Circuits Without Clocks: What Makes Them Tick?

Jo Ebergen
Sun Microsystems Laboratories
jo.ebergen@sun.com

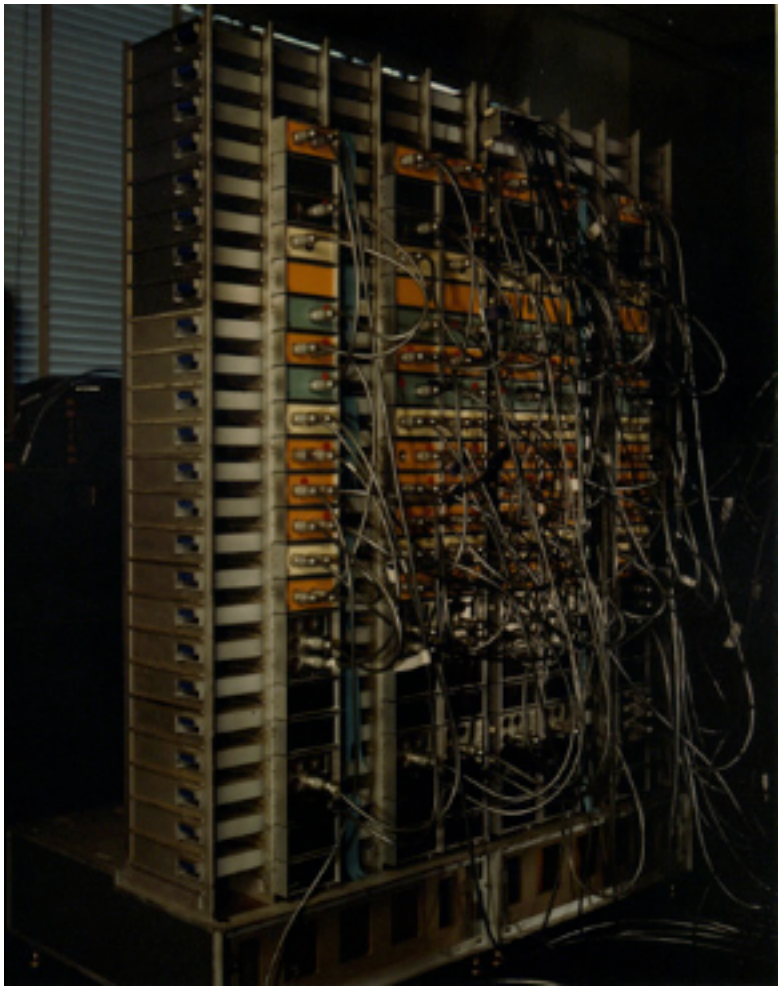
A Historical Perspective

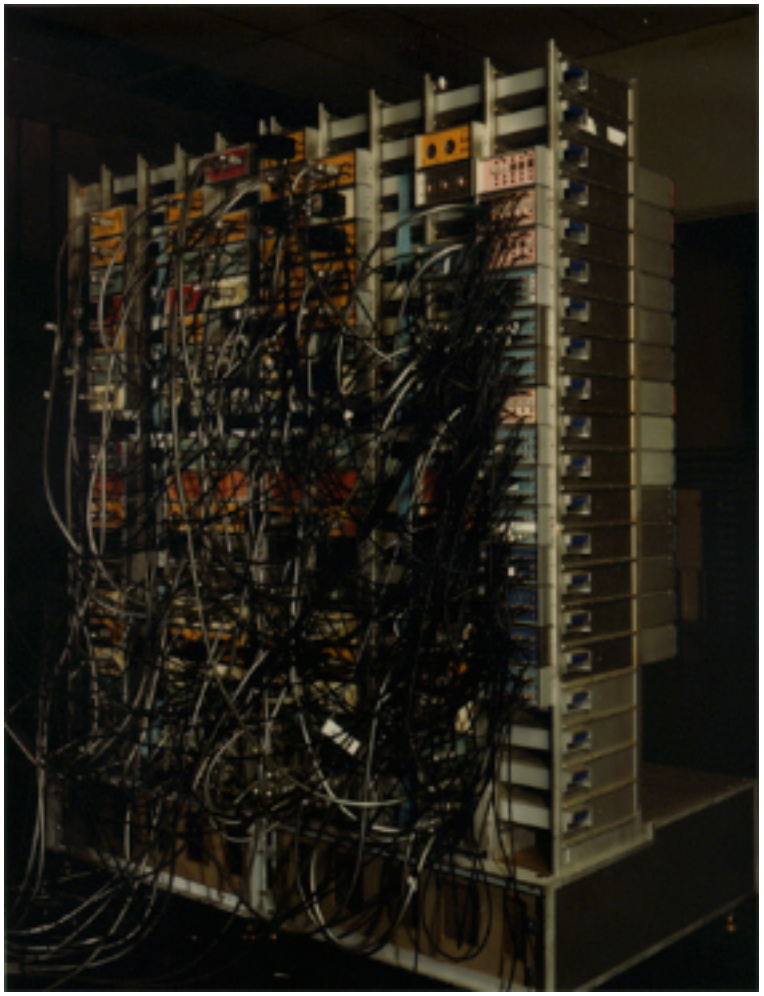
- Clocked versus clockless
- The first computers: clocked or clockless?
- ENIAC(’46), ACE(’46), EDVAC (’46)
- Enablers: Switching Theory and FSMs
- A BIG problem: making reliable computers
- Clock is convenient in design and verification
- Terminology: non-synchronous, asynchronous, speed-independent, self-timed, delay-insensitive

Why No Clock?

- Speed
  - Average vs worst-case behavior
- Avoid timing problems related to clock
  - Synchronization problem
  - Clock distribution, skew, jitter
- Modularity
  - Reusability
  - Composability
  - Incremental improvement

An Example: Macromodules ('66-'74)



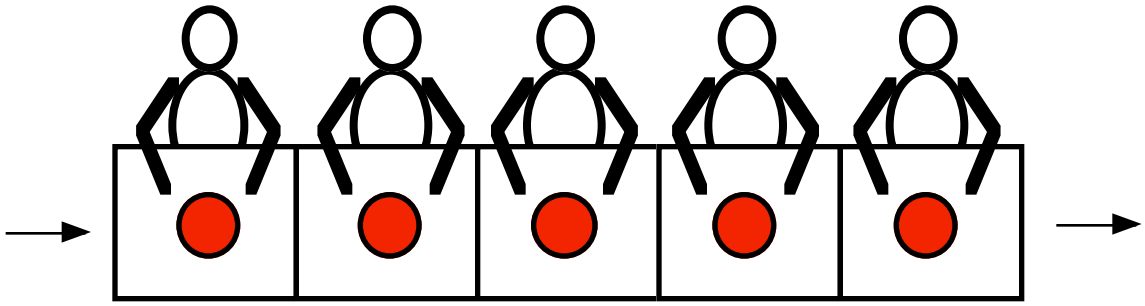


More: Why No Clock?

- Low power
  - Portable electronics
  - Budget constraint: total energy, not cycle time
- Architectural freedom
  - Concurrency at any grain size
- Robustness
  - Correct operation over a large range of process, voltage, and temperature variations
- Reduced electromagnetic radiation
  - Spread spectrum versus isolated peaks

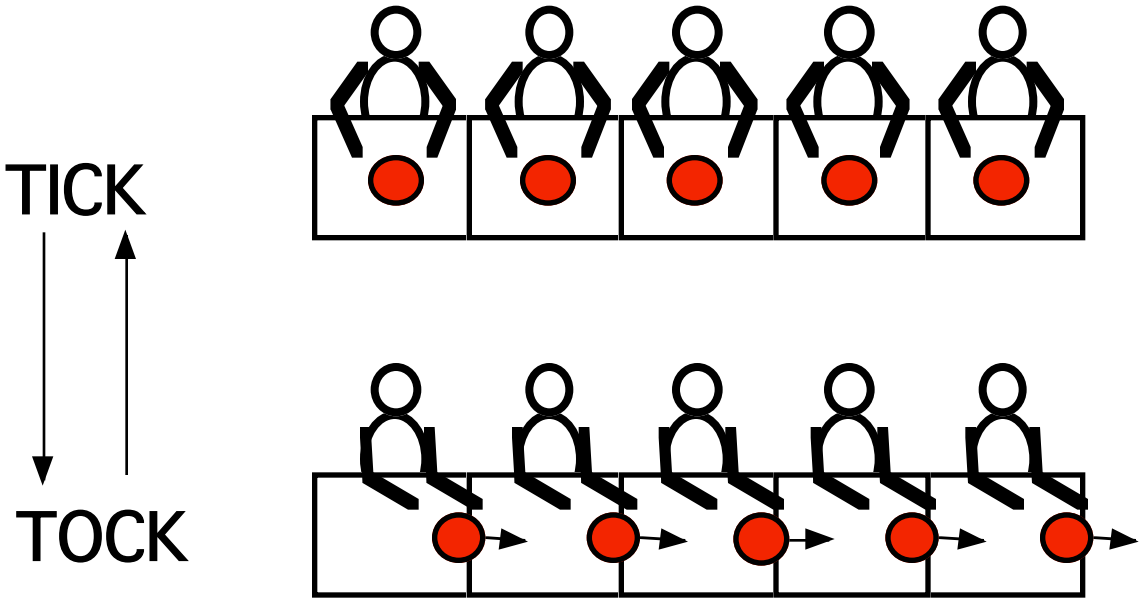
A Live Demo

- Pipelined “food” processing
- Clocked and clockless version



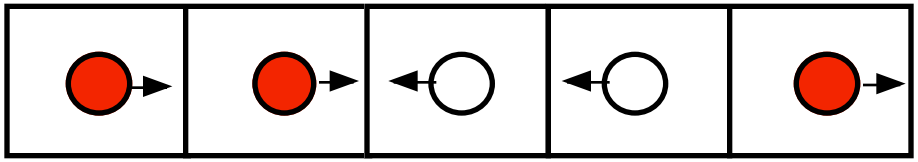
Clocked “Food Processing”

- **TICK: process food**
- **TOCK: move plate to successor**



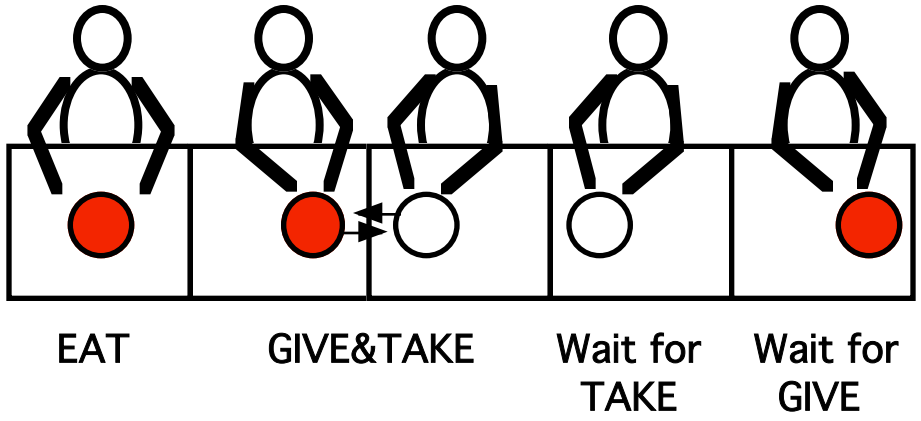
Clockless “Food Processing”

- Two types of plates
- Red plate = stage with data item
- White plate = empty stage
- Red plates move forward
- White plates move backward



Repeat These Actions

- **TAKE: Take RED plate from predecessor (and give WHITE plate)**
- **EAT: Process food**
- **GIVE: Give RED plate to successor (and take WHITE plate)**



To Notice

- **Clock period must be long enough for slowest person for worst food**
- ***Clockless pipeline adjusts speed to person and food being processed***

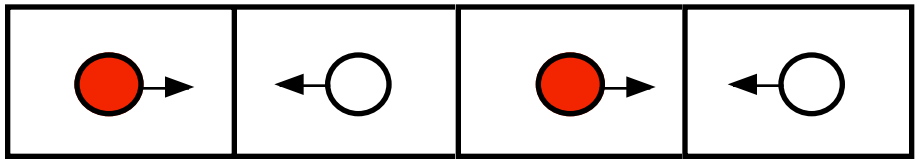
- **Clocked pipeline spends energy with every tick or tock, even when there is no food on plate**
- ***Clockless pipeline spends energy only when food must be processed***

More To Notice

- **Clocked pipeline must have the same frequency as the source and sink, to prevent overflow or underflow**
- ***Clockless pipeline has automatic overflow and underflow protection***

FIFOs

- FIFOs as a basis for pipelines
- Simple local communication

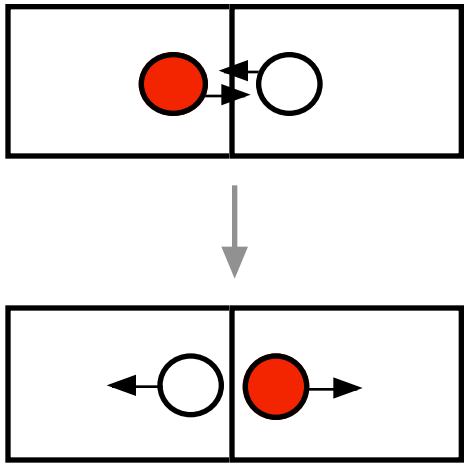


- Red plates move to the right
- White plates move to the left

Local Communication

- Synchronization at each boundary
- Only one rule:

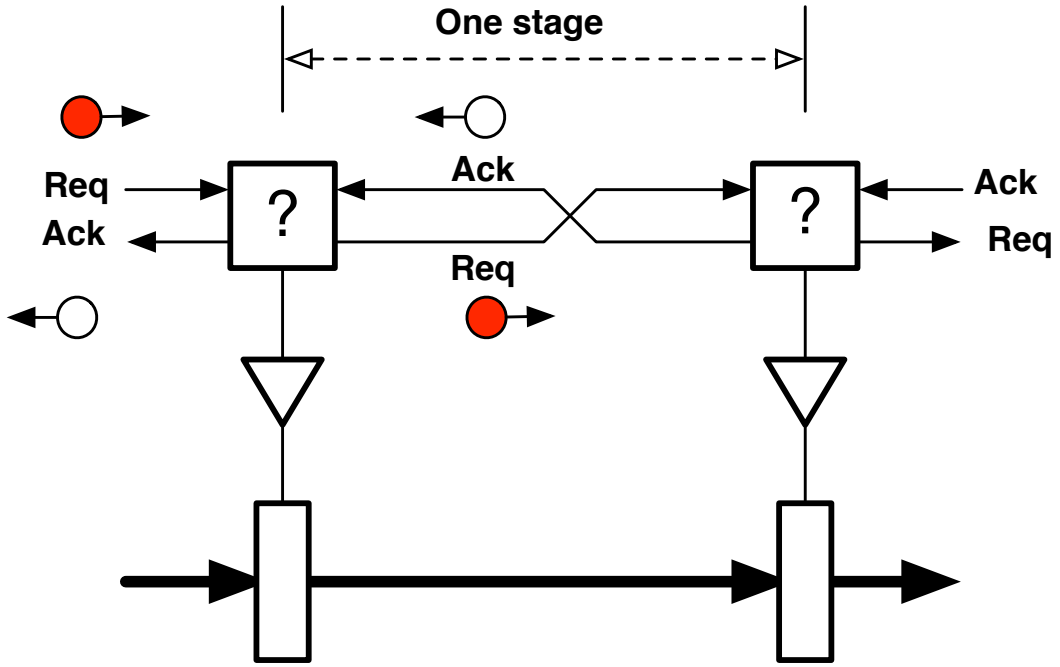
  IF (left plate is red AND right plate is white)
  THEN exchange plates



instantaneous event

An Implementation Scheme

- **Using bundled data signaling**



About Bundled Data Signaling

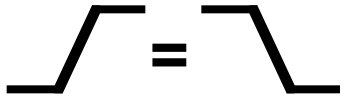
- Similar to clocked design, but with local “clocks”
- Control modules produce “clock” signal
  - when needed
  - where needed
- “Clock” signal can be
  - level signal
  - transition signal
  - pulse signal
- Type of latch or flip-flop depends on “clock” signal

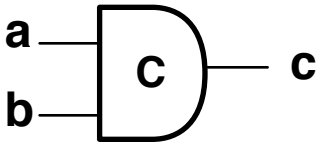
Many Implementations

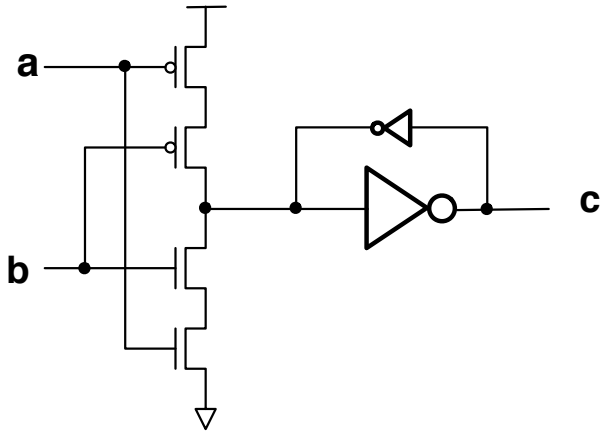
- Many circuits can implement local synchronization
  - Muller C-elements
  - asP* circuits
  - GasP modules
  - Many others...
- Challenge is to find circuits that
  - have small delay
  - have small area
  - consume little energy
  - can do pipelines with Branch, Merge, Fork, Join, etc

Muller C-Element

- Also known as Rendezvous or JOIN
- David Muller (60s)
- If both inputs are 1 (0), then output becomes 1 (0)
  - If inputs differ, output remains the same
- C-element = “AND” of transitions

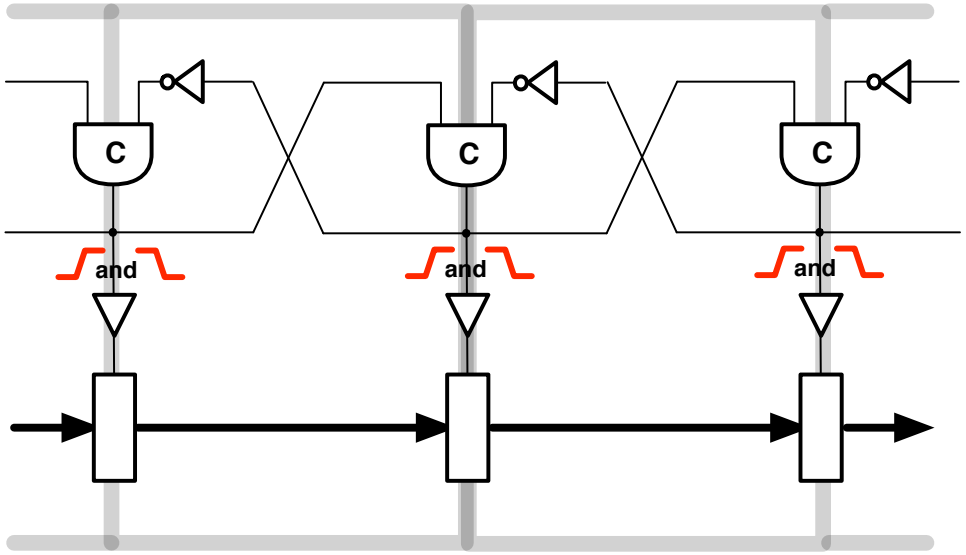






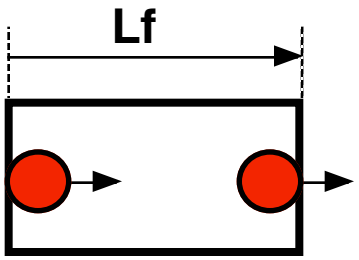
A C-element FIFO

- A simple micropipeline (Sutherland 88)
- Data must be stored with every transition
  - Makes data path more complex
- Cycle time is 5 gate delays

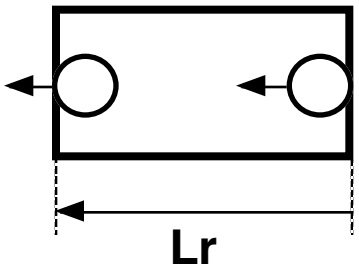


About Speed

- Forward latency of stage



- Reverse latency of a stage

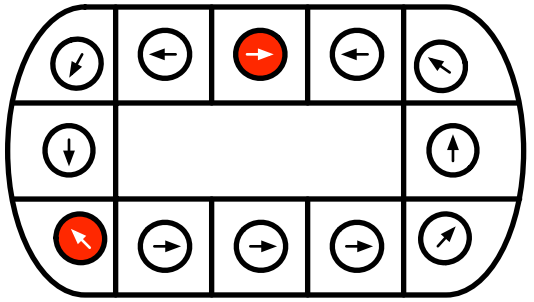


- Cycle time of a stage: Lf+Lr
- Forward latency of linear FIFO: n*Lf
- Reverse latency of linear FIFO: n*Lr
- Throughput of FIFO: 1/(Lf+Lr)

Measuring Speed

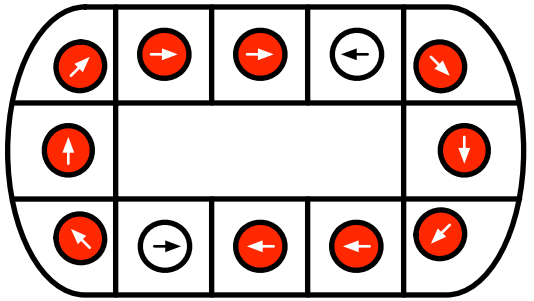
- **Throughput of ring FIFOs**
- **“Token-limited”**

  Red plates never have to wait



- **“Bubble-limited”**

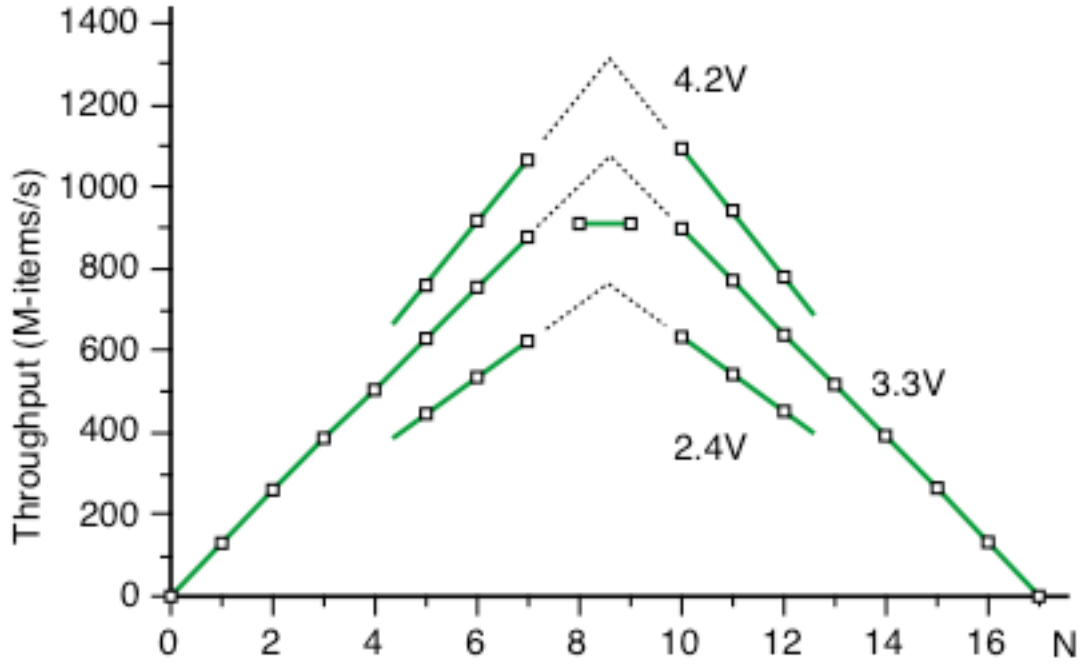
  White plates never have to wait



- **Limited by an interface stage**

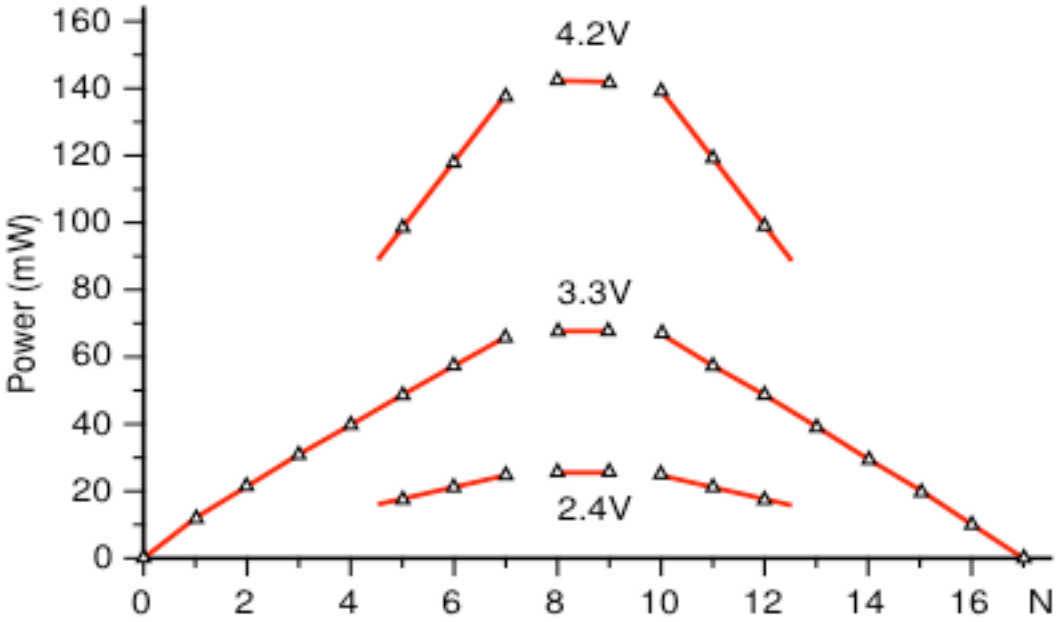
Speed Example

- **Measurement from chip, asP* FIFO (600nm, 3.3V)**

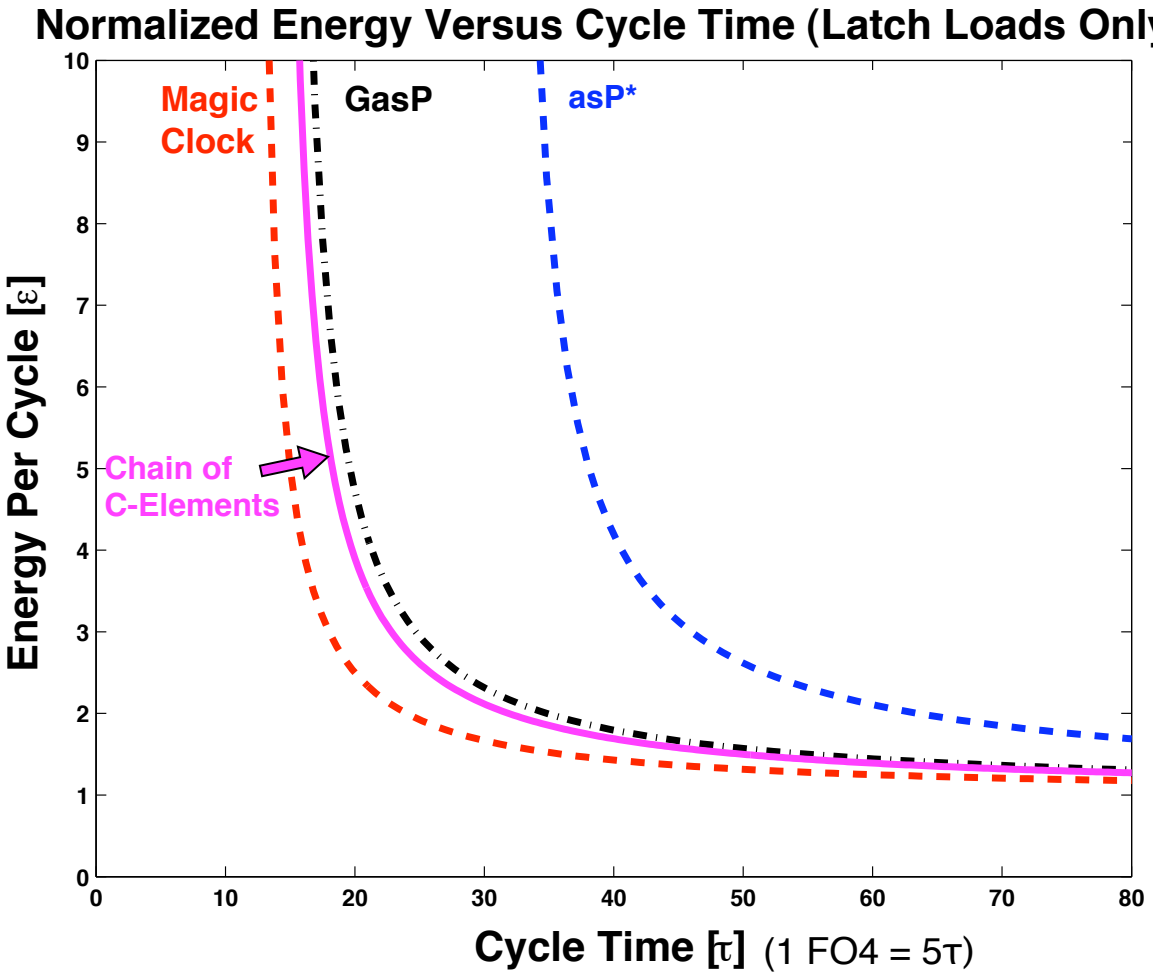


Power Example

- **Measured from chip, asP* FIFO (600nm, 3.3V)**
- **Power proportional to throughput!**



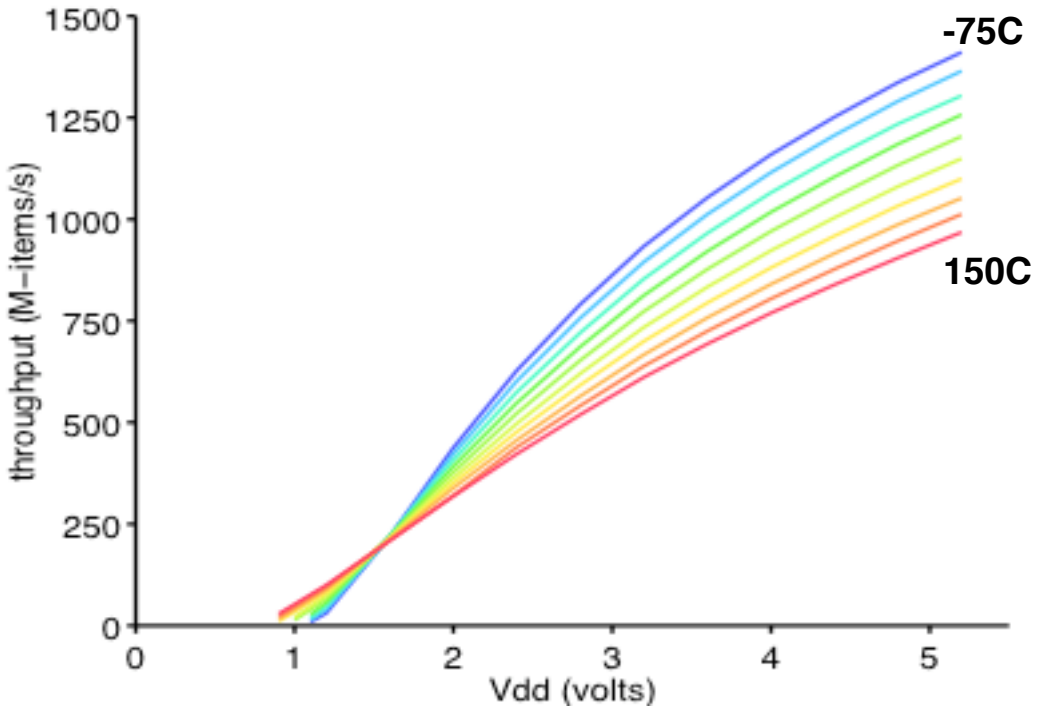
A Energy/Performance Comparison



1. Assumption: All gates have equal delay
2. Magic clock is infinite string of inverters with min cycle time of 6 gate delays
3. C-element FIFO has cycle time of 3 gate delays
4. In TSMC 180nm, τ = 14ps, ε=2.6fJ.

Robustness

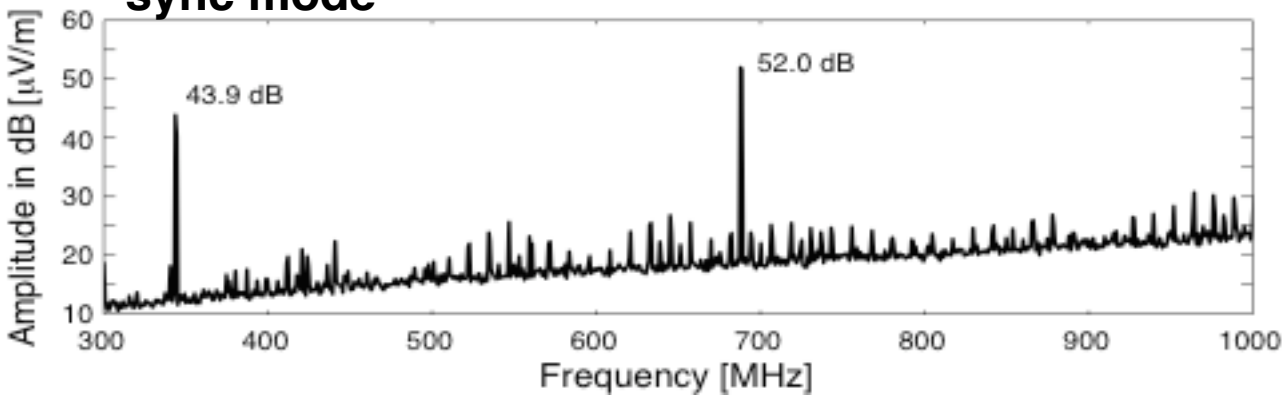
- Max throughput vs supply voltage and temperature
- C-element FIFO (600nm, 3.3V)



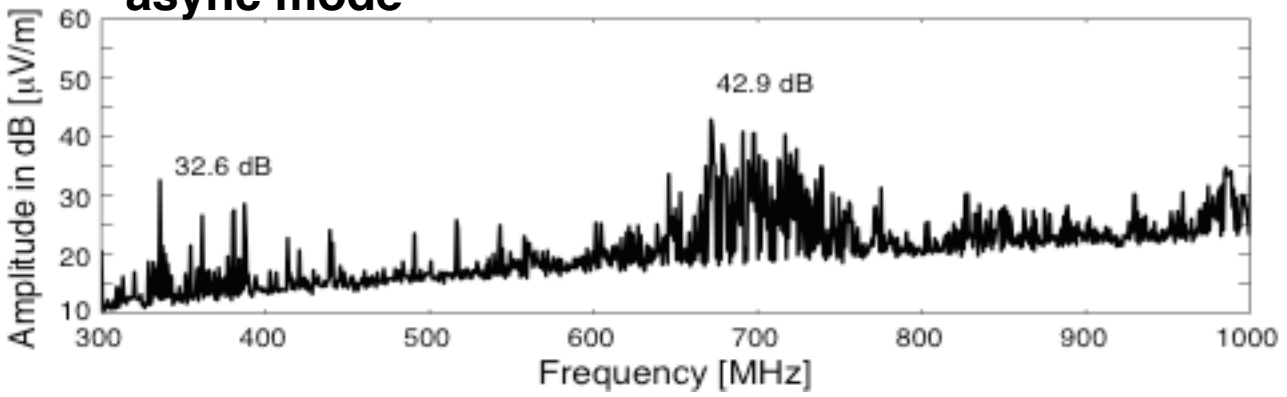
Reduced Radiation

- **Frequency spectrum of the same chip operating in sync and async mode**

sync mode

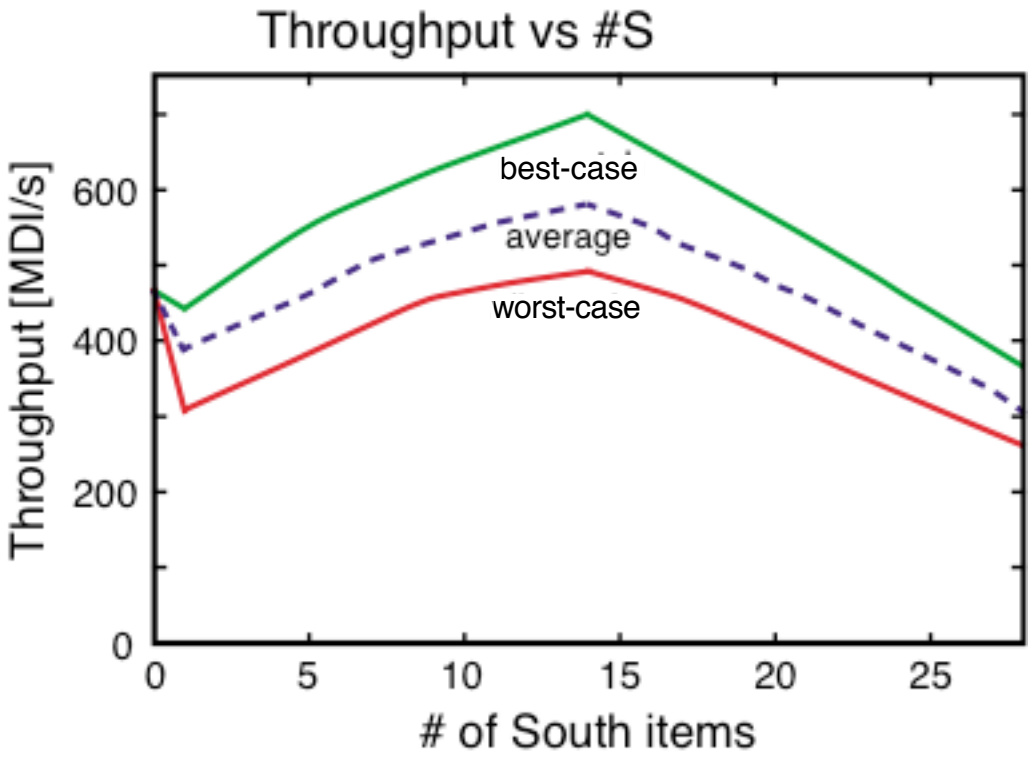


async mode



Average-Case Speed

- **From asP* counterflow pipeline chip**

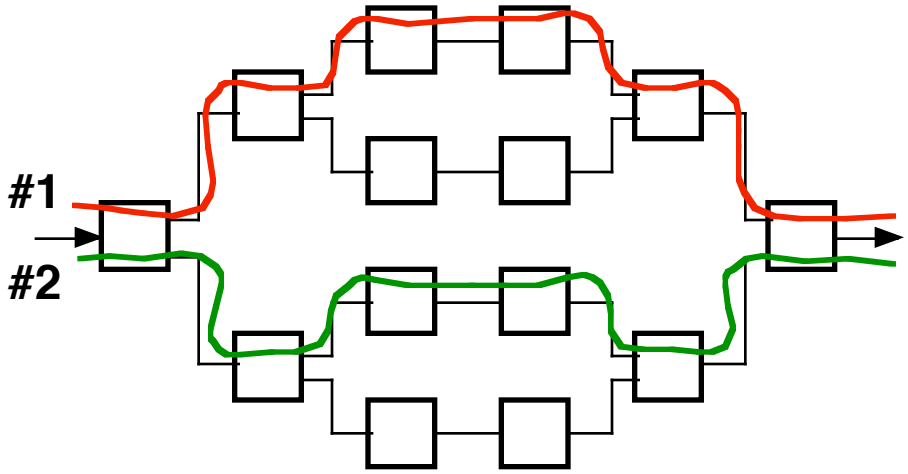


Architectural Freedom

- Can more freedom lead to better FIFOs?
- Better = lower power?
- Better = less latency?
- More concurrency?
- Different grain sizes (i.e. cycle times)?

A Non-Linear FIFO

- **The idea:**



- **What is energy consumption?**
- **What is latency?**
- **What is cycle time of leaf cell (compared to root cell)?**

Concluding Remarks

- Sun's UltraSPARCIIIi has async FIFOs
- Elsewhere..
  - Philips, IBM, Intel, ..
  - Academia
  - Start-ups
- Many challenges remain
  - CAD tools
  - Education
  - Creating order in "chaos of concurrency"

Resources

- Async Design Group at Sun Labs
  - research.sun.com/async/
- Async Logic Home Page
  - www.cs.man.ac.edu/async/
- Ivan Sutherland's Turing Award lecture 1989
- "Computers Without Clocks" Scientific American, Aug. 2002
- ASYNC conference series