
Lecture 11

Clocking High-Performance Microprocessors

Computer Systems Laboratory
Stanford University
horowitz@stanford.edu

Copyright © 2007 Ron Ho, Mark Horowitz

Overview

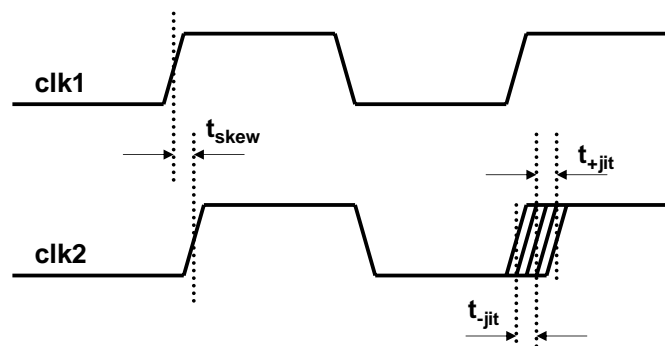
- Readings (Actually for today's lecture)
 - Gronowski Alpha clocking paper
 - Restle Clock grid paper
 - Harris Variations paper
- Today's topics
 - Overview of clock distribution methods and challenges
 - Three schemes of clock distribution
 - Overcoming and exploiting clock skew and jitter

Latches and Flops Need Clocks

- Previous lecture on latches and flops assumed an input clock
 - We talked briefly about imperfections in that clock: skew, jitter
 - In a chip, we can have n clocks for the latches and flops
 - $n=1$: one clock for the whole chip
 - $n<10$: one clock per major chip portion (e.g., core and cache)
 - $10<n<100$: one clock per major functional block (e.g., GALS)
 - $100<n<10000$: one clock per datapath stage
- ↓ asynchrony
- This lecture will focus on the globally synchronous systems
 - One or a few clocks distributed across the chip
 - How do we do this, and why is it so hard?

Principal Goal of Clock Distribution

- We need to distribute a **stable** clock across a die
 - Clock should be phase-aligned with an external input clock
 - Skew should be zero, or at least well-characterized
 - Jitter should be zero
 - Both skew and jitter should be insensitive to PVT
 - Duty cycle should be 50%-50%, or at least well-characterized

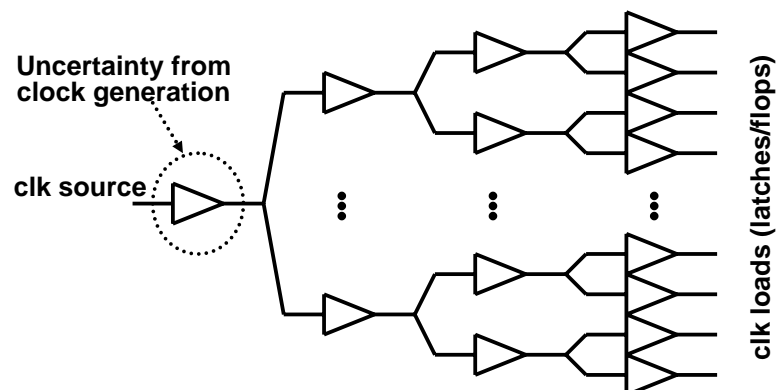


Other Goals of Clock Distribution

- Clock network should be low-power
 - But remember that most of the power is burned in the latches/flops
- Clock distribution should be simple to design and verify
- Clocking should allow external buses at slower frequencies
 - Bus/system clock is much slower than on-chip clock (why?)
 - Example: 533MHz bus clock, 3.0GHz chip clock
 - Often at a non-integer fraction of the CPU clock (3/5, 2/9...)
 - Every n^{th} CPU clock is phase-aligned with every m^{th} bus clock
- Clock generation should support test and debug
 - Control over frequency, skews, and duty cycle

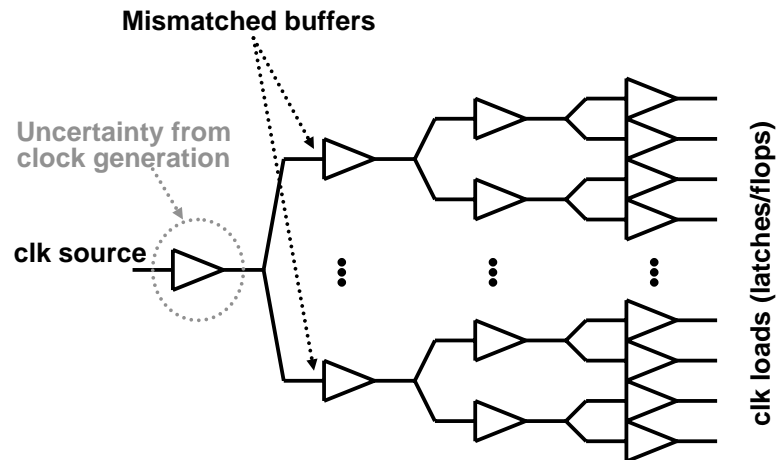
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



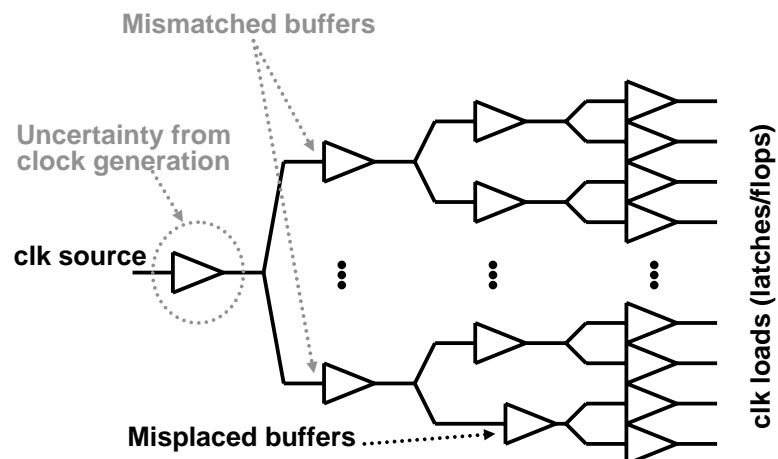
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



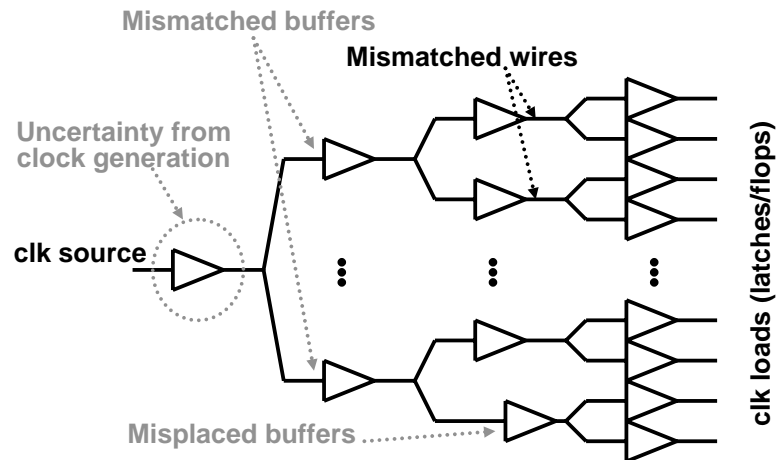
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



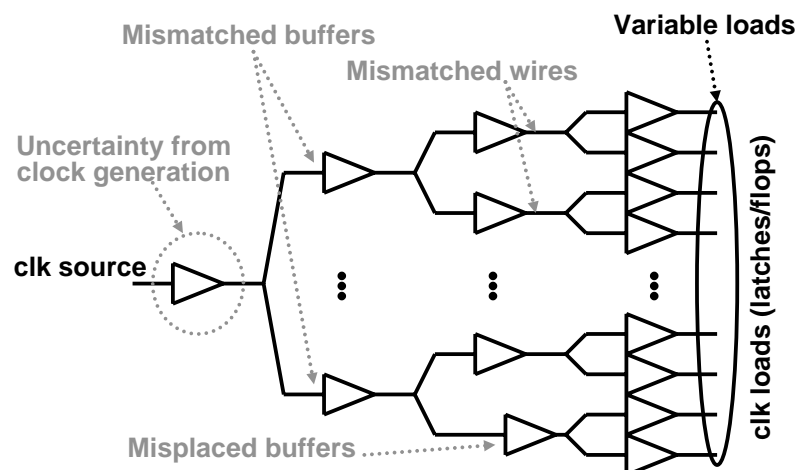
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



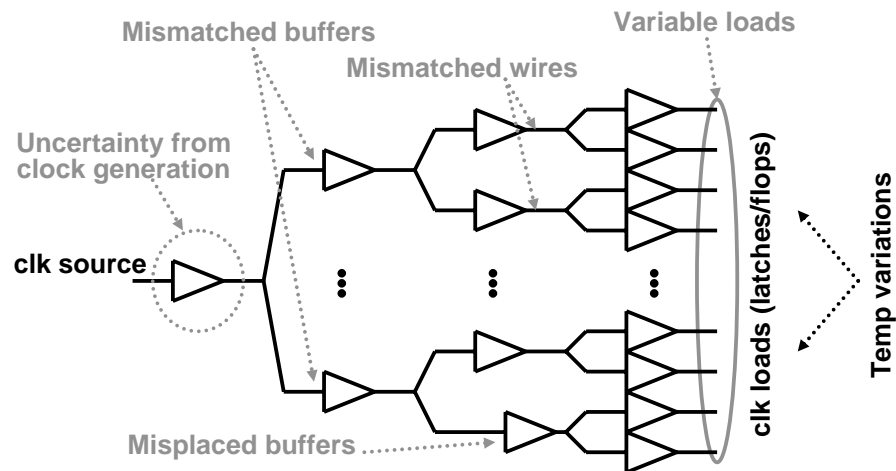
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



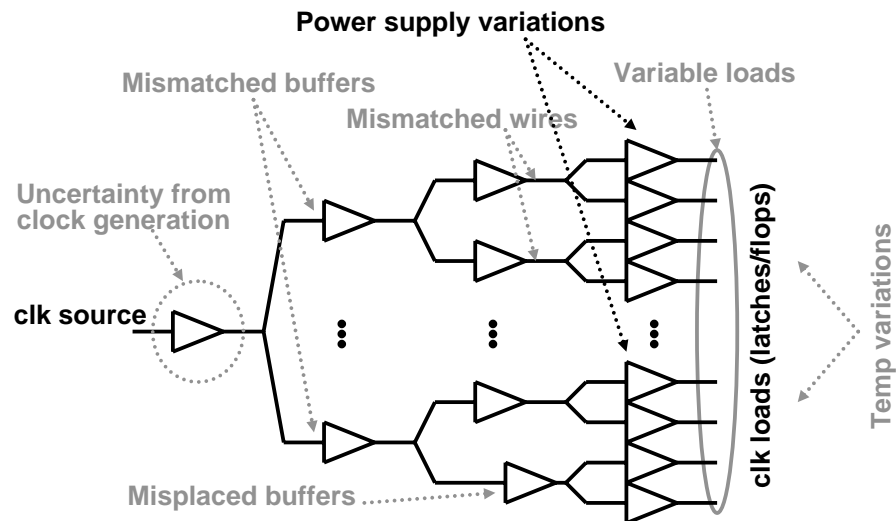
Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



Principal Goal is Difficult

- Uncertainty in the clocking network appears everywhere!
 - Affects skew, jitter, and duty cycle of the delivered clock



Mismatched Buffers

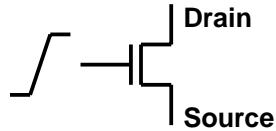
- Earlier we talked about process corners (FF, SS, TT, SF, etc.)
 - This represents variation from one die to another die
- There is also significant cross-die variation: “process tilt”
 - ITRS predicts 3σ of L_e will grow to be about 20% of L_e
 - Two devices on opposite chip sides will behave differently
 - Won’t show up in process corners: all devices identically fast in FF
- There is also local systematic variation: “proximity effects”
 - Local wiring density affects etch rates and thus dimension control
 - Use dummy structures to equalize and match environments
- Both process tilt and proximity effects must be considered

Mismatched Wires and Misplaced Buffers

- We would prefer to put buffers in “exactly the right places”
 - But usually we cannot due to congestion (middle of the cache?)
 - Leads to mismatched wire loads
- Wires don’t always fit “exactly right” either
 - They also have to fit in the middle of other layouts
 - You get different capacitive components (M5-M4, M5-M5, fringe...)
- But even if we put everything in the right places
 - (Expensive, but perhaps worth it for the clock)
 - Still have process tilt and proximity effects for wires
 - Again, not counted in process corner simulations
 - Designers often use shield wires and then ignore this

Variable Loads

- Clocks drive latches and flops, which can be passgate loads
 - Load depends on the data going through the passgate



Drain	Fall	Fall	Low	High	High	High	Rise
Source	Fall	Low	Low	Low	High	Rise	Rise
Gate load	1.3x	1.1x	1.0	0.8x	0.42x	0.31x	0.13x

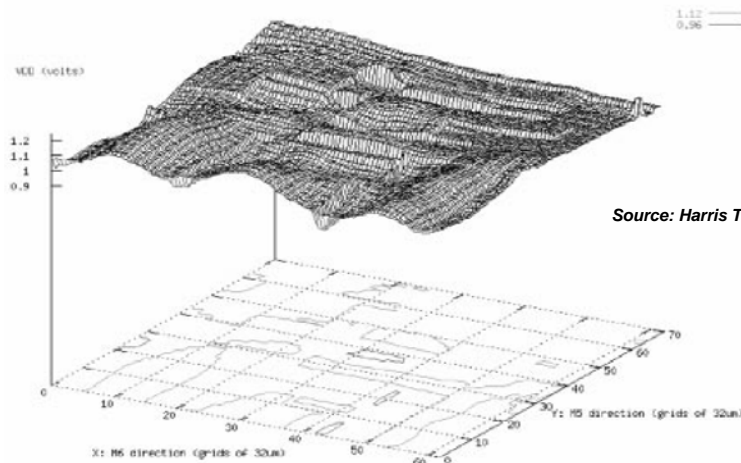
Source: Bailey JSSC 1998

2:1 variation

10:1 variation

Power Supply Variations

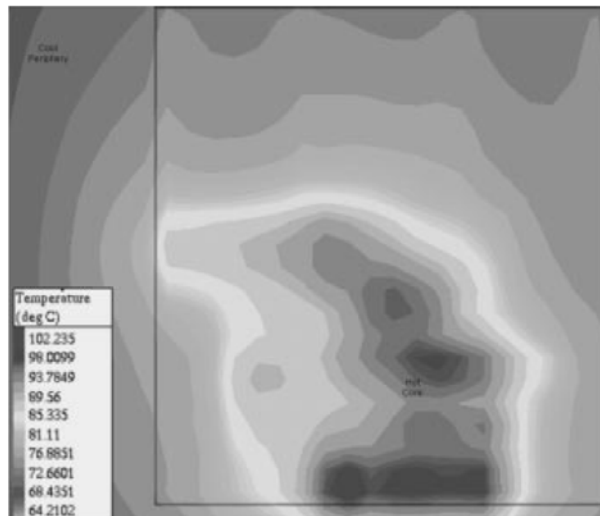
- RTL-based power traces can drive a global grid simulation
- Example of Itanium II power supply variations (snapshot in time)



Source: Harris TransVLSI 12/01

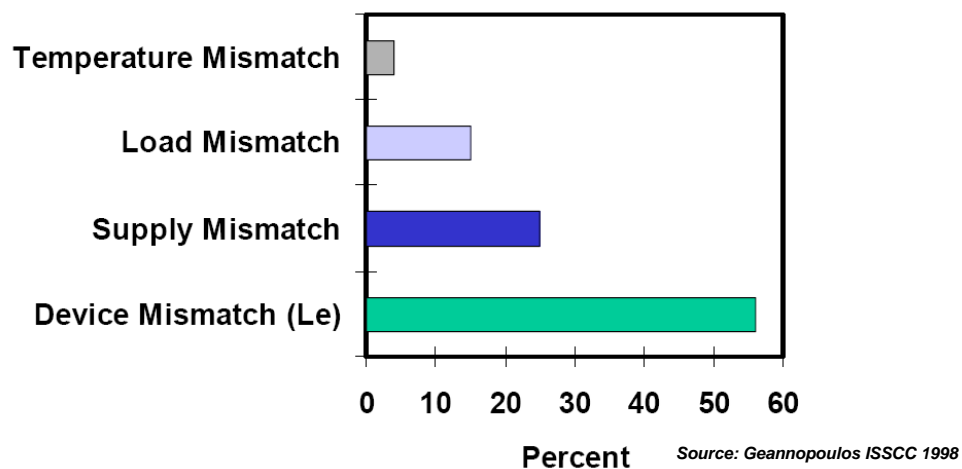
Temperature Variations

- Same for temperature variations, but these are slower effects
 - Long-term skew and/or jitter



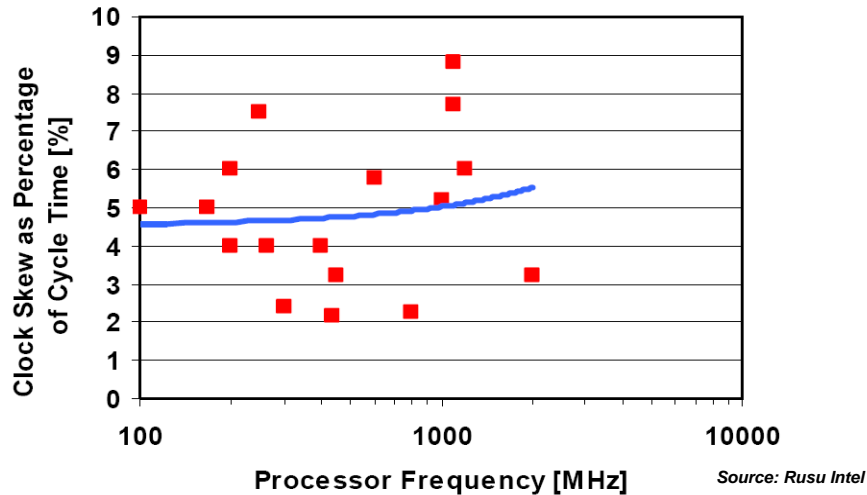
So What Makes Up Skew?

- If we eliminate systematic offsets we care mostly about devices



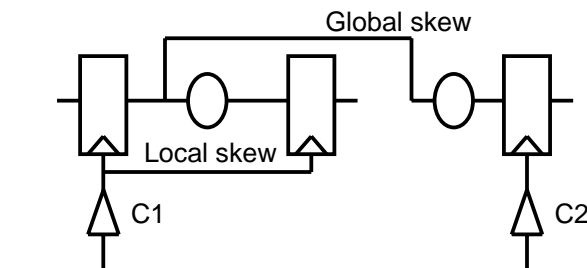
How Much Skew Do We Have?

- Historically, skew budgets have been about 5% of the cycle time
 - What people generally try to hit in designs



Simulating Skew

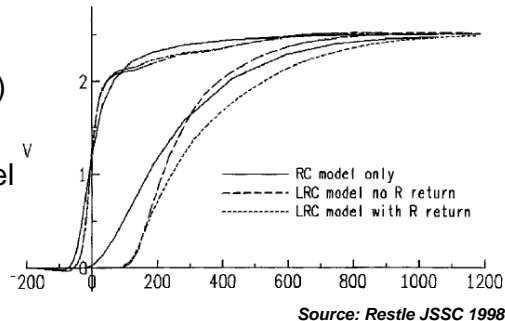
- Do not just gather all 3σ numbers in the worst possible way
 - Way too pessimistic; you'll never get the design done
 - You'll never see 3σ variation in clock buffers across a single datapath
- Statistical simulations figure out the “right” skew budget
 - Key point: skew from process variation affects **both** clock and data
 - Data variation makes local skew budgets worse...
 - ...but makes the difference in global and local skew smaller



Simulating Skew: A Note on Inductance

- Inductance doesn't matter for skinny signal wires
 - Short wires: driver resistance swamps out wire inductance
 - When $R_{\text{driver}} > 2Z_0$, where $Z_0 = \sqrt{\frac{R + j\omega L}{j\omega C}}$
 - Long wires: wire resistance swamps out wire inductance
 - When $R_{\text{wire}} > 2Z_0$ (i.e., if attenuation factor $\alpha > 1$)
 - RLC models are exceedingly complicated and time-consuming

- But: model L for clock networks
 - Wires are wide (much less R_{wire})
 - We care about pS of skew
 - Only a few wires (100s) to model



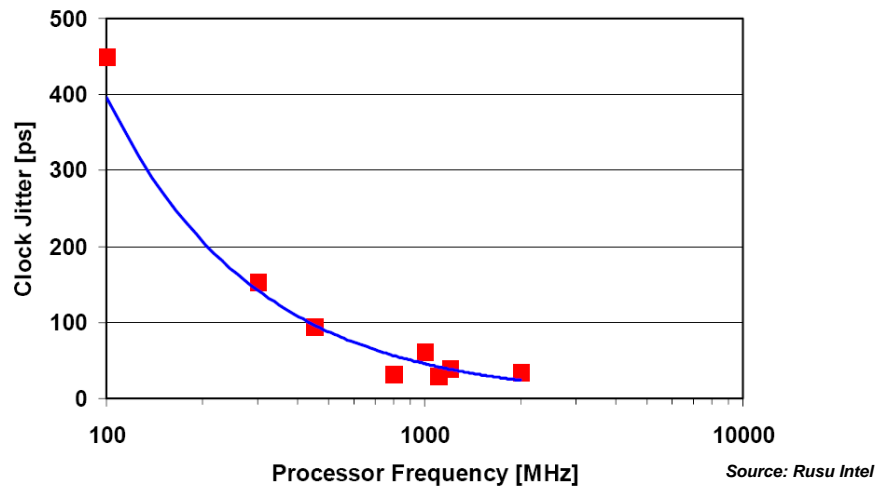
- L noise also worth modeling

What About Cycle-to-Cycle Variation (Jitter)?

- Jitter comes from temporal variations in the clock network
 - Some comes from the PLL/DLL, but not very much
 - Power supply changes mostly; temperature variations are slow
 - Also data coupling noise, but can design this away with shielding
- Jitter depends linearly on the delay through the clock network
 - “Insertion delay” is the time from PLL to the final latches
 - Around 2 cycles in CPUs today
- Note that jitter is primarily a max-path problem
 - Min-path problems are at the same clock edge
 - Although beware “differential jitter” between different clock nets

How Much Jitter Do We Have?

- Also around 5% of the cycle time (statistical simulations)
- But you should ask: how do people measure jitter? More later...

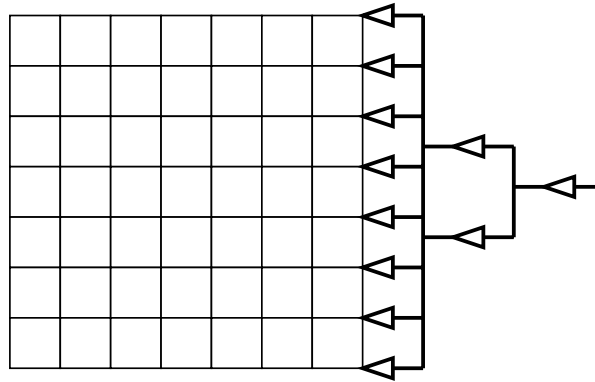


Clock Distribution Schemes

- Three principal ways of distributing clocks around a chip
 - Grid (DEC Alpha processors)
 - H-tree (IBM S/390, Intel Itanium)
 - Spines (Intel PentiumII, PentiumIII, P4)
 - Tree-Grid hybrids (IBM Power, Sun Ultrasparc)
- Other methods also being researched
 - Optical distribution
 - Resonant clocks

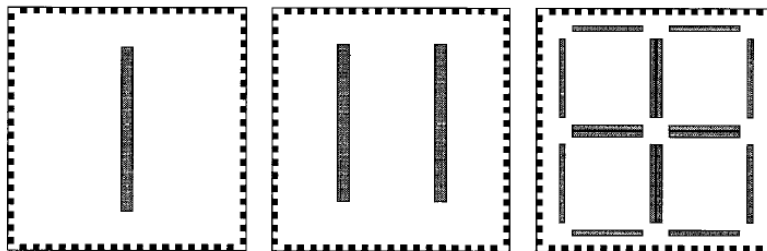
Clock Grids

- Cover the entire chip with a grid that carries clock
 - Good: simple, low skew (because everything is shorted together)
 - Bad: lots of power and metal resources consumed
- Need to figure out from where you will drive the grid



Three Generations of Gridded Alphas

- DEC Alpha used clock grids, driven by “panes” of drivers
 - 21064 (1993): 200MHz, 0.75 μ m process, one driver pane
 - 21164 (1996): 333MHz, 0.5 μ m process, two driver panes
 - 21264 (1997): 600MHz, 0.35 μ m process, sixteen driver panes



Source: Gronowski JSSC 1998

Alpha Gridded Skews

- Skews dropped: 240pS to 120pS to 75pS
 - As fraction of T_{cycle} : 4.8%, 3.6%, 4.5%
 - Simulated only, not measured

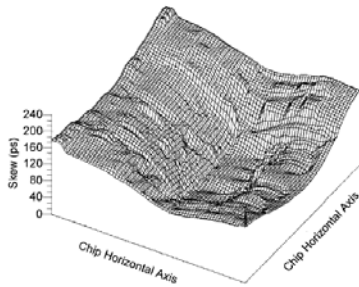


Fig. 14. 21064 clock skew.

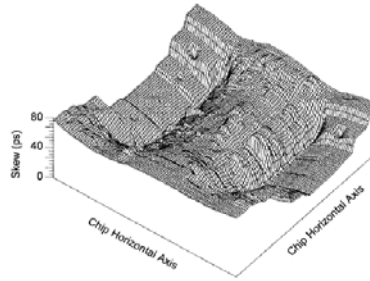


Fig. 17. 21164 clock skew.

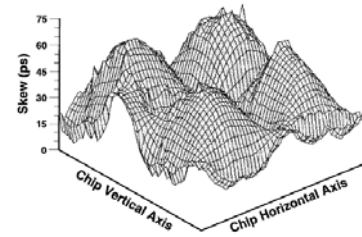
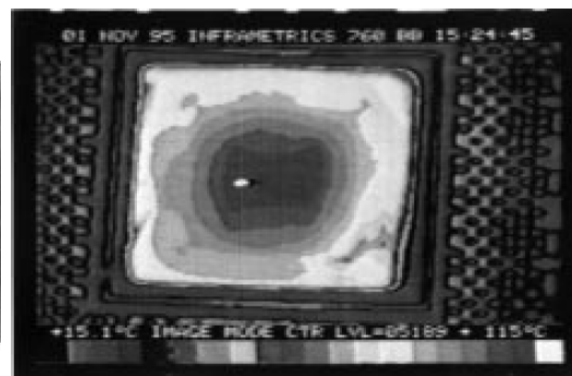


Fig. 20. 21264 GCLK skew.

Source: Gronowski JSSC 1998

Thermal Image of 21064 and 21164

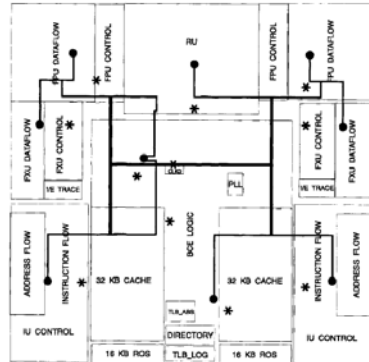
- Single pane driver on 21064 was definitely a hotspot
 - Splitting into two panes reduced thermal gradients significantly



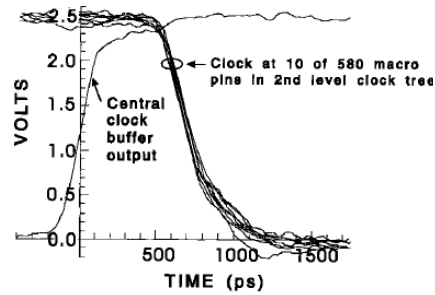
Source: Gronowski JSSC 1998

Clock H-Trees

- Balanced H-Trees are complex to design but are appealing
 - Good: Low power, low routing resources
 - Bad: Skew between divergent branches, complex design
- Example: IBM S/390 used balanced RC paths in the tree



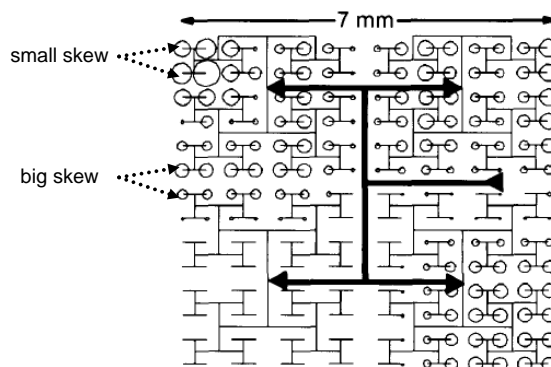
• Clock Sector Buffer
* Clock Waveform Measurement Point



Source: Webb JSSC 1997

H-Trees and Points of Divergence

- Skew between nodes on an H-Tree depends on location
 - Their last common node is their “Point of Divergence” (POD)
 - Two nodes with a near POD = low skew (10% s+j budget)
 - Two nodes with a far POD = high skew (20% s+j budget)
- Trees must balance RC delays with non-uniform loads

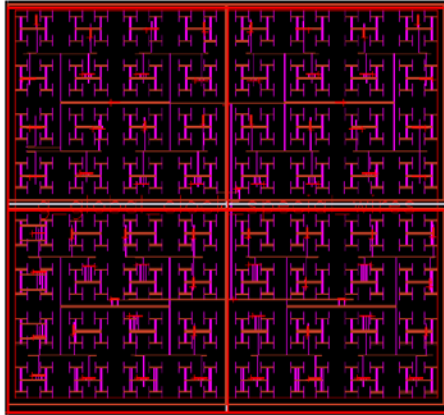


IBM testchip with loads
shown by diameter

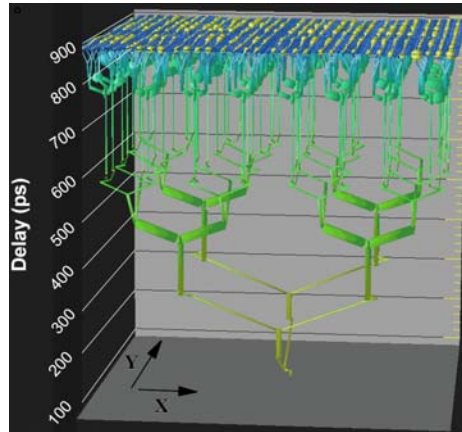
Source: Restle VLSI 2000

Hybrid: Trees Driving Grids

- Eliminates skew differences due to POD
 - But consumes lots of power, like grids
 - IBM Power4 in 0.18 μ m consumes 70% power in clocks and latches



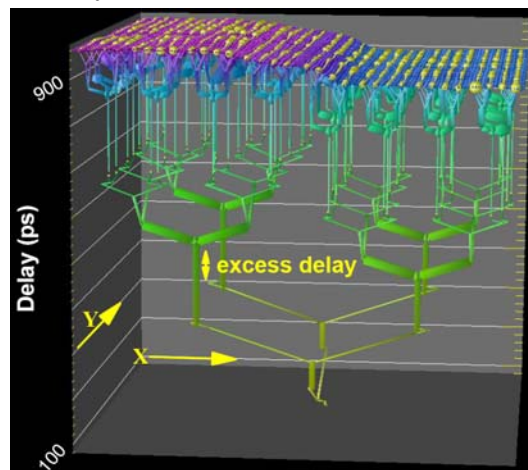
Source: Anderson ISSCC 2001



Source: Restle ISSCC 2001

Hybrid: Trees Driving Grids

- Simulations of skew-smoothing that you get with grids
 - IBM Power4 with intentional skew added to one buffer side
 - Simulations only

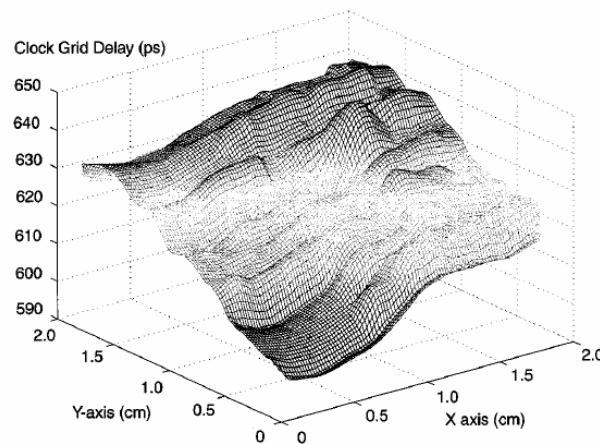


Animations of Grid Clocks

- Restle (IBM Research) does interesting work visualizing circuits
 - In particular, clock animations
 - <http://researchweb.watson.ibm.com/people/r/restle/Animations/DAC01top.html>

Hybrid: Trees Driving Grids

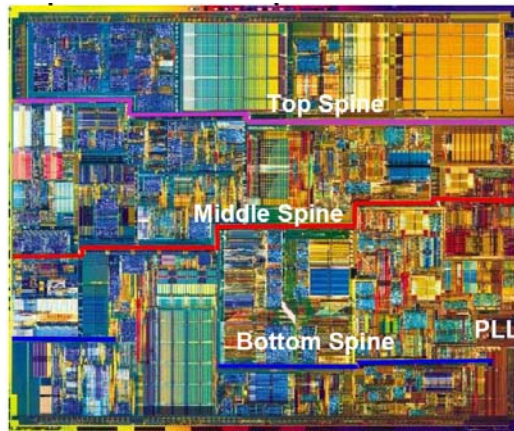
- Sun UltraSparcIII: balanced tree to global grid (skew simulated)



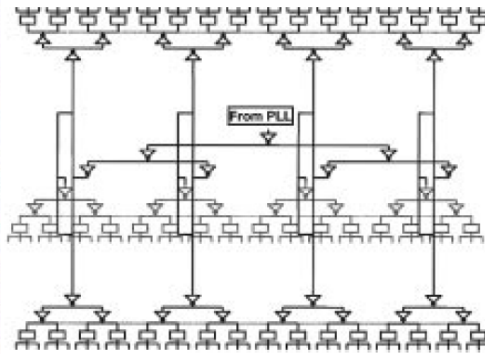
Source: Heald JSSC 2000

Clock Spines

- Spines look like a collapsed tree
 - Each spine is equipotential and can radiate outwards
- Example: First Intel Pentium4 microprocessor used three spines



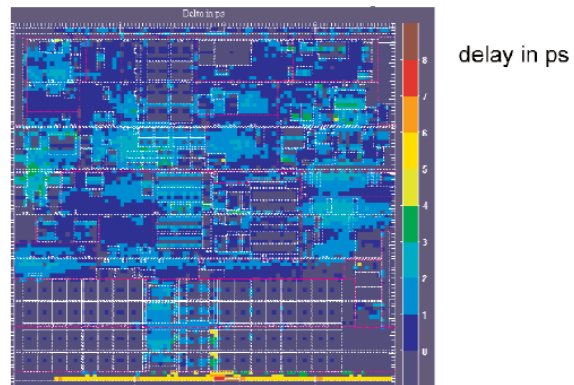
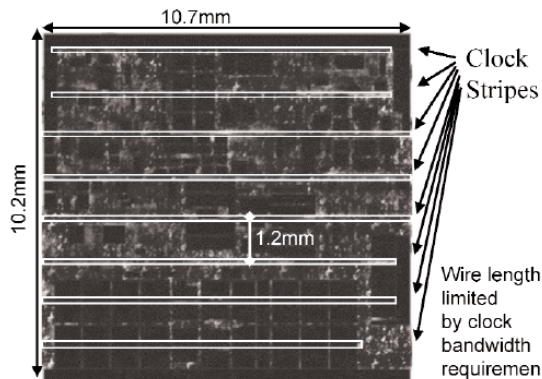
Source: Kurd ISSCC 2001



Source: Kurd JSSC 2001

Clock Spines Skew

- Later Intel's 90nm P4 added more spines
 - Skew was simulated to be under 10pS

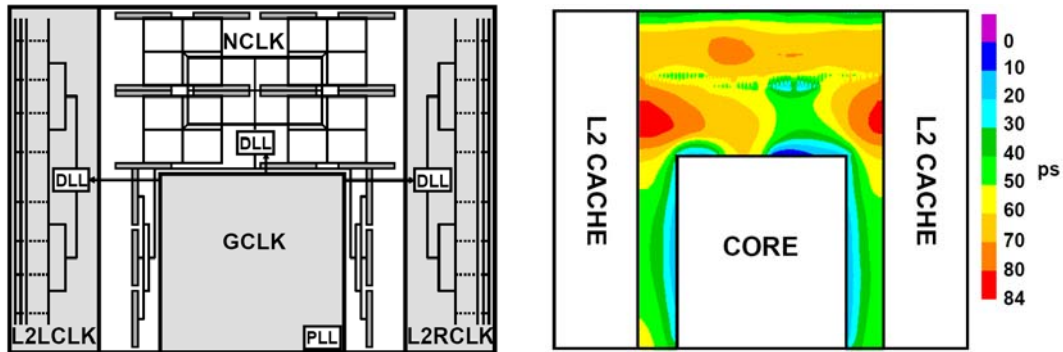


- Clock grid delay is modeled in timing
- Clock skew gradient is gradual

Source: Bindal ISSCC 2003

A True Hybrid

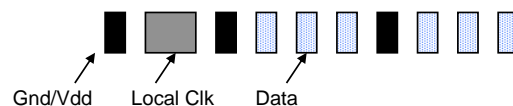
- Compaq Alpha (EV8, r.i.p.)
 - Grid in core, panes and trees in memory/network, trees in cache
 - Skew in the “glue” memory/network controller < 90pS



Source: Xanthopoulos ISSCC 2001

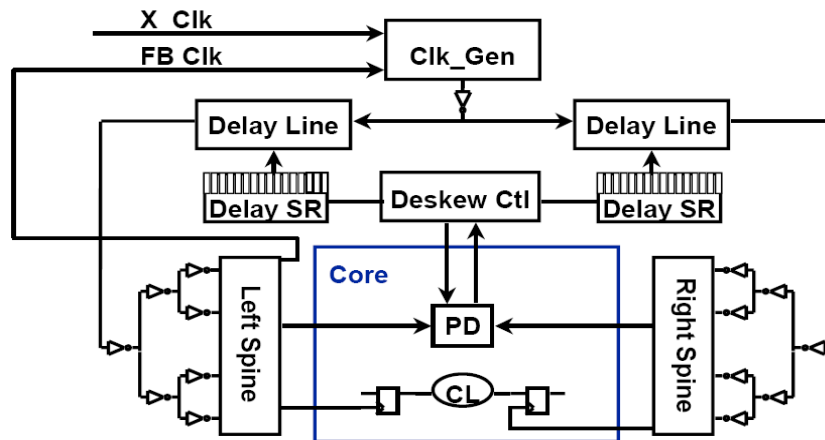
Minimizing Skew In Design

- Follow basic heuristics to match all gates
 - Single cell design with dummy structures around it, replicated
 - Same cell orientation in each instance
- Similar heuristics for wires
 - Consider and account for return path resistance
 - Heavily shield clock routes: makes C uniform and L minimal
 - Limit maximum wire dimensions: avoid skin effect
 - Simplify the problem by using templates
 - All of M5 looks like the following, repeated over the entire die



Active Skew Cancellation (Insertion)

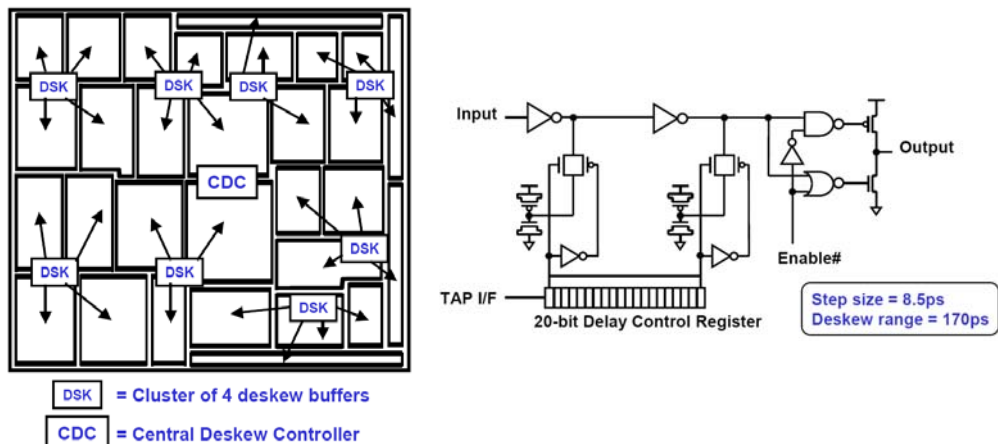
- Look at clocks and adjust them
 - Need to be careful about the loop bandwidth and stability concerns
- Example from Intel PentiumIII (dual spine design)



Source: Geannopoulos ISSCC 1998

Multi-zone Deskew Circuits

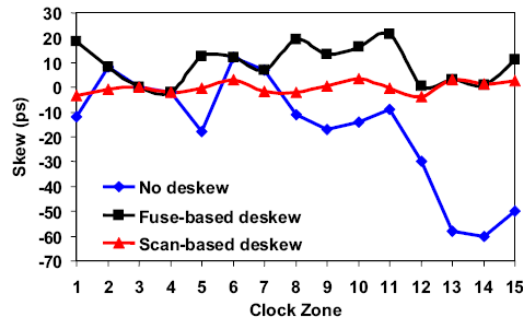
- Example from Intel Itanium clocking
 - Distribute an array of deskew buffers, each tying together 4 zones



Source: Rusu ISSCC 2000

Deskew Effectiveness

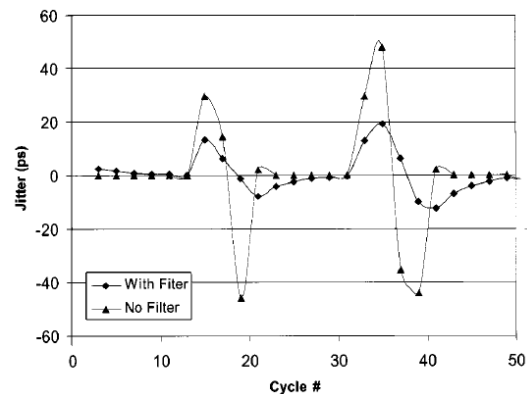
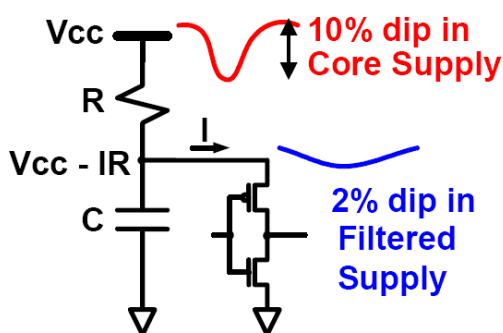
Chip	Publication	Zones	Skew	After	Stepsize
PentiumIII	ISSCC98	2	60pS	15pS	12pS
Itanium	ISSCC00	30	110pS	28pS	8pS
Pentium4	ISSCC01	47	64pS	16pS	8pS
Itanium (3 rd G)	ISSCC03	23	60pS	7pS	7pS



• Worst case clock skew is 24ps in fuse mode and 7ps in scan mode *Source: Stinson ISSCC 2003*

Minimizing Jitter In Design

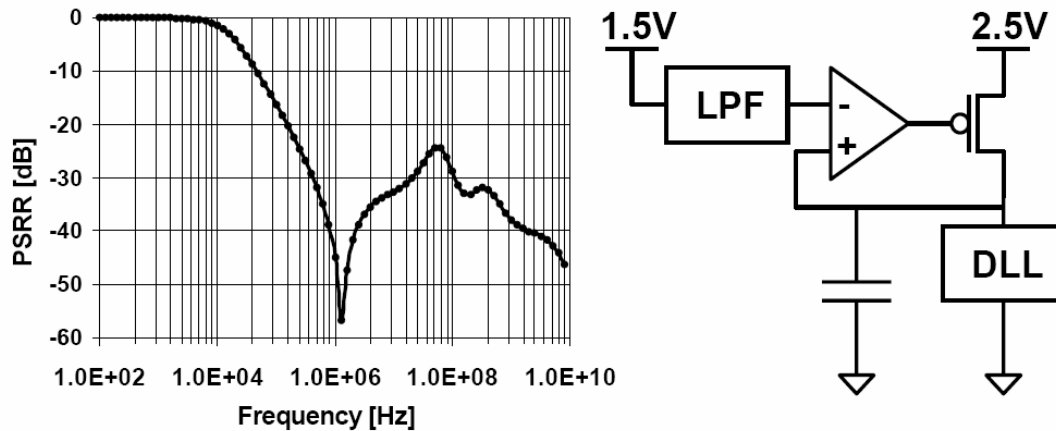
- Jitter predominantly comes from power supply noise
 - So filter the clock power supply
- Example from Pentium4 clock network (simulated)



Source: Kurd JSSC 2001

Minimizing Jitter In Design, con't

- Compaq Alpha EV8 used essentially the same idea (simulated)

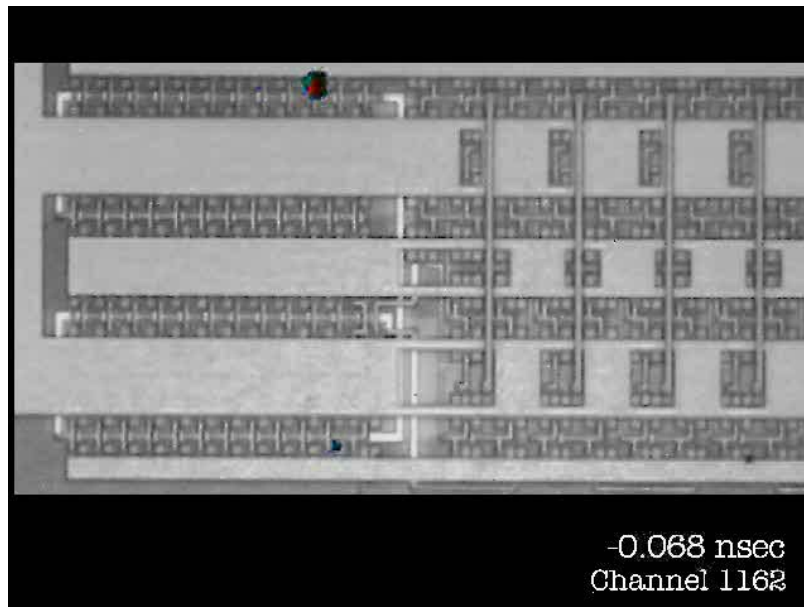


Source: Xanthopoulos ISSCC 2001

Measuring Skew and Jitter

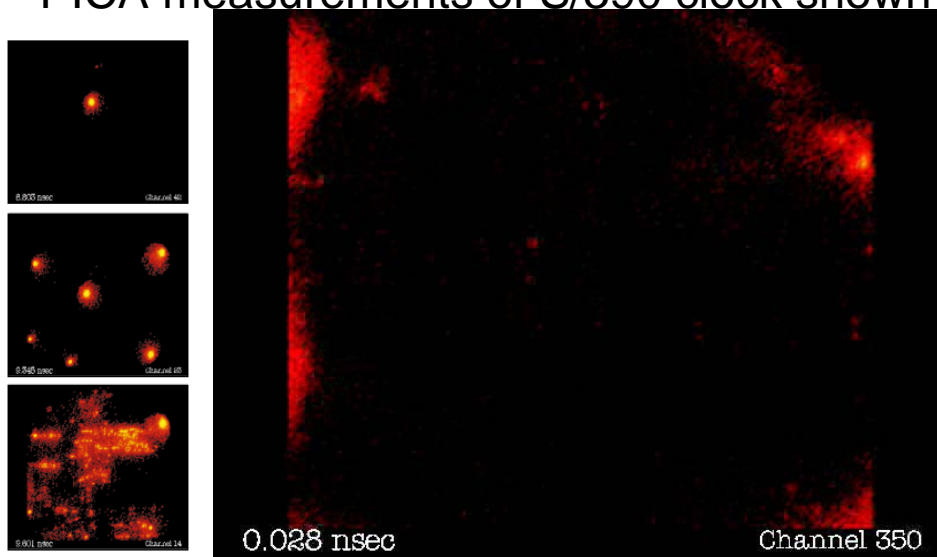
- There's a good reason why skew numbers are usually simulated
 - E-beam and picoprobe don't work with flip-chip packaging
- IBM PICA: Picosecond Imaging Circuit Analysis
 - Saturated nMOS emit photons in the channel
 - Very rarely; photons photomultiplied exponentially to be seen
 - Subsample to "probe" the whole chip at once over a long time
 - Can measure skew by imaging XOR of two different clocks
- Intel LVP: Laser Voltage Probe
 - Shine a laser onto the diffusion (from the back of the substrate)
 - Need a good transparent heat-spreader (diamond)
 - Reflectivity depends on voltage bias.
 - Probes only one point at a time, but rapidly

PICA Movie of Ring



PICA Measurements

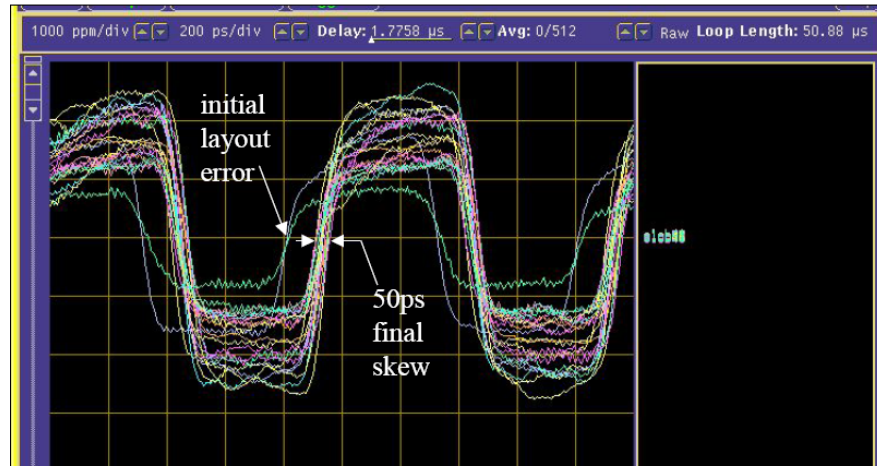
- PICA measurements of S/390 clock shown



Source: Tsang IBM JourResDev 2000

LVP Measurements of Skew

- Voltage levels not well characterized, but transitions are
 - Good for teasing out skew numbers from the edges
 - Shows a layout error that was fixed in a subsequent stepping



Measuring Jitter

- Almost impossible to do, so everybody estimates it
 - You can try to measure total skew + jitter (Restle ISSCC 2005)
- Both PICA and LVP are averaging measurements (no jitter)
- Typical schemes drive a clock node to a pin to expose it
 - But need to buffer up to drive the pin
 - Ramp-up chain introduces jitter itself
- Measurement scopes have source jitter, too
 - Measured jitter: 150pS. Source trigger jitter: 150pS. Hmm...
 - Commercial systems try to cancel trig jitter (M2, Wavecrest)
 - But still end up with on-chip buffer jitter