Comparison of channel protocols for low latency, low energy communication over transmission lines

Shomit Das Kenneth S. Stevens
University of Utah

Abstract— **The eminence of communication costs over computation costs in current systems-on-chip (SoCs) has led to a change in the assumptions and methodology of the deep submicron design process. The emergence of non-traditional global signaling techniques such as optical interconnects, wireless communication, and transmission lines adds further complexity to the design exercise. There is a pressing need for a better understanding of communication overheads, especially in long links. Previous research has been performed that mathematically models various synchronous and asynchronous pipeline protocols for long diffusive wires. The work presented in this paper adapts these throughput, latency, and energy models for fast, low power communication over on-chip transmission line interconnects. Using common parameters and metrics to characterize the channel protocols enables analogous comparisons among them. First order models are created and populated with SPICE values to collate performance metrics for global channels. The paper then compares the behavior of transmission lines and diffusive wires under these signaling protocols. Finally, the effect of wire length on channel performance is studied.**

I. Introduction

A. Scaling

One of the major consequences of physical scaling and increased performance targets is the reversal of the cost of communication and computation. To put the magnitude of these effects in perspective, consider the delay difference scaling has produced for a transistor to calculate a logic function compared to the propagation of a signal over 1mm of interconnect. One can compare 10 scaling generations by examining the 1.0m and 35nm technology nodes. Native transistor delay (τ) in the 1.0μm process is 30ps, while propagation delay down a 1mm wire is a small fraction of the transistor delay at 1ps. This 30:1 delay relationship has been dramatically reversed in the 35nm node. Transistor delay is projected to be 1ps while wire delay for the same 1mm wire balloons to 250ps. This is nearly four orders of magnitude difference. The energy relation is equally stunning. In the 35nm process technology, energy expenditures necessary to propagate a 64b signal 2mm is approximately equal to performing a 64b floating point operation [1]. One of the major changes in design methodology brought about by scaling is the addition of pipeline stages in the interconnect. This communication pipelining results in some circuit and architecture advantages and overheads. Various communication methodologies have differing local and architectural performance characteristics. It is useful to model pipelines using various synchronous and asynchronous protocols and compare their power-performance relationship [2].

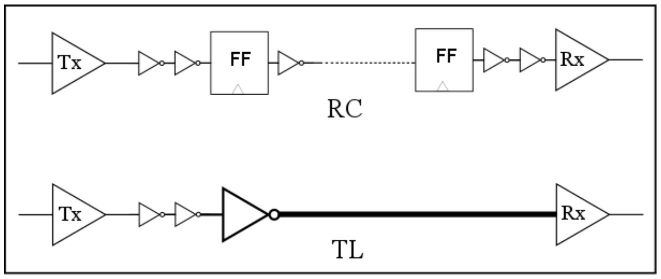


Fig. 1: No repeaters in TL channels

B. Transmission Lines vs Diffusive Wires

Transmission line interconnects transmit data at high bandwidths with low latency while consuming much less energy as compared to diffusive RC wires. There are fundamental differences in the technique of signal propagation between RC and RLC interconnects. Diffusive wires in modern process nodes have poor signal carrying properties. They have high resistance, which escalates the latency and diminishes the available bandwidth. Lesser supported bandwidth means that pipeline stages need to be added along the line to maintain throughput. Also, repeaters need to be introduced at regular intervals to limit signal distortion and maintain a linear delay-length relationship. This increases the power drawn by the wires, especially for long distances. On the other hand, transmission lines can support higher frequencies, and have better characteristics when they remain free of discontinuities, including vias. As seen in Fig. 1, the absence of signal boosting repeaters and pipeline stages are an important change while utilizing TL signaling links over RC wires. Communication protocols have been modeled and evaluated extensively for long RC wires. In this work, we update the models to apply to transmission line channels and report the effects of using various synchronous and asynchronous communication protocols on these disruptive interconnects.

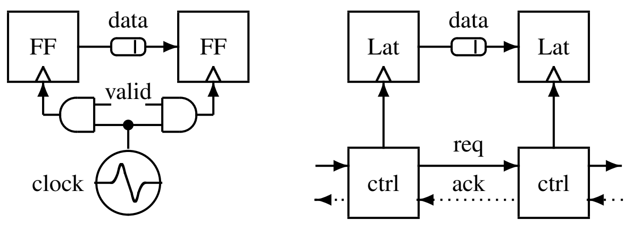
II. Background

A. Handshaking

Clock tree networks distributing a low skew clock for a global synchronous system are inefficient in terms of area and power consumption. Given the heterogenous and multi-frequency nature of modern SoCs, imposing a single timing reference on the entire chip leaves a lot of performance on the table. On the other hand, there are multiple benefits of using asynchronous protocols for communication. Self-timed blocks can be designed to operate at optimal frequency and power without the shackles of a global clock. The inherent modularity of asynchronous circuit blocks enables higher design reuse, faster time to market, and proliferance of custom IP blocks. Also, asynchronous designs are more robust due to their adaptive nature. However, there is a bandwidth penalty due to the overhead introduced by handshaking. In synchronous design, data is normally transmitted every clock cycle between latches or flops. In asynchronous design, the clock signal is replaced by handshaking between neighboring latches to indicate that data is ready to be transmitted. Data can be represented as tokens flowing in conjunction with handshake channels, transparent to combinatorial circuits. This data-flow abstraction separates the circuit behavior from its implementation details. Therefore, there may be multiple ways to implement each handshake protocol.

B. Protocols

Data transmission in asynchronous domains is based on handshake protocols (`req` and `ack`), allowing transactions to be reactive and only when data is valid. Single-rail (bundled data) protocols, shown in Fig. 2, have similar implementation to clocked designs but employ latches rather than flip-flops [3]. The other main family of asynchronous protocols are the 1-of-N codes which encode data validity within data wires but use a shared acknowledgment. Handshaking ensures that activity on the wires takes place only when data is valid, which is similar to clock gating, but is implicit to the protocol operation. The inherent energy efficiency of asynchronous protocols comes at the added cost of the back-propagating `ack` signal that places a limit on the communication throughput.



(a) Synchronous (b) Asynchronous

Fig. 2: Synchronous and asynchronous topologies

Similar to the setup-and-hold restraints in clocked methodologies, bundled data protocols need to ensure sufficient time separation between the control and data signals [4]. Instead of the time-step restriction in synchronous communication, the `ack` signal specifies the readiness of the receiving side to accept new data.

Communication costs are overshadowed by computation costs in local data paths. Short wires imply small delays, and control and data operations are almost concurrent. However, in global wires, the overhead of communication cannot be concealed. Intuitively, handshaking is inefficient for long wires due to the effect of added control wires between the sender and the receiver. Throughput, latency, and energy in asynchronous communication are more expensive relative to clocked communication, especially in complex systems where clock distribution costs are amortized over a large number of blocks. Conventional wisdom suggests that data communication is the most *inefficient* application for asynchronous protocols and creates a worst-case comparison between synchronous and asynchronous implementations.

Das et.al. introduce a self-timed communication protocol called Source Asynchronous Signaling (SAS) for long wires [5]. This protocol makes the throughput agnostic to the wire delay. Therefore, data transmission is possible at rates comparable to synchronous systems, rather than at one-fourth the propagation time between the latches as in the case of a four-phase protocol. SAS replaces distributed pipeline stages along a long link with FIFOs at the sender and receiver. This is fundamentally different from traditional handshake techniques, and therefore requires extra modeling effort.

There are seven protocols studied in this work. Three synchronous ones – flopped (Clock_f) and latched (Clock_l) clocking, and source synchronous (SrcSync). The asynchronous family of protocols is represented by the 2-phase (BD2) and 4-phase (BD4) bundled data, dual rail delay insensitive design, and source asynchronous (SAS) methodology.

Name	Description	FO4 Value
V_c	Cross coupling variation of wire	0.4
V_p	Process variation	0.16
V_{vt}	Voltage & temperature variation	0.12
T_{cq}	Delay from clock to output	1.8
T_{sul}	Latch setup time	0.48
T_{suf}	Flop setup time	0.96
T_{skj}	Clock skew and jitter	1
T_h	Hold time	0.25
T_{dqf}	Flop data-to-output delay	2.8
T_{dql}	Latch data-to-output delay	1.4
T_{pw}	Minimum pulse width that can be propagated through a critical distance	2.2
T_c	TL Channel Delay	5.0 – 6.3
T_{c+}	$T_c(1+\frac{V_c}{4})$	5.5 – 7.0
T_{c-}	$T_c(1-\frac{V_c}{4})$	4.5 – 5.7
T_{fsm+}	Controller delay per phase	2.55 – 6.64
T_{fsm-}	$T_{fsm+}(1-\frac{V_p}{2})$	2.34 – 6.10
T_{lat+}	Forward latency of async controller	1.18 – 3.95
T_{cfifo+}	Cycle time of 0-bit tree FIFO	12.12
$T_{latfifo+}$	Forward latency of 32-bit tree FIFO	4.33
E_{ctl}	Controller energy per phase per bit	0.072 – 0.90
E_{tl}	Energy of a TL link	1.32 – 2.12
E_{serdes}	Energy of a SERDES apparatus	2.15
E_{datsas}	Energy of a SAS data link	1.42
E_{ctlsas}	Energy of a SAS control logic	5.0
E_{rep}	Energy of a single repeater	1
E_{datf}	Data energy through flop	2.02
E_{datl}	Data energy through latch	1.78
E_{clkf}	Clock energy for flop	0.22
E_{clkl}	Clock energy for latch	0.2
E_{clkdef}	Clock energy for DETFF	0.42

TABLE I: Parameter variables and derivatives.

Name	Equation	Value
A_w	Activity factor of bus wires	0.18
A_b	Activity factor of bus	0.05
O_{wd}	Control wire delay optimization	0.85
O_{wa}	Control wire area optimization	1.85
W_b	Width of bus	32
$E_{clktree}$	Clock tree energy per transition	16.84
P_{dc}	% of clock tree energy for communication	10%

TABLE II: Parameter variables and derivatives.

C. Metrics

We calculate throughput, latency, and energy cost of implementing clocked and unclocked communication protocols over a 5000μm long transmission line in a 65-nm technology. The various parameters used in our models have been obtained using circuit simulations with HSPICE. Closed-form expressions are developed for these metrics in a similar fashion to those of diffusive wires in [2]. The effects of variation are modeled in these parameters, as are the overheads attributed to the bundling-data constraints. The serialization costs of using transmission lines are also included. The high frequency, speed-of-light signal propagation nature of transmission lines makes them a high bandwidth, low latency medium. The absence of intermediate repeaters as well as bandwidth-maintaining pipe stages renders them a low energy cost solution as well.

D. Terminology

Tables I and II show the parameters and their associated values that are used in this paper. First-order effects are modeled, including coupling, process variation, voltage and temperature variations (V_c, V_p, and V_{vt} respectively).

Values in the right column of Table I are scalar delays. All delay and energy values are relative to the fan-out of 4 (FO4) of an inverter in the process node. The transmission line link delay (T_c) is specified for three different values – 3, 5, and 7 mm. These values were calculated using SPICE simulations for a fixed drive strength and frequency. The T_{fsm} parameter in Table I is based on the worst case cycle time of the asynchronous controller. The total cycle time is averaged between the four (two) phases of the four (two) phase asynchronous controller. Different protocols exhibit different cycle time overheads. For example, 2-phase NRZ protocols generally require more complicated control with larger delays compared to 4-phase protocols. The T_{lat} parameter represent the forward propagation latency of the handshake control signals in the asynchronous protocols. Intuitively this represents the delay associated with propagation of the incoming request signal to outgoing request signal. T_{cfifo+} and $T_{latfifo+}$ represent the cycle time and forward latency of the tree FIFO designs used for the implementation of the SAS protocol. The E_{ctl} parameter represents the average energy per phase per bit for the control logic for the asynchronous protocols. The energy spent per bit during a transfer over a transmission line is denoted by E_{tl}. A 4:1 serialization ratio is chosen for the data bus implementation. Consequently, the total number of transmission lines is $W_b/4$. The energy overhead added by the SERDES is represented by E_{serdes}. The energy consumed by the flop / latch to transfer data the critical distance of a communication link is represented as the parameter E_{dataf} / E_{datal}. The average energy consumed per transition by the flop / latch due to switching of the clock is represented as parameter E_{clkf} / E_{clkl}. Similar to control delays, different asynchronous protocols exhibit different energy requirements. For a fair comparison T_{fsm}, T_{lat}, and E_{ctl} have been characterized through simulation for the different protocols and are listed in Table III. The sizing and spacing of the control wires in the asynchronous protocols may be optimized differently than the data wires. The parameters O_{wd} allow a reduced control wire delay for a larger area O_{wa}.

The parameter A_b is the activity factor of the bus, or the percentage of clock cycles during which data is transmitted across the bus. Each flop is considered to be gated during idle cycles in the clocked protocols. A_w is the activity factor of data on the bus, or the probability that any bit will switch values for an average bus transaction.

Name	$\mathbf{T}_{fsm}$	$\mathbf{T}_{lat}$	$\mathbf{E}_{ctl}$
4-phase bundled data	4.28	3.45	0.075
2-phase bundled data	4.77	3.45	0.072
DI 1-of-2	6.64	1.18	0.90

TABLE III: Asynchronous control parameters from 65nm SPICE simulations of the designs

III. Models

A. Cycle Time

The minimum cycle time of a flop represented by equation C_{ff+} presented in Table IV depends on the pulse width of the minimum sized pulse that can be safely propagated along the transmission line, as well as the sum of communication delay T_{c+} and the setup time, skew, and jitter overheads $T_{su} + T_{skj} + T_{cq+}$. The clocked latch protocol imposes an overhead of $2T_{dql}$ to account for the alternate phase control of the latches. Parameter T_{fsm+} represents the cycle time of the controller divided by the number of phases. Typically, dual rail protocols use a return-to-zero implementation; therefore the cycle time depends on $4T_{fsm+}$. It is evident that 4-phase bundled data protocol performs almost twice as badly as the 2-phase version. As mentioned earlier, the bandwidth of a SAS protocol is independent of the wire delay. As long as the sender FIFO can accept tokens, the system can keep sending data along the channel. Since the wire latency of a transmission line interconnect is lesser than the cycle time of the FIFO used in the experiment, no wavepipelining is required. Therefore, C_{sas+} is limited only by the FIFO cycle time, T_{cfifo+}.

B. Latency

The latency of clocked protocols is the same as its cycle time and is represented in Table V. For asynchronous protocols, the delay depends on the forward latency of the control logic and the latency across the channel. Bundled data protocols have an added overhead imposed by the margin between the arrival of the data and control signals, T_{cdel}. A data token in a SAS protocol has to travel across the FIFO at the receiving end. Therefore, the total latency is the sum of the TL communication latency T_{c+} and the forward latency of the receiver FIFO $T_{latfifo+}$.

C. Energy

The energy consumed by the flop / latch to transfer data the critical distance of a communication link is represented as the parameter E_{datf} / E_{datl}. The average energy consumed

Name		equation	description
T_{ps}	$\leq$	$(T_{fsm+} + T_{cq+} + T_{c+} + T_{su}) - (T_{fsm+} + T_{c-} + T_{fsm-})$	Handshake to data margin
T_{cdel}	$\leq$	$(1 + \frac{V_p}{2})\max(0, T_{ps}) - (1 - \frac{V_p}{2})\max(0, T_{ps})$	Robust delay element size
C_{ff+}	$\geq$	$\max(2T_{pw}, (T_{c+} + T_{su} + T_{skj} + T_{cq+}))$	Clk_flop throughput
C_{l+}	$\geq$	$\max(2T_{pw}, T_{c+} + 2T_{dql})$	Clk_latch throughput
C_{di+}	$\leq$	$4T_{fsm+} + 2T_{c+} + 2O_{wd}T_{c+}$	2-rail and 1-of-4 DI
C_{bd2+}	$\leq$	$2T_{fsm+} + T_{c+} + T_{cdel} + O_{wd}T_{c+}$	2-phase bundled data cycle time
C_{bd4+}	$\leq$	$4T_{fsm+} + 2T_{c+} + T_{cdel} + 2O_{wd}T_{c+}$	4-phase bundled data cycle time
C_{ss+}	$\leq$	$\max(2T_{pw}, (T_{cq+} + T_{c+} + T_{su}),\ (T_{c+} + T_{fsm+}))$	src-sync cycle time
C_{sas+}	$\leq$	T_{cfifo+}	SAS

TABLE IV: Cycle time equations. Cycle time measured in FO4 delays.

Name		equation	description
L_{ff+}	$\geq$	$\max(2T_{pw}, (T_{c+} + T_{su} + T_{skj} + T_{cq+}))$	Clk_flop latency
L_{l+}	$\geq$	$\max(2T_{pw}, T_{c+} + 2T_{dql})$	Clk_latch latency
L_{di+}	$\leq$	$T_{lat+} + T_{c+}$	2-rail and 1-of-4 DI latency
L_{bd2+}	$\leq$	$T_{lat+} + T_{c+} + T_{cdel}$	2-phase bundled data latency
L_{bd4+}	$\leq$	$T_{lat+} + T_{c+} + T_{cdel}$	4-phase bundled data latency
L_{ss+}	$\leq$	$\max(2T_{pw}, (T_{cq+} + T_{c+} + T_{su}),\ T_{c+} + T_{fsm+})$	src-sync latency
L_{sas+}	$\leq$	$T_{latfifo+} + T_{c+}$	SAS latency

TABLE V: Latency per stage. Delay measured in FO4 delays.

Name	equation	description
E_{gf}	$\frac{1}{A_b}\frac{E_{clkf}}{12} + \frac{E_{clkf}}{4}$	Flop clock gate energy
E_{gl}	$\frac{1}{A_b}\frac{E_{clkl}}{12} + \frac{E_{clkl}}{4}$	Latch clock gate energy
E_{fe}	$\frac{1}{A_b}2P_{dc}E_{clktree} + \frac{W_b}{4}(2(E_{gf} + E_{clkf}) + A_w(E_{datf} + E_{tl} + E_{serdes}))$	Clk_flop energy
E_{le}	$\frac{1}{A_b}2P_{dc}E_{clktree} + \frac{W_b}{4}(4(E_{gl} + E_{clkl}) + A_w(2E_{datl} + E_{tl} + E_{serdes}))$	Clk_latch energy
E_{4e}	$4(E_{ctl} + E_{tl})$ $+ \frac{W_b}{4}(2E_{clkl} + A_w(E_{datl} + E_{tl} + E_{serdes}))$	4-phase bundled data energy
E_{2e}	$2(E_{ctl} + E_{tl})$ $+ \frac{W_b}{4}(E_{clkdef} + A_w(E_{datl} + E_{tl} + E_{serdes}))$	2-phase bundled data energy
E_{ss}	$2(E_{ctl} + E_{tl})$ $+ \frac{W_b}{4}(2E_{clkf} + A_w(E_{datl} + E_{tl} + E_{serdes}))$	src_sync energy
E_{di2}	$\frac{W_b}{4}(4E_{ctl} + 2E_{tl}) + 2E_{tl} + 2E_{serdes}$	DI 1-of-2 energy
E_{sas}	$(E_{ctlsas} + E_{tl} + \frac{W_b}{4}(E_{ctlsas} + A_w(E_{tl} + E_{serdes})))$	SAS energy

TABLE VI: Models for average energy per transaction per pipe stage.

per transition by the flop / latch due to switching of the clock is represented as parameter E_{clkf} / E_{clkl}. The average energy per transaction consumed by the clock distribution network is given as $\frac{1}{A_b}2P_{dc}E_{clktree}$ where $\frac{1}{A_b}$ accounts for the amount of energy consumed by the clock distribution network when the data bus is idle. The energy consumed by a flop is $\frac{W_b}{4}(2(E_{clkf} + E_{gf}) + A_w E_{dataf})$ and the energy consumed by the transmitter $\frac{W_b}{4}A_w(E_{tl} + E_{serdes})$ to drive the data. For the case of latch based communication the energy consumed by the latches is $\frac{W_b}{4}(4(E_{clkl} + E_{gl}) + A_w(2E_{datal}))$ due to two latches per pipeline stage. Note the 4 in the denominator to indicate a 4:1 serialization scheme. The energy consumed per transaction per pipeline stage is the sum of the controller energy and the energy consumed for data transfer through latches. The controller energy for the 4-phase protocol is equal to $4*(E_{ctl} + E_{tl})$ and $2*(E_{ctl} + E_{tl})$ for the 2-phase protocol. The energy consumed by the latch in a 4-phase protocol is calculated as $\frac{W_b}{4}(2E_{clkl} + A_w(E_{datal} + E_{tl} + E_{serdes}))$ and for a 2-phase protocol energy is calculated as $\frac{W_b}{4}(E_{clkdef} + A_w(E_{datal} + E_{tl} + E_{serdes}))$. Dual Rail protocol energy is calculated in similar fashion (Table VI). For the SAS protocol, the sum of the FIFO energies at the sender and the receiver is taken into account for calculations.

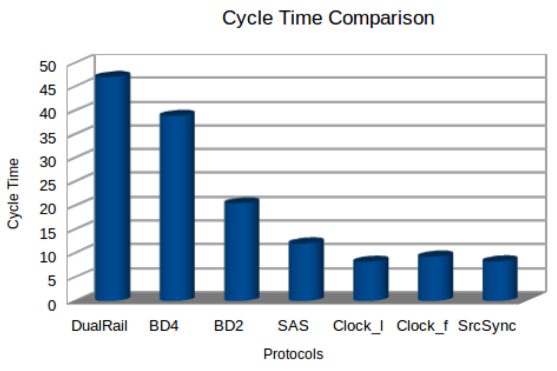


Fig. 3: Cycle Time for various protocols

IV. Comparisons

A. Protocols

We compare various pipeline protocols to determine the optimal choice for long distance communication over transmission lines. All numbers in the graphs are normalized to the fan-out of 4 (FO4) of a typical inverter.

It is clear from Fig. 3 that the asynchronous handshaking protocols are the least efficient for communication, with

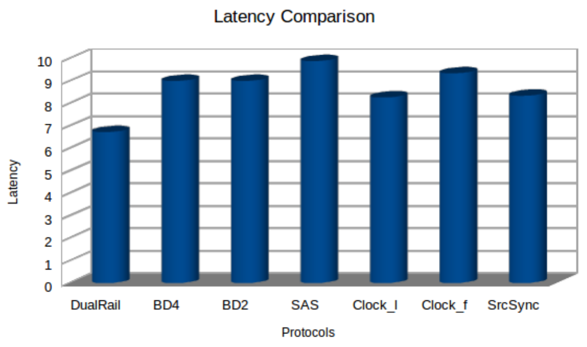


Fig. 4: Latency for various protocols

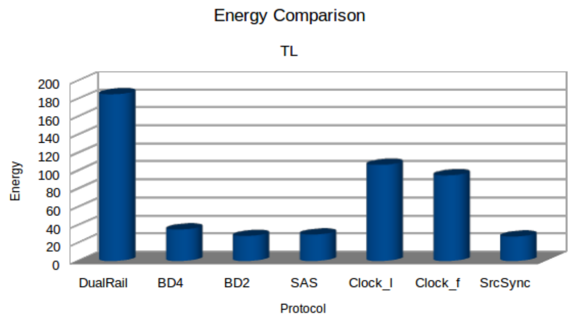


Fig. 5: Energy for various protocols

the 4-phase protocols being the worst. The 2-phase protocols perform better due to the reduction in the number of times the control signals travel across the channel. The clocked protocols are extremely efficient due to the simpler handshake logic and absence of multiple signal traversals per transmitted datum. SAS also performs better than other asynchronous methods because its throughput is wire-delay agnostic.

Fig. 4 indicates that the dual rail protocol provides the lowest latency of all the protocols. This is due to the lack of bundling data constraint overhead of the other protocols and the use of high speed domino logic in the datapath. The SAS protocol suffers from a slightly higher latency due to the presence of large FIFO buffers in its forward path, albeit the effect is slightly mitigated by using non-linear FIFO arrangements like the parallel FIFO, or in our case, the tree FIFO.

In a TL signaling regime, wire energy over long distances is a smaller fraction of the total communication link energy due to the higher frequency of operation coupled with the lack of signal repeaters. The asynchronous dual rail protocol has poor power characteristics due to its activity factor. We see in Fig. 5 that the bundled data protocols perform much better. In a clocked scheme, energy required to correctly distribute the clock tree with low skew is an energy intensive operation with a heavy switching profile. Also, energy is required in the gating logic even during idle phases, while asynchronous protocols provides ideal clock gating at no extra cost. SAS again benefits from the bundled data nature of its implementation.

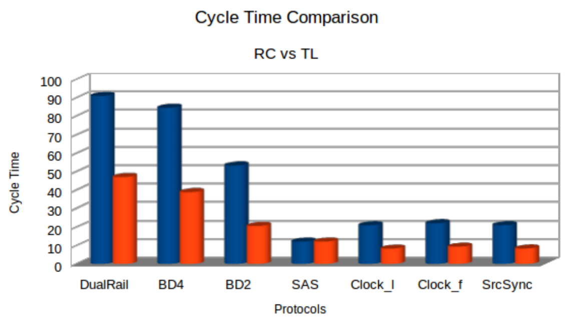


Fig. 6: Cycle Time RC vs TL

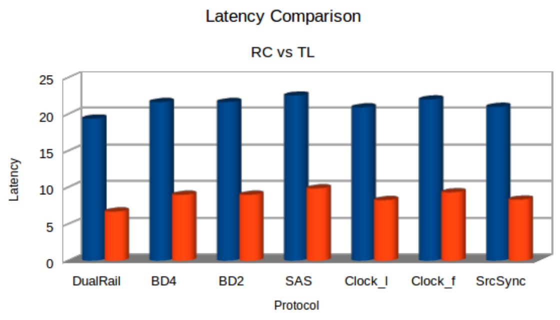


Fig. 7: Latency RC vs TL

B. Diffusive Wires

We now evaluate the effect of using low latency, low energy transmission lines for long range communication by comparing the metrics to an equivalent 5mm RC line consisting of a single pipeline stage, but augmented with repeaters for signal boosting [2]. Obviously, without intermediate pipeline stages, this experiment yields pessimistic numbers for the throughput metric. However, skipping intermediate pipeline stages means that the associated energy and latency overheads are drastcally reduced, especially for long wires. Fig. 6 shows over 60% improvement in bandwidth when migrating from an RC interconnect to a transmission line for a 2-phase bundled data protocol. Other protocols show similar trends. The exception is the SAS protocol which has a throughput that is independent of wire delays, and thus remains the same. Therefore, SAS protocol provides the same high throughput irrespective of the physical medium

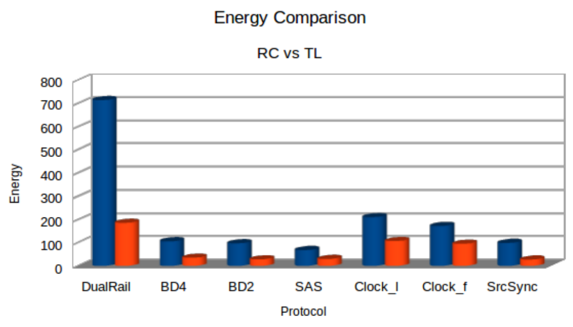


Fig. 8: Energy RC vs TL

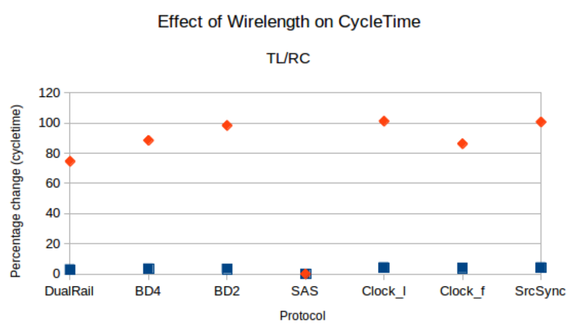
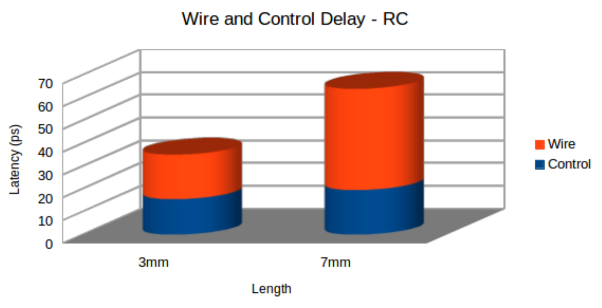


Fig. 9: Wirelength effect on cycle time



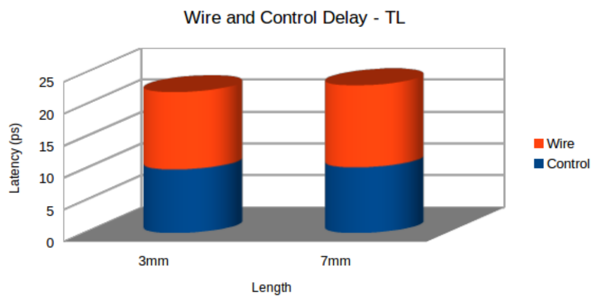


Fig. 10: Contribution of wire delay to cycle time

of signal propagation. The latency comparison in Fig. 7 shows the difference between the two channels. There is a higher variance in latency in the case of transmission lines which serves to display the difference between the various protocols. This effect gets masked by the higher contribution of wire delay to total latency in diffusive RC wires. Transmission lines make energy efficient interconnects. Fig. 8 shows how communication energy per bit is reduced by using transmission lines.

C. *Link Length*

So far, we have investigated the case of a single wire length (5mm). We now demonstrate the effect of changing the wire length from 3mm to 7mm on both transmission lines and RC wires for different protocols. In Fig. 9, we map the percentage change in the channel bandwidth when the length is increased from 3mm to 7mm for a single pipeline stage link. The blue squares reflect the TL data, while the orange kites show the corresponding RC wire data. It can be seen that if the line length is increased without adding intermediate pipeline stages, the throughput reduces drastically (up to 2×) for RC channels. The sole exception is

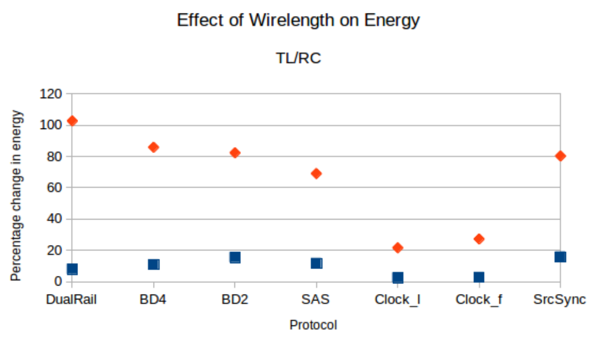


Fig. 11: Wirelength effect on energy

the SAS protocol, which as mentioned earlier, is agnostic to wire delay. On the other hand the transmission line system is a lot more impervious to this change (maximum 4.2%). This is due to the higher contribution of wire delay to cycle time in case of diffusive wire links. Fig. 10 shows the contribution of wire and control logic delay to the total cycle time for RC and TL links. It can be seen that the transmission line system is almost insensitive to the increase in wire delay. Therefore, the effect of longer wires on channel cycle time is minimal for transmission lines. Similar arguments can be made for the total link energy, as seen in Fig. 11. This means that scaling long links is much easier with transmission lines than RC wires as the overall effect on the communication channel performance is modest at best.

V. Conclusion

In this work, we have parameterized and compared first order models for cycle time, latency, and energy associated with communication protocols over a transmission line. Optimal methodologies for each metric have been identified. It is observed that the SAS protocol enjoys favorable throughput and energy characteristics, while retaining the modularity of an asynchronous methodology. Also, we observe that transmission lines have significantly better communication characteristics as compared to RC wires for all metrics considered in the study. Additionally, transmission lines can be used for short, mid-range, and long wires without much loss in performance, where as diffusive wires need pipeline stages to attain comparable throughput at the cost of extra latency and energy. For high performance, low energy, scalable data transfer, we recommend using SAS along with transmission lines.

References

[1] B. Dally, “Low-power architecture,” 2005.
[2] K. S. Stevens, P. Golani, and P. A. Beerel, “Energy and Performance Models for Synchronous and Asynchronous Communication,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 369–392, March 2011.
[3] I. E. Sutherland, “Micropipelines,” *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, June 1989, turing Award Paper.
[4] C. L. Seitz, “System timing,” in *Introduction to VLSI Systems*. Addison Wesley, 1980, ch. 7.
[5] S. Das, V. Vij, and K. S. Stevens, “Sas: Source asynchronous signaling protocol for asynchronous handshake communication free from wire delay overhead,” in *Proceedings of the 2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, 2013.