Self-Timed Circuits

Ivan Sutherland
Portland State University

ivans@cecs.pdx.edu

Asynchronous Research Center
Portland State University
October 2011

Offering freedom from the tyranny of the clock

Calibration

- This font size appears in the slides

- **If you cannot read the line above, consider moving closer to the screen**

About the ARC at Portland State

- People
  - Marly Roncken (ex Philips & Intel) now PSU – CS Dept.
  - Ivan Sutherland (ex Sun) now PSU – ECE Dept.
  - Willem Mallon (ex Philips & NXP) now PSU – CS Dept.
- Sponsorship
  - College of Engineering, PSU
  - Oracle
  - DARPA
  - Seeking additional sponsors

Outline

- Kinetic Learning Activity (KLA)
- Pipelines
- Control protocols
- Performance & Logical Effort
- Data formats
- Gate models
  - semi modular versus inertial delay

KLA rules

- Predecessor and successor
- Use only one hand
- Pred has object AND you don't
- Take from predecessor
- NEVER PUT to successor

Pipeline action

- Conditions for action
  - > predecessor proffers data
  - > successor has space
- Three part atomic action:
  - > copy data
  - > make successor FULL
  - > make predecessor EMPTY

Pipeline is:

- Logic stage “L”
- Wires “w”

- w L w L w L w L w L w L w

- Wires may hold data (or not)

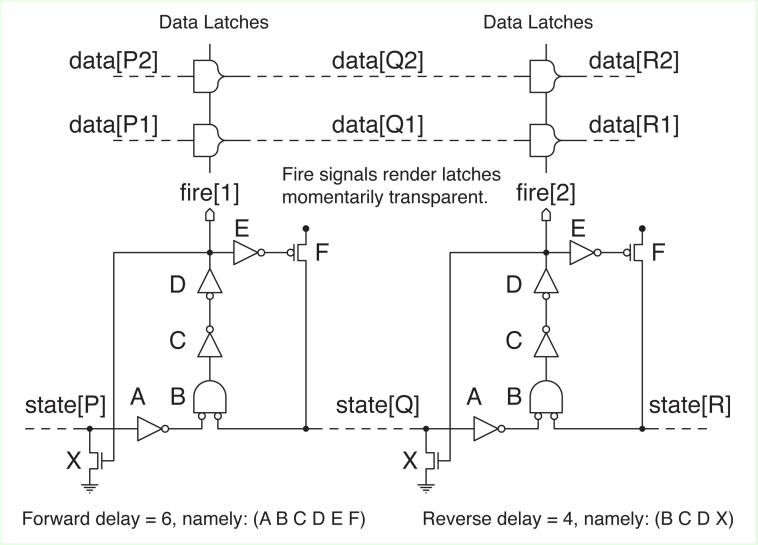
Control protocols

- Single wire (GasP)
  - > HI = data valid = FULL
- Two wires – four phase
  - > request HI = data valid
  - > acknowledge HI = data accepted
- Two wires – two phase NRZ
  - > differ = data valid = FULL
  - > double data rate (DDR)

6-4 GasP circuit

- Single control wire plus data
  - > HI is FULL convention
  - > bundled data
  - > control wire is bidirectional
- Least logical effort => fastest
  - > forward latency = 6
  - > reverse latency = 4
- Normally opaque data latches

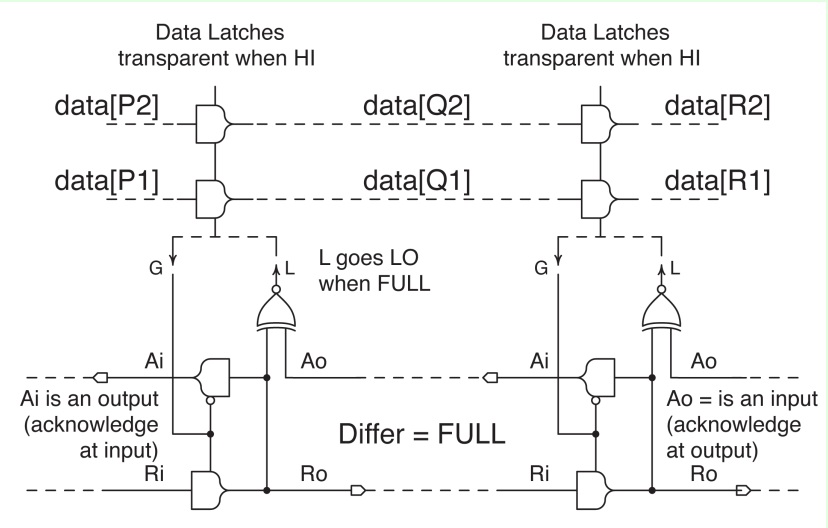
GasP circuit diagram



Charlie box

- Two wires between stages
  - > differ = FULL
  - > bundled data
  - > acknowledge after capture
- Fast in forward direction
  - > forward = 2 via one latch
  - > reverse = 4 via XOR + latch
- Normally transparent data latches

Charlie box circuit diagram



Pipeline essentials

- One AND function
  - pred FULL *and* succ EMPTY
- Data captured in latches or flip flops
- Some relative timing assumptions

- Compare with source clocking

Source and sink clocking

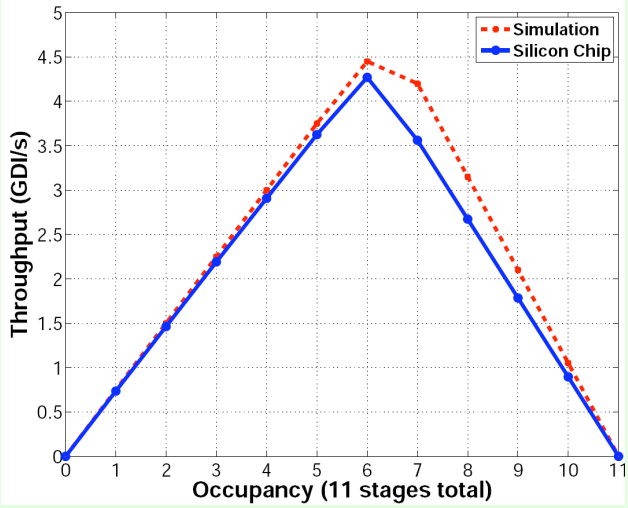
- Source clocking
  - clock moves forward with data

- Sink clocking
  - clock moves back with space

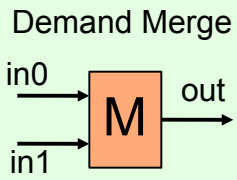
Canopy graph

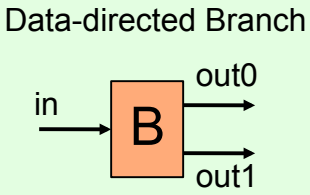
- Throughput versus occupancy
  - throughput in items/time: GDI/s
  - occupancy in items/length
- Maximum throughput
- Elastic = variable occupancy
  - can insert if not full
  - can withdraw if not empty

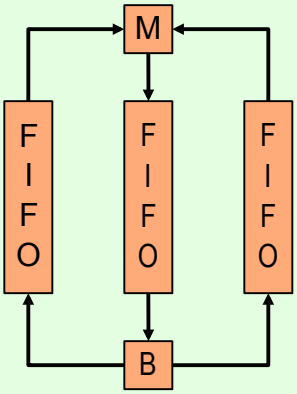
Throughput vs Occupancy for ring of 11 stages



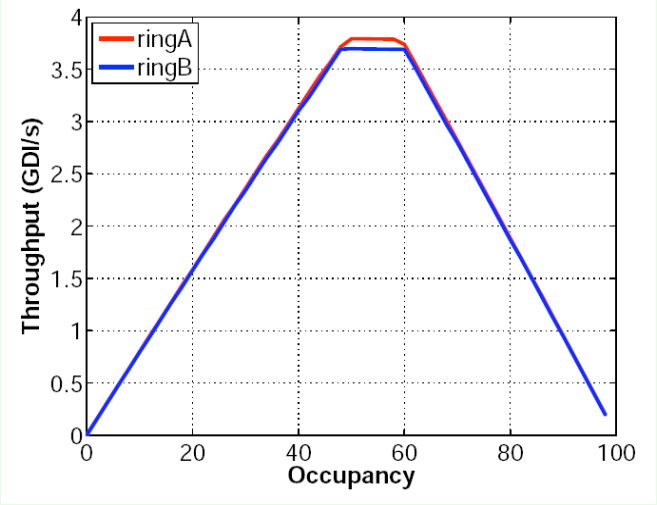
Infinity test







Infinity: Throughput vs Occupancy

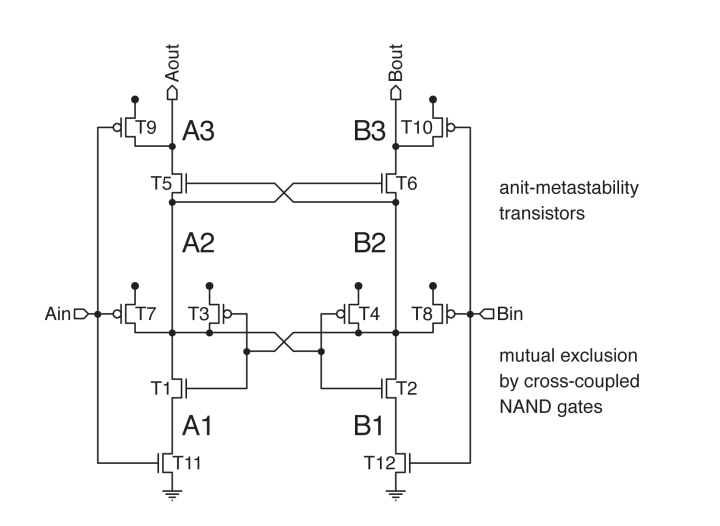


Mutual exclusion

- Two events at “same” time
  - > which choice doesn’t matter
  - > but choice must be clean
- Flip flop can hang metastable
  - > exit is Poisson distributed
  - > may take a long time, but rarely will
- Asynchronous system can wait

Continental divide

Mutual exclusion (Seitz)



Blend control with data

- Two wires per bit (like domino)
  - LO + LO = invalid
  - LO + HI = one
  - HI + LO = zero
- Plus one acknowledge wire
  - four-phase, NRZ, or single track
  - Fulcrum – four phase 18 FO4 cycle

Validation issues

- Loops in each pipeline stage
  - > interlocked with AND functions
  - > may not have latches or flip flops
  - > hard to verify with conventional tools
- Slope and delay model
  - > slope converges quickly
  - > delay converges if slope is known
  - > not available in conventional tools

Global state

- Is an unnecessary fiction
- Handshakes isolate local actions
- Transactions are what matters
- Pipelines are easy to think about
  - > local transactions tell all
  - > avoid state explosion
  - > painless concurrency

Fork and join pipes

- Fork sends same data both ways
- Join combines two data inputs
- Fork and joint parallel pipelines
  - > storage capacity of the shorter
  - > latency of the slower
  - > *slack matching* avoids
    - excess storage
    - excess latency

Unique verification tasks

- Combinational Loops
- Slack matching
- Working with local state
- Deadlock
  - > wormhole networks OK if no loops
  - > other cases?

Discussion