TICA: A 0.3V, Variation-Resilient 64-Stage Deeply-Pipelined Bitcoin Mining Core with Timing Slack Inference and Clock Frequency Adaption

Jieyu Li[1], Weifeng He[1], Bo Zhang[2], Guanghui He[1], Jun Yang[3], Mingoo Seok[2]

[1]Shanghai Jiao Tong University, Shanghai, China, [2]Columbia University, New York, USA, [3]Southeast University, Nanjing, China

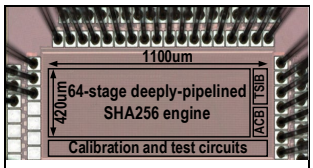
Energy-efficient bitcoin mining cores have gained significant attention since the energy cost for computing dominates the mining expenses [1]. Ultra-low-voltage (ULV) digital circuits have emerged as an attractive approach to improve the energy-efficiency. However, they demand a large timing margin for the worst-case process, voltage, and temperature (PVT) variations, undermining a significant portion of energy savings. Recent works, including multi-phase latch pipeline [1], tunable replica circuits [2-3], *in-situ* error detection and correction (EDAC) [4-6], and dynamic timing enhancement [7], can reduce the pessimistic margin. However, it is not straightforward to adopt those techniques in mining cores due to their deeply-pipelined architecture (up to 128 stages [1]). For example, to adopt EDAC, the deep pipeline requires inserting many bulky error detectors as it has many critical paths. Our experiment with a 0.3V 28-nm mining core shows >18.9% registers need to be replaced with error detectors, considering 6σ local process variation only. Also, multiple stages can have timing errors simultaneously, making an error correction process (e.g., clock gating [5], VDD boosting [6]) complex and costly.

To address those challenges, we propose a new technique that can infer a pipeline's timing slack (t_{slack}) and adaptively set clock frequency to minimize the margin over time. Specifically, it employs a latch-based pipeline and counts how many cycle borrowings the latches did during an observation window (t_{ob}). This cycle borrowing count (N_{CB}) decreases monotonically with t_{slack}. Therefore, we can infer t_{slack} by N_{CB}, and using the inferred t_{slack} we can dynamically set the clock frequency such that it has only a minimal amount of margin. Furthermore, it does not need error correction since we can keep t_{slack} positive, eliminating the overhead associated with the correction. Also, by leveraging the error tolerance of the mining core, we can configure it to operate under a target error rate (1E-6% to 5%) for additional throughput or energy-efficiency gains. Existing EDAC cannot do this because it can only detect timing errors but cannot infer the magnitude and sign of t_{slack} it has from timing errors. The latter, however, is necessary for creating feedback control.

With this proposed **t**iming slack **i**nference and **c**lock frequency **a**daption technique, we prototyped TICA, a variation-resilient bitcoin mining core. Fig. 1 shows TICA's microarchitecture consisting of a SHA256 engine and a TICA controller. The engine implements a fully-unrolled 64-stage SHA256 datapath with 3-phase latches to better balance the path delay. Also, the 3-phase latch pipeline helps to eliminate min-delay buffers [1]. In TICA, we replace 288 latches (out of 49,152 latches, 0.59%) with cycle borrowing detectors. The detector is based on the double sampling latch pair with an XOR gate, which can assert CB if it does cycle borrowing in a cycle.

Based on the CB signals, TICA controller can infer t_{slack}. Fig. 2 shows the block diagram of the TICA controller. It consists of a timing slack inference block and an adaptive clocking block. The inference block employs a CB accumulator to count how many CB signals are high within an observation window (t_{ob}), producing a value called N_{CB}. N_{CB} decreases monotonically with t_{slack}. Therefore, we set up feedback to regulate t_{slack} by constraining N_{CB} within a set-point zone. If N_{CB} is between $N_{CB,refp}$ and $N_{CB,refn}$ (i.e., $N_{CB,refn} \le N_{CB} \le N_{CB,refp}$), the current t_{slack} is considered moderate, where no clock frequency tuning is performed/required; if N_{CB} is smaller (larger) than $N_{CB,refn}$ ($N_{CB,refp}$), the current t_{slack} is considered loose (tight).

Based on the t_{slack} inference, the adaptive clocking block generates $clk_{elastic}$ while minimizing the timing margin over time. It consists of (i) a local ring-oscillator (RO) and (ii) a clock stretch/compress unit. The RO, which shares the same voltage domain with the SHA256 engine, produces clk_{ref} to track fast voltage variations. On the other hand, the clock stretch/compress unit generates 64 phase-modulated clocks from clk_{ref} and synthesizes $clk_{elastic}$, whose frequency can be set between 0.75X to 1.5X of clk_{ref} [4] to minimize the redundant timing margin adatively. At last, from $clk_{elastic}$, the 3-phase clock generator produces clk1, clk2, and clk3 for the engine.



Die micrograph.

We have verified the monotonic relationship between N_{CB} and t_{slack}. We developed an analytical model for a simple case where only one receiver detector/latch is considered (Fig. 3 top) and later extended it to a more complex case (Fig. 3 bottom). For the simple case, t_{slack} can be formulated as $t_{slack}=t_{cbw}-(t_{crit}-t_{clk})$, where t_{cbw} is the cycle borrowing window of a latch, t_{crit} is the delay of the critical path that leads to the detector, and t_{clk} is the clock period. If we define Δt_{crit} as a random variable representing critical path delay variation, we can formulate $t_{crit}=t_{clk}+\Delta t_{crit}$ and thus $t_{slack}=t_{cbw}-\Delta t_{crit}$. Then, suppose f(x) is the delay probability function of the paths leading to the detector, which depends on the circuit implementation and input data. We can model f(x) with a lognormal function since a digital pipeline typically has many critical and near-critical paths after delay/power optimization (Fig. 1 top right). Then, if we assume PVT variations affect every path delay uniformly by Δt_{crit}, the post-variation delay probability function $f_{pv}(x)$ becomes $f_{pv}(x)=f(x-\Delta t_{crit})$. Furthermore, the integration of $f_{pv}(x)$ from t_{clk} to $(t_{clk}+\Delta t_{crit})$ is equal to the expected value of CB, E(CB). By solving this equality, we can get the simplified solution of $t_{slack}=t_{cbw}-t_{clk}\cdot k\cdot E(CB)^{\alpha}$, where k and α are parameters that depend on the mean and standard deviation of the lognormal f(x). Then we extended the t_{slack} equation for the SHA256 engine, which has N_{det} detectors in it. We did it by substituting E(CB) with $N_{CB}/N_{det}/t_{ob}$, which finally gives $t_{slack}= t_{cbw}-t_{clk}\cdot k\cdot(N_{CB}/N_{det}/t_{ob})^{\alpha}$.

We verified the above t_{slack} analytical model through simulation and measurement. Fig. 4 top shows that the model matches with the simulation results of the SHA256 engine. We considered four cases each having different numbers of detectors. While we need to slightly modify the parameters (k, α) for different cases, the inference error is small, only 1-3%. The model also matches the measurement results with the similar 1-3% t_{slack} inference error (Fig. 4 bottom).

The number of detectors (N_{det}) (or insertion rate, i.r.) poses a trade-off between t_{slack} inference accuracy and area and power overhead. Thus, we optimized N_{det}. We posit that we do not need detectors in every critical path since the cycle borrowing ability of a latch-based pipeline can self-heal the delay increase of a critical path in one stage with non-critical paths in the following stages [6]. Therefore, we tried to insert detectors roughly every several stages and not to insert detectors into some critical paths that have a high chance of self-healing. We decided to insert a total of 288 detectors (i.r.=0.59%) as it incurs only ~3% inference error which can be easily compensated by adding ~3% margin to the clock frequency.

On the other hand, we need to set the observation window (t_{ob}) for different target error rates (ERs). For a small target ER, the pipeline performs only a few cycle-borrowings, which makes N_{CB} generally small. To capture a statistically stable N_{CB} value, therefore, we have to observe the CB signals for a more extended period. We have found that t_{ob} of ~5,000 cycles is necessary for a target ER of 0%. Meanwhile, if we target a large ER, we can use a small t_{ob}. We summarize the optimal t_{ob} across different target ERs (Fig.4 bottom).

Based on the proposed technique, we prototyped TICA in a 28nm. We measured our chip at different VDDs from 0.21V to 0.9V (Fig. 5). Compared with the baseline margined for 10% VDD drop, TICA achieves 4.2X clock frequency gain or 19.3% energy savings at 0.3V, with ER=0% while executing 2^{32} cycles. We also measured 17 dies and observed a similar level of energy savings, confirming the robustness of our technique over process variations. By setting the target ER to 1%, TICA achieves additional throughput gain of 35.7% or energy savings of 10.6%. Compared to prior works, TICA achieves a similar level of energy savings yet at the smallest area overhead of only 1.4%, thanks to the relible t_{slack} inference with ultra-low detector insertion rate and no requirement on correction hardware. Also, uniquely from the prior EDAC works, TICA can be configured to operate under a target ER for additional throughput/energy gains.

References: [1] V. Suresh et al., VLSI Symp., 2019. [2] X. Sun et al., VLSI Symp., 2020. [3] M. Cho et al., ISSCC, 2016. [4] W. Shan et al., JSSC, 2020. [5] R. Uytterhoeven et al., JSSC, 2021. [6] S. Kim et al., JSSC, 2015. [7] T. Jia et al., ISSCC, 2020.

978-1-6654-0756-4/22/$31.00 ©2022 IEEE

2022 IEEE Custom Integrated Circuits Conference (CICC) | 978-1-6654-0756-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/CICC53496.2022.9772803

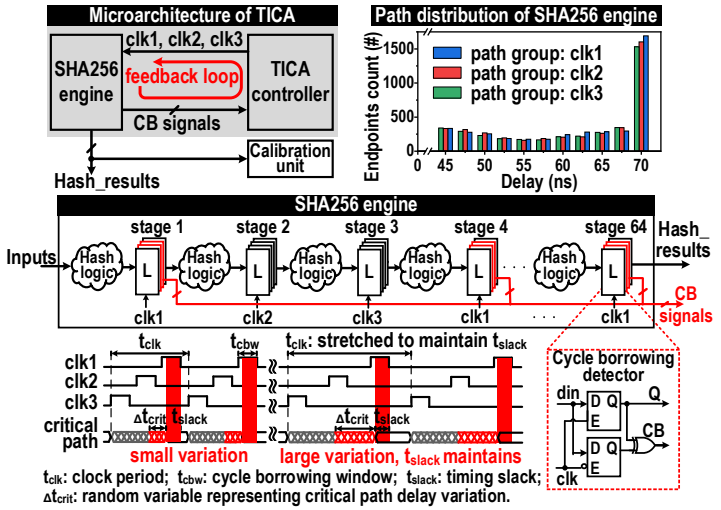


Fig. 1. The microarchitecture of TICA. The feedback loop between SHA256 engine and TICA controller anchors the core's runtime timing slack within a small level across PVT variations adaptively by dynamic clock frequency tuning.

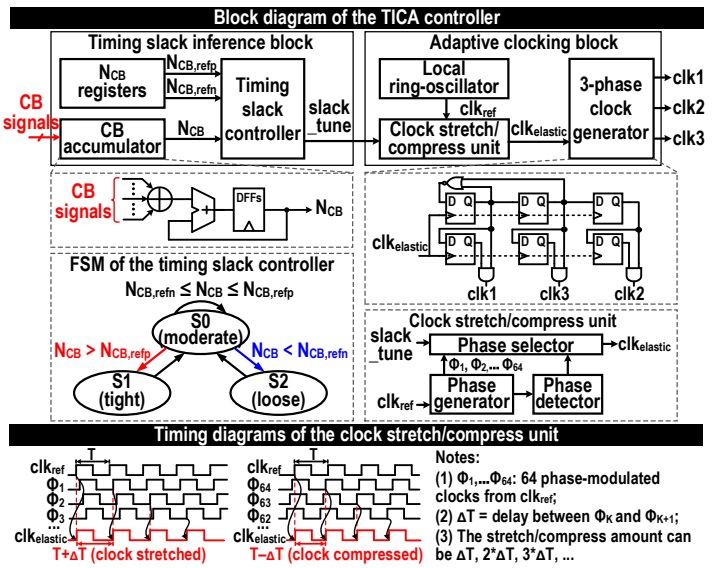


Fig. 2. Block diagram of the TICA controller and timing diagrams of the clock stretch/compress unit.

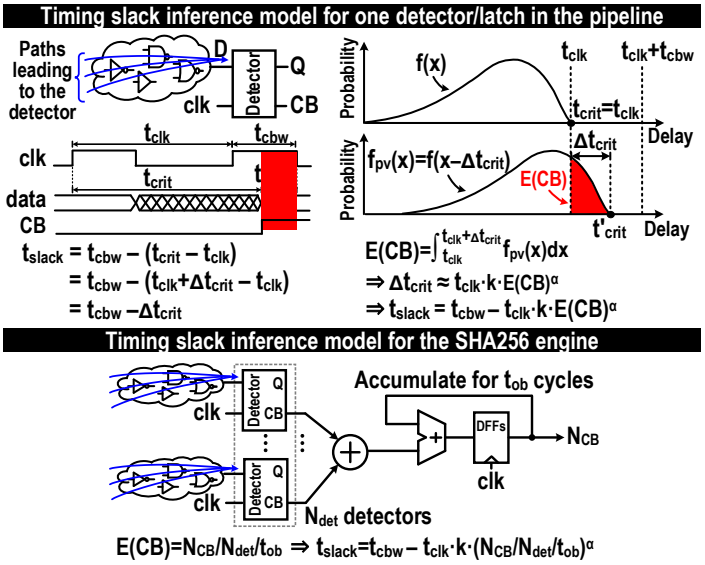


Fig. 3. The analytical timing slack inference model for one receiver detector/latch and the SHA256 engine with N_{det} detectors.

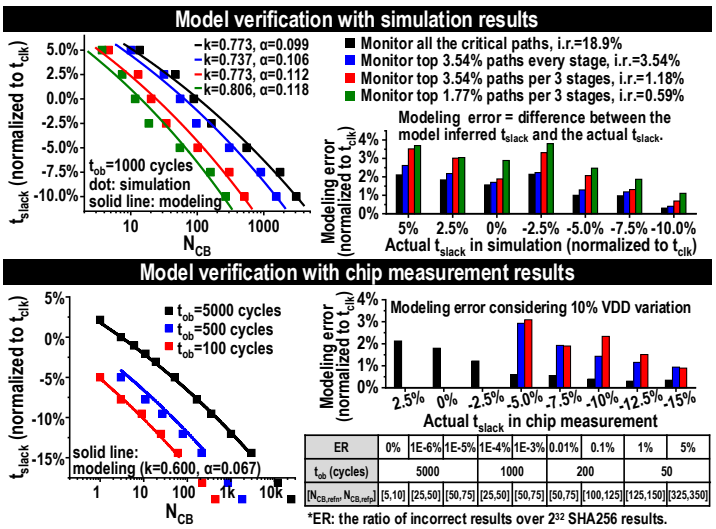


Fig. 4. Verification of the timing slack inference model. Simulations and chip measurements confirm i.r.=0.59% with ~3% timing margin ensures robust inference accuracy under a 10% VDD variation.

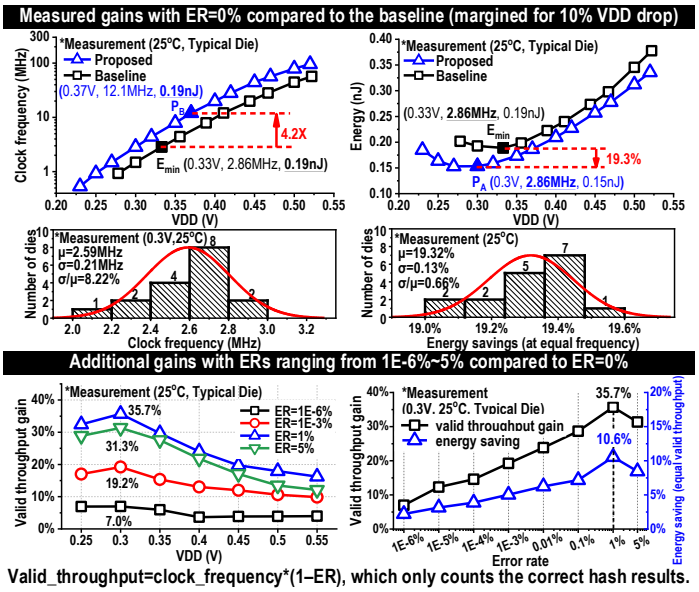
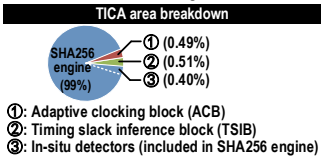


Fig. 5. Chip measurement results.

	[6] JSSC'15	[5] JSSC'21	[1] VLSI'19	This work
Technology	65nm CMOS	28nm CMOS	14nm FinFET	28nm CMOS
Architecture	5-stage microprocessor	3-stage CoolFlux DSP	120-stage bitcoin mining core	64-stage bitcoin mining core
Supply voltage	0.26 – 0.6V	0.25 – 0.65V	0.23 – 0.9V	0.21 – 0.9V
Guardband reduction technique	EDAC and DVS	Completion detection and adaptive clocking	3-phase latch based pipeline	Timing slack inference and clock frequency adaption
Error rate configurable	NO	NO	NO	YES (ER=0%, and 1E-6% ~ 5%)
In-situ detector insertion rate	13% (57/445)	5% (1120/22500)	–	0.59% (288/49152)
Area overhead	8.3%(1)	4.9%(2)	–	1.4%(3)
Throughput gain(4)	2.3X (ER=0%)	–	–	4.2X (ER=0%), and additional 35.7% (ER=1%)
Energy savings(4)	38% (ER=0%)	33% (ER=0%)	–	19.3%(ER=0%), and additional 10.6% (ER=1%)

(1) Including error detectors, min-delay buffers, and the VDD boosting circuits.
(2) Including completion detectors and clock control circuits.
(3) Including cycle borrowing detectors, timing slack inference block and adaptive clocking block.
(4) The baseline of [6] and [7] are margined for 10% VDD drop, temperature and SS process corner; the baseline of this works are margined for 10% VDD drop.



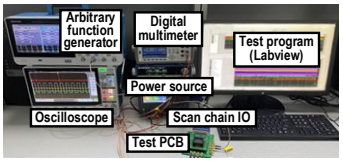


Fig. 6. Comparison with prior works, area breakdown of TICA, and the measurement setup.