Title
Impolite High Speed Interfaces with Asynchronous Pulse Logic

Authors
Miller, Merritt
Segal, Carrie
Carthy, David Mc
et al.

Publication Date
2018-05-30

DOI
10.1145/3194554.3194592

Impolite High Speed Interfaces with Asynchronous Pulse Logic

Merritt Miller
ECE Department, UCSB
Santa Barbara, California
merrittmiller@ece.ucsb.edu

Carrie Segal
ECE Department, UCSB
Santa Barbara, California
chsegal@umail.ucsb.edu

David Mc Carthy
ECE Department, UCSB
Santa Barbara, California
davidmc@ece.ucsb.edu

Aditya Dalakoti
ECE Department, UCSB
Santa Barbara, California
adityadalakoti@ucsb.edu

Prashansa Mukim
ECE Department, UCSB
Santa Barbara, California
prashansa_mukim@umail.ucsb.edu

Forrest Brewer
ECE Department, UCSB
Santa Barbara, California
forrest@ece.ucsb.edu

Abstract

We present a design solution that allows design of higher-than-core rate operation with techniques that avoid PLL/DLL blocks to provide higher speed timing. Many modern integrated circuits (ICs) have high speed interfaces which operate at higher cycle rates than the core of the IC. As a result of the higher-than-core rate, these interfaces are not directly representable in the core sequential logic. Asynchronous pulse logic offers an alternative design method for high speed interfaces with similar performance, simpler circuitry and without resorting to high-power logic cells such as emitter coupled logic. Formal and practical considerations for constructing high-speed interfaces are described. Gate designs and timing information for example cases are presented. These cases suggest that 80% improvements on rate compared traditional clocked logic are possible.

ACM Reference Format:
Merritt Miller, Carrie Segal, David Mc Carthy, Aditya Dalakoti, Prashansa Mukim, and Forrest Brewer. 2018. Impolite High Speed Interfaces with Asynchronous Pulse Logic. In *Proceedings of 2018 Great Lakes Symposium onVLSI (GLSVLSI '18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3194554.3194592

1 Introduction

This paper presents a design methodology for creating asynchronous pulse circuits. The design style uses circuits that do not rely on hand-shakes, hence we call them impolite. This methodology offers a practical approach to energy-efficient, high-performance design. The approach not only allows interfaces to run faster than core logic but also easily enables interfaces between structurally different sources. Timed asynchronous automata can use high-frequency signals without the overhead of high-performance clock distribution and recovery. The methodology is an adaptation of existing asynchronous design approaches. It is targeted to the production of high-rate circuits embedded in slower fabrics. Circuits with such a rate disparity are not only common in data link and high-speed communication systems, but are especially useful in neuromorphic systems, distributed IOT sensor networks, and other applications with data primarily in bursts. This methodology promises to ease the design and extend the capability of such systems.

Interfacing between high and low speed clocking regimes is a common design problem in integrated circuits where serial interface and communication speeds frequently exceed core clock rates. Complex communication link circuits involve operating a metal wire near the rate where substantial amplitude loss and symbol timing jitter are serious issues. Correction of symbol-dependent timing errors are difficult to mitigate.

Synchronous circuits have limited capacity to handle timing issues. Specifically, time domain synchronization is difficult, and is commonly relegated to specialized blocks, such as DLLs, PLLs, and skew compensators, whose behaviors are outside of the synchronous domain.

The presented design implements asynchronous logic blocks with a style to handle the slow-parallel to fast-serial domain interface independent on the behavior of a PLL or timing circuit. Timed asynchronous logic increases tolerance of timing variance. Fully unconstrained asynchronous design has high design and verification complexity. In this work, a set of composition rules and a variety of pulse-logic gates are presented that allow a limited set of classical timing constraints to close both the low-speed and high-speed design behaviors.

Unlike existing asynchronous systems, the proposed "Impolite" scheme primarily uses feed-forward construction, meaning performance with picosecond timing resolution is possible. Feed-forward logic does come at a cost: the timing of the system needs to be verified as part of the construction procedure. One of the novel contributions of our methodology to the research community is the means to limit the complexity of the timing verification.

1.1 Related Asynchronous Design Paradigms

Existing asynchronous design paradigms fail to meet the intended performance requirements of interfaces operating at least 2X faster than core speed. In particular, classical feed-back based delay independent techniques are problematic. In the case of physically long transmission media (starting at the mm scale for multi-GHz signals) time of flight for the electromagnetic wave carrying the signals adds substantially to the system delay, reducing performance. On-die

GLSVLSI '18, May 23–25, 2018, Chicago, IL, USA

ACM ISBN 978-1-4503-5724-1/18/05…$15.00
https://doi.org/10.1145/3194554.3194592

scale structures (100μ scale) have substantial propagation delay and at lengths of 1mm have signal integrity issues. These considerations defeat design styles based on feed-back such as GasP[1] and Null Convention Logic[2].

Instead, the circuits presented here are a limited sub-class of self-resetting CMOS circuits – a design style that has a reset circuit assigned to small clusters of domino-like logic. SRCMOS circuits work in an inherently pulsed manner but have the down side that SRCMOS circuits require pulses to arrive nearly simultaneously for proper function[3]. Timing analysis for SRCMOS is presented in [4, 5]. SRCMOS circuits have seen application in asynchronous circuits in [6, 7] but use is commonly restricted to systems that have feed-back (polite handshakes) to confirm correct behavior.

2 On-Chip Signaling: Pulse VS. Edge

A key feature of this work is that pulses (as opposed to edge based signals) are used for communicating timing critical events. The one-at-a-time pulse model makes the event timing check the dual of the SRCMOS timing check, *pulses must not overlap* for correct behavior. Since the use of a pulse is critical for the presented methodology to be successful, we now make the case that signaling with self resetting buffers permits an event cycle rate as fast or faster than edge event rates.

Jitter and propagation behavior of pulses and edges are nearly identical for practical interconnect cases. Pulsed signals, with a single characteristic width, allow the use of pulse gates, such as in [6, 8–10] and similar to [1]. These gates are known to maintain stable, narrow pulse widths, whereas logic without feedback would accumulate edge-to-edge uncertainty, widening the minimum pulse width.

In edge-based signaling, a rising edge must be followed by a falling edge, requiring separate types of event detection. Since different devices are involved, systems relying on edge-based signaling have inherently higher sensitivity to process variance but have been used successfully (e.g.) [11].

Since edge-based communication has a theoretical advantage in both power and bandwidth, it is important to demonstrate that pulsed signaling does not come at a high cost relative to edges in practical on-chip design. A case study using a 130nm process node interconnect wire is used to demonstrate, because it is in this process node where wire dimensions became a limiting factor[12] in signaling rate.

2.1 Case Study 5mm wire 130nm process node

A 5mm wire in a metal layer of a 130nm process is used to compare pulsed and edge encoding for an event signal. For this metal a conductor thickness of .3μm and a inter-layer dielectric thickness of .3μm is typical. The wire width and spacing is chosen to optimize the cost function of $delay \times wire\ pitch$ giving a wire width of .55μ and a wire spacing of .38μ. This configuration gives a fringing capacitance of $100fF/mm$, a side coupling capacitance of $31fF/mm$, and, assuming a copper conductor, a resistance of $133\Omega/mm$.

With an inverter size of 26μ NMOS, minimal worst-case delay (with an even number of stages) occurs with 4 internal repeaters (5, 1mm long wire segments). The single-stage worst-case delay time constant in this configuration is 68.4ps.

Table 1: Propagation times for edge and pulse.

	(100ps rise) Edge		(165ps width) Pulse	
Coupling noise	Average	σ	Average	σ
No coupling	251.3 ± 1.0	8.6 ± 0.7	239.5 ± 0.9	7.7 ± 0.7
Fastest	215.6 ± 0.9	7.5 ± 0.7	208.5 ± 0.8	6.9 ± 0.5
Slowest	285.9 ± 1.1	9.5 ± 0.7	276.7 ± 1.0	8.7 ± 0.6

All values shown with 95% confidence interval marked.

2.2 Edge-communicated signal

Arrival jitter is approximated by Monte-Carlo simulation consisting of 1k runs, sufficient to gain 95% confidence values for most measures. Process variation is taken from a vendor (IBM) model for the 130nm process. Power variation is estimated to have a global, correlated variation of 30mV power to ground, modeling power regulator noise and a local, uncorrelated variation of 30mV is added to each power and ground node, modeling IR noise internal to the IC.

When there is a single fast edge (<100ps) of a slow signal ($f < 200MHz$) the average propagation time is projected to be 251.3 ± 1.0ps, close to the value that $\#stages \times stage\ delay \times ln(\frac{1}{2})$ predicts. The sample standard deviation (σ) of the arrival time in this experiment is 8.6 ± .7ps. Assuming a Gaussian distribution, a 5σ interval gives a delay between 203.5ps to 298.5ps for a 5mm wire.

Due to the high coupling capacitance (~38% of the total) the impact of neighbor wires must be considered. The worst-case jitter occurs when both neighbors are correlated in the same or opposing direction as the main signal. Same direction switching has delay 215.6 ± .9ps with a sample σ of 7.5 ± .7ps. The 5σ fast arrival time under these circumstances is 173ps. Opposite direction switching delay is 285.9 ± 1.1ps with $\sigma \approx 9.5 \pm .7ps$, giving a 5$\sigma$ slow case of 338ps. This is a range of 165ps of environmental jitter that an ideal latching strategy cannot compensate for. A more common single clock phasing strategy would need a flip-flop with $\tau_{su} + \tau_{clk-Q} < 162ps$ to run at 2GHz.

2.3 Pulse-communicated signal

Table 1 compares the propagation times for edges and pulses. Using 165ps as the full-width, half-maximum measure of the pulse, propagation times are similar to edges. The minimum pulse period is twice the pulse width – 330ps, marginally faster than the non-skew-compensated rate of edges.

Using self-resetting gates to create a regenerative buffer, improves performance. Self-resetting gates protect pulse widths, and thus jitter cannot destroy a pulse. For systems using self-resetting buffers, two conditions must be met: First, the pulse width and its reset time must be obeyed. Second, the pulses must arrive in order. The pulse width is set locally within the buffer since it is self-timed. Native self-resetting pulse width in 130nm was determined to be 64ps with $\sigma \approx 4ps$, giving 84ps for a maximum pulse width, and 168ps for a safe pulse interval. Jitter for the self-timed buffer case is slightly higher than for the inverter buffer case at an estimated total jitter (5σ+pattern dependence) of 188ps. This extra jitter, as compared to the inverter buffer case, is due to the extra logic required in a self-resetting buffer. This extra logic extends

maximum propagation time, in this case to 405*ps*. The arrival order uncertainty limit of 188*ps* is the dominant of the two restrictions.

2.4 Pulse vs. edge for marking an event

The relative propagation timing of pulses and edges are very similar, even when the width of the pulse is small. Jitter dominates gain/bandwidth in limiting performance, and a pulse because it is atomic and unambiguous in its arrival, need not be correlated to any other signal. Thus pulse-based event detection can operate as fast or faster than edge event detection correlated to another signal or state.

Table 2: Event detection times for various signaling methods

Method	Period	Notes
Handshake	597p	Interleaving forward & backward wires (minimizes worst-case propagation)
Clocked Edge	338p	No Phase compensation
Edge with DLL or PLL	223p	Requires DLL/PLL of ~20mW (assumes 15ps RMS jitter ≈ -112dBc/Hz phase noise)
Pulse	*330p*	*Not using self-reseting buffers*
Pulse	*188p*	*With self-reseting buffers*

Table 2 shows a number of cases, and the associated minimum period of operation for a BER of 6×10^{-7} (corresponding to $\pm 5\sigma$ variance). For the purposes of comparison, the minimum latch timing is left out of the presented period. In a clocked system, the latch sample interval adds 50-200ps. In asynchronous systems, both pulsed and handshake, latch sampling time need not add to the minimum period, as both techniques have sampling times built into their respective operating mechanisms.

3 Composition Rules

The following construction techniques give rise to a class of systems that are delay sensitive, and are easier to verify than a system completely free in timing specification.

The lack of delay insensitivity means timing verification is required for behavioral closure. The construction only admits designs that can be verified with static timing. Static timing analysis is a common part of verifying clocked systems and numerous extensions, such as yield estimation, have been formulated[13]. This method uses a constraint system with the same analytic methodology, similar to clocked SRCMOS[5].

3.1 Basic Rules

There are a handful of rules to enable simple, closed examination of these event-driven circuits. The circuit can be specified hierarchically, where each level of the hierarchy obeys these rules. For the purposes of discussion a *Block* is a functional component at one level of the hierarchy, while a *Gate* is specifically at the lowest level. The rules for blocks are as follows

(1) There is a strong typing of *Event* and *Data* signals. A signal will be of exactly one of those two classes.
(2) Specification, and verification occurs within a time *Frame*. The time frame is marked by a start event and a finish event.
(3) Within a *Frame*, Data values will update at most once.
(4) For any *Block*, and most *Gates* only a single Event can occur at a time.
(5) When an event occurs, Data cannot be in transition.
(6) The *Consensus* gate is exempt from rule # 4. The behavior of this gate is handled as a special case.

The Frame requirement from these rules enforces a data-path check that is similar to traditional clocked logic. The model fits well with a system matching high-rate and low-rate regimes – the low-rate timing can easily provide the Frame.

3.2 Timing Check Complexity

For this acyclic-within-frame logic, the complexity of the static timing check is simply bound by the number of delays along the path and thus scales as $O(n)$ given n delay bearing nodes including latches. For a circuit with m unconstrained events $m!$ constraints are needed in worst case although this is practically limited by gate fan-in. The basic rules are placed on gates to ensure predictable (inertial) delays apply and have complexity $O(m^2 \cdot n)$. This has the consequence that some kinds of circuits fall outside of the constraints.

However, circuits including near-miss arbiters and all circuits necessary for communication links are permitted.

3.3 Logical Constraints on Construction

Signals are partitioned into two classes or types: Events and Data. The Event class serves to mark time, and is analogous to a clock. The state of a gate can only change with an event, communicated by a pulse of fixed width. The other class, Data informs state updates in the presence of an event. Data is communicated as traditional digital levels.

This leads to the first constraint, *the value of a Data signal must be stable between the setup and hold time* for each gate relative to the arrival of an updating event.

The second constraint is that a block interface may have only *one active event at a time*. One-at-a-time events prevent complex timing issues from arising within the relative timing check. In order for two events to be processed simultaneously they must act on separable parts of the system, or be passed into a Consensus gate for processing.

For two events to inform interacting parts of the system they must have a fixed timing relation. The relationship can be confirmed by verification, and event re-timing can be part of the construction. The designer should describe any two-event behavior as a set of one-at-a-time actions. Fair arbitration is not a feature of the methodology, instead the designer must describe the arbitration method. This can be aided by a Separating Arbiter gate, that while it cannot perform ideal arbitration, it can separate vanishingly-close (1ps) pulse interactions.

Because of these interaction rules the timing can be checked in $O(m^2 n)$ time complexity, involving two different timing checks limited by this complexity class. The timing checks are of $Event \rightarrow Event$ relationships, and $Event \rightarrow Data \rightarrow Event$ relationships,

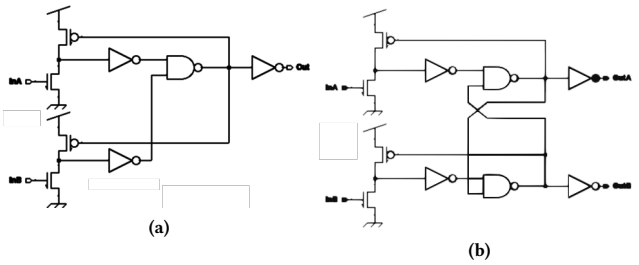


Figure 1: (a) Pulse Consensus gate. Gate fires output out once at least one pulse arrives on each input (since the last firing). (b) Separating Arbiter gate. Gate recreates input pulse sequence with restored separation.

since no more complex relation can exist. The Event simultaneity check needs to be performed at each gate to confirm that no two events arrive simultaneously; This allows the event-as-a-clock equivalent model. Then data timing is checked on $Event \rightarrow Data \rightarrow Event$ sequences, similar to a clocked system where the sequence is $Clock \rightarrow Data \rightarrow Clock$. Since events are one-at-a-time, event-data and event-event checks are orthogonal.

In practice, the timing check is further simplified by the limited number of events used and the limited fan-in and fan-out of practical designs.

3.4 Event timing

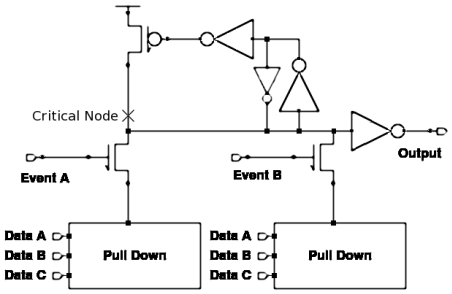
The one-at-a-time pulse model makes the event timing check the dual of the SRCMOS timing check, *pulses must not overlap* for correct behavior. Pulse Gating is done with a self-resetting gate structure and data latching utilizes a Set-Reset latch style. The pulse arriving at the gate serves as the sampling aperture for the pull-down network; this sets the data hold window to the actual pulse width of an event. The minimum pulse-width is thus set by the need to reliably sample, and is a trade-off with the complexity of the pull-down network.

There is a special case to the non-overlap verification: the Arbiter circuit. An arbiter circuit is allowed, logically, to have two events occur at its inputs as long as the events are of known order and avoid exact overlap. This non-overlap constraint is of practical importance, as arbitration circuits near meta-stability have ill-defined behavior and can take an infinite time to resolve[14]. In the contexts of these checks, arbitration is reduced to describing near-miss cases.

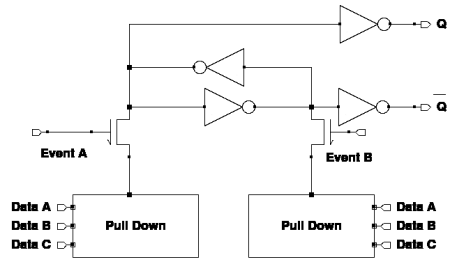
4 Gate construction

The strict classification of signals is crucial for timing analysis and simplifies design of the gates. This structure allows a gate to act on a given event given a set of data guard values since the data is known to be stable on event arrival. Logic functions are incorporated into the front-ends of Latches and pulse Gates, giving fast, small designs.

There are two classes of gate in this construction paradigm, pulse *Gates* and *Latches.* Pulse Gates, Figure 2a, have a pulsed output, and thus are used to create Events, while Latches, Figure 2b, have a level output creating Data signals. Pulse Gates use a self-resetting logic to (re)-create the event pulse.



(a) The pulse-gate, outputs a pulse given a condition



(b) Pulse-Triggered SR Data-Latch - outputs a data level that changes on a pulse given a condition

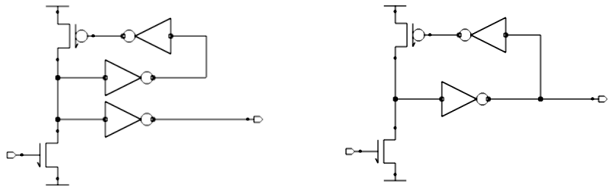
Figure 2: Pulse Gates and Latches

4.1 Pull-down network timing

The structure of the pull-down network, combined with the typing rules for Events and Data, create a timing constraint set for each pull-down network. The behavior of these networks are similar for pulse gates and latches, and the analysis holds for both. Correct functioning of the pull-down network determines the values for the timing constraints in the composition rules (Section 3). The prohibition on event overlap ensures that the action due to any event is not preempted by another event, a critical requirement for timing verification to not be forced to check all event combinations. Consider the timing of as SR latch, like the one shown in Figure 2b. Event A triggers setting this latch, while event B triggers reset. In both cases, the action is contingent on the logic of Data A, B, and C encoded into the respective pull-down networks. Electrically, the pull-down network is assumed conducting or non-conducting when the associated event pulse arrives. In the case of the SR latch, the set condition must be stable for the set pulse and the reset condition must be stable for the reset pulse.

4.2 Drive and Feed-back network

The characteristic pulse width of a self-resetting gate is set by the propagation delay through the feedback path. The output amplitude is dependent on both the output driver and its load. Pulse detection requires that the delay of the feedback path must match or exceed the amount prescribed by driver and load environment.



(a) Load isolated feed-back gate. (b) Load sensitive feed-back gate.

Figure 3: Two self-reset feed-back network options.

The feed-back network for a pulse gate can either be connected or isolated from its load as shown in Figure 3. The isolated-load model (Figure 3a) is faster than the load sensitive feedback and has more reliable timing. This feed-back network is well suited when the gate load is either constant, so that a correct pulse width can be selected, or for cases with small fan-out where pulse attenuation is not a concern. Load sensitive feedback (Figure 3b) is useful in a production environment where more universal cells are desired. The load sensitive gate's pulse-width and timing are derived from an estimate of pulse detectability. For reasonable levels of output loading, the feedback increases driver on-time until the local detectability threshold is met. This model keeps the impolite fire-and-forget model, by approximating detectability, rather than using a handshake. To illustrate the difference, we consider two different cases that arose in real designs executed in the 130nm process node. First with $7fF/\mu_{driver}$ of load capacitance – a typical value for timing-critical, local wires. In this case, the two feedback networks perform similarly. In the second a load of $50fF/\mu_{driver}$ represents a typical drive configuration for moderate-distance interconnect or a heavily loaded reset signal. In this case, load-sensitive feedback produces a slower, longer, pulse. This results in a pulse that is nearly 4x as detectable.

4.3 Two-Pulse Gates

Gates where two pulses arrive with the potential of overlap are special cases. There are two types of these gates: The *Consensus* gate, which produces an event after seeing two events is shown in Figure 1a and the *Arbiter*, shown in Figure 1b can resolve pulses that are near complete event overlap (<5ps timing difference).

The Charlie diagram, a visualization of output timing vs input timing[15, 16], for these two gates is shown in Fig 4. It is important to note the unstable behavior of the arbiter circuit near complete event overlap (<5ps timing difference). This establishes the limiting constraint for event-event timing checks. Two architectural solutions exist for cases when operating in this regime. First, is to insert arbitration up-stream to correct order. Second is to design a system that is tolerant of this hard-to-arbitrate condition, for example creating a parallel data-path dependent on the consensus gate that will time-out an ambiguous arbitration.

4.4 Pulse Gate Timing Stability

Pulse gates have a higher timing stability when compared to standard CMOS gates. This phenomenon can be be seen in the high timing stability of ring oscillators made form pulse gates[17]. The

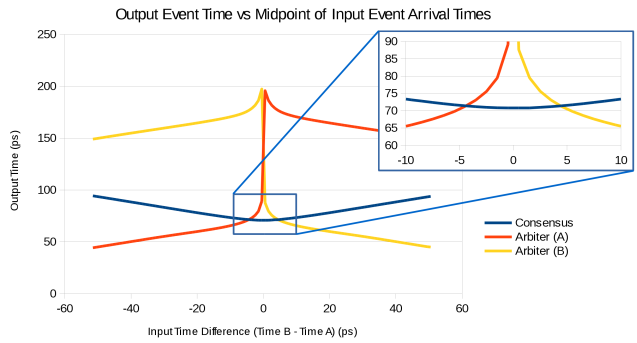


Figure 4: Consensus and Arbiter gates

stability of the timing may be due to lowered noise in MOS transistors for inputs in some state transitions[18]. In the case of pulse logic decisions are made by NMOS transistors turning on for all decision cases. To confirm the applicability of these results to the case of logic cells, timing is confirmed for the well-characterized 130nm process node, using noise models that take bias-condition into account. Figure 5 shows the timing stability of post-layout performance of pulse logic as compared to foundry-provided cells – in this case a buffer cell for driving a long wire.

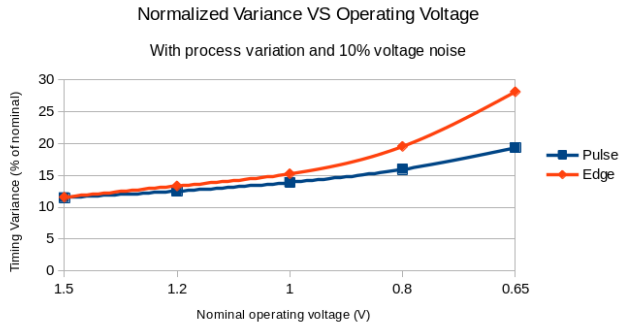


Figure 5: Variance of Pulse Gates, CMOS gates. Pulse gates have much lower variance at low operating voltage

4.5 Pulse Gate Library

Table 3 shows timing from a number of pulse gates. Computed from post-layout extracted designs. The gate set is an equivalent of a standard cell set for pulse-logic, and is used to build the designs of Section 5.The pull-down networks are sized for equal on current. The gate measured by simulation is triggered by a gate with similar drive (Fanout of 1) and loaded with a 4x pulse-repeater load (Fanout of 4, FO4). The decision threshold for measurement is set at VDD/3, an approximation of the pulse-gate decision threshold.

5 Data Link Performance Estimation

A hypothetical 4-bit serializer(SER) and deserializer(DES) are analyzed to demonstrate the verification procedure and construction

Table 3: Pulse Gate Timing in 130nm process

Gate	FO4 Delay	Delay Std. Dev.
Pulse Repeater	29.9ps	2.2ps
Or of two pulses	31.9ps	2.7ps
Pulse Single-condition	31.5ps	2.7ps
Pulse 2-Condition AND	29.4ps	2.6ps
Pulse 2-Condition OR	29.1ps	2.3ps
Or of 2 pulses, w/conditions	33.3ps	3.5ps
SR Latch	37.0ps	3.3ps

methodology For a high-speed interface. The 2-line encoding of [8, 9] is used. The SER uses 4 delayed copies of the data valid signal, which need not be locked to any other circuit. The DES produces 1 data vector per packet and a single data valid pulse. Given these constraints the timing results computed are shown in Table 4.

Table 4: Performance Estimates for 4-bit SER/DES system

	130nm	65nm	45nm
Trigger Time (Pulse Width)	93.8ps	52.1ps	23.0ps
Max Bit-Bit Time	187.6ps	104.2ps	46.0ps
Minimum DDR Clock Period	592MHz	960MHz	2.17GHz
Max Data Rate	4.74Gbps	7.68Gbps	17.4Gbps
Comparable Max Data Rate	2Gbps[19]	4.8Gbps[20]	10.5Gbps[21] [1]

5.1 Static Timing Analysis Complexity

For the serializer(SER) and deserializer(DES) from Section 5 there are 9 blocks in the DES and 13 blocks in the SER. For the SER there are 16 events, while the DES has 10 events. The composition rules require checking events at each block. A simplistic algorithm can verify timing with only 4228 path sums. This check not only allows verification with much less computation than a mixed-signal simulation, but also allows designers to use asymmetric or varying components without resorting to an exponential number of cases that need to be checked. The rules and verification become an enabling technology for systems checked in $O(m^2n)$ time, compared to free-for-all asynchronous systems bound by $O(m!n)$.

6 Conclusion

Since the "Impolite" methodology permits the exploration of a design space containing a much larger number of events, it is practical to consider what those sorts of systems could be developed to solve. The most obvious use is the embedding of simple mathematical functions in communications.

The Internet of things and the growing influence of practical artificial intelligence offers a platform where localized computational models can be of great benefit. The communications between these many diverse systems will require many channels variable, and burst data rates as well as unique light-weight interfaces. The logic family presented here shows promise for creating unique, and possibly asymmetric solutions that remain easy to verify.

Additionally, this composition style can implement Race Logic, which shows promise as a new computational model[22]. The required temporal functions (MIN, MAX, and COMPARE) are permissible using the composition rules discussed in Section 3.

In short, Impolite Asynchronous Pulse Logic enables verifiable construction of high-timing-performance circuits from small cells and simple composition rules. The fire-and-forget model, as well as timing rules enable static-logic style checking at decision rates typically reserved for mixed-signal style design. The resulting circuits maintain precise timing and have good idle behavior.

[1] Design uses even and odd channels, and when combined are limited at 21Gbps

References

[1] I. Sutherland and S. Fairbanks. Gasp: a minimal fifo control. In *ASYNC 2001*, pages 46–53, 2001.

[2] K.M. Fant and S.A. Brandt. Null convention logictm: a complete and consistent logic for asynchronous digital circuit synthesis. In *ASAP 1996*, pages 261–273, Aug 1996.

[3] Gunok Jung, V.A. Sundarajan, and G.E. Sobelman. A robust self-resetting cmos 32-bit parallel adder. In *ISCAS 2002*.

[4] Ayoob E Dooply and Kenneth Y Yun. Optimal clocking and enhanced testability for high-performance self-resetting domino pipelines. In *Advanced Research in VLSI, 1999. Proceedings. 20th Anniversary Conference on*, pages 200–214. IEEE, 1999.

[5] Vinod Narayanan, Barbara A Chappell, and Bruce M Fleischer. Static timing analysis for self resetting circuits. In *ICCAD 1996*.

[6] Mika Nyström and Alain J Martin. *Asynchronous pulse logic*. Springer, 2002.

[7] T. Säntti and J. Isoaho. Modified srcmos cell for high-throughput wave-pipelined arithmetic units. In *ISCAS 2001*, volume 4, pages 194–197 vol. 4, May 2001.

[8] Brian D Winters and Mark R Greenstreet. A negative-overhead, self-timed pipeline. In *ASYNC 2002*, pages 37–46. IEEE, 2002.

[9] Mark R Greenstreet and Jihong Ren. Surfing interconnect. In *ASYNC 2006*.

[10] Merritt Miller, Greg Hoover, and Forrest Brewer. Pulse-mode link for robust, high speed communications. In *ISCAS 2008*.

[11] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[12] Ron Ho, Kenneth W Mai, and Mark A Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, 2001.

[13] Anirudh Devgan and Chandramouli Kashyap. Block-based static timing analysis with uncertainty. In *ICCAD 2003*, page 607. IEEE Computer Society, 2003.

[14] Thomas J Chaney and Charles E Molnar. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on computers*, 22(4):421–422, 1973.

[15] Jo C Ebergen, Scott Fairbanks, and Ivan E Sutherland. Predicting performance of micropipelines using charlie diagrams. In *ASYNC 1998*.

[16] Anthony Winstanley and Mark Greenstreet. Temporal properties of self-timed rings. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pages 140–154. Springer, 2001.

[17] Aditya Dalakoti, Merritt Miller, and Forrest Brewer. Pulse ring oscillator tuning via pulse dynamics. In *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pages 469–472. IEEE, 2017.

[18] I Bloom and Y Nemirovsky. 1/f noise reduction of metal-oxide-semiconductor transistors by cycling from inversion to accumulation. *Applied Physics Letters*, 58(15):1664–1666, 1991.

[19] Rashed Zafar Bhatti, Monty Denneau, and Jeff Draper. 2 gbps serdes design based on ibm cu-11 (130nm) standard cell technology. In *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, pages 198–203. ACM, 2006.

[20] Peng Wang, Ziqiang Wang, Chun Zhang, and Zhihua Wang. Data lane design for transmitter of 4.8 gbps serdes in 65nm cmos. In *Electron Devices and Solid-State Circuits (EDSSC), 2014 IEEE International Conference on*, pages 1–2. IEEE, 2014.

[21] Jonathan E Proesel and Timothy O Dickson. A 20-gb/s, 0.66-pj/bit serial receiver with 2-stage continuous-time linear equalizer and 1-tap decision feedback equalizer in 45nm soi cmos. In *VLSI Circuits (VLSIC), 2011 Symposium on*, pages 206–207. IEEE, 2011.

[22] Advait Madhavan, Timothy Sherwood, and Dmitri Strukov. Race Logic: A hardware acceleration for dynamic programming algorithms. *ISCA 2014*, pages 517–528, 2014.