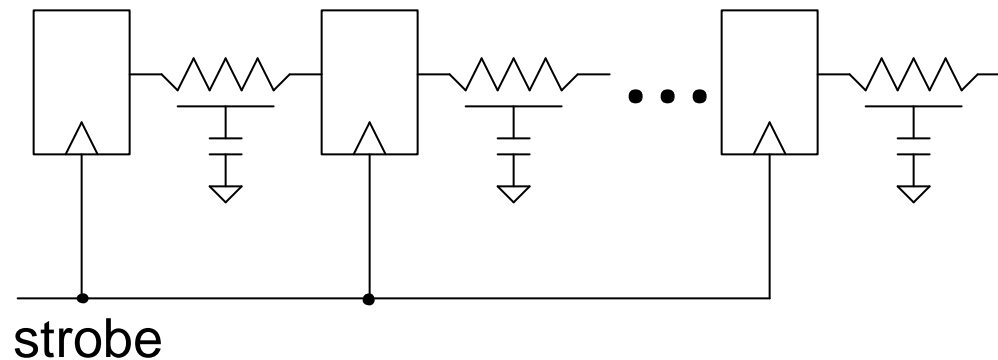


- **But they do not improve bandwidth**
 - **Unless we wave-pipeline them**
 - **Scary with {device,wire} {static,dynamic} variations**

Speeding up global wires

Latched repeaters

- Latched repeaters improve latency and bandwidth
 - Latency a little worse due to internal delays

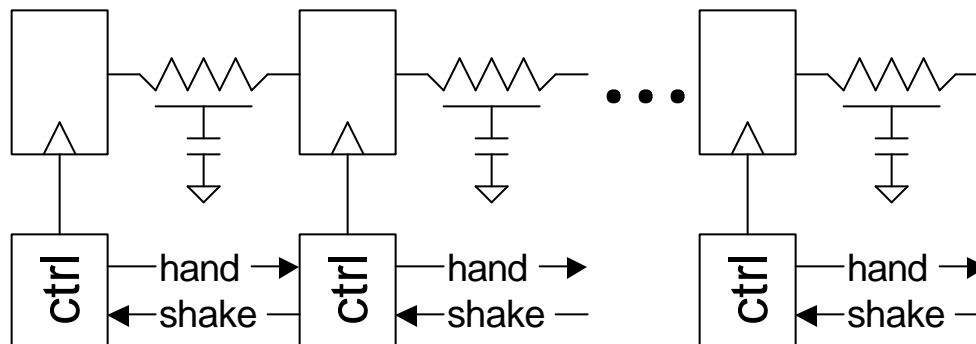


- The problem: they need a fast strobe (~5 FO4s)
 - Can't use CPU clock (no faster than ~15 FO4/cycle)
 - Local fast clock generation adds complexity

Speeding up global wires

Asynchronous latched repeaters

- So control the latched repeaters asynchronously
 - Better latency, better bandwidth, don't need clock
 - Allows for GALS: asynchronous compute modules

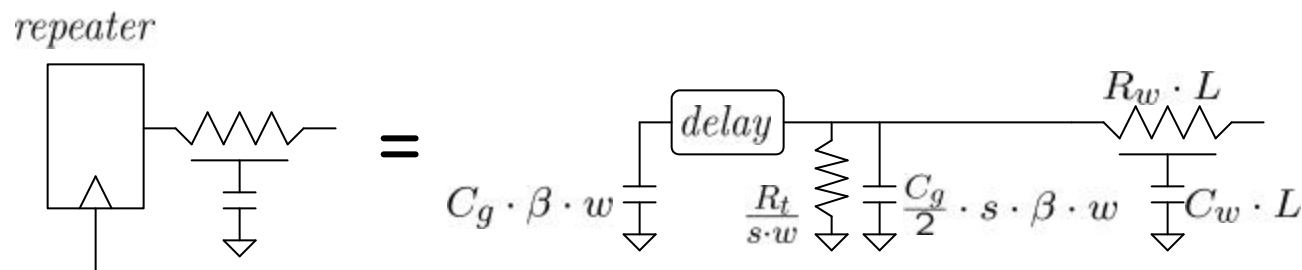


- Treat global wires as flow-through FIFOs
 - So: how do we optimize latency and bandwidth?

Optimizing wire latency

Analytic models

- **Leverage well-known circuit analysis techniques**
 - **Use dominant time constant (Elmore) models**
 - **Not specific to asynchronous circuits**
 - **But assume source-limited data patterns**
- **Turn repeater and wire into component Rs and Cs**
 - **Parameterize by driver width (w), wire length (L)**
 - **Latch design sets delay, p/n ratios (b), stepup (s)**



Optimizing wire latency

Analytical formulation leads to optimization

- Formulate RC delay and optimize
 - Partial derivative w.r.t. driver width (w) = 0
 - Partial derivative w.r.t. segment length (L) = 0
- Example: latch with tristate-able output
 - For minimal delay:

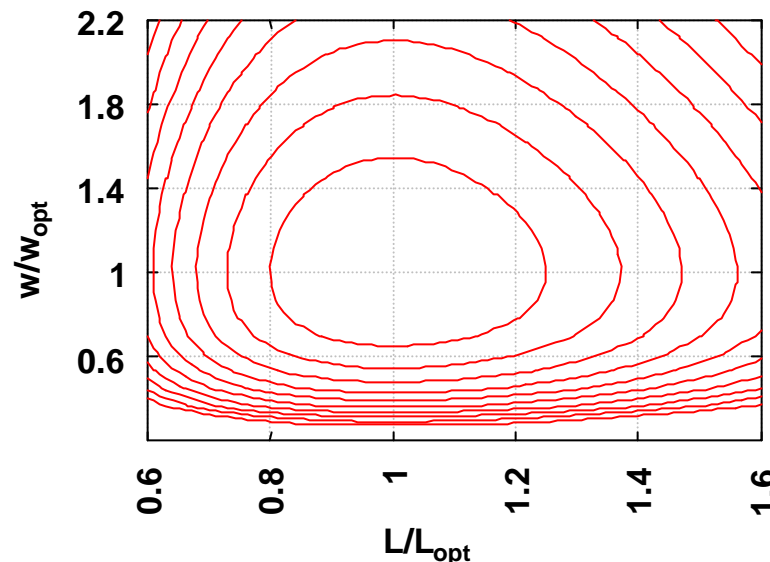
$$L_{opt} = 9.33 \sqrt{\frac{R_t C_g}{R_w C_w}} \quad w_{opt} = \frac{1}{\sqrt{6}} \sqrt{\frac{R_t C_w}{R_w C_g}}$$

- In a TSMC 180nm logic process, using M5 wires
 - Delay-minimal $L = 3.8\text{mm}$, $w = 20\text{mm}$

Optimizing wire latency

Sensitivities

- What about sensitivities to L and w?
 - Normalize to their delay-optimal values



2% delay contours
Very flat contours!

- So for datapaths, best latency is $\sim 3\text{mm}$ to 4.6mm
- What about bandwidth?

Optimizing wire bandwidth

Transactional nature of controls

- **Asynchronous circuits are *transactional***
 - **Each cycle requires a request and a response**
 - **During the request, data flows**
 - **During the response, no data flows**
 - **Control circuit families reflect this imbalance**
 - **In GasP ACKs (2 gates) are faster than REQs (4)**
 - **ACKs would be zero, except for hold times**

Optimizing wire bandwidth

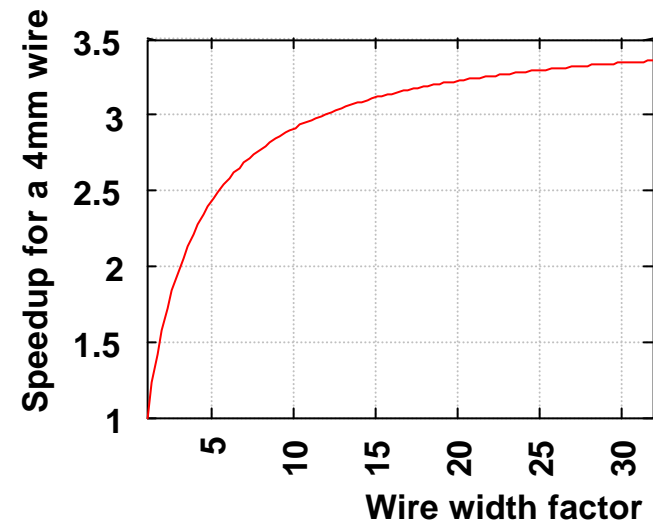
Implications for wires

- **Long wires exacerbate transaction delays**
 - **Both REQ and ACK require wire RC delay**
 - **REQ delay matches data delay: useful**
 - **ACK delay is dead time for datapath: useless**
- **Can wire engineering help?**
 - **Fatten ACK wire**
 - **Lower its RC delay**
 - **Get 2.5x speedup easily**
 - **Much more is too costly**

Optimizing wire bandwidth

Implications for wires

- Long wires exacerbate transaction delays
 - Both REQ and ACK require wire RC delay
 - REQ delay matches data delay: useful
 - ACK delay is dead time for datapath: useless
- Can wire engineering help?
 - Fatten ACK wire
 - Lower its RC delay
 - Get 2.5x speedup easily
 - Much more is too costly



Optimizing wire bandwidth

Control protocol implications for long wires

- **Level-sensitive control (RZ) is a poor choice**
 - Uses four phases: two wire transitions per token
 - Has *twice* the transactional penalty
- **Transition-encoded control (NRZ) is better**
 - Uses two phases: average one transition per token
 - Still has transactional bandwidth limitation
- **Pulse-encoded control (GasP) also okay**
 - Has same energy as NRZ, same bandwidth penalty
 - Has the advantage that we're familiar with GasP

Optimizing wire bandwidth

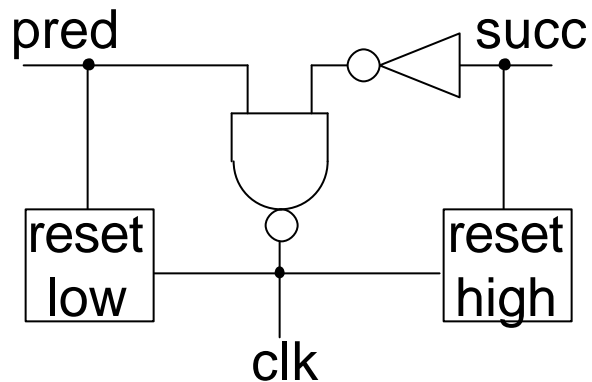
Pulse-encoded control challenges

- **By the way, GasP control of long wires isn't trivial**
 - **Control wires are bidirectional, data wires are not**
 - **Capacitance asymmetry between control, data**
 - **Requires a bit more timing margin**
 - **Pushing pulses on a moderately long wire is hard**
 - **Must overcome the “wet noodle” effect**
 - **Logical effort theory can help CAD sizing**
 - **But for now, size things manually via spice**

Optimizing wire bandwidth

Modified GasP for long wires

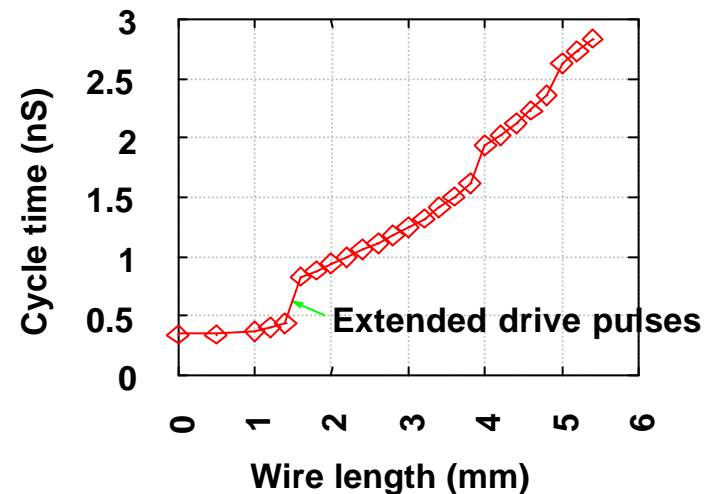
- A simplification of GasP
 - High = full, or “token present”
 - Low = empty, or “no token present”
 - If (pred==high && succ==low) then
 - Flip the clk, and reset both pred and succ



Optimizing wire bandwidth

Simulations of GasP

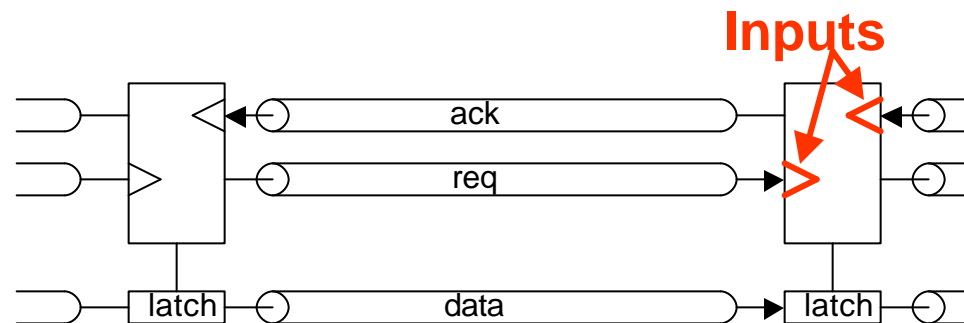
- **Simulate long wires under GasP control**
 - Use M5 wires on a TSMC 180nm logic process
 - Clearly see quadratic effects of long wires
 - Steps: added delays for extended drive pulses
- **Slow signaling rate**
 - At 3.8mm, $T_c=1.6\text{nS}$
 - Transactional control penalty damages BW



Optimizing wire bandwidth

Dual-path control GasP

- We can eliminate the ACK's dead time
 - Key notion: Let datapath do work during the ACK
 - If we keep datapath busy, we double the bandwidth

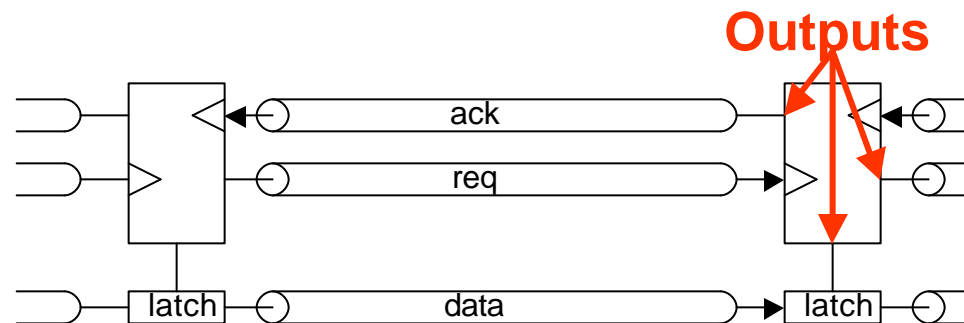


- Control drawn with two wires for simplicity
- GasP uses a single wire driven by both ends

Optimizing wire bandwidth

Dual-path control GasP

- We can eliminate the ACK's dead time
 - Key notion: Let datapath do work during the ACK
 - If we keep datapath busy, we double the bandwidth

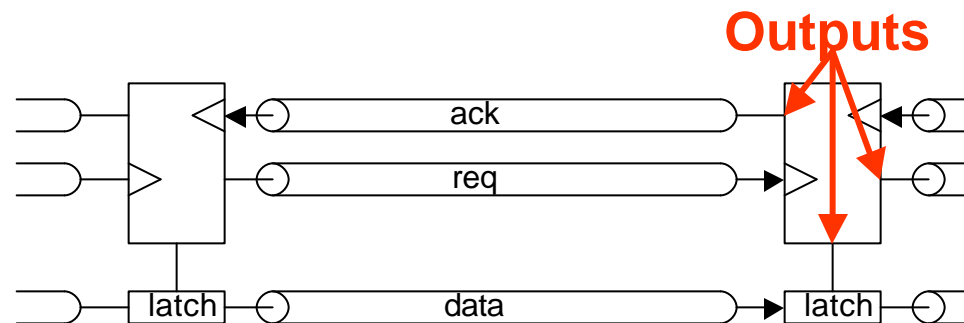


- Control drawn with two wires for simplicity
- GasP uses a single wire driven by both ends

Optimizing wire bandwidth

Dual-path control GasP

- We can eliminate the ACK's dead time
 - Key notion: Let datapath do work during the ACK
 - If we keep datapath busy, we double the bandwidth



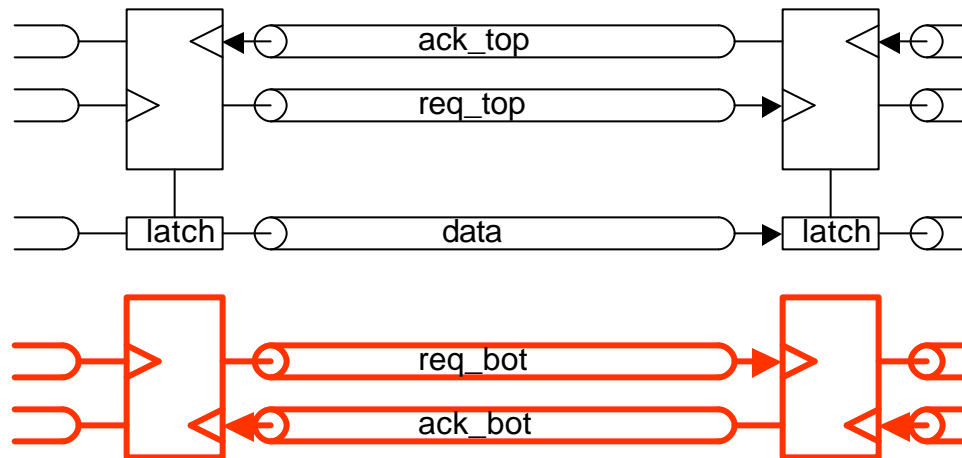
Outputs
fire iff
all inputs
arrive

- Control drawn with two wires for simplicity
- GasP uses a single wire driven by both ends

Optimizing wire bandwidth

Dual-path control GasP

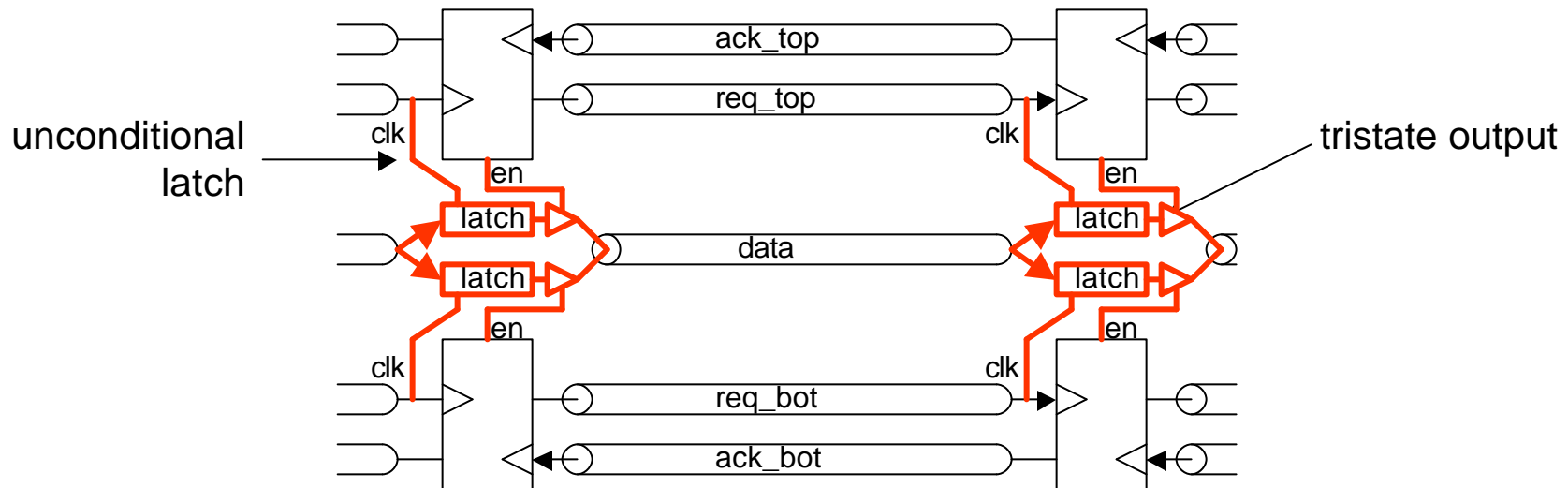
- **Dual, alternating control paths (top and bot)**
 - **When top is ACK-ing, bot is REQ-ing, & vice versa**
 - **But what does the bottom control path drive?**



Optimizing wire bandwidth

Dual-path control GasP

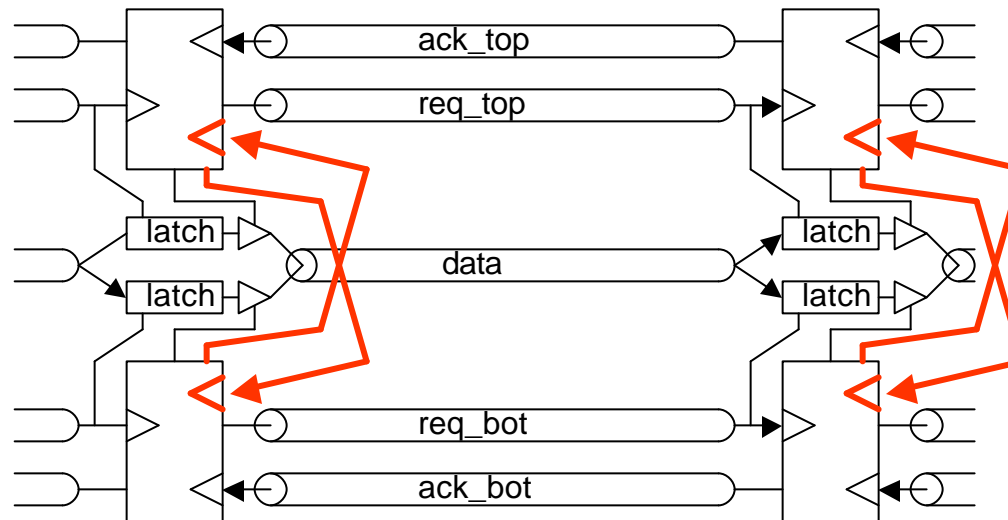
- **Answer: we double the datapath latches**
 - **Latches are muxed so use a tristate output**
 - **Latch inputs are unconditionally latched by REQ**



Optimizing wire bandwidth

Dual-path control GasP

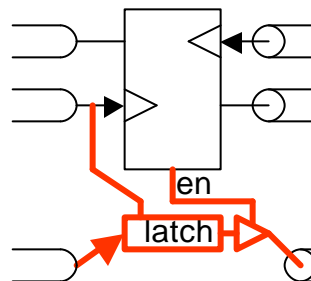
- **Not quite right: two paths must truly alternate**
 - **Otherwise one path's data can clobber the other's**
 - **So insert an alternation token between paths**
 - **Alternation path delay should match data delay**



Optimizing wire bandwidth

It's slower for short wires

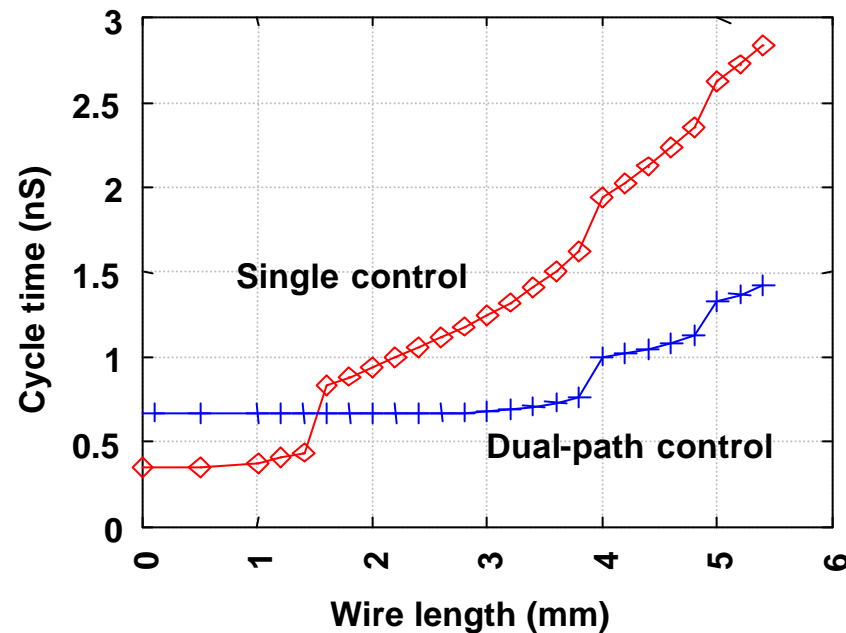
- Recall we used an unconditional latch
 - Causes a critical path in the control
 - Data must flow through latch before control reaches the GasP stage
 - To fix this, delay the reset of the GasP stage
 - *Same* tweak we did earlier to drive long wires



Optimizing wire bandwidth

Simulations of dual-path control GasP

- **Dual-path control has ~2x bandwidth gain**
 - **No extra delay at 1.6mm: already added for latches**
 - **Dual-path control hides wire effects up to ~4mm**
 - **Datapath best ~3.8mm; control best < 4mm**



Optimizing wire bandwidth

What's the cost?

- Looks promising, but beware the NFL* principle
- Area and power overhead
 - Added extra control wires and GasP module
 - Minor compared to a 64bit datapath
 - Added extra latch for *every* datapath bit
 - Not a big deal if the wires span 4mm lengths
- Reliability (noise) concern, dual path or not
 - Long pulse-encoded control wires are scary
 - We can always trade off area for reliability

*NFL = “no free lunch”

Optimizing wire bandwidth

Other costs

- **Performance overhead**
 - Only for very short wires
 - Bandwidth improvement for typical length wires
- **Complexity overhead**
 - Very large – lots of manual spice simulation
 - Lack of CAD tools and flows complicates design
 - Restricts usage to homogenous, regular wires
 - Design-once, use-many

What about power?

Wire optimizations

- **Datapath wires consume lots of power**
- **Minimizing transitions by coding: helps a little**
 - **One-hot encoding trades more area for less power**
 - **E.g., 8-bit bus goes from 4 to 1 avg transitions**
- **Minimizing voltage swing: helps a lot**
 - **1.8v to 0.1v saves 10x in power**
 - **Not 20x due to need for differential signaling**
 - **A lot more savings if we have a reduced supply**

What about power?

Data-bundled protocols

- **Reduced-swing data wires mandate data-bundling**
 - **Data must be amplified back to full logic levels**
 - **Amplification must be triggered**
 - **Flow-through amplifiers are inefficient**
- **Lots of literature for on-chip low-swing signaling**
 - **Not much on doing it asynchronously**
 - **An active area of work for us**

Conclusions

- **Long wires forcing us to rethink design issues**
 - **Motivate exploration of asynchronous repeaters**
- **Latency: use well-known repeater analytic models**
 - **Provide lowest-latency datapath design**
- **Bandwidth: dual-path control is promising**
 - **Reduces handshaking transactional penalty**
- **Power: Perhaps the most important parameter?**

Acknowledgments



Many thanks to:

- **Anonymous reviewers**
- **Bill Coates**
- **Jo Ebergen**
- **Bob Proebsting**
- **Ivan Sutherland**
- **DARPA, HPCS contract NBCH30390002**