Shared-Staticizer for Area-Efficient Asynchronous Circuits

Samira Ataei and Rajit Manohar
Computer Systems Lab, Yale University, New Haven, CT 06520
{samira.ataei, rajit.manohar}@yale.edu

***Abstract*—Quasi-Delay-Insensitive asynchronous designs can simultaneously provide higher performance, lower energy consumption and less sensitivity to the process variations compared to their clocked counterparts. However, these circuits normally exhibit more silicon area overhead. In this paper, a shared-staticizer solution is presented, to eliminate some part of this area overhead. Staticizers, also known as keepers, are one of the most widely used primitives is asynchronous datapath and control design. Hence, reducing staticizer gate area can result in great area reduction for entire design. Effectiveness of the proposed shared-staticizer method is evaluated in several technology nodes and different asynchronous designs. Results show this technique works correctly down to subthreshold voltage and is superior to other staticizer implementations with respect to area consumption with no impact on performance and power. Shared-staticizer handles output congestion and arbitrary input rates, safely.**

I. Introduction

Synchronous designs are more power hungry due to their clock distribution overhead, precharging and discharging in portions of a circuit unused in the computation and their pipeline storage elements; i.e. flipflops. On contrary, asynchronous design are clock-less, have transitions only in areas involved in the useful computation and their pipeline stages use transparent latches or no explicit latches at all. Therefore asynchronous pipeline designs potentially can result in less energy consumption.

Lower energy consumption and less sensitivity to the process variation in asynchronous designs make them competitive rivals to their clocked counterparts in emerging technologies that need long battery life-time such as medical implants, remote sensors and portable devices. However, to ensure timing robustness in certain encodings, asynchronous circuits require more silicon area. The area of asynchronous control logic that implements the handshaking normally exceeds clocked control logics. This area overhead may result in increased power consumption especially, if the underlying process has poor leakage properties.

Quasi-Delay-Insensitive (QDI) asynchronous designs are robust to gate delay and very careful about energy consumption. However they introduce significant area overhead especially in wide datapaths. Therefore, design techniques for reducing the silicon area overhead in QDI asynchronous design (without sacrificing the energy and performance) are of great interest. It helps to pay-off the tremendous design effort needed to ensure the robustness in asynchronous circuits. It allows to design area-efficient asynchronous circuits of some complexity with superior robustness, performance and power metrics compared to synchronous designs where area/energy is always scarified for latency/throughput. This paper focuses on the area reduction of an asynchronous circuit primitive: the *Staticizer*, also known as keeper.

In asynchronous pipeline design, each functional block stores its results using staticizers. Staticizers are lightweight storage elements widely used in different types of asynchronous circuits, from arithmetic blocks to microprocessors. Different implementations for staticizers has been proposed for both asynchronous and domino circuit families [1], [2], [3], [4]. Also, it has been demonstrated that these lightweight storage elements work correctly at subthreshold voltages with good immunity to leakage and noise [5]. However, one of the challenges of using staticizers is their area overhead as they are widely used on both datapath and control path.

This paper introduces the shared-staticizer approach to address the challenge of designing asynchronous circuits with less area overhead. Shared-staticizer reduces the gate area and hence the total area of the design. A comparative evaluation on completion-detection, asynchronous constant-time counter, a digit-serial adder and a Dadda multiplier demonstrate that the shared-staticizer technique works correctly with area usage superior to traditional weak feedback staticizer approach with no impact on power consumption and performance. Simulations show that shared-staticizer is a robust design and the right choice for high-performance QDI asynchronous designs.

The rest of this paper is organized as follows. Section II explains the operation and limitations of staticizer gate. Sections III and IV provide guidance for staticizer sizing and introduce the shared-staticizer technique. Simulation results and comparisons are shown in section V. Section VI concludes the paper.

II. Staticizer Design

Asynchronous designs by definition have no globally distributed clock to control their timing. Hence, always-on staticizers in datapaths and C-elements for completion detection in control paths are necessary primitives for correct and robust operation in this design paradigm. For correct operation, a staticizer attached to a combinational or state holding gate [1],

[1]A gate is said to be combinational when either the pull-up or pull-down network is conducting and as a result, the gate's output is always driven. Otherwise, if there is a state where the pull-up or pull-down network is not conducting and the gate's output is not driven, gate is said to be state-holding.

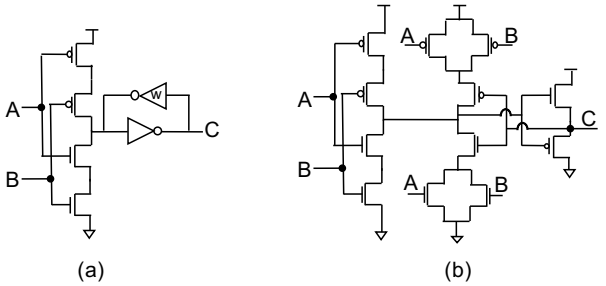


Fig. 1: 2-input C-element with staticizer: (a) weak feedback staticizer, (b) combinational feedback staticizer

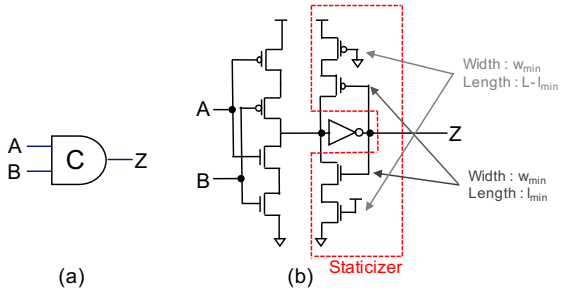


Fig. 2: C-element (a) symbol, (b) circuit implementation with weak feedback staticizer. Transistors of feedback inverter are splitted to reduce the capacitive load at node Z.

must satisfy two requirements: it must hold, or *staticize*, the data when the gate is not driven, and float when the gate is driven.

The two most popular CMOS implementations of staticizer in asynchronous designs are shown in figure 1. Staticizer in figure 1(a) is attributed to A. J. Martin [3] by [5] and [6] and is commonly called weak feedback staticizer. This staticizer has been used in many asynchronous designs including the Caltech asynchronous microprocessor [7] and Manchester University low-power asynchronous ARM processor [8]. Staticizer in figure 1(b) is introduced by I. E. Sutherland [4] and is commonly called combinational feedback staticizer and is used in high-performance micropipelines.

Both staticizers make sure that state-holding gate's output is always driven to V_{DD} or GND. Both schemes work correctly and show robust operation even at ultra-low-voltage (subthreshold) designs with process variation being considered. However, each of these staticizers comes with some disadvantages. Combinational feedback staticizer uses more transistors and consumes more area compared to a weak feedback design. Although this implementation of staticizer is ratio less (no constraint on the size of transistors), it can result in circuits that are more complex and bigger than the state-holding operator itself and is not always applicable.

Weak feedback staticizer is simpler but needs careful transistor sizing not to interfere the gate's operation (race problem). Without appropriate feedback strength, feedback inverter may oppose any changes in state-holding output and result in erroneous operation. However, with proper feedback strength, opposition of feedback inverter not only does not interfere with state-holding output, but also helps to combat noise. Staticizer with right strength for weak feedback inverter improves the noise margin of the gate because when the inputs of gates are slightly above the threshold voltage, staticizer can supply enough current to staticize the output. Transistor of staticizer must be weak enough since they serve only to retain an already-established value but also strong enough to compensate for the leakage current drawn when the gate is not driven. There are two ways to make the feedback inverter of staticizer weaker than a minimum-sized transistor: (i) using a smaller supply voltage for feedback inverter (e.g. $VDD_{feedback} = VDD/2$) and (ii) reducing the driving current of feedback inverter by increasing the transistor length (e.g $L_{feedback} = 10 * l_{min}$). In the first technique, adding a second supply voltage for staticizers increases the design complexity by adding voltage level-shifters and extra routing for second supply voltage. Second technique which proposes a large transistor length can increase the output capacitive load ($C_{gate} \propto W.L$) and also results in more area consumption. Capacitive load increase can be avoided by splitting the transistors of weak feedback , see figure 2(b), and area reduction is the focus of this paper.

TABLE I: 2-input C-element truth table and Boolean function.

a	b	z
0	0	0
0	1	previous state ($\hat{z}$)
1	0	previous state ($\hat{z}$)
1	1	1

$z = a.b + \hat{z}.(a + b)$

Here in this paper, first we provide the guidance for staticizer sizing selection in weak feedback scheme and then proposes a circuit modification technique that reduces the area overhead of weak feedback staticizer. The proposed technique is simple and does not add any complexity to circuit design while greatly reduces its layout area. Simulation results in following sections demonstrate proposed technique works correctly in QDI asynchronous designs at different technology nodes .

III. Staticizer Sizing

Using a 2-input C-element (consensus element), we describe the constraints for appropriate staticizer size selection. The C-element is introduced by D. E. Muller [9] and if often called Muller C-element. It is an important element in asynchronous circuit design and is widely used for control synchronization. In general, a C-element is a state holding gate which is transparent when all its inputs are equal and holds the previous output value otherwise. Table I shows the truth table of this gate and its Boolean function. The Muller C-element can easily be generalized to three or more inputs, requiring that all inputs reach a new logical state before copying that new state as output [4].

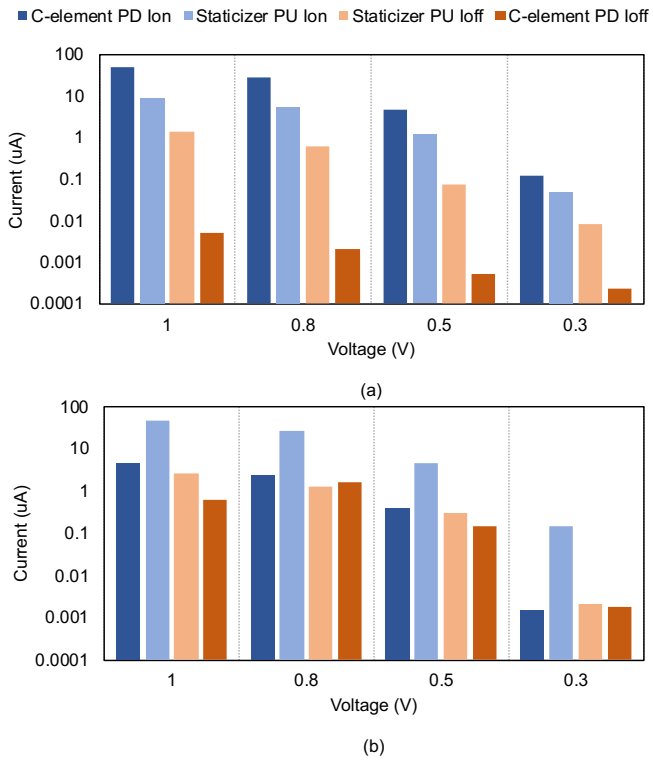


Fig. 3: Driving and leakage current of C-element and weak feedback staticizer at different voltages. Weak feedback staticizer with (a) proper sizing ($L = 10l_{min}$) works correctly at different voltages and satisfies the current conditions i and ii, (b) improper sizing ($L = 2l_{min}$) fails and doesn't satisfies the current conditions at small voltages.
(i) staticizer $I_{off} + 3\sigma <$ C-element $I_{on} - 3\sigma$
(ii) staticizer $I_{on} - 3\sigma >$ C-element $I_{off} + 3\sigma$

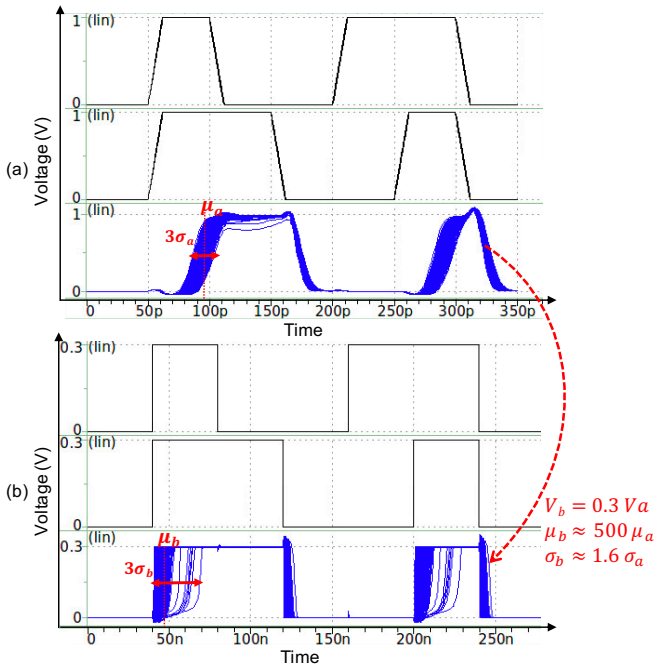


Fig. 4: Monte Carlo simulation results for C-element with weak feedback inverter staticizer: (a) at nominal voltage of technology and (b) at subthreshold voltage.

In order for C-element to hold its output while inputs are not equal, a staticizer must be used at the output of this gate. Without the staticizer and with complementary inputs being persistent, the charge stored on output of C-element can drift over time due to subthreshold, gate and junction leakage currents thereby changing the value of output and result in an incorrect execution. Adding a weak feedback staticizer to C-element gate may introduce interferences between the original production rules of gate and the rules added by the feedback inverter. This interference can be resolved by adjusting the relative strengths of feedback inverter and the switching networks this staticizer is connected to. Figure 2 shows the symbol we use for the C-element (a standard AND logic symbol with a large C inside) and its circuit implementation in CMOS technology with weak feedback staticizer to maintain the gate's output while it is not driven. To reduce the extra capacitive load at output Z, introduced by long keeper transistors, these transistors are splitted as shown in Figure 2(b). Minimum sized transistors are closer to output and the transistor with bigger length are tied ON to act as a current source with a very small current value ($I \propto w_{min}/(L - l_{min})$).

Staticizer's pull-down (PD) transistors sizing can be identified by capturing the leakage (I_{off}) and driving current (I_{on}) of C-element pull-up (PU) network. Similar measurement with the C-element PD network will identifies ratio constraint of the staticizer's PU transistors. C-element PD I_{on} and I_{off} define the bounds for staticizer PU feedback strength: (i) staticizer's PU drivability (I_{on}) must be greater than C-element PD I_{off} to hold the value, and (ii) staticizer's PU I_{off} must be smaller than C-element PD I_{on} current to avoid any opposition (race condition) during transitions.

IV. Shared-Staticizer Approach

Following experiments are performed using an industrial 65nm bulk CMOS technology. The effect of process variation is included in all the tests through 1,000 point Monte Carlo simulations[2]. For staticizer transistor we use the sizing shown in figure 5(a). Figure 3 shows the current values of C-element PU network and staticizer PD network at selected V_{DD} values from 1.0V (nominal voltage of technology) down to the subthreshold 0.3V (transistor threshold voltage for this technology is around 380 mV) for $K = 10$. Current deviation (σ) is added to all the current values of figure 3, i.e. $I_{off} + 3\sigma$ and $I_{on} - 3\sigma$ for C-element PD currents and $I_{off} - 3\sigma$ and $I_{on} + 3\sigma$ for staticizer PU currents. Staticizer must be

[2]Inter-die and intra-die variation in all process corners (TT, SS, FF, SF, FS) is used in Monte Carlo simulations to predict the silicon distribution accurately

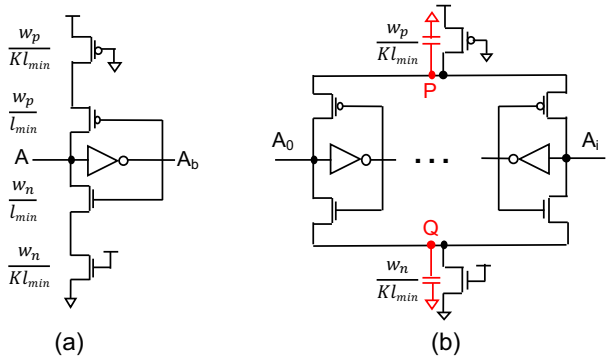


Fig. 5: (a) Staticizer transistor sizing for correct functionality, and (b) proposed shared-staticizer design. Multiple staticizers share their long channel transistors at nodes P and Q.

able to handle 3σ of intra-die variation to have negligible impact on yield when design has many staticizers. As shown in figure 3(a) a transistor with 10 times less driving current compared to a minimum size transistor, satisfies the I_{on} and I_{off} relations between staticizer and C-element at different voltages. Figure 3(b) shows with improper sizing ($K = 2$) current relations are not satisfied at low voltages which results in wrong operation.

C-element delay depends on the arriving time and order of the inputs. Extensive simulations by altering the arriving time and arriving order between inputs show that weak feedback staticizer with $K = 10$ is a robust design. Figure 4(a) and (b) show the Monte Carlo simulation results of C-element input and output voltages for selected input orderings. At sub-threshold, figure 4(b), there is a 500X performance degradation and %60 more variation in output delay. However, with weak feedback staticizer, C-element shows a robust operation; it can correctly write new values and hold its previous value.

Results of above experiments show that staticizer with properly ratioed feedback path can guarantee the robust operation of state-holding gate. However, with ever increasing I_{off}/I_{on} ratio and process variability in advanced process nodes, only very long staticizers ($> 10l_{min}$) can satisfy the current bounds defined by the pull-up and pull-down networks of the gate. Very long staticizer has implications for the physical design of the circuit and may not always be practical.

Here we propose a shared-staticizer design where multiple weak feedback staticizer can share the long foot transistors in order to reduce the state-holding gate area. Proposed scheme is shown in figure 5(b). As shown in this figure both long channel NMOS and PMOS devices which contribute to more than %40 area of staticizer (see figure 7), can be shared between multiple staticizers.

Figure 6 shows a C-element tree, commonly used as completion detection in dual-rail asynchronous designs. In this design the output z cannot change until all n dual-rail inputs are equal, i.e $A_0...A_n = n'b0$ or $A_0...A_n = n'b1$. We applied the shared-staticizer design to an 8-bit C-element tree and compared the delay, power and area values with the same size

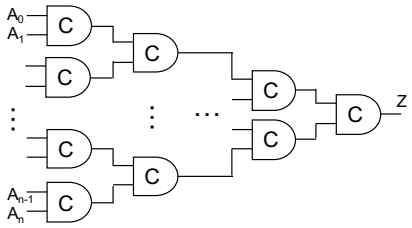


Fig. 6: N-input C-element tree

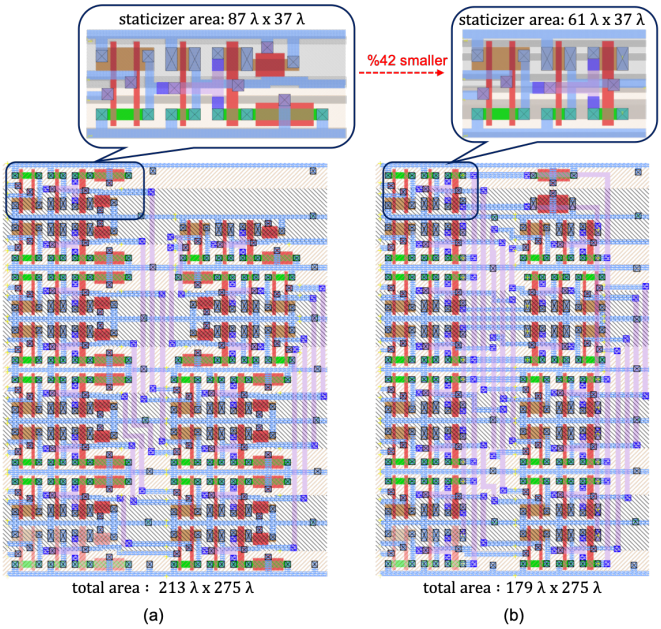


Fig. 7: Area comparison for 8-bit C-element trees: (a) without sharing the staticizers and (b) with shared-staticizer design.

C-element tree where staticizers are not shared in multiple technology nodes. Simulation results in table II show that C-element tree with shared-staticizer has almost the same delay and power consumption as C-element tree with weak feedback staticizer in different process nodes. However, the layout area is significantly smaller when long transistors of feedback inverter, are shared. The layout area numbers of table II are calculated using standard-cell-like C-elements and automated routing. Figure 7 shows even with custom layout, C-element tree with shared-staticizer consumes %20 less area compared to other C-element tree. [3]

Shared staticizer is a simple structure and easy to implement. However, there is a ceiling for number of staticizers that can share their foot transistors which should be defined by designer. This ceiling value is technology dependent and is related to the capacitors at nodes Q and P (C_Q and C_P), shown in figure 5(b). C_Q and C_P go high with area of source/drain. Higher number of shared staticizers, increases these capacitance as every staticizer contribute some diffusion

[3]Both layouts are drawn manually in SCMOS 0.5um process node with same number of metal layers.

TABLE II: power and delay comparisons for 8-bit C-element trees in different technology nodes, with shared-staticizer and conventional weak feedback staticizer.

Technology	0.5um		65nm		28nm	
Design	shared-staticizer	conv-staticizer	shared-staticizer	conv-staticizer	shared-staticizer	conv-staticizer
VDD (V)	5.0	5.0	1.0	1.0	1.0	1.0
power (mW)	2.88	2.9	0.34	0.37	0.16	0.16
Delay (nSec)	1.16	1.2	0.13	0.14	0.096	0.1
Layout Area (normalized)	1	1.5	1	1.43	1	1.46

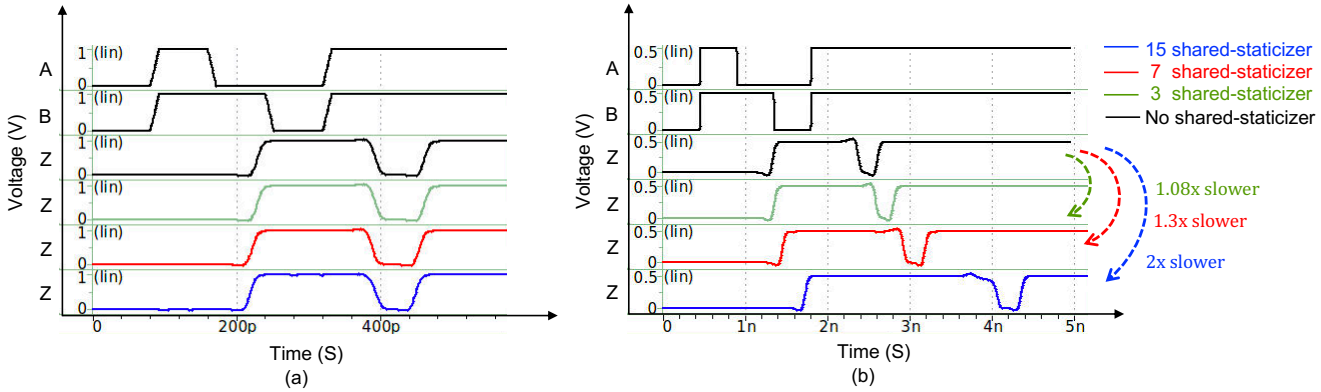


Fig. 8: Performance of a 8-bit C-element tree at (a) $V_{DD} = 1.0V$ and (b) $V_{DD} = 0.5V$ with different number of shared staticizers. At low-voltages 15 shared-staticizer slows down the design bt 2x while 2 shared-staticizer has negligible effect.

capacitance on these nodes . Large C_Q and C_P reduce the voltage headroom and become the source of delay by shifting the switching point of the gate to right.

Figure 8(b) shows 2x slowdown in performance of a 8-bit C-element tree in 0.5V supply voltage when 15 staticizers share their foot transistors while with 7 staticizer sharing there is 1.3x slowdown and with 3 staticizer sharing there is only 1.08x slowdown. Figure 8(a) shows at nominal voltage there is no slowdown because the voltage swing is big enough that is not effected by values of C_Q and C_P.

As a side note, reduced voltage swing improves the propagation delay of the gate when V_{DD}/V_{TH} ratio is big, however it slows down the circuit and might lead to incorrect results in low-voltage designs where V_{DD}/V_{TH} ratio is small.

SOI (Silicon-on-Insulator) technologies have smaller parasitic capacitance and virtually eliminate the diffusion capacitance, hence allow more staticizers to share their foot transistors.

V. Experimental Evaluation

To evaluate the effectiveness of the proposed shared-staticizer scheme, this technique is applied to three different asynchronous designs using an industrial 28nm process node. Both shared and regular weak feedback implementations are examined in following asynchronous designs and for each design, we measure the latency, cycle time, power consumption and layout area. In implementation, simulation and layout generation of following asynchronous circuits, we used the ACT framework developed at Yale University [10].

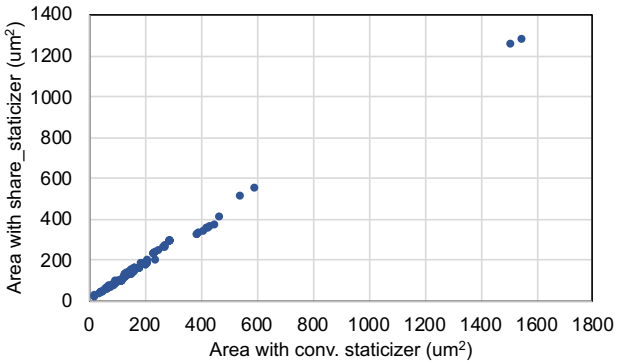
Asynchronous counter: Counters play critical role in the control logic and pipeline management of both synchronous and asynchronous designs. We picked an 8-bit counter as our first case study. This counter has a constant response time and is implemented in the most robust asynchronous logic family, QDI circuits. There are a total of 14 staticizer in the implementation of this counter where all staticizers are used in weak-conditioned half buffer (WCHB) handshake circuits. In this design every 4 and 6 staticizers share their foot transistor. Simulation results summarized in tableIII show sharing the staticizer does not have any impact on latency, cycle-time and power consumption of this design. However the layout area is decreased by %16.

Asynchronous adder: The second case study is an asynchronous digit-serial adder implemented with the combination of bundled data datapath and QDI control. This digit-serial adder with arbitrary-length is based on LSB first serial addition algorithmand has the opportunity to decrease the overall energy usage while increasing the throughput/area efficiency. There are a total of 397 staticizer in the implementation of this adder. Staticizers are used in WCHB FIFOs and 4-bit C-element trees. In this design every 6 staticizers in FIFOs and every 7 staticizer in C-element trees share their foot transistors. Simulation results in tableIII show similar to counter, sharing the staticizer does not have any impact on performance and power consumption of this design. However, the total area is reduced by %21.

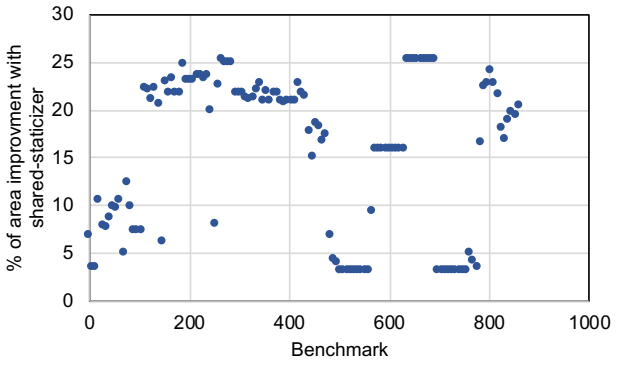
Asynchronous multiplier: Multipliers are key hardware components in a wide range of application from signal processing

TABLE III: Power, performance and area comparisons for three asynchronous circuits in 28nm technology node, with shared-staticizer and conventional weak feedback staticizer. All designs are simulated at the nominal voltage of the technology.

Design	8-bit Counter		16-bit Adder		64-bit Multiplier	
	shared-staticizer	conv. staticizer	shared-staticizer	conv. staticizer	shared-staticizer	conv. staticizer
power (mW)	0.94	0.97	0.74	0.75	39.8	39.7
latency (nSec)	0.38	0.38	0.97	0.97	0.95	0.96
Cycle Time (nSec)	0.63	0.64	0.57	0.59	1.01	1.02
Layout Area (Normalized)	1	1.16	1	1.21	1	1.16



(a)



(b)

Fig. 9: (a) Design layout area with conventional staticizer vs. shared-staticizer in 28nm process node and (b) percentage of layout area improvement using shared-staticizer.

to machine learning accelerators. Multiplication is a hardware intensive operation and less silicon area in its design is of the great interest in VLSI design. Asynchronous multipliers have higher throughput and consume less energy but are not compact in terms of area. In our third case study we apply shared-staticizer technique to a 64 x 8 bit Dadda multiplier implemented with a dual-rail encoding and QDI control.

There are a total of 1365 staticizer in the implementation of this multiplier where staticizers are used in 8-bit, 16-bit and 32-bit C-element trees, WCHB FIFOs and single-bit full-adders of fine-grain asynchronous precharge half buffer (PCHB) pipelines . In this design every 7 staticizers in C-element trees, every 6 staticizers in WCHB FIFOs and every 4 staticizers in full adder modules share their foot transistors which reduces the number of long transistors from 2730 to 436 and helps to reduce the silicon area of this multiplier by %16. Simulation results in tableIII show sharing the staticizer does not have any impact on performance and power consumption of this multiplier.

Simulation results of above case studies demonstrate staticizers with shared transistors have no impact on performance and energy consumption while significantly decrease the silicon area of the design. Figure**??** shows the layout area for over 800 benchmarks. All these benchmark have QDI control circuits. As shown in this figure shared-staticizer reduces the layout area from %3 up to %26.

Shared-staticizer is an efficient technique that can be used in asynchronous circuits with QDI controls. This technique may not be as efficient in asynchronous circuit families that use bundled-data encoding.

VI. Conclusion

This paper presents shared-staticizer method to reduce the layout area of QDI asynchronous circuits. Shared-staticizer shows area usage superior to other implementations of staticizer. It is a robust and easy to implement technique that has no implication on performance and energy consumption. Staticizers with shared transistors can hold a stages output under arbitrary congestion and remain stalled and safely handle any uncertainty in the arrival time of circuit's inputs. We can say that the weak feedback shared-staticizer is definitely the right choice for high performance QDI asynchronous designs. Shared-staticizer is applied to multiple asynchronous designs and evaluated in several technology nodes and simulation results verify the effectiveness of the proposed technique. Shared-staticizer is an improvement only for QDI circuit family and can be a useful technique in asynchronous architectures where parallel throughput is maximized through spatial parallelism. It allows to fit more functional blocks within a given area budget that determines the amount of parallelism.

References

[1] C. H. Kim, K. Roy, S. Hsu, A. Alvandpour, R. K. Krishnamurthy, and S. Borkar, "A process variation compensation technique for sub-90nm dynamic circuits," in *Symposium VLSI Circuit Digest of Technical Papers*, pp. 205–206, June 2003.

[2] Y. Lih, N. Tzartzanis, and W. W. Walker, "A leakage current replica keeper for dynamic circuits," in *IEEE Journal of solid-state circuits*, no. 1, pp. 48–55, January 2007.

[3] A. J. Martin, "Programming in VLSI: from communicating processes to delay-insensitive circuits," in *Caltech Report* (W. J. Dally, ed.), pp. 1–66, 1989.

[4] I. E. Sutherland, "Micropipelines," in *Communications of ACM*, pp. 720–738, June 1989.
[5] Y. Chen, M. Seok, and S. M. Nowick, "Robust and energy-efficient asynchronous dynamic pipelines for ultra-low-voltage operation using adaptive keeper control," in *IEEE Symposium on Low Power Electronics and Design*, pp. 267–272, January 2013.
[6] M. Shams, J. C. Ebergen, and M. I. Elmasry, "A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element," in *IEEE Symposium on Low Power Electronics and Design*, pp. 93–96, August 1996.
[7] A. J. Martin, A. Lines, R. Manohar, M. Nystroem, P. Penzes, R. Southworth, and U. Cummings, "The design of an asynchronous MIPS R3000 microprocessor," in *Advanced Research in VLSI*, pp. 164–181, 1997.
[8] S. Furber, "Computing without clocks: Micropipelining the ARM processor," in *G. Birtwistle and A. Davis Asynchronous Digital Circuit Design Workshops in Computing*, pp. 211–262, 1995.
[9] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proceedings of an International Symposium on the Theory of Switching*, pp. 204–243, Harvard University Press, April 1959.
[10] R. Manohar, "ACT: Asynchronous circuit toolkit." https://github.com/asyncvlsi/act, 2019.
[11] N. Bingham and R. Manohar, "QDI constant-time counters," in *IEEE Transactions on Very Large Scale Interation (VLSI) Systems*, no. 1, pp. 83–91, January 2019.
[12] N. Bingham and R. Manohar, "Self-timed adaptive digit-serial addition," in *IEEE Transactions on Very Large Scale Interation (VLSI) Systems*, no. 9, pp. 2131–2141, September 2019.
[13] L. Dadda, "Some schemes for parallel multipliers," in *Alta Frequenza*, no. 5, pp. 349–356, May 1965.