A Fine-Grained GALS SoC with Pausible Adaptive Clocking in 16 nm FinFET

Matthew Fojtik, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Stephen G. Tell,
Brian Zimmer, Tezaswi Raja, Kevin Zhou, William J. Dally, Brucek Khailany
NVIDIA Corporation
{mfojtik, benk, aklinefelter, npinckney, stell, bzimmer, traja, kevinz, bdally, bkhailany}@nvidia.com

***Abstract*—Modern SoCs suffer from power supply noise that can require significant additional timing margin, reducing performance and energy efficiency. Globally asynchronous, locally synchronous (GALS) systems can mitigate the impact of power supply noise, as well as simplify system design by removing the need for global timing closure. This work presents a 4 mm^2 distributed accelerator engine with 19 independent clock domains implemented in a 16 nm process. Local adaptive clock generators dynamically tolerate and mitigate power supply noise, resulting in a 10% improvement in performance at the same voltage compared to a globally-clocked baseline. Pausible bisynchronous FIFOs enable low-latency global communication across an on-chip network via error-free clock domain crossings. The SoC functions robustly across a wide range of voltages, frequencies, and workloads, demonstrating the practical applicability of fine-grained GALS techniques for modern SoC design.**

I. Introduction

Demand for increased computational performance has motivated the implementation of large, reticle-limited SoCs that feature billions of transistors over many hundreds of square millimeters of silicon [1]. Many sources of unwanted variation, such as voltage noise, temperature differentials, and spatial process gradients, are exacerbated in these massive systems, making their design and implementation a significant engineering challenge. Current systems often address these difficulties by increasing clock frequency margin, which guards against worst-case variation at the cost of reduced performance and energy efficiency.

Fine-grained globally asynchronous, locally synchronous (GALS) design has been proposed to mitigate the effects of variation [2], [3]. In this approach, a design is broken into many small, synchronous clock domains, each of which operates on an independent clock. Because each domain is asynchronous relative to the others, their clocks can be tuned independently, so timing margin can be reduced as each domain can operate at a frequency best suited to local conditions. GALS clocking has also been shown to reduce peak switching current by spreading switching energy in time [4].

Timing margin in GALS systems can be further reduced with the use of adaptive clocking. Adaptive clocks do not operate at a frequency derived from a fixed reference. Instead, adaptive clocking circuits generate clocks with a frequency that varies over time, speeding up or slowing down in response to changes in local operating conditions [5]. While adaptive clocks can mitigate fixed or slowly-changing variation effects, such as process, temperature, and aging, their primary benefit is their ability to respond to rapid changes in local supply voltage. By quickly reducing clock frequency in response to a voltage noise event, adaptive clock circuits allow digital logic to maintain timing guarantees with less static margin [6]. Fine-grained GALS systems can implement an independent adaptive clock for each synchronous clock domain, enabling local frequency responses to local voltage noise events. Recent analysis has found that replacing coarse-grained clock domains with fine-grained GALS adaptive clocks in simulation can reduce expected power consumption by up to 15% due to improved noise tracking [7] and reduce voltage noise margin by up to 83% [8].

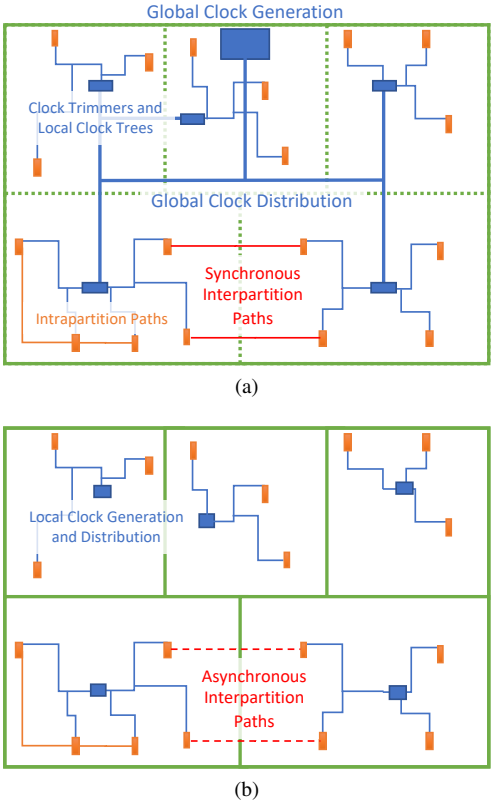


Fig. 1. Fine-grained GALS designs simplify design by eliminating the need for global timing closure. In traditional designs (a), a single clock is distributed to multiple place-and-route partitions, resulting in synchronous interpartition timing paths. In fine-grained GALS designs (b), all interpartition paths are asynchronous and no global clock distribution is required.

This research was, in part, funded by the U.S. Government, under the DARPA CRAFT program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

The fine-grained GALS approach has the added benefit of eliminating the need for the laborious process of global timing closure (see Figure 1). In designs with large, synchronous clock domains, all paths within these large domains must be analyzed for setup time and hold time violations. This task is made even more challenging because physical design is typically performed on *partitions* much smaller than these clock domains, so any netlist or layout adjustments can impact multiple partitions simultaneously, and any changes may therefore require a respin of multiple partitions through place-and-route. By restricting synchronous timing paths to those within the small GALS domains, which need not exceed the size of place-and-route partitions, the task of timing closure at higher levels of hierarchy is eliminated entirely, decreasing design cost and reducing time to market.

This paper presents an 87 M-transistor SoC designed to demonstrate the benefits of the fine-grained adaptive GALS clocking scheme. 19 independent clock domains contain local adaptive clock circuits that generate clocks for their internally-synchronous digital logic. Each clock domain corresponds to a single place-and-route partition, allowing the system to be assembled without global timing closure. The system features the first silicon implementation of pausible bisynchronous FIFOs, an implementation of pausible clocking that easily integrates into standard design flows to achieve low-latency asynchronous communication between clock domains [9]. Measurement and noise-generation circuitry allow the effects of voltage noise to be adjusted and quantified.

Taken together, this fully-functional silicon implementation is a strong technical demonstration of the feasibility and benefits of GALS design.

II. Background

Fine-grained GALS system design should mitigate the effects of supply voltage noise while minimizing system overheads. This section describes the problem of power supply noise, the applicability of adaptive clocking to address it, and the utility of pausible clocking to achieve low-latency interface crossing in a GALS design.

A. Power Supply Noise

A central purpose of fine-grained GALS design is to mitigate the effects of power supply noise. An ideal SoC would deliver a zero-impedance supply throughout the chip, but real systems have parasitic resistance, inductance, and capacitance that can cause supply voltage to fluctuate both locally and globally as current demand varies over time. This power supply noise is split into two classifications, resistive IR drop and resonant di/dt droop.

IR drop refers to the reduction in power supply voltage due to resistances encountered along the power delivery network, and it can vary both spatially and temporally. A measured example of IR drop is shown in Figure 2. Activating digital logic only in the shaded portion of the floorplan causes a larger static IR drop of up to 30 mV in that region, compared to smaller perturbations further away from the current load. The magnitude of local IR drop is determined by the power density of the digital logic and the resistance between the regulator supply and the standard cells.

Resonant supply noise, or di/dt droop, is voltage noise caused by rapid changes in current through the parasitic LC

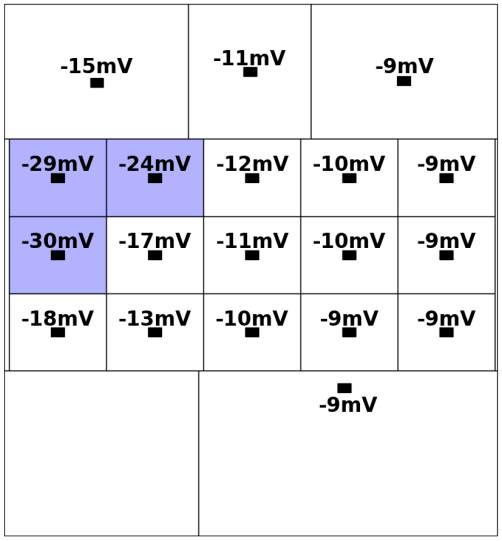


Fig. 2. Testchip with with spatially varying activity. Activating the shaded portion of the logic induces a non-uniform IR drop.

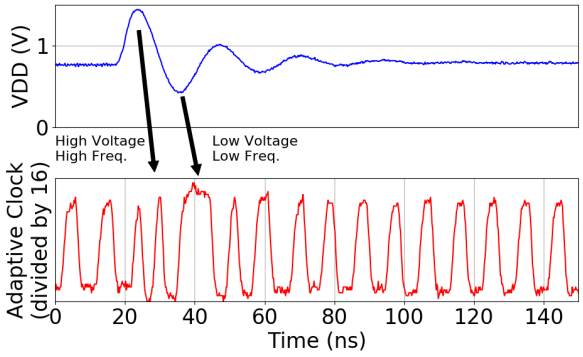


Fig. 3. A sudden decrease in load current induces first droop di/dt noise. An adaptive clock generator adjusts its frequency to compensate.

oscillators in the power delivery network. The most difficult LC-induced droop to address is the so-called "first droop," which is primarily determined by the values of on-die bypass capacitance and package inductance, causing rapid voltage changes at nanosecond timescales.

The blue line in Figure 3 is a measurement of an extreme example of first droop, caused by the sudden halting of a maximum-activity workload on a die with deliberately weakened package power delivery. This rapid decrease in load current causes a voltage spike, followed by a damped oscillation at the parasitic first droop resonant frequency. The worst-case voltage droop is nearly 50% of the nominal voltage in this worst-case scenario.

B. Adaptive Clocking

Conventional digital circuits typically use fixed-frequency clocks, often employing phase-locked loops (PLLs) to multiply external references into the gigahertz range. These clock sources are designed to synthesize a tightly-controlled waveform with good frequency stability and low jitter. Although

this type of clock generator can robustly output a stable clock in the presence of voltage noise, the delay of the digital logic using that clock is still highly sensitive to supply voltage, so voltage noise creates a mismatch between the clock rate and the speed of the logic being clocked. Typically, this mismatch is dealt with by employing safety margins, adding a guardband to ensure that the circuitry will work correctly even under worst-case voltage droop. For example, a digital circuit using a fixed clock and operating under the supply noise shown in Figure 3 would require a slow enough clock to safely operate at 0.4 V, even though the average voltage is much higher. These margins reduce performance and energy efficiency, since the circuit is operating more slowly than the unperturbed supply voltage would allow.

To reduce timing margin, several clock generation and distribution techniques have been proposed that adapt to changing supply voltage by detecting droop events and slowing or gating clocks in response [10]–[16]. These circuits treat frequency changes during droop events as infrequent deviations from an otherwise stable target frequency. Many existing droop detection techniques produce signals that must be synchronized before they are acted on. This synchronization latency limits the effectiveness of such schemes [16]. In contrast, a simple, low-overhead adaptive clock generator can be designed by assembling a ring oscillator with delay elements operating on the same power supply as the logic being clocked [17], [18]. Such adaptive clock generators do not lock to an external reference, even during typical noise-free operation. Instead, the clock frequency automatically adjusts to voltage and temperature changes as the logic speeds or slows, without the need for complicated droop detection circuitry and without any synchronization delay. This avoids the need for complicated PLLs or droop-detection circuitry, allowing for lightweight implementations that can be included in small clock domains with minimal overhead. Figure 3 shows one of these adaptive clocks (divided by 16 for measurement purposes) automatically adjusting frequency in response to supply noise.

C. Pausible Bisynchronous FIFOs

A key challenge introduced by the use of fine-grained GALS with adaptive clocking is the difficulty of reliable, low-latency communication between asynchronous clock domains. Traditional "brute-force" (BF) synchronizers pass asynchronous signals through several synchronizing flip-flops (FFs) in series, reducing the probability of metastability to infinitesimal levels but imposing a multi-cycle latency penalty for each clock domain crossing. An alternative approach known as pausible clocking combines adaptive clocking with fast synchronization [19]–[21]. By adding a condition to the generation of each clock edge, a pausible clock generator only produces the next clock edge when it is safe to do so, guaranteeing a metastability-free transaction (see Figure 4). A mutual-exclusion (mutex) circuit ensures that the synchronized output cannot toggle simultaneously with the rising clock edge. In the rare case that the asynchronous input and the previous clock edge arrive simultaneously, the clock pauses while the metastability resolves, but regardless of the direction of resolution, the circuit will eventually resume safe operation with no data loss.

The pausible clock generator can be integrated into a pausible bisynchronous FIFO as shown in Figure 5. This

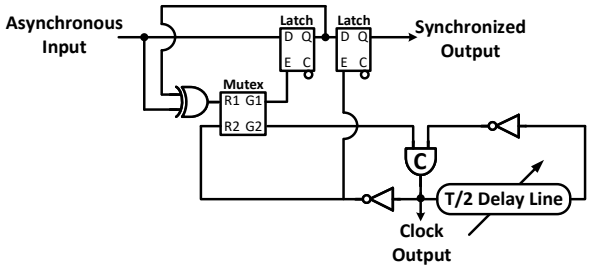


Fig. 4. The pausible adaptive clock generator integrates the adaptive clock generator with synchronizer circuitry.

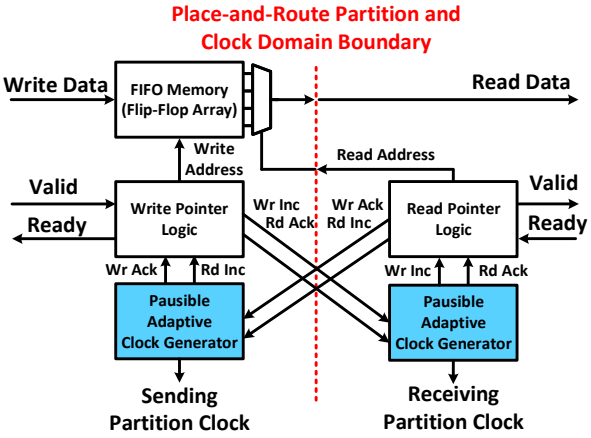


Fig. 5. Pausible bisynchronous FIFOs allow for error-free, low latency clock domain crossings.

circuit is similar to a standard BF bisynchronous FIFO and requires no asynchronous circuit elements beyond those already described, allowing straightforward integration with existing toolflows [9]. Asynchronous increment and acknowledge lines are used to synchronize FIFO pointer updates between the sending and receiving clock domains. This low-overhead synchronization technique imposes minimal area and latency overhead compared to a synchronous FIFO while allowing fully asynchronous GALS operation.

III. Prototype SoC

To demonstrate the efficacy of fine-grained GALS, a prototype SoC was designed that allows direct measured comparison between different clocking and synchronization modes. The design, a programmable machine learning accelerator, consists of a spatial array of processing elements (PEs), a global buffer split into two physical partitions, and a RISC-V microcontroller [22], all connected by a mesh network-on-chip (NoC). The accelerator architecture is described in detail in [23]. The design makes extensive use of latency-insensitive communication at partition interfaces, in which ready/valid signals are bundled with message data and functional correctness is ensured regardless of communication latencies. Latency insensitivity allows the seamless insertion of additional GALS clock domains that have nondeterministic latencies at interface crossings. Each of the 19 synthesized partitions corresponds to an independent clock domain and can generate a local adaptive clock, allowing the system to operate in a GALS fashion

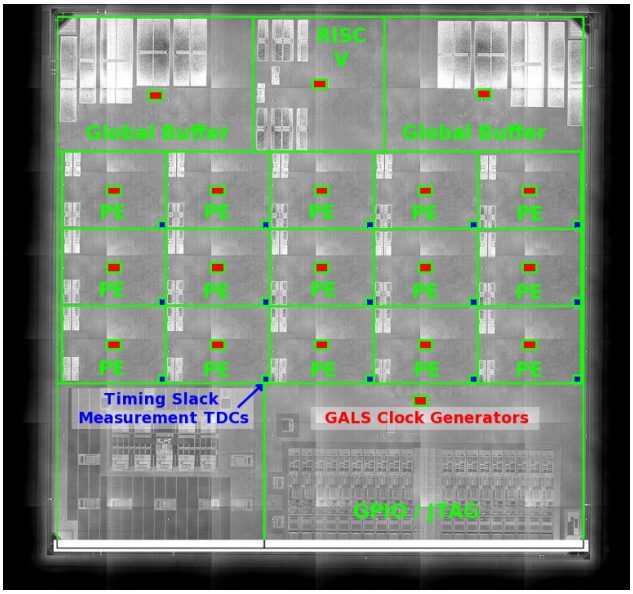


Fig. 6. Annotated die plot of the 16 nm testchip.

using pausible synchronizers to communicate with neighboring partitions. Test and measurement circuitry enables quantitative comparison of the clocking and synchronization options.

The 87 M-transistor design was fabricated in TSMC 16 nm FinFET technology; an annotated die plot is shown in Figure 6. Clocking and synchronization logic is located centrally in each partition, with timing slack measurement circuits placed in the lower right corner of each PE. The total area overhead of per-partition clock generators and pausible FIFOs is 3.6%. A second 16 nm testchip with the same synchronization and clocking scheme was used to obtain some of the measurement results presented in Section IV.

A. Clock Generator

The adaptive clock generator integrates a tunable replica delay line with the necessary asynchronous circuitry to implement the pausible clock generator described in Section II-C. The tunable delay line must track overall system performance, which may depend on a combination of gate-dominated and interconnect-dominated paths [5]. Accordingly, the delay line was designed to closely match the voltage-frequency relationship from a previous chip in the same process to enable accurate power supply noise tracking. The delay of the path can be tuned via programmable codes to adjust the frequency higher or lower at a particular operating condition.

The entire pausible clock generator shown in Figure 4, including mutexes, feedback, and C-element, was hardened in the clock generator macro. These gates were placed in a regular grid, and their timing was verified using post-layout extracted SPICE. Custom layout permitted tighter timings on the critical feedback loop, while avoiding the need to codify the complex timing constraints required to synthesize similar logic. The circuit was provisioned to support up to 16 independent unidirectional latency-instensitive interfaces for each clock domain. Prior analysis shows that each interface requires independent synchronization of three increment and three acknowledge signals to achieve full throughput [9], so a total of 96 independent asynchronous lines are fed into the

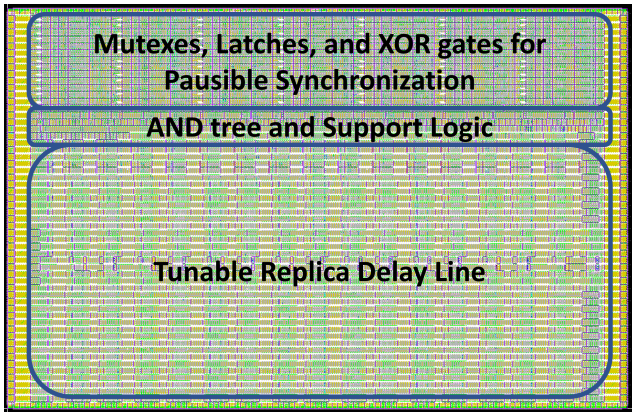


Fig. 8. Layout of the pausible adaptive clock generator.

AND tree ahead of the C-element. Note that because the synchronized signals are not the data words themselves, but instead pointer update signals, each independent interface can transfer an arbitrarily wide data word per cycle, and so the technique does not limit the total bandwidth that can cross each interface.

A behavioral schematic of the clock generator is shown in Figure 7. For the sake of simplicity, buffer trees used to properly distribute signals with large fanouts are not shown. The pausible synchronization logic must be replicated many times because each pausible clock generator must handle synchronization between all neighboring clock domains, and each neighboring domain requires the synchronization of multiple increment and acknowledge signals. Additional configuration bits allow the clock generator to be disabled and reset to a known state or cleanly halted and resumed by external test logic. The delay of the replica path can be changed by programming control bits that adjust the strength of the internal drive cells, allowing each clock generator to be tuned to match the frequency of the actual critical paths in the clock domain. The pausible synchronization path can be bypassed entirely, allowing brute-force synchronization to be employed at the interfaces instead.

Figure 8 shows the layout of the 51.8 µm×33.4 µm clock generator. The replica delay path consumes a majority of the macro area.

B. Clocking and Synchronization Modes

The testchip supports both experimental and legacy operating modes to allow for direct comparisons in the same system. The adaptive clock generator in the lower-right partition can be used as a global clock that is distributed in a conventional manner to each partition via large repeaters and routing on upper metal layers. Each partition can use its own local clock or this global clock, with the selection made via a mux at the root of each local clock tree. The insertion delay of each partition can be artificially increased via a programmable delay line, simulating the effects of larger partition sizes. In addition to pausible synchronization, the inter-partition interfaces also support alternative synchronization using brute-force FFs of either two stages (common in academia) or three stages (common in industry), allowing direct comparison between synchronizer modes.

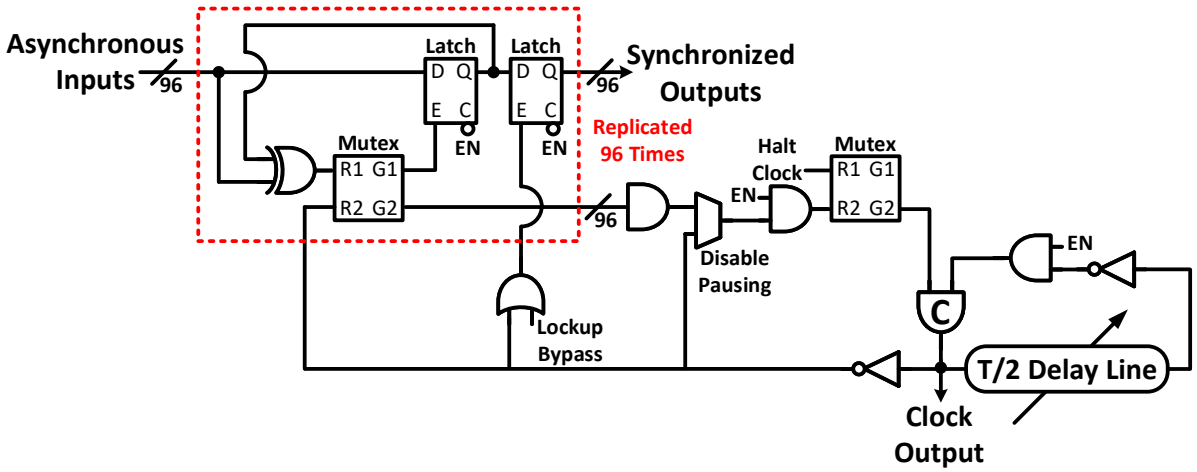


Fig. 7. Behavioral schematic of the pausible adaptive clock generator.

C. Timing Constraints and Design Flow

Each partition was run through a conventional synchronous place-and-route flow using standard timing constraints and tools. Pausible bisynchronous FIFOs were added to all inter-partition interfaces, and the FIFO logic was split across partitions such that each partition clock never leaves its respective partition as shown in Figure 5.

When operating in GALS mode, the only inter-partition timing paths that must be constrained for correctness are the paths from the read address to the read data that both start and end on the receiving side of the FIFOs but pass through the transmitting side. At the top level, these paths are automatically analyzed as typical single-cycle flop-to-flop paths without requiring additional constraints. At the partition level, interface timing constraints were added to ensure these paths would be fast enough to meet single-cycle timing when composed at the top level.

The asynchronous increment and acknowledge signals were also constrained at the partition level. Although their timing does not impact correctness, it does impact interface latency, so should be minimized. The clock generator macro's timing model included worst case arrival times, both early and late, of its synchronized output signals.

Top-level timing was also closed for the globally clocked comparison mode. The tunable insertion delays at the root of each partition clock tree were used to counteract clock skew and align the partition clocks. If this globally-clocked legacy mode was not implemented for comparison purposes, the additional complexity of top-level clock distribution and skew alignment could be avoided entirely.

D. Noise Generators

To provoke worst-case noise events, the logic in the PEs can be repurposed as a controllable voltage noise generator by running scan shift at gigahertz speeds and feeding the logic with repeating programmable 64-bit patterns. The number of 0-to-1 and 1-to-0 transitions in the 64-bit pattern sets the activity factor of the logic and thus the average power consumption of each partition. The clock gate enables in the PE partitions can also be overridden by a programmable, repeating 128-bit pattern to create periods of high and low activity at fine time scales, allowing the construction of artificial di/dt droop events.

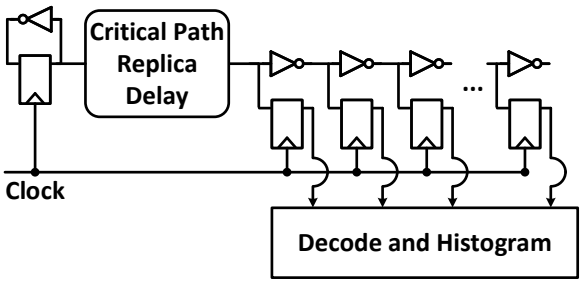


Fig. 9. Schematic of a time-to-digital converter.

E. TDC-based slack measurement

Time-to-digital converters (TDCs) were included in the lower-right corner of each PE. These TDCs measure the amount of setup time slack each cycle for a 1-cycle path in the TDC as shown in Figure 9 [24]. This information is stored in a small memory to be read out at the completion of a test, allowing the measurement of local time-series data corresponding to cycle-by-cycle operating conditions in each partition.

F. Packaging

The test chip was assembled in a flip-chip package with power and ground C4 bumps in a standard checkerboard pattern above the core logic area. To emulate the power delivery to logic in a large SoC, only power and ground balls directly below the die were used to provide power to the package. For a test chip of this size, the package power delivery would have been substantially over-provisioned by using all available resources outside the footprint of the die but within the package footprint.

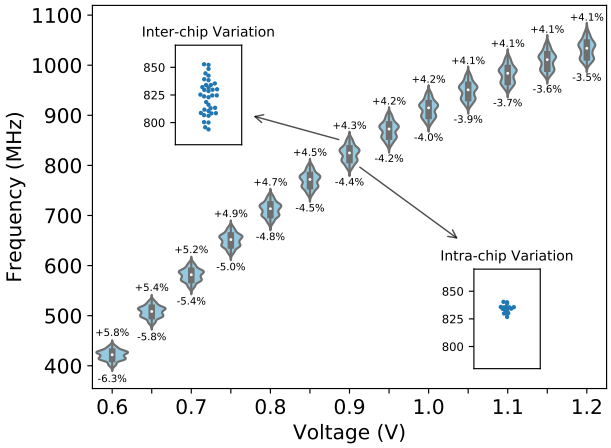


Fig. 10. Variation in average adaptive clock frequencies. The upper-left inset shows inter-chip variation at 0.9 V; each data point is the average of all partition frequencies for one die. The lower-right inset shows intra-chip variation at 0.9 V; each data point represents one partition within the same die. The tuning code of the adaptive clock delay path was reduced to a lower setting to improve measurement accuracy, so the frequencies shown are lower than those of the system under nominal conditions.

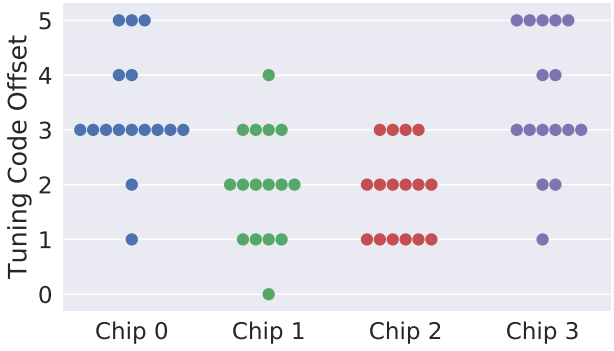


Fig. 11. Variation in replica path tracking across four different chips. Each data point represents the best achievable tuning code for one partition.

IV. Measurement Results

This section summarizes the measurement results from the testchips. Average frequencies of adaptive clocks were measured via on-chip frequency counters that record the number of system clock edges over a time window determined by a fixed reference clock used for this frequency measurement.

Time-series supply voltage measurements of V_{DD} and GND were collected using an oscilloscope to probe the chip supply inputs, relative to the board GND. The difference between V_{DD} and GND, as seen by the chip, is used when reporting on voltage noise.

The noise generators were configured to switch between periods of low activity, in which all the PE clock gates were disabled, and high activity, in which all the PE clock gates were enabled and the logic was driven with a 20% data activity factor. Spatial variation in static IR drop was measured by characterizing the per-partition voltage-frequency relationship under minimal load. The reduction in average frequency at high load could then be correlated to a local average voltage change as observed by each clock generator.

A. Adaptive Clocks and Process Variation

Frequency variation between and within individual dice motivates the need for per-partition clocking. Figure 10 shows measured clock frequencies of all per-partition generated clocks across 36 dice. The delay line configuration was set to a consistent value for each clock domain, so variation in frequency is mostly a result of process variation in the standard cells that make up the replica delay paths. At low voltages, frequency variation of $\pm 12.1\%$ across all dice was observed. As shown in the insets, most of the variation is due to inter-chip variation, though small variations (typically 1–2%) within individual dice are also observed.

Even when adaptive clocks are employed, their effectiveness depends on fine-tuning their replica paths to match critical paths for each clock domain. This replica path tracking also varies with process. Figure 11 shows the best achievable tuning code relative to the worst-case baseline for the adaptive clock replica paths of the PEs of four different chips when running the same workload. By tuning each adaptive clock independently, rather than using a single tuning code for all clock domains, some partitions can see up to five tuning codes of improvement, which corresponds to a frequency improvement of roughly 4%.

B. Pausible FIFOs

Figure 12 shows measured latencies of the pausible interfaces compared to those of the brute-force synchronizers. As the latency of the pausible interface varies depending on data arrival times, measurements were averaged over millions of transactions and multiple hops of the network. The pausible bisynchronous FIFO achieves an average latency reduction of 2.02 cycles compared to three-stage brute-force synchronization. While a definitive stress test is beyond the capability of the experimental system, the pausible FIFOs functioned reliably under test, with no functional issues observed over days of continuous traffic and months of intermittent testing. Pausible clocking can theoretically cause degradation in average frequency due to repeated clock pauses, but pause events remained rare enough in the measured system that no reduction in average frequency could be measured within the limits of on-chip instrumentation (roughly 0.1% precision), even when driving all pausible links at their maximum throughput. The use of pausible FIFOs can therefore improve system performance by reducing network latency without reducing clock rates. Figure 13 shows measured cycle counts from benchmark programs that were executed on the RISC-V core but stored in memory in the PE furthest from the processor. The reduced latency associated with instruction cache fills resulted in a performance improvement of up to 15.5%. The measured latency is comparable to prior published results of pausible synchronizers [25], [26].

To reliably guard against timing errors, pausible clocking circuits must be able to propagate a halted clock to interface leaf nodes with less than one cycle of delay, limiting the amount of insertion delay allowed in the synchronous area [25]–[27]. Even the longest insertion delays in the testchip implementation were well below this limit due to the small clock domain sizes, but the insertion delays of each partition can be artificially increased to determine the amount of

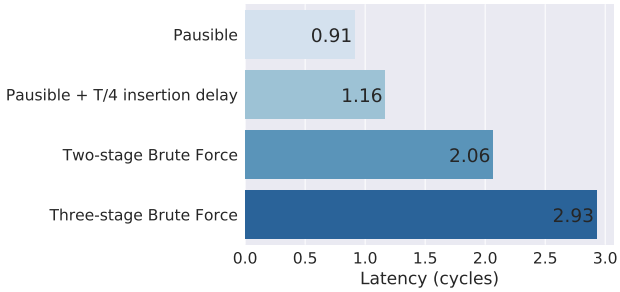


Fig. 12. Measured latency of asynchronous interfaces.

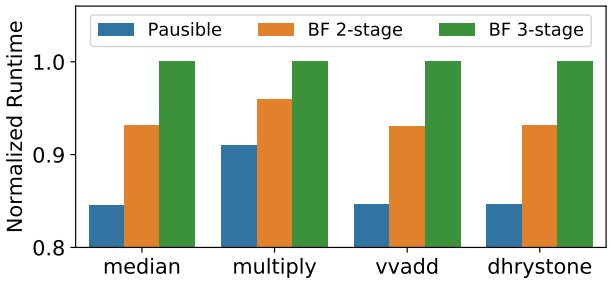


Fig. 13. Impact of synchronization on performance.

remaining margin. Figure 14 shows that roughly a quarter-cycle of insertion delay can be added to the clock tree of one clock domain before provoking pausible synchronizer failures. Since permissible insertion delay is a function of clock period, operating at slower clock rates allows the interface to function with larger insertion delays. The system presented here targets energy-efficient operation at moderate clock rates (1–1.5 GHz at nominal voltage), and so synthesized clock trees were used to minimize clock tree power. Production designs targeting higher performance and frequency often employ structured trees or clock meshes that reduce both skew and latency, ameliorating the insertion-delay limitations of pausible clocking associated with higher-frequency operation.

Insertion delay also impacts latency over the interface, as a larger insertion delay increases the time required for signals that have been safely synchronized to be clocked in the receiving flip-flop. Figure 12 shows the latency impact of an additional quarter-cycle of insertion delay on the pausible interface. Issues with large insertion delays reinforce the importance of using small clock domains to achieve low-latency pausible synchronization.

C. Improved Local Noise Tracking

In order to quantify the benefit of fine grained GALS adaptive clocking over global adaptive clocking, the noise generators were configured to generate identical noise workloads and the TDCs were used to measure per-cycle setup time slack. Since voltage noise is highly dependent on the clocking mode, as described in the next section, all but one partition was driven by the adaptive global clock, and the one partition under test measured its slack when using its own local GALS clock versus the global clock.

In order to exploit the improved tracking of local noise in GALS mode, a non-uniform noise generator workload was

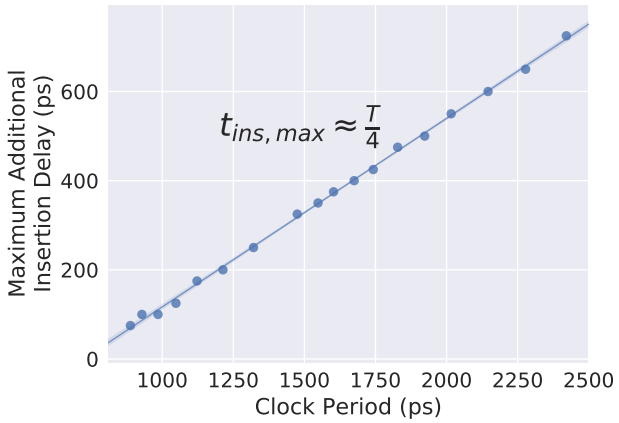


Fig. 14. Maximum insertion delay as a function of frequency.

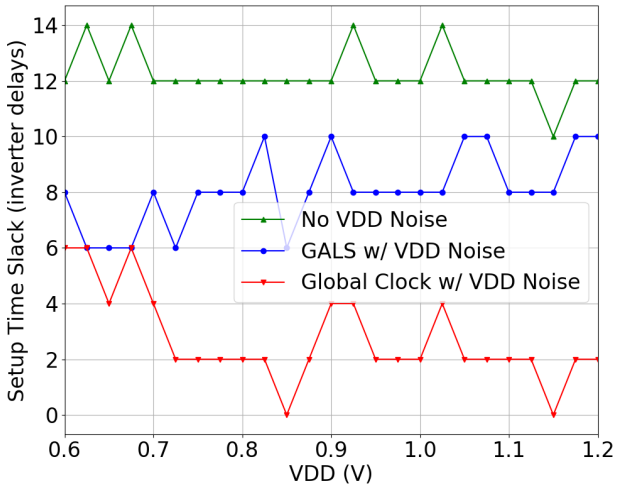


Fig. 15. Setup time slack for various clocking modes, under identical voltage noise

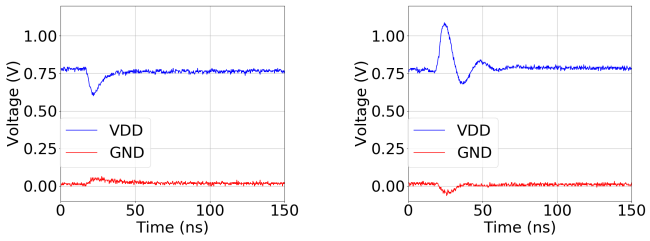
chosen that consumes more power on the left side of the SoC than the right, by only exercising the left two columns of PEs. Slack measurements were taken from the top-left PE, the furthest observable partition from the global clock source.

Figure 15 shows the amount of setup time slack in inverter delays. At higher V_{DD}, more noise is generated, which increases the spread between fine grained GALS and global clocking. The use of the local adaptive clock resulted in a best case improvement of 10 inverter delays of setup time slack over the global adaptive clock, and was within 2 inverter delays of the noise-free case.

D. Worst-Case Noise Mitigation with Adaptive Clocks

Not only do adaptive clocks *respond* to worst-case noise events, they can also *mitigate* these events by reducing their magnitude. First, adaptive clocks act as negative feedback, dampening di/dt oscillations. Second, local adaptive clocks drift out of phase, limiting the extent to which large di/dt events can occur simultaneously across the die.

Figure 16a shows the di/dt noise resulting from a sudden increase in circuit activity under global adaptive clocking. As V_{DD} begins to droop, the clock generator automatically slows



(a) (b)

Fig. 16. Damping of V_{DD} noise with adaptive clocking after a (a) sudden increase in current and (b) sudden decrease in current.

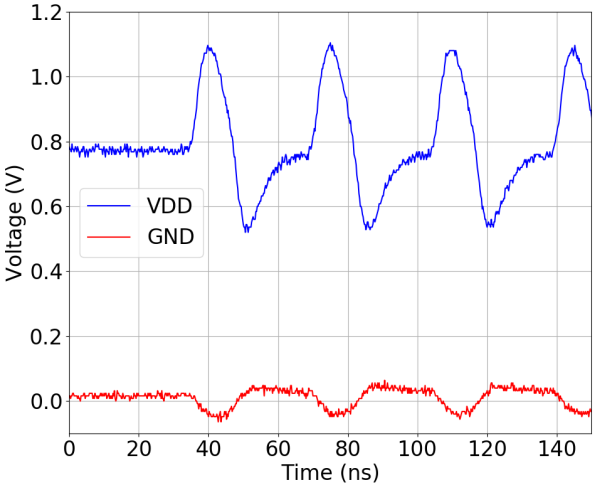


Fig. 17. Package resonant frequency excited by power virus workload.

the clock frequency, which in turn reduces the magnitude of the sudden increase in current, damping the resonant oscillation that would have otherwise been seen in the LC network and eliminating the voltage spike following the initial droop. Figure 16b shows the inverse behavior, in which activity is suddenly decreased. In this experiment, logic is partially clock gated to achieve the step decrease in current. The response of the clock generator cannot therefore provide as strong of a negative feedback response, as the clock is no longer supplying all of the digital logic in the die. Accordingly, the response is less damped than in Figure 16a, and a droop, albeit a reduced one, follows the initial voltage spike. This shows that when it is active, the adaptive clock circuitry is effective at quickly damping resonant oscillations from di/dt events.

Local adaptive clocking can also mitigate worst-case noise events by greatly reducing the likelihood of simultaneous noise events across the system. Worst-case di/dt noise can be generated by switching between periods of low and high activity at the first droop frequency, exciting the parasitic package resonance as shown in Figure 17. When running this noise virus workload in a globally clocked system for longer timescales (shown in Figure 18), the magnitude of noise increases over time until it reaches a maximum droop of 336 mV below nominal.

By virtue of its asynchronous execution, fine-grained GALS clocking reduces the magnitude of this worst-case resonant droop. Figure 19 shows the same noise virus executed on the

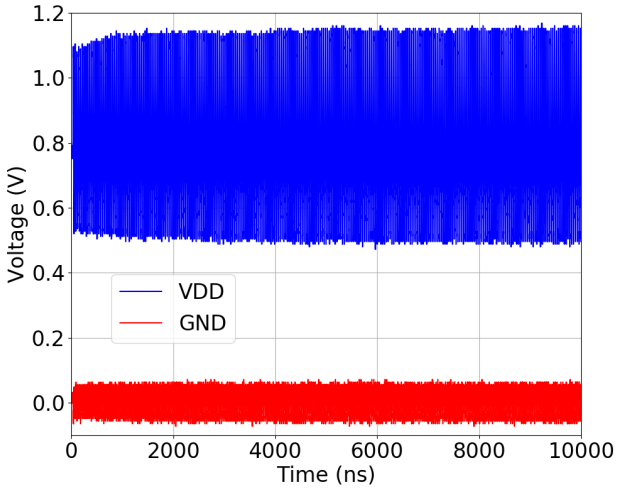


Fig. 18. Effects of a noise virus executed with a global clock.

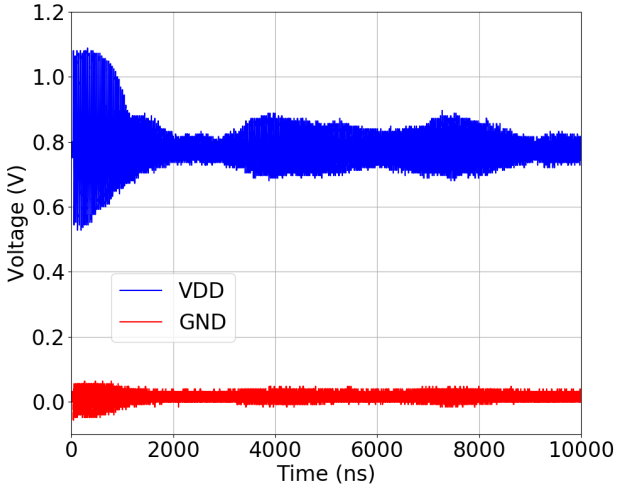


Fig. 19. Effects of a noise virus executed with fine-grained GALS clocks.

system using per-partition adaptive clocking. As the partition clocks drift out of sync, each PE ends up working on a different part of the workload at different times. This reduces the noise magnitude compared to the globally-clocked case in which sustained maximum noise is generated. The adaptive clocks reduce the worst-case measured droop by 64 mV.

Figure 20 shows the measured margin reduction achieved from this effect as translated into an improvement in average frequency. The TDCs were used to detect setup time errors during the noise virus workload, and maximum error-free frequency was measured for the two clocking modes as well as a noise-free baseline. In this extremely noisy environment, a 10% frequency benefit is obtained by reducing the total amount of V_{DD} noise due to this workload-spreading effect.

V. Conclusion

Fine-grained GALS design is a compelling solution to the many design challenges faced in deeply scaled process nodes. This work presents a fully-featured accelerator SoC with fine-grained clock domains in 16 nm FinFET. Per-partition adaptive

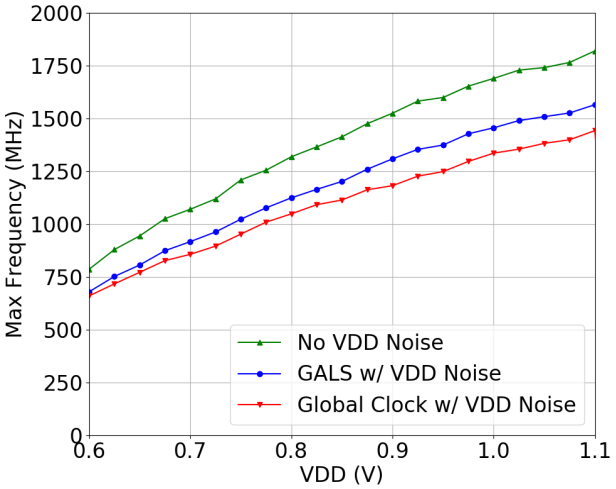


Fig. 20. Frequency improvements from fine-grained GALS clocking.

clocks simplify physical design while both tolerating and mitigating power supply noise, allowing significant reductions in timing margin to improve performance and energy efficiency. Pausible bisynchronous FIFOs achieve error-free asynchronous boundary crossings with a multi-cycle latency reduction. This system provides a template for fine-grained GALS design in modern SoC implementations.

References

[1] J. Choquette *et al.*, "Volta: Performance and programmability," *IEEE Micro*, vol. 38, no. 2, pp. 42–52, Mar. 2018.

[2] D. M. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Stanford University, Oct. 1984.

[3] C. L. Seitz, *Introduction to VLSI systems*. Reading, MA: Addison-Wesley, 1980, ch. 7.

[4] X. Fan *et al.*, "GALS design for spectral peak attenuation of switching current," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, May 2013, pp. 83–90.

[5] T. D. Burd *et al.*, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, Nov. 2000.

[6] Y. Zhou *et al.*, "Modeling and measurement of noise aware clocking in power supply noise analysis," in *Proc. IEEE Conference on Electrical Performance of Electronic Packaging and Systems*, Oct. 2014, pp. 7–10.

[7] D. A. Kamakshi *et al.*, "Modeling and analysis of power supply noise tolerance with fine-grained GALS adaptive clocks," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, May 2016, pp. 75–82.

[8] L. Machado *et al.*, "Voltage noise analysis with ring oscillator clocks," in *IEEE Computer Society Annual Symposium on VLSI*, July 2017, pp. 1–6.

[9] B. Keller *et al.*, "A pausible bisynchronous FIFO for GALS systems," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, May 2015, pp. 1–8.

[10] A. Grenat *et al.*, "Adaptive clocking system for improved power efficiency in a 28nm x86-64 microprocessor," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, Feb. 2014, pp. 106–107.

[11] K. A. Bowman *et al.*, "A 22 nm all-digital dynamically adaptive clock distribution for supply voltage droop tolerance," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 4, pp. 907–916, Jan. 2013.

[12] M. S. Floyd *et al.*, "Adaptive clocking in the POWER9 processor for voltage droop protection," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, Feb. 2017, pp. 444–445.

[13] T. Singh *et al.*, "Zen: A next-generation high-performance x86 core," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, Feb. 2017, pp. 52–53.

[14] H. Mair *et al.*, "A 10nm FinFET 2.8 GHz tri-gear deca-core CPU complex with optimized power-delivery network for mobile SoC performance," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, Feb. 2017, pp. 56–57.

[15] C. Vezyrtzis *et al.*, "Droop mitigation using critical-path sensors and an on-chip distributed power supply estimation engine in the z14 enterprise processor," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, Feb. 2018, pp. 300–302.

[16] P. N. Whatmough *et al.*, "Analysis of adaptive clocking technique for resonant supply voltage noise mitigation," in *Proc. International Symposium on Low Power Electronics and Design*, July 2015, pp. 128–133.

[17] B. Keller *et al.*, "A RISC-V processor SoC with integrated power management at submicrosecond timescales in 28 nm FD-SOI," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1863–1875, July 2017.

[18] J. Cortadella *et al.*, "Ring oscillator clocks and margins," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, May 2016, pp. 19–26.

[19] K. Yun and R. Donohue, "Pausible clocking: a first step toward heterogeneous systems," in *Proc. IEEE International Conference on Computer Design*, Oct. 1996, pp. 118–123.

[20] E. Tuncer *et al.*, "Enabling adaptability through elastic clocks," in *Proc. ACM/IEEE Design Automation Conference*, July 2009, pp. 8–10.

[21] R. Mullins and S. Moore, "Demystifying data-driven and pausible clocking schemes," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, Mar. 2007, pp. 175–185.

[22] K. Asanović *et al.*, "The Rocket Chip Generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr. 2016.

[23] B. Khailany *et al.*, "A modular digital VLSI flow for high-productivity SoC design," in *Proc. ACM/IEEE Design Automation Conference*, June 2018.

[24] C. R. Lefurgy *et al.*, "Active management of timing guardband to save energy in POWER7," in *Proc. International Symposium on Microarchitecture*, Dec. 2011, pp. 1–11.

[25] R. Dobkin *et al.*, "Data synchronization issues in GALS SoCs," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, Apr. 2004, pp. 170–179.

[26] X. Fan *et al.*, "Performance analysis of GALS datalink based on pausible clocking," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems*, May 2012, pp. 126–133.

[27] A. E. Sjogren and C. J. Myers, "Interfacing synchronous and asynchronous modules within a high-speed pipeline," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 5, pp. 573–583, Oct. 2000.