Adaptive Clocking Techniques for SoC Supply Droop Response in Predictive 7nm CMOS

Dan Fritchman, *Member, IEEE* and Wahid Rahman, *Member, IEEE*

***Abstract*—Adaptive clock generation techniques have emerged in recent generations of high-performance SoCs to mitigate timing failure due to supply voltage droops. Resilient techniques vary from analog voltage mixing to digital sensing and clock actuation. This work identifies a taxonomy for adaptive clock generation systems: adaptive clock distribution (ACD) and adaptive PLL-based schemes. Reported realizations in state-of-the-art processor SoCs are reviewed and key performance metrics for comparisons are identified. A PLL-based design is evaluated in a predictive 7nm CMOS technology and compared with prior works.**

***Index Terms*—Adaptive clocking, adaptive frequency, power efficiency, supply-droop mitigation, supply-voltage droop.**

I. Introduction

Power management techniques in modern system-on-chips (SoCs) are critical for energy-efficient processors ranging from data servers to mobile devices. SoC thermal dissipation constraints and energy-saving modes necessitate system-level power management to decrease the power supply or reduce the number of active processing cores. Such techniques exhibit a decrease (i.e. droop) in the SoC supply voltage (V_{DD}) due to: the controlled decrease of V_{DD} from a supply regulator or DC-DC converter; and the transient $L\frac{di}{dt}$ supply voltage ripple due to sudden current changes through package inductances when dynamically enabling or disabling on-chip processing cores. To maximize processing throughput, SoCs are designed to operate close to the maximum possible clock frequency (f_{MAX}) for the target (V_{DD}) with minimal guardbanding. Decreasing V_{DD} reduces the drain currents of CMOS transistors, thereby increasing propagation delays in the critical paths of digital logic. If sufficient timing margin is not available in the design, these critical paths can fail to meet timing and cause unrecoverable errors.

To mitigate such failures, adaptive clock generation techniques have emerged in recent generations of high-performance SoCs [1]–[5]. Adaptive-clock systems detect transient supply events and dynamically adjust clock frequencies to not exceed a changing f_{MAX} as V_{DD} decreases. Transient response of adaptive clock techniques is vital. The external package routes supplying power to the SoC impact supply voltage settling during changes to on-chip core utilization [1], as illustrated in Fig. 1. The range of inductances & capacitances in this network induces a medley of fast and slow time constants. Resilient adaptive clock generation circuits react quickly to fast ripples in V_{DD} ranging from 50-100MHz [1], [3] while also tracking slower transient settling at 1MHz and below [1], [3], [5].

Several schemes exist to realize supply-induced clock adaptation ranging from directly controlling the clock-synthesizing phase-locked loop (PLL) [1], [2] to modulating a tunable delay line or phase rotator downstream in an adaptive clock distribution (ACD) network [3]–[6]. In this work, we survey and contrast the adaptive PLL- and ACD-based mechanisms presented in [1] and [3], respectively. This brief

Date of publication May 3, 2020.

D. Fritchman and W. Rahman are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94720 USA (e-mail: dan_fritchman@berkeley.edu; wahid.rahman@berkeley.edu).

This work was supported by Professor Borivoje Nikolić.

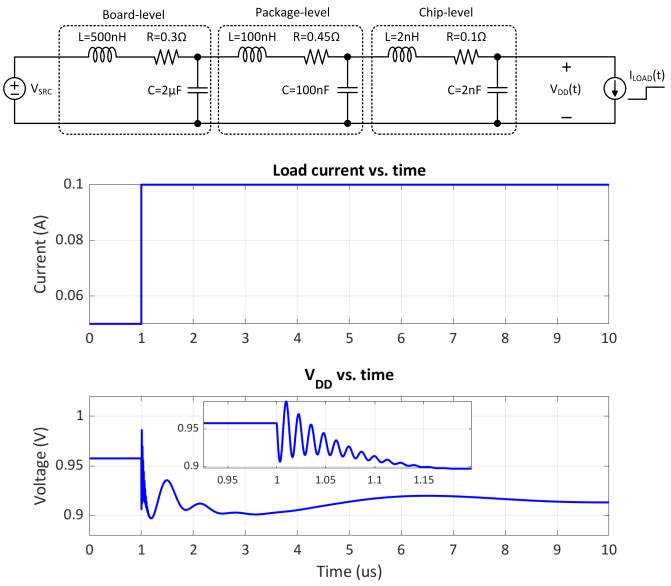

Fig. 1. Supply droop transients due to package inductances and load current (adapted from [1]).

is organized as follows: Section II provides an overview of adaptive clocking schemes and defines the scope of this work; Section III discusses the operating principles of mechanisms presented in [1] and [3]; Section IV provides a comparison of [1] and [3]; and finally Section VII discusses planned work and concludes this brief.

II. Adaptive Clocking Schemes

Adaptive clocking systems include two fundamental components:

(a) a *power-supply sensor*, which measures and reports transient droops in supply voltage; and

(b) a *clock period actuator*, which modulates the system clock period in response to reports from the power-supply sensor.

During a supply drop, low-latency sensing and actuation is crucial to detect and adjust the clock frequency quickly to ensure error-free operation continues throughout the event. If and when V_{DD} recovers from this event to its nominal value, additional circuitry can assist with managing the corresponding f_{CLK} recovery.

ACD- and adaptive PLL-based systems differ in their implementation of the clock period actuator. ACD-based systems decrease the clock frequency by extending clock periods using an ACD circuit placed between the clock-synthesizing PLL and the global clock distribution network as illustrated in Fig. 2. A digital code $CTRL_{FCLK}$ from the power supply sensor controls the ACD circuit such that it extends the period of the synthesized clock, CLK_{PLL}, to the desired frequency given the sensed change in V_{DD}. This lower-frequency clock, CLK_{SOC}, is then distributed through the global clock network to the SoC processing cores. Examples of this ACD circuit include tunable-length delay [5] and phase rotators [3], of which the latter is discussed in Section III-B.

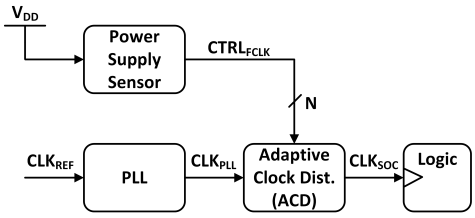

Fig. 2. Adaptive clock distribution (ACD) based system.

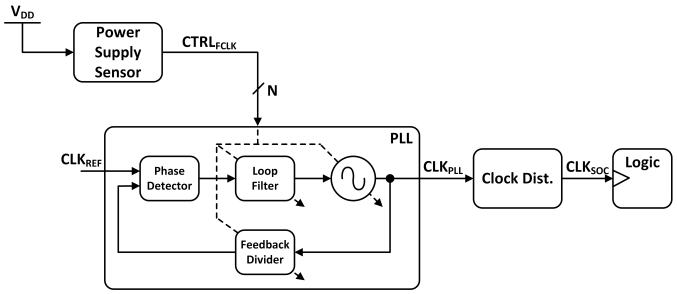

Fig. 3. Adaptive PLL-based system with possible intra-PLL control variants.

Adaptive PLL-based systems, in contrast, incorporate information from the power-supply sensor to dynamically change the behaviour of PLL sub-components. Typically, these schemes affect oscillator control on top of the traditional PLL loop. In [2] for example, the oscillator is controlled indirectly through a loop filter modified for droop response: when a droop is detected, the traditional proportional- and integral-loop filter switches to proportional-only to form a type-I PLL that tunes the oscillator to frequency lock without hazardous over- or underclocking. In [1], this idea is further extended by interrupting the PLL feedback loop to modify the oscillator directly, reacting to a supply droop event with relatively low latency. The feedback divider is then slowly modified to restore PLL lock. This latter work is further discussed in Section III-A.

The focus of this brief is to compare the design and effectiveness of clock period actuators in ACD and PLL-based schemes. While high-resolution power-supply sensors and their low-latency integration are critical to the overall adaptation time of such schemes, it remains beyond the scope of this work.

III. Operating principles

The two main works for comparison are the PLL- and ACD-based adaptive clocking schemes in [1] and [3] respectively. The operating principles of each are described in this section.

A. PLL-based adaptive clocking

The scheme reported in [1] presents a PLL-based adaptive clocking control for three SPARC processor cores in a 20nm CMOS testchip. A sense point placed near one of the SPARC cores transmits the core V_{DD} through low-impedance on-package (off-chip) routing to the on-chip power-supply sensor. This sensor converts V_{DD} droops into a thermometer-coded quantized digital signal, $q[n]$, by use two calibrated delay lines and a 8-bit time-to-digital converter (TDC).

A major contribution of [1] is to use the instantaneous value of q[n] to immediately reduce the PLL's oscillator frequency to correct for high-frequency droops and use a filtered version of $q[n]$ to correct for low frequency droops and ultimately re-lock the PLL feedback loop. As shown in Fig. 4, $q[n]$ is processed by the frequency control logic to generate two control signals: ΔF_{code} that can instantaneously respond to changes in $q[n]$ to directly change the digitally-controlled oscillator (DCO) frequency, and N_{code}^{sync} that relies only on a filtered value of $q[n]$ to track low-frequency droops and re-lock the PLL loop at the new frequency by changing the feedback division value. The minimum function and the non-linear filter together effectively construct an all-pass filter when V_{DD} decreases and a low-pass filter when V_{DD} increases. They allow the loop to actuate the clock with minimal latency during a V_{DD} droop event. During droop recovery, they low-pass filter V_{DD} transients and overshoots as the voltage ultimately settles to its nominal value.

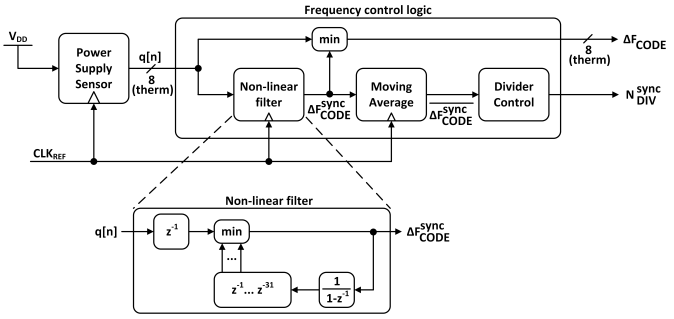

Fig. 4. Frequency control logic for PLL-based adaptive clocking in [1].

A fast response to high-frequency supply droops is an important criterion in [1], significant design efforts were made to ensure low-latency clock period actuation. The reported design target is 8× faster than the first-droop frequency of 50MHz. This permits a time window of 2.5ns from the time V_{DD} crosses the first supply sensor threshold to the time the SoC clock frequency is corrected. This narrow timing constraint is met by generating and propagating the critical oscillator control signal ΔF_{code} asynchronously. Crucially, the 8-bit thermometer-coded ΔF_{code} controls a DCO that can tolerate asynchronous changes in its code without glitching its output clock. The DCO reported in [1] is a bank of nine ring inverters with eight of the rings independently enabled by each thermometer bit of ΔF_{code}. Such a structure can tolerate asynchronous arrival of ΔF_{code} when suddenly decreasing frequency, as the case of a droop. Note that frequency increases are restricted: the combination of non-linear filter memory with the masking of minimum function ensures ΔF_{code} only increases one bit at a time after droop recovery [1]. This obviates synchronization of ΔF_{code}, reducing adaptation latency.

The reduction in adaptation latency directly correlates with reported experimental results. Fast response to droops allows for tighter guardbanding and therefore higher f_{MAX}. While [1] does not report the exact clock adaptation latency, the work improves f_{MAX} by 7.5%, from 4.65GHz without adaptive clocking to 5.00GHz with the proposed scheme. This is the largest frequency gain reported in compared works.

B. ACD-based adaptive clocking

The scheme reported in [3] presents a ACD-based adaptive clocking control for AMD's Steamroller CPU in 28nm CMOS. The power supply sensor consists of a DLL-based droop detector acting as a delay line connected to the SoC core V_{DD}: as V_{DD} changes, the DLL phases change with respect to a regulated PLL's output clock. A programmable threshold selects one of four DLL phases to compare against the reference PLL phase to signal a 1-bit droop activity.

Fig. 5 illustrates the reported scheme. A DLL-based phase rotator realizes the ACD circuit. When a droop activity is detected and $DroopDetected$ is asserted, phase rotation between the 40 DLL phases occurs to effectively "stretch" the clock period and reduce the clock frequency. To ensure glitchless phase rotation, small positive pre-programmed phase increments occur at each step, and the phase

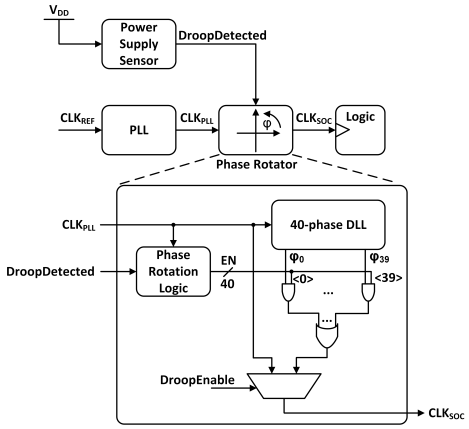

Fig. 5. Frequency control logic for ACD-based adaptive clocking in [3].

rotator rotates through all 40 phases before continuing the cycle. Rotation can be bypassed entirely by de-asserting $DroopEnable$.

This scheme reports to target fast initial droops of 100MHz and slower 1-5MHz secondary droop transients by means of the programmed droop detector threshold [3]. As the detector signal is binary, the ACD-based reduces the SoC clock frequency by a fixed amount (phase rotator steps through a fixed phase step per cycle) or does not change the frequency at all. Voltage recovery is triggered when V_{DD} crosses back beyond the detector threshold. The work reports a total adaptation latency of 3 cycles at 3.4GHz from droop detection to clock frequency reduction. While experimental results do not report frequency gain as in [1], silicon results demonstrated a reduction of 3-6% in V_{MIN}, the minimum supply voltage tolerable at a given frequency for the SoC. No comparison was provided with prior works.

IV. Comparison of ACD and PLL Actuators

A. Comparison Scope

As discussed in Section II, clock adaptation consist of various sub-components and their latencies. This is illustrated in Fig. 6. Design of a complete adaptation system requires a V_{DD} sense point, routing delay, detector and actuation latencies, and finally clock propagation delay. This comparison focuses on the detection logic, clock actuation, and their combined actuation latency. There are indeed important specifications beyond this when designing adaptation circuits. For example, sensor resolution, bandwidth, and ease of design integration with digital SoCs are important considerations: highly digital sensors [1], [3] to custom analog voltage mixing [7] have their bandwidth and integration tradeoffs. However, the scope of this comparison focuses on PLL- and ACD-based adaptive clocking presented in [1] and [3] respectively, agnostic to extraneous design considerations.

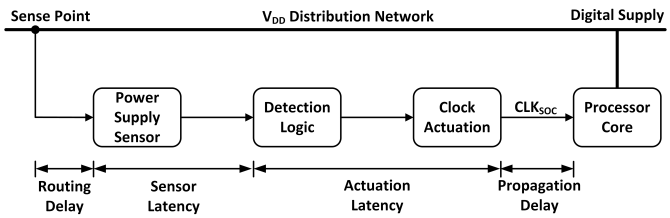

Fig. 6. Various latencies in adaptive clock system (adapted from [1]).

In comparing the PLL- and ACD-based adaptive clocking schemes, key metrics are identified: actuation latency, power/area, calibration, and frequency gain/V_{MIN} reduction. Forecasts for these metrics in the predictive 7nm technology ASAP7 [8] are also discussed.

B. Actuation Latency

Minimizing actuation latency helps reduce overall system response time. In [1], the PLL-based actuation latency varies depending on the direction of V_{DD} change due to the non-linear adaptation filter. As previously discussed, for decreases in V_{DD}, the latency is less than one clock cycle and limited by asynchronous propagation delay through the minimum function in Fig. 3 and the control bandwidth of the oscillator. The control port time constant is approximately $\frac{T_{VCO}}{2N}$ for the reported oscillator [1]. Assuming the routing parasitic from the minimum logic to the oscillator control port is negligible, this allows a latency below one clock cycle at the reported f_{MAX} = 5GHz, i.e. $t_{act} < 200$ps.

Conversely, the ACD-based actuation latency is reported as 2 clock cycles at 3.4 GHz [3], implying $t_{act} = 588$ps. This is due to synchronizing $DroopDetected$ to the PLL clock domain and generating the corresponding phase rotation enable signal as was shown in Fig. 5.

We predict these results will be comparable in ASAP7. The actuation latency in [3] will remain 2 clock cycles by design. That in [1] will remain less than one cycle again by design, with the potential for t_{act} to be reduced due to increased f_T. One drawback may be higher resistance routing which could decrease oscillator control bandwidth if not carefully designed.

C. Power/Area

While neither works report measured power and area overhead of the actuation or overall adaptation circuits, the use of a DLL in [3] is estimated to cost more. Consider phase mismatch and jitter amplification in the DLL [3]. The 40-phase DLL consists of 20 pseudo-differential delay cells, each designed as a pair of cross-coupled inverters with tunable MOS capacitor (MOSCAP) loads. In theory, the k^{th} and $(k+20)^{th}$ phases have a standard deviation of $\sqrt{k}\sigma_u$ where σ_u is one stage's delay variation. The DLL must, at a minimum, ensure phase monotonicity to guarantee glitchless phase rotations. σ_u is directly correlates to V_t and thereby area.

D. Clock Tree Insertion Delay

Modern SoCs incur several clock cycles of delay in clock distribution. When a supply droop occurs, it can affect both the data-path logic and clock buffers. PLL and ACD-based actuators mitigate these delays differently. Adaptive PLLs correct for droops at the clock source. Delay and supply dependence in clock distribution is typically beyond scope. ACD circuits, in contrast, can be distributed deeper into clock distribution network. Compact implementations such as [6] can be inserted many times per SoC clock domain, each cleaning up a small sub-block's supply noise. Such distributed sensors may also benefit from the phenomenon of *clock-data compensation* (CDC), originally reported in [9]. This can relax the required actuation range. However, these distributed actuators typically rely on the PLL clock frequency as time-domain sensors [6]; any adaptation to PLL or ACD-based clocking schemes changes this frequency and thereby sensor accuracy.

E. Silicon Performance

The key performance metric reported in both adaptation schemes is reducing voltage/frequency margin. As [1] claims, this is directly proportional to adaptation latency: the shorter the adaptation response time, the closer the SoC can operate to safety limits. Both systems achieve similar results. In terms of V_{MIN}, the PLL-based scheme achieves 5% reduction at 4.65GHz [1] while the ACD-based scheme reports up to 6% reduction [3] at 4GHz. The PLL-based scheme also reports a 7.5% gain in f_{MAX} to 5GHz if V_{DD} is unchanged [1]. These results require integration of the adaptation system with the complete entire SoC, which was not feasible in ASAP7 for this project scope: adaptation latency was used as a proxy.

V. Design of PLL-based Adaptive Clocking in ASAP7

The PLL presented in [1] was selected as a candidate for further feasibility study in FinFET technologies due to the reduced actuation latency and the lack of immediate power, area, and calibration overhead for similar reduction in V_{MIN} reported in planar technologies [1], [3]. A digital PLL with synthesized loop filters and dividers and a custom DCO were implemented in ASAP7 predictive 7nm technology [8]. The following sections describe its design at the system-, logic-, and transistor-level implementations.

A. System-level Design

The top-level PLL architecture is shown in Fig. 7. A bang-bang phase detector in the PLL digitizes the phase error between CLK_{REF} and CLK_{FB}. A proportional-integral loop filter then tunes the DCO through a 9-bit 1st-order $\Delta\Sigma$ converter, which reduces the DCO quantization noise within the bandwidth of the PLL. A 5-bit 2nd-order $\Delta\Sigma$ single-modulus divider then divides the DCO output clock, CLK_{PLL}, to generate CLK_{FB} and closing the loop. The frequency control logic, as discussed previously, tunes the DCO direct control port ΔF_{CODE} and the feedback division ratio N_{DIV}^{sync}. For this design, the CLK_{DCO} was chosen to have an f_{MAX} of 5GHz to match the reported f_{MAX} in [1]. This PLL was designed with a CLK_{REF} frequency of 500MHz and a nominal division ratio N_{DIV}^{sync} of 10. The DCO gain of 60MHz/LSB was used, correlating with the transistor-level design detailed in Section V-C. Combined with an inverse bilinear transform of the loop filter with $K_P = 1$ and $K_I = 2^{-9}$ clocked at f_{REF} of 500MHz yields a phase margin of 79° and a bandwidth of 810kHz. A moving average of length 32 in the frequency rebound controller, similar to [1], set its bandwidth to roughly $10\times$ that of the PLL.

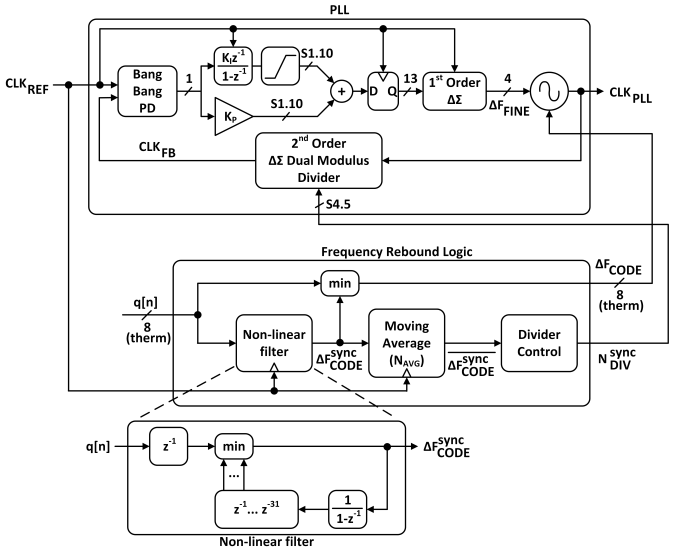

Fig. 7. Top-level implementation of PLL-based scheme from [1].

The response to a droop event simulated in MATLAB is shown in Fig. 8. The critical latencies from transistor-level simulations (Section V-C) are assumed. A short, maximum droop impulse is injected by q[n]. The frequency rebound controller then tunes ΔF_{CODE} that, after an actuation latency, directly changes the DCO frequency. The controller also slowly tunes the divider ratio to re-lock the PLL at the new DCO frequency and ramp the DCO back up to its nominal value. The PLL thereby recovers from the droop impulse after 540ns.

B. Logic Design

Save for the DCO and phase detector, the PLL was designed in IEEE1800-standard SystemVerilog. A parameterized phase-detector

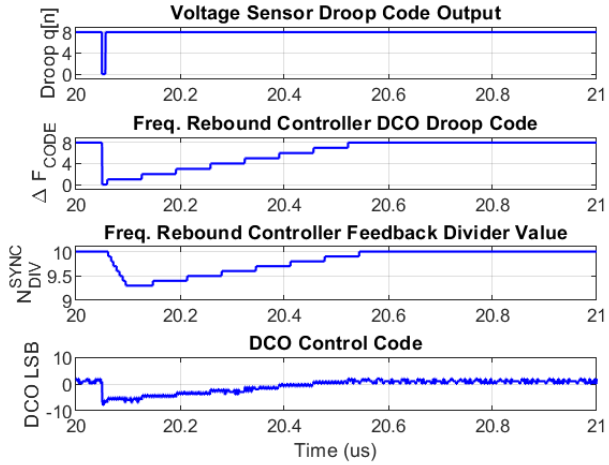

Fig. 8. PLL and frequency control transient response to maximum droop.

interface allows integration of the loop filter with either multi-bit TDCs or single-bit bang-bang PDs. The loop filter gains were also configurable.

As discussed in [1], the frequency control logic latency must be minimized to reduce the overall droop actuation latency. To do so, the asynchronous min path was thermometer-encoded logic to follow the implementation in [1]. This reduces the propagation delay to a single AND logic gate plus associated wiring delays. The post-place-and-routed timing-annotated simulation, shown in Fig. 9, measures a 40ps latency through this path at the SS 0.63V 100°C corner.

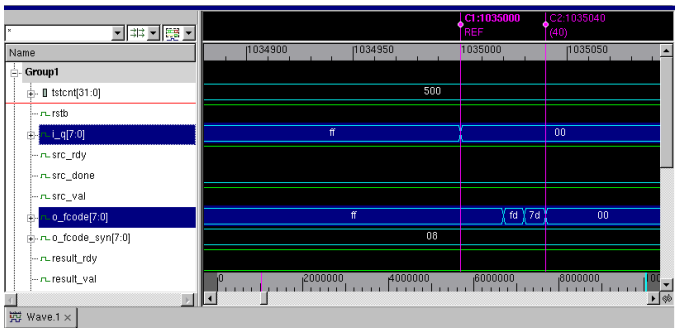

Fig. 9. Post-place-and-route timing-annotated simulation of input to output delay in frequency rebound logic ($q[n]$ to ΔF_{CODE} at SS corner).

PLL lock is achieved through a three-step process: coarse frequency calibration, fine phase locking, and droop response. The latter two interactions have been discussed and demonstrated above. The coarse frequency calibration, conducted to initially bring the DCO within the PLL frequency capture range, sets the DCO's coarse oscillator code. The nominal coarse step is 250MHZ/LSB. This code is frozen following coarse lock, after which point the PLL assumes phase locking control through the fine DAC. The segmentation of the coarse and fine DACs are discussed in V-C.

A sample RTL simulation in Fig. 10 shows the PLL initially locking to a target f_{DCO} of 4.0GHz, then responding to a droop event and temporarily reducing its frequency to 3.5GHz before recovering and restoring DCO frequency.

C. Circuit Design and Implementation

The PLL layout, generated by standard EDA tools and the HAMMER [10] back-end toolchain, is shown in Fig. 11. Its area is 10,000 μm^2 ($100\mu m \times 100\mu m$). This figure excludes the custom-designed DCO, which was estimated to comprise $100\mu m^2$ or less than 1% of the overall PLL area.

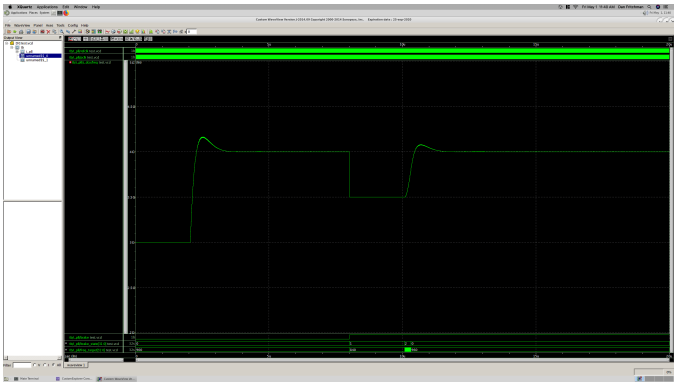

Fig. 10. Logic simulation of initial lock, droop response, and droop recovery from 4.0GHz to 3.5GHz and back to 4.0GHz. Main waveform shows DCO frequency vs. time.

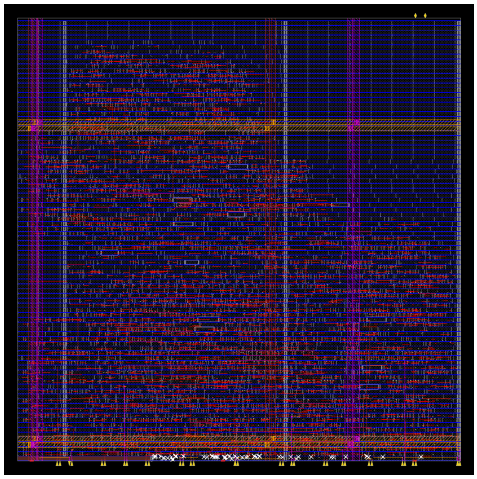

Fig. 11. PLL layout in ASAP7 predictive technology ($100\mu m \times 100\mu m$). DCO is omitted.

The top-level DCO schematic, shown in Fig. 12, was simulated with cell-level RC extractions using Calibre and 60fF of estimated wire capacitances added to each node. For this prototype design, the CLK_{DCO} was chosen to have an f_{MAX} of 5GHz to match the reported f_{MAX} in [1]. This PLL was designed with a CLK_{REF} frequency of 500MHz and a nominal division ratio N_{DIV}^{sync} of 10. The DCO consisted of custom C2MOS cells as shown in Fig. 13.

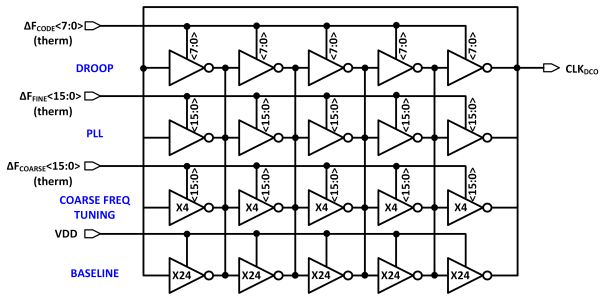

Fig. 12. DCO schematic using custom C2MOS cells.

The design objective was to achieve an octave (2:1) frequency tuning range across corners with sufficiently small resolution for PLL dynamics. This was achieved with segmented coarse- and fine DACs. The fine DAC was designed to span approximately four coarse LSBs: even after initial frequency calibration froze the coarse codes, the PLL could span that range through the fine DAC if lock dynamics

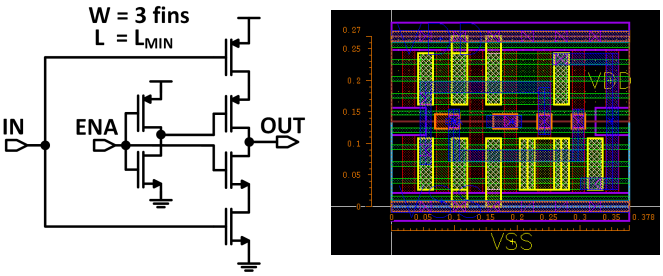

Fig. 13. Custom C2MOS schematic and layout for DCO implementation.

required. This segmentation then obviates gain calibration between the two DACs. For brevity, tuning curves for only the typical corner are shown in Fig. 14.

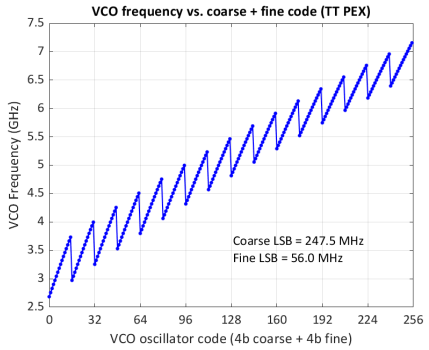

Fig. 14. DCO frequency vs. coarse + fine DAC code at TT corner.

The smaller droop control DAC was designed to tolerate a 10% change in frequency at an f_{MAX} of 5GHz. At high V_{DD}, this corresponds to roughly a 10% droop in supply, corresponding to the maximum droop transient shown previously in Fig. 1. Eight thermometer bits were used for the total droop range, mapping to 60MHz/LSB. The response of this droop control is shown in Fig. 15.

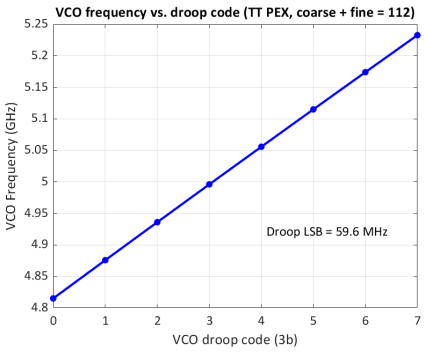

Fig. 15. DCO frequency vs. droop code at TT corner (coarse + fine centred at 5GHz).

The actuation time of the droop input control of the DCO to the time the DCO output clock frequency settles in frequency is shown in Fig. 16 for TT corner and extracted C2MOS cells. For a full-scale change at the droop input requires 267ps to settle to the new frequency, or $6.675\times$ higher than the frequency control logic latency of 40ps reported in Section V-B.

VI. Discussion

A. Actuation Latency Penalty to f_{MAX}

As previously discussed, actuation latency is the time delay from a supply droop to the the clock frequency change. This serves as a proxy to evaluate how f_{MAX} is improved by an adaptive clocking technique for an SoC. More precisely, consider the droop example

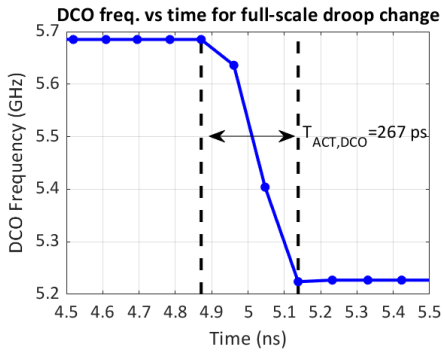

Fig. 16. DCO frequency vs. time with full-scale DCO droop code change.

illustrated in Fig. 17. A supply droop occurs where V_{DD} suddenly decreases from its ideal 1.0V, and the adaptive clocking technique adjusts SoC clock frequecy f_{CLK} after actuation latency T_{ACT}. To mitigate timing failure during this latency, the SoC f_{MAX} can only operate assuming $V_{DD,ACT}$, the supply voltage at the time f_{CLK} changes. Actuation latency limits SoC performance.

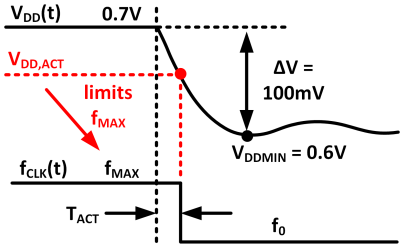

Fig. 17. Effect of droop activation latency on f_{MAX}.

If the supply voltage initially follows $V_{DD}(t) = -\Delta V_d sin(2\pi f_d t)$ from the linear RLC supply network, where ΔV_d and f_d are the voltage drop and frequency of the worst-case droop, then the slope of $V_{DD}(t)$ near the start of the droop is $-2\pi\Delta V_d f_d$. Using T_{ACT}, it can be shown that $V_{DD,ACT} = V_{DD} - 2\pi\Delta V_d f_d T_{ACT}$. Assuming high V_{DD} and a linear transistor model, the achievable f_{MAX} is:

$$f_{MAX} = (1 - \frac{2\pi\Delta V_d f_d}{V_{DD}} T_{ACT}) f_{MAX,ideal} \quad (1)$$

where $f_{MAX,ideal}$ is the maximum SoC clock frequency when V_{DD} is at its maximum voltage. It is apparent from Eq. 1 that, to minimize impact on performance, either the actuation latency, droop voltage, or droop frequency must be reduced or the nominal supply voltage must be increased. In ASAP7 for example, the simulated logic & DCO latencies total T_{ACT} = 40ps + 267ps = 307ps and V_{DD} = 0.7V nominally. To target a 100mV first droop with 50MHz ripple as in [1], a 1.4% f_{MAX} penalty is incurred from $f_{MAX,ideal}$. Note that clock tree insertion delay can significantly change this: simulated delays in ASAP7 for a chain of 45 FO4 inverters incur a delay of 951ps, increasing T_{ACT} to 1258ps and thus f_{MAX} penalty to 5.6%.

B. Comparison to Prior Works

Table I shows a comparison of this feasibility study with prior works discussed. A key point in the analysis from Eq. 1 above is that prior works do not report the theoretical $f_{MAX,ideal}$, but only the comparisons between f_{MAX} achieved with and without their reported adaptive clocking schemes. This includes clock tree insertion delay, which may dominate T_{ACT} and thus the f_{MAX} penalty.

The prototype implemented here based on [1] suggests that design is indeed feasible in ASAP7: as Table I shows, ASAP7 achieves a 5× lower logic latency in the frequency rebound controller from [1]. However, the reported benefits of low-latency frequency rebound control logic is likely small. The DCO actuation time was 267ps in this study: the logic latency accounted for only 15% of the total T_{ACT} even before clock tree insertion. While [1] does not report their DCO latency, logic latency plays an increasingly small role compared to other clocking latencies in ASAP7. Future research can incorporate DCO actuation and clock insertion latencies to achieve a lower effective T_{ACT} and a lower f_{MAX} penalty from ideal.

TABLE I
COMPARISON TO PRIOR WORKS

	[3]	[1]	This work
Process	28nm CMOS	20nm CMOS	7nm Predictive CMOS
Detection Method	Droop sensor	Droop sensor	Droop sensor (assumed)
Adaptation Method	DLL & Phase Rotator	Adaptive PLL	Adaptive PLL
Logic Latency	588ps	200ps	40ps
Clock Freq. Latency	294ps (phase rotator)	Not reported	267ps (DCO)
Power	Not reported	Not reported	1.65mW (DCO only)
Area	Not reported	Not reported	10,100 μm^2

VII. CONCLUSION

This work compares the costs and effectiveness of ACD- and PLL-based adaptive clocking systems [1], [3]. ACD-based systems exchange design complexity for higher power and area, but both systems achieve low-latency actuation and performance. To study feasibility in FinFET, the PLL-based system was implemented in a predictive 7nm CMOS, reporting power, area, and a 5× lower detection logic latency. However, other delays such as the DCO and clock distribution dominate the overall actuation latency, and a theoretical penalty is presented. Both suggest that further FinFET speed-ups in detection logic will have diminishing returns; adaptive clocking systems must consider other latencies in the overall design.

REFERENCES

[1] T. Hashimoto *et al.*, "An Adaptive-Clocking-Control Circuit with 7.5% Frequency Gain for SPARC Processors," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 1028–1037, Aug. 2018.
[2] F. Ahmad *et al.*, "A 0.5-9.5-GHz, 1.2-μs Lock-Time Fractional-N DPLL with ±1.25%UI Period Jitter in 16-nm CMOS for Dynamic Frequency and Core-Count Scaling," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 21–32, Jan. 2017.
[3] K. Wilcox *et al.*, "Steamroller Module and Adaptive Clocking System in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 24–34, Jan. 2015.
[4] M. S. Floyd *et al.*, "Adaptive clocking in the POWER9™ processor for voltage droop protection," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, San Francisco, USA, Feb. 2017, pp. 444–445.
[5] K. A. Bowman *et al.*, "A 16nm All-Digital Auto-Calibrating Adaptive Clock Distribution for Supply Voltage Droop Tolerance Across a Wide Operating Range," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 8–17, Jan. 2016.
[6] J. Kwak and B. Nikolić, "A self-adjustable clock generator with wide dynamic range in 28 nm fdsoi," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 10, pp. 2368–2379, 2016.
[7] N. Kurd *et al.*, "Next Generation Intel Core™ Micro-Architecture (Nehalem) Clocking," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1121–1129, Mar. 2009.
[8] V. Vashishtha *et al.*, "ASAP7 predictive design kit development and cell design technology co-optimization," in *IEEE/ACM Int. Conf. on CAD Dig. Tech. Papers*, Irvine, USA, Nov. 2017, pp. 992–998.
[9] K. L. Wong, T. Rahal-Arabi, M. Ma, and G. Taylor, "Enhancing microprocessor immunity to power supply noise with clock-data compensation," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 749–758, 2006.
[10] E. Wang, A. Izraelevitz, C. Schmidt, B. Nikolic, E. Alon, and J. Bachrach, "Hammer: Enabling reusable physical design."