Active GHz Clock Network Using Distributed PLLs

Vadim Gutnik, *Member, IEEE,* and Anantha P. Chandrakasan, *Member, IEEE*

***Abstract*—A novel clock network composed of multiple synchronized phase-locked loops is analyzed, implemented, and tested. Undesirable large-signal stable (mode-locked) states dictate the transfer characteristic of the phase detectors; a matrix formulation of the linearized system allows direct calculation of system poles for any desired oscillator configuration. A 16-oscillator 1.3-GHz distributed clock network in 0.35-μm CMOS is presented here.**

***Index Terms*—Clock network, multiple oscillator system, phase-locked loop.**

I. Introduction

THE CLOCK distribution network of a modern microprocessor uses a significant fraction of the total chip power and has substantial impact on the overall performance of the system. For example, the 72-W 600-MHz Alpha processor [1] dissipates 16 W in the global clock distribution, and another 23 W in the local clocks: more than half the power goes to driving the clock net. The clock uncertainty budget for a global clock is 10% of a clock period, which translates to a 10% reduction in maximum operating speed; as argued below, this penalty is likely to increase for currently popular clock architectures.

Most conventional microprocessors use a balanced tree to distribute the clock [1]–[3]. Because the delays to all nodes are nominally equal, trees may be expected to have low skew. However, at gigahertz clock speeds a large fraction of skew and jitter comes from random variations in gate and interconnect delay. The majority of jitter in a clock tree is introduced by buffers and inter-line coupling to the clock wires; a relatively small amount comes from noise in the source oscillator [4]. Therefore, a primary consideration in clock design is matching delay along the clock path.

As clock speed increases, signal delay across a chip becomes comparable to a clock cycle. For example, a 2-cm-long wire in a 0.25-μm process has a delay of 0.86 ns, while the clock might be as high as 1 GHz; scaling to 4 GHz, the same wire (with optimal buffering) will have a delay of approximately 0.43 ns, compared to a clock period of 0.25 ns. In all practical cases a signal that takes longer than a clock cycle to propagate would be pipelined, and hence re-clocked. The fundamental weakness of tree distribution (and networks that depend on tree matching) is that skew is only relevant between communicating latches, but the clock path is always the length of the chip. Clock speeds increase with gate delay, and processor architectures can exploit both locality of blocks and pipelining to avoid penalty due to long signal paths, but the error in a global clock scales with the total path delay, and is thus a growing fraction of a clock cycle.

In this paper, we consider the effects of static and dynamic mismatch on a few representative clock networks in Section II and propose a distributed generation scheme that needs only local synchronization to generate a global clock. Large and small-signal stability of the proposed network is analyzed in Section III. This clock was implemented on a test chip; circuit details and results are presented in Sections IV and V.

Manuscript received March 24, 2000; revised June 24, 2000. This paper was supported by the MARCO Focused Research Center on Interconnects, which was funded at the Massachusetts Institute of Technology through a subcontract from the Georgia Institute of Technology, and supported in part by a Graduate Fellowship from the Intel Corporation.

V. Gutnik was with M.I.T. Microsystems Technology Lab, Cambridge, MA 02139 USA. He is now with Silicon Laboratories, Austin, TX 78749 USA (e-mail: gutnik@mit.edu).

A. P. Chandrakasan is with M.I.T. Microsystems Technology Lab, Cambridge, MA 02139 USA (e-mail: anantha@mtl.mit.edu).

Publisher Item Identifier S 0018-9200(00)09441-5.

II. Modeling Random Skew

A. Assumptions

Given sufficiently accurate models, systematic skew can be corrected at design time. Therefore, the primary interest is random zero-mean variations. For the sake of comparing architectures, we make several simplifying assumptions.

1) Delay mismatch, both static and dynamic, is proportional to total delay.
2) Wire *RC* delay is independent of gate delay (d).
3) The clock period proportional to gate delay.
4) Chip size is independent of gate delay.
5) In 0.25-μm technology, signal delay across a die equals one clock period.

Assumption 1 is inaccurate, but convenient. Mismatch due to gradients scales as delay squared; purely random short-distance mismatch scales as the square root of delay. For the sake of analysis, however, we will assume that uncertainty scales linearly. Assumptions 2, 3, and 4 are approximately true, given historical data: as the geometries scale the resistance increase in clock wires is offset by lower capacitance; processor cycle time is generally on the order of 8–16 gate delays; and chip sizes hover around 15×15 mm^2.

Assumption 5 serves to normalize signal delay, chip size, and clock speed. It is not coincidental that random variation has become a noticeable issue at about the time when cross-die signal delay is comparable to one clock cycle: as a heuristic, 10% of a clock cycle is allocated for unmodeled skew and jitter margin, and delay uncertainty is about 5%–10% of delay. Hence, when delay across a chip is comparable to clock cycle time, random delay is a considerable fraction of the total clock error budget.

B. Tree

To keep internal clock skew low, a tree is generally made deep enough that a tile driven by a single leaf is small compared to the size of the chip [5], [6]. In turn, this means that the path from the

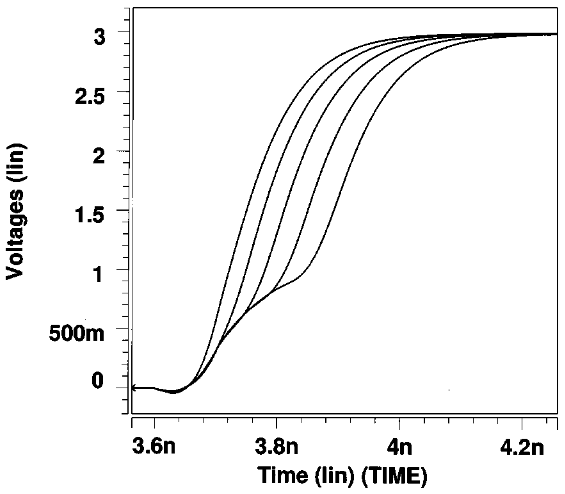


Fig. 1. Simulated edge in a grid with skew to the drivers.

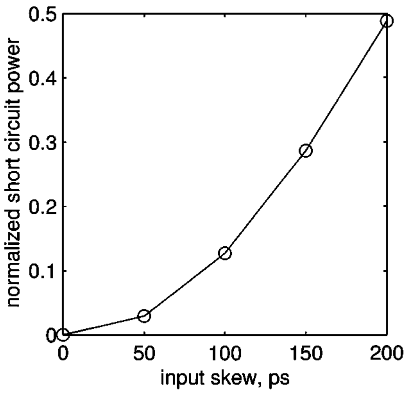


Fig. 2. Short circuit power in a grid vs. input tree skew.

clock source to the load is comparable to the size of the entire die. Because the worst-case skew occurs between two adjacent leaves for which the clock path was completely different, worst case mismatch depends on the entire source-to-leaf delay. And worse, the problem becomes worse with process scaling. Because *RC* delay does not scale, delay along an optimally buffered line scales only as $\sqrt{d}$; hence the skew as a fraction of the clock period grows as $1/\sqrt{d}$ with falling d.

C. Grid

Modern grids are H-tree-grid hybrids: a short H-tree distributes clock to a few (4 or 16, for example) buffers around a chip, and those buffers drive a clock grid in parallel. Shorting the buffers together helps drive down some of the uncertainty at the cost of increased short-circuit power during switching and somewhat slower edge rates. However, rise time scales linearly with d, so by the same reasoning as applied to the tree scaling arguments, skew as a fraction of rise time will increase with $1/\sqrt{d}$ as gate delay falls. When the tree skew exceeds rise time, short circuit power dissipation increases rapidly, and the clock edges begin to show an unacceptable kink. Fig. 1 shows simulated edge shapes with increasing input skew for a grid driven from a 4-level tree with skews from 0 to 200 ps, and Fig. 2 shows the corresponding short-circuit power dissipation, plotted as a fraction of CV^2f-power for the clock grid.

D. Active Feedback

As is evident from the given examples, most of the skew comes from the initial long-distance distribution of a clock to relatively small loads. A delay-locked loop (DLL) could be adapted to measure and cancel out wire variations, as shown in Fig. 3. If the round-trip delay is tuned to an even number of clock cycles, the wire has nominally 0 delay.

Unfortunately, despite the apparent symmetry, the forward and reverse paths do not match well for two reasons. First, "matched" buffers are physically separated. In Fig. 3, b_1 should match b_7, although it would be physically near b_8. b_1 is not as far away from its matched pair as it might be in a tree, but it will still typically be millimeters away. Second, there is no temporal correlation. The clock signal passes w_1 at a different time than it passes w_7, so any time-dependent variations, including those due to power supply and signal coupling, do not match.

Another approach, proposed by Intel, is shown in Fig. 4 [7]. Here, a DLL matches delays to two half-trees; an obvious generalization, with four DLLs matching quarter-trees is shown in Fig. 5. Static delay variations of some nearest neighbors are canceled out by the DLL to within the precision of the matching of the comparators. The drawback is that some neighboring nodes, as A and B in Fig. 5, are only related through multiple DLLs. A much better result can be obtained by using DLLs that take multiple reference inputs, and adjust output phase to be aligned exactly between the two inputs. The network can then be redrawn somewhat more symmetrically, as Fig. 6. (For clarity, the local tree was not drawn, and the connections to the comparators are abstracted.)

Optimization of the number of tiles is straightforward. Internal skew scales with tile area, so as the number of tiles increases, internal skew falls. However, every boundary between tiles introduces some skew because of mismatch in the phase detector (PD). Hence, as the number of tiles increases, the number of boundaries increases. Fig. 7 shows the optimization curves calculated for this clock metric. As in other clock networks, faster clocks require a more finely grained architecture. Jitter in a DLL network will rise in exactly the same way as it increases in clock trees, and for the same reasons. Skew scales linearly with d because it is comprised of comparator mismatches and delays across each leaf-patch. Note, however, that in a phase-locked loop (PLL) the noise can be expected to scale with d; a PLL network like the one in Fig. 6 would have total clock uncertainty that is a constant fraction of the clock period.

III. Stability

We propose a distributed clock network comprised of an array of synchronized PLLs. Independent oscillators generate the clock signal at multiple points ("nodes") across a chip; each oscillator distributes the clock to only to a small section of the chip ("tile") (Fig. 8). PDs at the boundaries between tiles produce error signals that are summed by an amplifier in each tile and used to adjust the frequency of the node oscillator. In general, the network need not be square or regular.

With locally generated clocks, there are no chip-length clock lines to couple in jitter; skew is introduced only by asymmetries in PDs instead of mismatches in physically separated buffers,

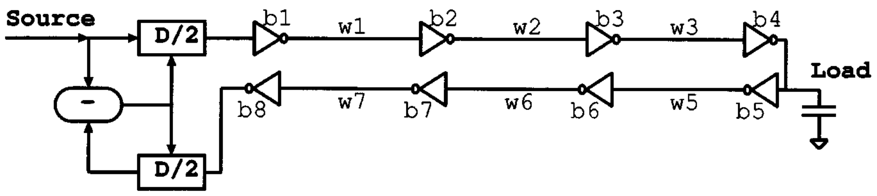


Fig. 3. Low-skew wire with DLL.

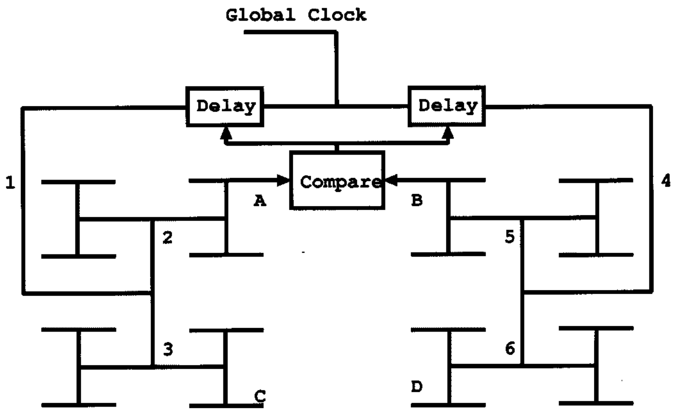


Fig. 4. Matching tree leaves with a DLL.

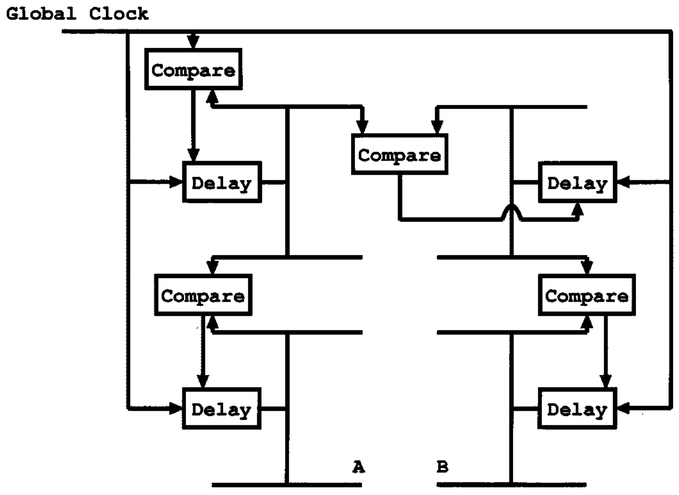


Fig. 5. DLL architecture.

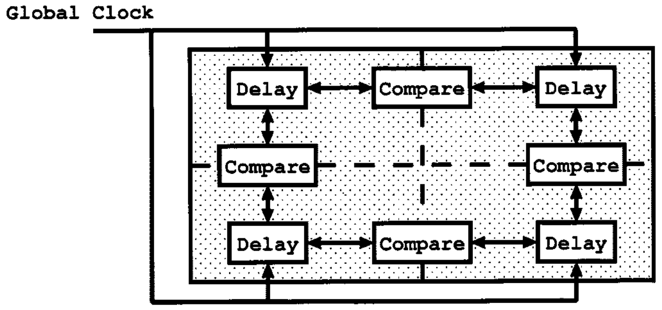


Fig. 6. Multi-input delay cell DLL architecture.

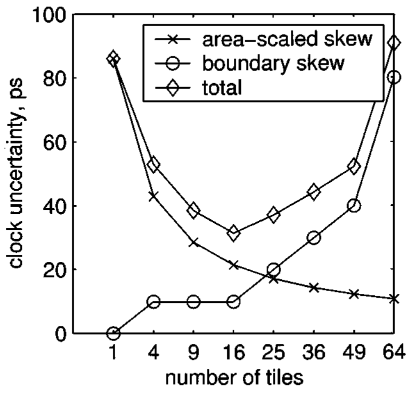


Fig. 7. Tile number optimization.

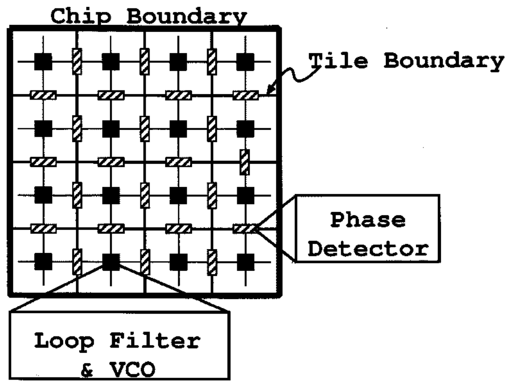


Fig. 8. Distributed clocking network.

and the clock is regenerated at each node, so high-frequency jitter does not accumulate with distance from the clock source. Unlike earlier work on multiple clock domains which suggested the use of multiple independent clocks [8], this approach produces a single fully synchronized clock. The rest of this section examines small and large signal stability of a distributed PLL.

A. Small Signal

In a multiple-oscillator PLL large-signal and small-signal behavior are interrelated. In normal operation, the oscillators are phase-locked, and jitter depends on the network response to noise. Because startup is expected to take a negligibly small fraction of time, the connection of the oscillators is optimized for small-signal behavior rather than to make initial acquisition more efficient. The linearized small-signal behavior, valid when the oscillators are nearly in phase, is analyzed first.

B. General Derivation

The block diagram (Fig. 9) of a multiple-oscillator PLL is essentially identical to the one for a conventional PLL, except that the connections between blocks are vectors instead of individual signals, and the gains and transfer functions are matrices instead of scalars. This means that the PD becomes matrix A_1,

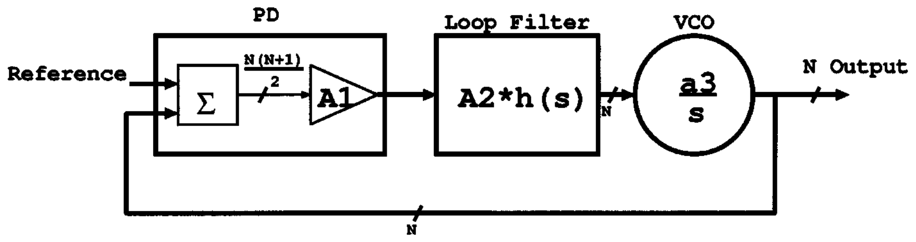


Fig. 9. Linear system model of a multi-oscillator PLL.

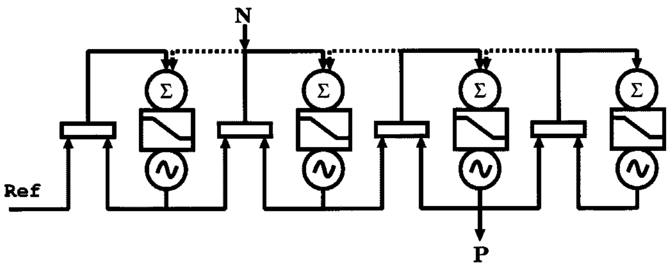


Fig. 10. One-dimensional PLL array; symmetrical with the dotted-line connections.

of size $N(N+1)/2 \times N$, and the loop filter becomes A_2, a corresponding $N \times N(N+1)/2$ matrix. $G = A_2A_1$ is an intuitively meaningful $N \times N$ matrix. The network of oscillators is similar to a lumped circuit C with a node for each oscillator and a branch for each connection between pairs of oscillators. Node voltages in C represent oscillator phase, and branch currents represent the error signals on the output of the PD. G is the conductance matrix for C with unity conductance branches. G for a four-oscillator network is shown in (1). Each off-diagonal entry g_{ij} is -1 if there is a PD between node i and node j; g_{ii} is the number of detectors attached to node i.

$$G = \begin{pmatrix} 3 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}. \tag{1}$$

DC gain in the loop can be lumped into a_3.

Writing the transfer function in matrix form gives

$$\Phi = [sI + a_3A_2A_1h(s)]^{-1}\, h(s)a_3A_2u \tag{2}$$

where u is the phase error input to each phase comparator. $u(1)$ is the reference phase, and $u(2)\cdots u(n)$ are the noise contributions from interconnect and PD mismatch.

C. Examples

Matrix A_1 is determined by the geometry of the tiles, and hence will constrained by the placement of clock loads, which for this problem is fixed. Assuming the simplest possible PLL, $h(s) = (s+z)/s$. This leaves A_2, a_3, and z as design variables.

There are still far too many choices to find the general optimum, but a few examples may help guide the search.

1) One-Dimensional Array: A one-dimensional array of oscillators with PDs between neighbors is the simplest generalization of a single PLL. In a perfectly asymmetrical array (call this system S_1), the output of PLL i is the input to PLL $i+1$, as shown in Fig. 10. S_1 is described by

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \quad A_{2,1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3}$$

This system has multiple poles at the same place where a single-oscillator PLL has single poles.

On the other hand, in a perfectly symmetrical array (call it S_2), the input to each oscillator i is the phase of oscillators $i-1$ and $i+1$ (Fig. 10, with the dotted-line connections). The A_1 matrix is the same because the physical arrangement of nodes is identical, but A_2 changes:

$$A_{2,2} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4}$$

To achieve the same phase margin in S_2 as in S_1, it is necessary to lower the gain a_3. This can be shown with a geometrical argument: in S_2, when the phase of oscillator i changes by $\Delta\phi$, the change is measured at two PDs, so oscillator i feels twice the feedback that it would have felt in S_1, and at the same time, oscillators $i-1$ and $i+1$ both adjust in the opposite direction, giving four times the effective gain. Hence, the gain must be decreased by a factor of approximately four. Mathematically, the largest eigenvalues of $A_{2,1}A_1$ is 1, but the largest eigenvalue of $A_{2,2}A_1$ is 3.5. Poles of the symmetrical system, solved via (2), are plotted in Fig. 12(a). The key difference between S_1 and S_2 is the systems' response to noise. In both cases, noise at frequencies higher than the unity gain frequency ω_0 are attenuated. For frequencies much lower than ω_0, the response can be calculated via (2). Fig. 11 shows a Bode plot of noise at node P in response to a noise source at node N. Noise performance of S_1 is much worse for intermediate frequencies because there is no feedback so errors propagate forever. In S_2, the feedback limits the influence of preceding stages, and this in turn attenuates noise. For this reason, networks with feedback are preferred, despite the more complicated stability calculation.

2) Two-Dimensional Array: A two-dimensional array is analyzed exactly the same as is a one-dimensional array, except that the gain has to decrease by another factor of two because the center oscillators see four neighbors rather than two. A 16-element array in a 4×4 grid is implemented in this thesis. Its poles are shown in Fig. 12(b).

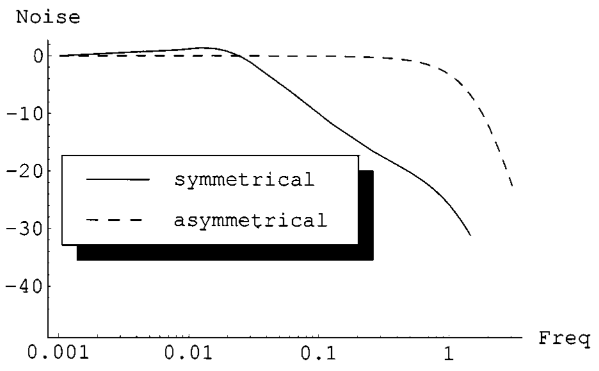


Fig. 11. Comparison of noise responses for symmetrical and asymmetrical networks.

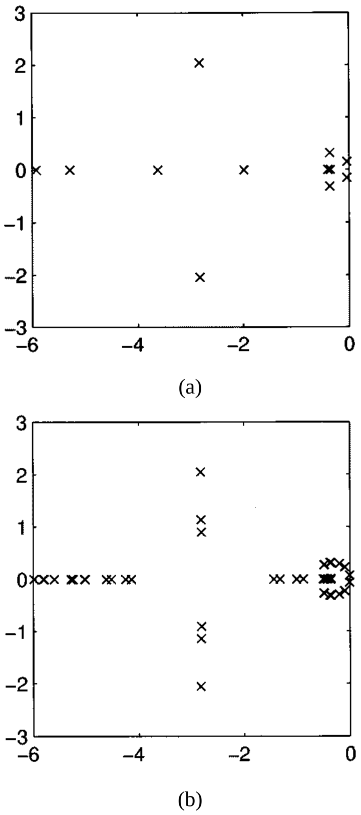


Fig. 12. Root locus for 1-D and 2-D PLL arrays. (a) 1-D array. (b) 2-D array.

D. Large Signal: Mode Locking

The analysis of the previous section indicates that fully connected networks should have a better noise response than asymmetrical networks. However, the feedback allows the possibility of undesirable large-signal modes. Consider the matrices for a 2×2 PLL network:

$$A_1 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

$$A_2 = A_1^T = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix}. \quad (5)$$

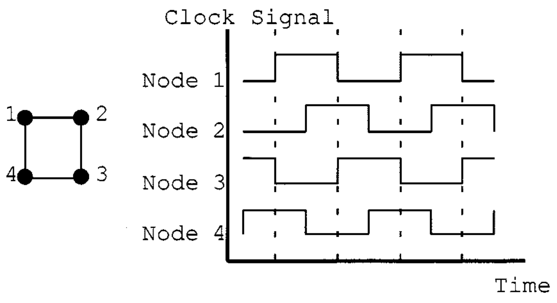


Fig. 13. Mode-locking example.

Because phase is periodic with period 2π, the phase measured at the PDs $\Delta\phi = A_1\phi \bmod 2\pi$. For small ϕ, $(A_1\phi \bmod 2\pi) = A_1\phi$, so the nonlinearity is irrelevant. However, with $\phi = \phi_x = [0, \pi/2, -\pi/2, \pi]^T$

$$A_2(A_1\phi_x \bmod 2\pi) = A_2[0, -\pi/2, \pi/2, -\pi/2, \pi/2]^T = 0 \quad (6)$$

so ϕ_x is a stationary point. This is intuitively easy to see, in reference to Fig. 13: each oscillator leads one neighbor, and lags behind another neighbor by exactly the same amount. The net phase error is zero, so clearly there is no restoring force to drive the phases to 0. Because the nonlinearity does not change for small deviations from ϕ_x, dynamics about ϕ_x are the same as those about 0 and hence this state is stable. The locking of a distributed oscillator to nonzero relative phases has been called *mode-locking* [9]. At startup, each oscillator in a distributed PLL starts at a random phase, so there is a nonzero chance of converging to a mode-locked state. Simulations show that for a network like the one shown here, the system ends mode-locked from $\approx 1/3$ of random initial states. The probability goes up rapidly with the size of the system; a 4×4 array ends up mode-locked well over 99% of the time.

Pratt and Nguyen proved several useful properties about systems in mode-lock [9]. The key result, generalized for non-Cartesian networks, is that *for a system in mode-lock, there must be a phase difference θ between two oscillators such that $\theta \geq 2\pi/n$* where n is the number of nodes in the largest minimal loop in the network and a *minimal loop* is a loop in the graph that cannot be decomposed into multiple loops

This result suggests a way to distinguish between mode-locked states and the desired 0-phase state: in mode-lock, there must be at least one branch with a large phase error. If the gain of the PD is designed to be negative for a phase difference larger than θ, then all mode-locked states are made unstable without affecting the in-phase equilibrium. Pratt and Nguyen suggest that XOR PDs preclude mode-lock in a rectangular network of oscillators because the response decreases for phase errors larger than $\pi/2$, [9]. This result follows directly from the result derived above: in a rectangular array, the largest minimal loop has four nodes, so $\theta = 2\pi/4 = \pi/2$. A PD described in the next section, with $\theta < \pi/2$, would be useful in nonrectangular networks, and where more gain near 0 phase is desirable.

IV. Implementation

The distributed clock network generates the clock signal with PLLs at multiple points ("nodes") across a chip, and distributes

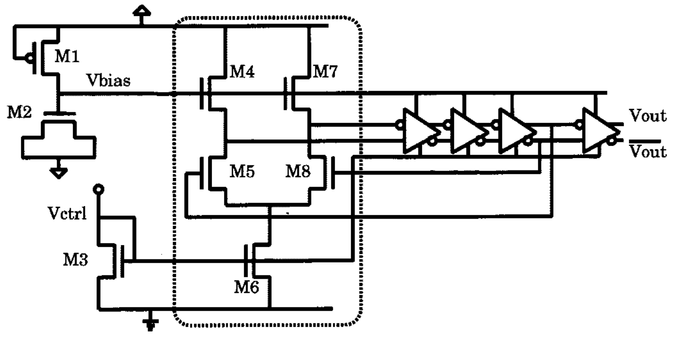


Fig. 14. Ring oscillator schematic

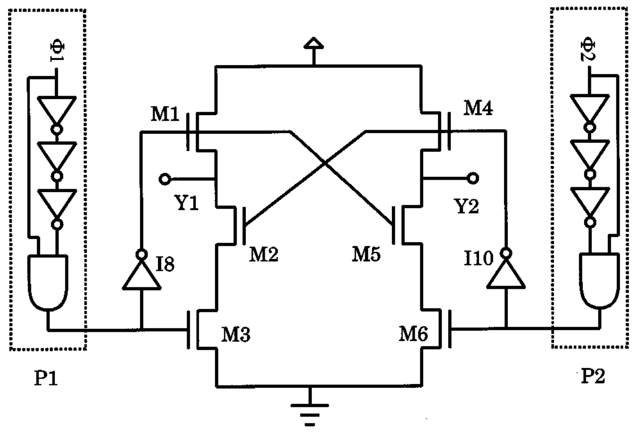


Fig. 15. Phase detector (PD).

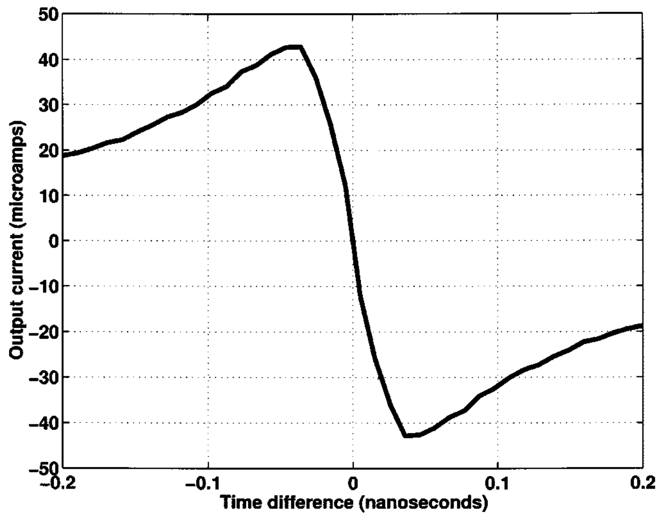


Fig. 16. Simulated phase transfer curve

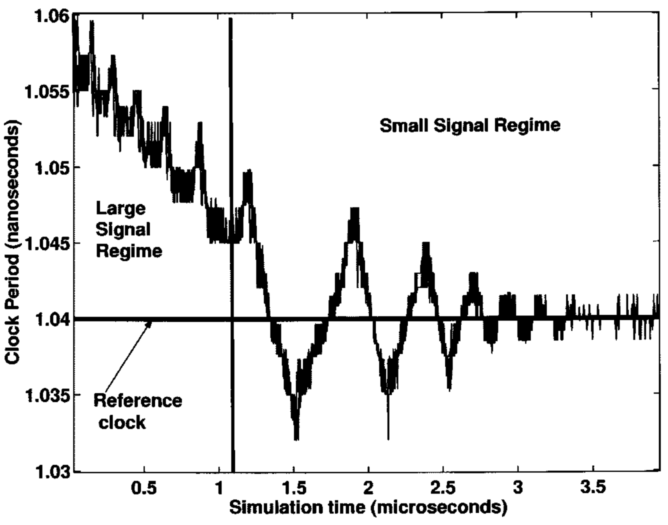


Fig. 17. Locking behavior of the PLL array

each only to a small section of the chip ("tile") (Fig. 8). PDs at the boundaries between tiles produce error signals that are summed by an amplifier in each tile and used to adjust the frequency of the node oscillator

Because the proposed network has many nodes, the power and size constraints on each node are even more stringent than the constraints on a single global PLL. The oscillator, PD, and loop filter of a working demonstration chip, fabricated in a standard 0.35-μm single-poly triple metal process, are considered in turn below.

A. Oscillator

The demonstration chip used an nMOS-loaded differential ring oscillator as a voltage-controlled oscillator (VCO) (Fig. 14). Transistors $M_4 - M_8$ comprise the differential inverter. The differential pair is $M_{5,8}$, the tail current is driven by M_6, and $M_{4,7}$ act as the nMOS load. The nMOS loads allow fast oscillation and shield the output signal from V_{DD} noise. V_{bias} is a low-pass version of V_{DD} generated by subthreshold leakage through PFET M_1; supply noise coupling in through C_{gd} of $M_{4,7}$ is bypassed by M_2. The oscillation frequency is only dependent on the supply voltage through capacitor nonlinearity and the output conductance of $M_{4,7}$, and feedback of the PLL compensates drift of V_{DD} and V_{bias}.

B. Phase Detector (PD)

The PD, shown in Fig. 15, has a sufficient nonlinearity, higher gain at small input phase difference and less high-frequency content than an XOR PD. The core ($M_1 - M_6$) is an nMOS-loaded arbiter which acts as a nonlinear PD. For no input phase difference, the output is balanced. As the phase difference increases from zero, one output will be asserted for the full duration of an input pulse, while the other output will be asserted for only the remainder of the input pulse duration after the first input pulse ends, which is equal to the input phase difference. Thus the detector has very high gain near zero phase error that drops off to zero as the input phase difference approaches the input pulse width (Fig. 16).

The pulse generators P_1 and P_2 enable this arbiter to give frequency-error feedback. If one input is at a higher frequency than the other, its output will be asserted for more input pulses than the other. Because the width of the pulses is independent of input frequency, the average output voltage corresponds to frequency. Unlike a typical phase-frequency detector, however, the strength of the error signal falls to zero as frequency difference goes to 0, so there can be no mode-lock problems, yet large signal frequency (and hence, phase) locking is enhanced. Fig. 17 shows the large signal correction and small signal behavior of the entire array of PLLs as the already internally locked array

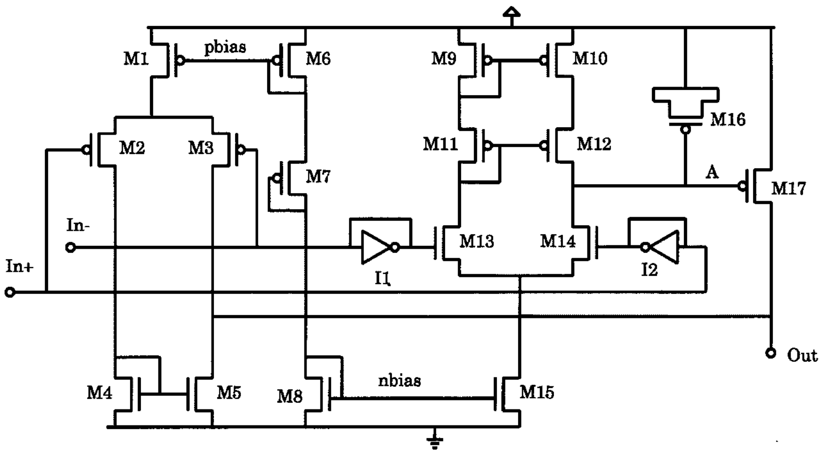
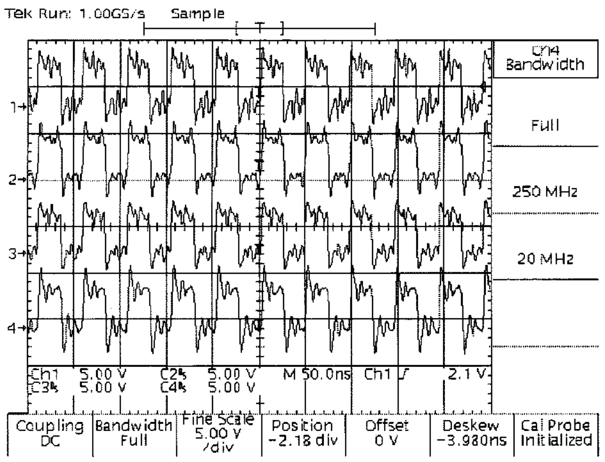


Fig. 18. Loop filter schematic.



Ch4 Bandwidth
Full
250 MHz
20 MHz

Ch1 5.00 V C2 5.00 V M 50.0ns Ch1 ∫ 2.1 V
C3 5.00 V C4 5.00 V

Coupling DC | Bandwidth Full | Fine Scale 5.00 V /div | Position -2.18 div | Offset 0 V | Deskew -3.980ns | Cal Probe Initialized

Fig. 19. Frequency-locked divider outputs.

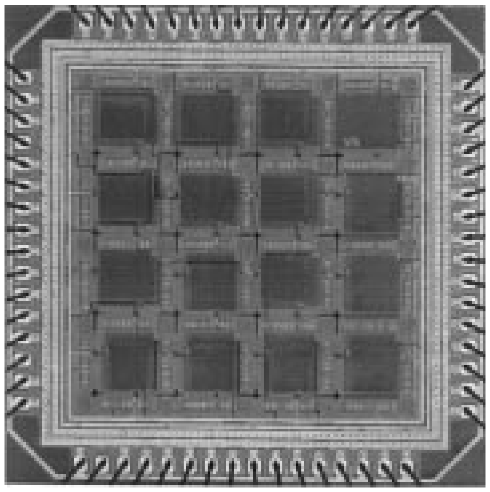


Fig. 20. Micrograph of the 16-oscillator 1.3-GHz chip.

approaches and locks to the reference clock. The detector fits in $30\ \mu\text{m} \times 30\ \mu\text{m}$.

C. Loop Filter

The loop filter is shown in Fig. 18. $M_1 - M_5$ make up amplifier A_1, while $M_9 - M_{17}$ make up A_2. The differential output currents from the PDs at the edges of each tile are summed at nodes $In+$ and $In-$, and drive both amplifiers. A_1 is a single stage differential pair so it has relatively low gain but a bandwidth limited by g_m/C_{gs}. A_2 has a high-gain cascoded stage driving a common source PFET M_{17}. M_{16} is a large gate capacitor which serves to set the dominant pole of A_2 such that the PLL network is stable. M_{15} is biased at very low current to boost gain and enable a low time constant (as low as 12 kHz) with a $15\ \mu\text{m} \times 15\ \mu\text{m}$ gate capacitor. The simple design and feed-forward compensation allow the loop filter to fit in only $15\ \mu\text{m} \times 45\ \mu\text{m}$. Each clock node, consisting of an oscillator and a loop filter, takes just $45\ \mu\text{m} \times 45\ \mu\text{m}$.

V. Results

A chip was fabricated with a 4×4 array of nodes and PD between nearest neighbors. Counting one node and two PDs the area overhead is approximately 0.0038 mm^2 per tile. Another PD was placed between one of the nodes and the chip clock input to lock the network to an external reference. The output of the 16 oscillators was divided by 64 and driven off chip. At $V_{DD} = 3$ V, the divided outputs were seen to be frequency locked at 17 to 21 MHz, corresponding to oscillator phase lock at 1.1 to 1.3 GHz. An oscilloscope plot of four locked output signals is shown in Fig. 19. Long-term jitter between neighbors is less than 30 ps. Cycle-to-cycle jitter is less than 10ps. The oscillators, amplifiers and all the biasing draws 130 mA at 3 V. A chip plot is shown in Fig. 20. (The rest of the area on the 3mm $\times$ 3mm chip is taken up by test circuits.)

VI. Conclusion

Design and measurements on this chip confirm that generating and synchronizing multiple clocks on chip is feasible. Neither the power nor the area overhead of multiple PLLs is substantial compared to the cost of distributing the clock by conventional means. Most importantly, a distributed clock network can take advantage of improved devices by shrinking the size of the cells, lowering the overall skew and jitter, so performance will scale with the speed of devices, rather than with the much slower improvement of on-chip interconnect speed.

References

[1] D. W. Bailey and B. J. Benschneider, "Clocking design and analysis for a 600-MHz Alpha microprocessor," *J. Solid State Circuits*, vol. 33, no. 11, pp. 1627–1633, Nov. 1998.

[2] C. F. Webb, "A 400-MHz S/390 microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 1997, pp. 168–169.

[3] T. Yoshida, "A 2-V 250-MHz multimedia processor," in *ISSCC Dig. Tech. Papers*, Feb. 1997, pp. 266–267.

[4] I. A. Young, M. F. Mar, and B. Bhushan, "A 0.35-μm CMOS 3-880-MHz PLL N/2 clock multiplier and distribution network with low jitter for microprocessors," in *ISSCC Dig. Tech. Papers*, Feb. 1997, pp. 330–331.

[5] H. B. Bakoglu, J. T. Walker, and J. D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ULSI and WSI circuits," in *IEEE Int. Conf. Computer Design*, NY, Oct. 1986, pp. 118–122.

[6] P. Zarkesh-Ha, T. Mule, and J. D. Meindl, "Characterization and modeling of clock skew with process variations," in *Proc. IEEE 1999 Custom Integrated Circuits Conf.*, pp. 441–444.

[7] G. Geannopoulos and X. Dai, "An adaptive digital deskewing circuit for clock distribution networks," in *ISSCC Dig. Tech. Papers*, Feb. 1998, pp. 400–401.

[8] F. Ançeau, "A synchronous approach for clocking VLSI systems," *J. Solid State Circuits*, vol. SC-17, no. 1, pp. 51–56, Feb. 1982.

[9] G. A. Pratt and J. Nguyen, "Distributed synchronous clocking," *IEEE Trans. Parallel and Distributed Systems*, Mar. 1995.

Vadim Gutnik (M'00) received the B.S. degree in electrical engineering and materials science from the University of California, Berkeley, in 1994, and the S.M. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1996 and 2000, respectively.

Previous research interests have included micromechanical resonators, and variable-voltage power supplies. He is currently working as a Design Engineer at Silicon Laboratories, Austin, TX.

Dr. Gutnik received an NDSEG fellowship in 1994, and the Intel Foundation Fellowship in 1997.

Anantha P. Chandrakasan (M'95) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since September, 1994, he has been the Analog Devices Career Development Assistant Professor of electrical engineering at the Massachusetts Institute of Technology, Cambridge. His research interests include the ultra-low-power implementation of custom and programmable digital signal processors, wireless sensors and multimedia devices, emerging technologies, and CAD tools for VLSI. He is a co-author of the book titled *Low Power Digital CMOS Design* (Norwood, MA: Kluwer, 1995). He has served on the technical program committee of various conferences including ISSCC, VLSI Circuits Symposium, DAC, ISLPED, and ICCD. He is the Technical Program Co-Chair for the 1997 International Symposium on Low-Power Electronics and Design and for VLSI Design'98.

He received the National Science Foundation Career Development Award in 1995, the IBM Faculty Development Award in 1995, and the National Semiconductor Faculty Development Award in 1996. He received the IEEE Communications Society 1993 Best Tutorial Paper Award for the *IEEE Communications Magazine* paper titled, "A Portable Multimedia Terminal."