Energy and Performance Models for Synchronous and Asynchronous Communication

Kenneth S. Stevens, *Senior Member, IEEE*, Pankaj Golani, and Peter A. Beerel

***Abstract*—Communication costs, which have the potential to throttle design performance as scaling continues, are mathematically modeled and compared for various pipeline methodologies. First-order models are created for common pipeline protocols, including clocked flopped, clocked time-borrowing latch, asynchronous two-phase, four-phase, delay-insensitive, single-track, and source synchronous. The models are parameterized for throughput, energy, and bandwidth. The models share common parameters for different pipeline protocols and implementations to enable a fair apple-to-apple comparison. The accuracy of the models are demonstrated for complete implementations of a subset of the protocols by applying 65-nm process simulated parameter values against the SPICE simulation of full pipeline implementations. One can determine when asynchronous communication is superior at the physical level to synchronous communication in terms of energy for a given bandwidth by applying actual or expected values of the parameters to various design targets. Comparisons between protocols at fixed targets also allow designers to understand tradeoffs between implementations that have a varying process, timing, and design requirements.**

***Index Terms*—Asynchronous, bundled-data, bundling data constraint, communication bandwidth, delay-insensitive, single-track.**

I. Introduction

A NUMBER of effects are impinging on the way we do design. One fundamental change is the requirement of adding pipeline stages in the interconnect. This increased complexity is due to a number of factors including physical scaling as well as the desire to increase performance. This communication pipelining results in some circuit and architecture advantages and overheads. Various communication methodologies have differing local and architectural performance implications and CAD requirements. This work models pipelines using various protocols, including clocked and asynchronous, and makes a first-order comparison of energy and performance of these models. These models show that some asynchronous communication methods can be comparable or superior to clocked communication, even under worst case conditions.

Manuscript received March 30, 2009; revised August 16, 2009.

K. S. Stevens is with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112 USA (e-mail: kstevens@ece.utah.edu).

P. Golani and P. A. Beerel are with the Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA 90089 USA (e-mail: pgolani@usc.edu; pabeerel@usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2009.2037327

Global synchronous design requires the expenditure of a large design effort to create a low skew clock. While this can result in many efficiencies, including no synchronization overhead and uniform performance targets, there are also significant drawbacks. A single global frequency may not be optimal for many architectures because each module has different optimal power/performance points. Further, a single global frequency requires repipelining every module as frequencies change. Furthermore the global clock distribution network also leads to more overhead to account for clock skew and higher energy consumption. If efficient asynchronous communication can be designed, three significant benefits arise. First, each module can be designed for its best frequency and power. Second, the ability to interconnect components with different frequencies will be vastly enhanced resulting in higher design reuse, faster time to market, and easier ability to customize designs. Third, the ability of the asynchronous designs to adapt themselves to physical properties can lead to more robust design.

The goal of this work is to study communication methodologies to determine if and when asynchronous communication can be competitive at the physical level with synchronous styles. A similar study is done in [6] where bit-rate, power, area, and latency are compared between register links, wave pipelined parallel links, and asynchronous serial links [7]. This work, instead, focuses on asynchronous parallel links, comparing them with synchronous methodologies.

II. Handshaking and Background

Fig. 1 shows the basic interconnection for synchronous and asynchronous design methodologies. In synchronous design, data is normally transmitted every clock cycle between the latches or flops. Data transmission in asynchronous domains is based on handshake protocols (`req` and `ack`), allowing transactions to be dynamic based on data validity and space availability. Single-rail (bundled data) protocols, shown in Fig. 1, use the same data path as clocked designs but employ latches rather than flip-flops for storage [21]. Data can also be transmitted using 1-of-N codes which encode data validity within data wires but use a shared acknowledgment. When data is not available, the communication links remain idle [3]. This is emulated in a synchronous design by applying clock gating, which passes a data valid bit with similar functionality to the asynchronous `req` signal indicating the validity of the data, allowing one to disable or "gate" the clock to the flops. This implicit clock gating that asynchronous design provides improves the energy efficiency of the data communication, especially when the sender is not ready to transfer data every clock cycle. The backward acknowledge (`ack`) signal in the

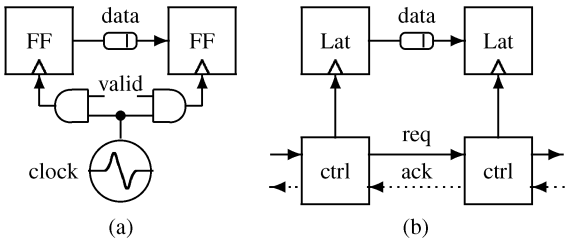


Fig. 1. Synchronous and asynchronous topologies. (a) Synchronous. (b) Asynchronous.

asynchronous domain restricts the max cycle time in a design dependent way that ensures min-delay hold-time failures can not occur or are explicitly modeled.

Synchronous methodologies and bundled data protocols place restrictions on how and when data can change relative to the clock or handshake signals [16]. The data must have sufficient time to become valid before the clock or request signal enables its usage. The `ack` signal is used in asynchronous protocols to indicate the data has been received, and that new data can be transmitted. The protocols are maintained in a synchronous system with setup and hold requirements.

Most asynchronous protocols have backwards handshakes and return-to-zero (RZ) phases. Two-phase (or NRZ) protocols halve the number of transitions per handshake but often require more logic overhead and delay. Timing assumptions can be made using pulsed and source synchronous protocols to entirely remove the backward handshake path [19].

When controlling local data paths, the communication delays for control signals are small and the handshaking overhead can largely occur concurrently with the data function, hiding the overhead. However, for communication, the `req` and `ack` control signals are exposed to the same power and delay costs as the data because they must also propagate from sender to receiver. Thus propagation of the control lines is an expensive operation for most asynchronous communication protocols in both power and delay when compared to clock methodologies where the distribution costs can be shared amongst many logic blocks. This makes data communication the most *inefficient* application for asynchronous protocols and creates a worst-case comparison between synchronous and asynchronous implementations.

Wire sizing and shielding can be used to somewhat mitigate control wire delay overhead by trading off increased area. Nonetheless, the handshaking significantly limits the throughput of asynchronous. For example, when the delays of the data and control wires are roughly the same, then a four-phase protocol can at best transmit data at a rate one-fourth the propagation time between the latches.

The remainder of this paper reports the comparison of common protocols against each other for a configuration typical of medium to long range point-to-point microprocessor communication link. Models are then developed that are used to compare cycle time, latency and energy in a hierarchical manner. These models include most first-order effects such as clock gating, coupling, process variation, voltage variations, data, and bus activity factors.

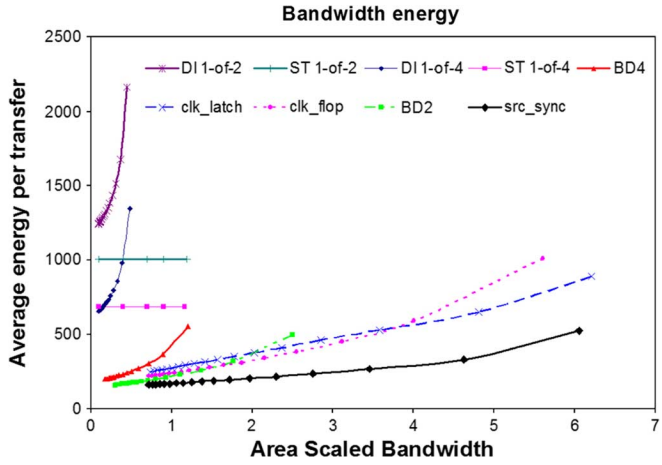


Fig. 2. Complete energy/bandwidth graph.

III. Overview

Eight representative protocols are modeled in this paper. This includes two synchronous protocols, five asynchronous protocols, and one source synchronous protocol. The synchronous protocols include the flopped design (clk_flop) and latched design with time borrowing (clk_latch). The asynchronous protocols include dual-rail (2-rail) and one-of-four (1-of-4) DI protocol, NRZ two-phase, and RZ four-phase bundled data protocols, and a two phase single-track protocol. Finally, a source synchronous protocol (src_sync) is modeled, which sends the time reference as a pulse along with the bundled data. The source synchronous protocol has no acknowledgment.

Other protocols with specific implementation styles such as the PC2/2 protocol [17], the Mousetrap protocol [18] and those using n-of-m codes [2], [15] can be similarly modeled but are not included in the paper for clarity and generality. Most protocols not explicitly included here can be accurately modeled with the protocols and parameters supplied in this paper. If not, new models can be created that are very similar to, or bounded by, the models presented in this paper in terms of throughput, energy, and bandwidth.

Figs. 2 and 3 show the final results of the models applied to a long 10 000 μm bus with a critical repeater distance of 555 μm. The distances and parameters are typical of what might be found on a microprocessor fabricated in a 65-nm process. A very long bus was chosen to allow a wide range of pipelining and frequencies to be graphed. The y-axis shows the average energy per transaction, measured in relation to the energy of driving a minimum sized inverter. The x-axis shows the effective bandwidth in terms of the number of concurrent transactions that can be sent down this path. The bandwidth is scaled for area by dividing the overall throughput by the number of wires in the control and data paths.

Each tick mark on every protocol line in the graphs indicates a specific pipeline granularity and thus a particular frequency. The parameter values in this paper use an optimized 10 000 μm wire with 18 repeaters. The far left point of each protocol is the condition where all of the repeaters are inverters. Pipelining is increased moving right along each protocol by replacing one

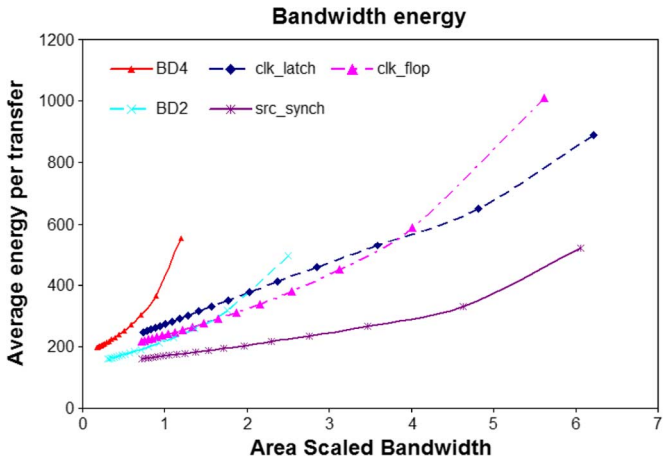


Fig. 3. Efficient protocols.

inverter with flops or latches. As more of the inverters of the communication path are replaced with pipeline latches, the frequency and bandwidth increases across the x-axis. The rightmost tick is where all of the inverters have been replaced with flopped repeaters or pipeline control, achieving the maximum bandwidth for the protocol.

Achieving a specific bandwidth target can come at different frequencies (or pipeline granularity) based on the protocol being used. In general, these results show that the asynchronous protocols require higher pipelining to achieve the same bandwidth as the synchronous protocols. For example, to achieve the area scaled bandwidth of 1 with the parameters used in these graphs, asynchronous four-phase bundled data communication requires more pipelining than synchronous flop based communication, hence consuming nearly two times more energy. This can be seen by determining the pipelining for every protocol that achieves a particular bandwidth target in the graphs.

Increased bandwidth comes at the cost of increased energy per transaction for a fixed bus width. Fine-grain pipelining, or the right-most point on each protocol, is the most energy-hungry operational mode for any protocol. Furthermore for synchronous and source synchronous designs there exists a limit on the clock frequency equal to the minimum pulse width that can be successfully propagated along a critical distance length of a wire. For optimal power and performance, the bandwidth should be matched by appropriate pipeline granularity or frequency.

The asynchronous protocols show a significantly reduced bandwidth range when compared with the synchronous and source synchronous protocols. This is due to the delay overheads in propagating request and acknowledge signals across long wires.

Fig. 3 zooms in on the energy efficient protocols. Synchronous protocols exhibit the best energy and bandwidth values for coarse grained pipelining having a low clock frequency. The source-synchronous protocol is the best for highly pipelined designs. The bulk of the energy for all protocols is dominated by the wires. However, as the data is increasingly pipelined, the average energy per transaction increases. The slopes of the four-phase asynchronous protocols are steeper, indicating a larger energy penalty for increased bandwidth.

The graphs show the worst case conditions. A vast majority of the asynchronous protocols will operate at a significantly improved frequency on a chip because they adapt to the current fabrication and environment conditions. Nominal values would reduce the slopes on the asynchronous protocols which makes them more competitive to synchronous design.

The DI and single-track protocols exhibit significantly higher average energy per transaction and the lowest bandwidth ranges. The bandwidth limitations are largely due to the inefficient use of wires. Each bit requires two wires in these protocols, effectively halving the area scaled bandwidth reported here. The high energy consumption of these protocols can mostly be attributed to the high activity factors that result from the data encodings. However, because these protocols are delay insensitive the CAD requirements are greatly reduced, allowing quicker time to market and higher robustness to operational parameters. Hence they may still be good choices depending on the overall design requirements.

IV. Comparison Methodology

The following methodology was employed to compare various modes of communication.

1) Create a set of parameterized first-order equations modeling the following for each protocol:
   a) delay and delay variations;
   b) energy and energy variations;
   c) cycle time and latch and flop overheads;
   d) bandwidth and wire area.
2) Design 65-nm circuits for all protocols and simulate them to:
   a) provide accurate delay, energy, and area values for the models;
   b) compare the first-order model results against SPICE simulations of the complete design.

A. Parameters

The timing of all fabricated circuit elements will be faster or slower than the scalar value of an "ideal" element due to process variations, capacitative coupling, and other effects. Variation from ideal devices and wires are considered an overhead in this work, whether it manifests itself as clock skew or device and wire delay variation.

Tables I and II show the parameters and their associated values that are used in this paper. All values come from simulation of the designs in 65-nm technology using predictive spice models [1], [5]. The parameters in the top section of Table I model variations from ideal delays that occur on wires and devices. 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. These delays are all relative to the fan-out of 4 (FO4) of a typical inverter, for example worst case clock skew is assigned a scalar delay of 1 FO4. The ideal stage delay, D_i, is varied from 27.8 FO4 to 1 FO4 to allow evaluating pipelines ranging from coarse

TABLE I
PARAMETER VARIABLES AND DERIVATIVES

Name	Equation	FO4 Value
V_c	Cross coupling variation of wire	0.4
V_p	Process variation	0.16
V_{pw}	Pulse width variation	0.2
V_{vt}	Voltage & temperature variation	0.12
V_{cad}	Margin for CAD (PV Accuracy)	0.05
D_i	Ideal stage delay (as FO4)	27.8 – 1
L_w	Latency of a 10,000μ interconnect with repeaters	27.8
T_{cq+}	Delay from clock to output	1.8
T_{cq-}	$T_{cq+}(1 - V_p)$	1.65
T_{sul}	Latch setup time	0.48
T_{suf}	Flop setup time	0.96
T_{skj}	Clock skew and jitter	1
T_{sk}	Clock skew	1
T_{skp}	Time borrow phase skew	0.5
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_{pw+}	$T_{pw}(1 + \frac{V_{pw}}{2})$	2.42
T_{dr}	Domino reset & propagation	0
T_{cad}	$D_i V_{cad}$	1.5–0.05
T_l	Part of D_i attributed to logic	0
T_{lc+}	$T_l(1 + \frac{V_p}{2} + V_{vt})$	0
T_{l+}	$T_l(1 + \frac{V_p}{2})$	0
T_c	$D_i - T_l$	27.8 – 1
T_{cc+}	$T_c(1 + \frac{V_c}{2} + \frac{2V_{vt}}{1+D_c})$	35.9 – 1.29
T_{c+}	$T_c(1 + \frac{V_c}{2})$	33.25 – 1.2
T_{c-}	$T_c(1 - \frac{V_c}{2})$	22.25 – 0.8
N_P	Number of pipeline stages	18 – 1
N_{rep}	Number of repeaters per pipeline stage excluding memory element	18 – 0
T_{csh+}	$T_c(1 + \frac{V_c}{4})$	30.59 – 1.1
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
E_{ctl}	Controller energy per phase per bit	0.072 – 0.90
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 II
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
X_c	Critical wire distance	555μm
W_b	Width of bus	32
D_c	Delay across repeated wire	1.6
$E_{clktree}$	Clock tree energy per transition	16.84
P_{dc}	% of clock tree energy for communication	10%

to very fine grain. This is the amount of ideal logic delay between latches or flops in a logic based design, and it determines the pipelining or frequency of a communication link. Latency

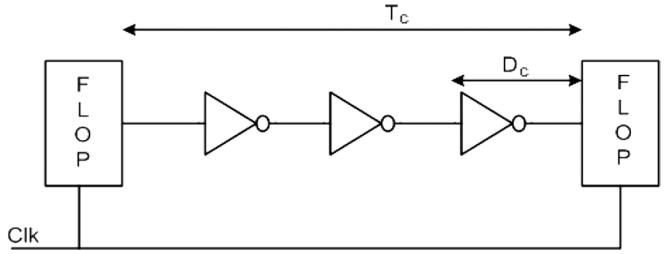


Fig. 4. Total communication delay in a pipeline stage delay T_c, versus repeated wire segment delay D_c.

L_w represents the ideal latency to transmit data across an unpipelined 10 000 μm wire.

The target delay of any pipeline stage can be distributed as either functional logic delays T_l or communication delays T_c, which includes both repeater and wire delay as shown in Fig. 4. For the designs simulated in this paper $D_i = T_c$. For generality our models include function logic delays, which significantly change the throughput results for some protocols. Variations are applied differently to logic and communication.

More accurate first-order models are achieved by including variation values. A single value V_c is used here to model crosstalk coupling variation. Delays with a maximum variant can append a "+" symbol, and min-delay values append a "−". For example, T_{c+} is the max communication delay and T_{cq-} is the minimum clock to data output delay of a flop or latch. T_{lc+} and T_{cc+} are the maximum logic delay and communication delays for synchronous systems. Synchronous protocols take first droop effects into account since the fixed frequency could otherwise cause the circuits to fail under the slower operation induced by the voltage droop. The asynchronous protocols will instead adapt itself to the operating conditions. Communication delay for the shielded links T_{csh+} of the DI and single-track protocols is smaller than for the non-shielded bundled data paths.

Variables N_{rep} and N_P in Table I represent the number of repeaters per pipeline stage excluding the memory element and the number of pipeline stages, respectively. The N_{rep} parameter is calculated as $\lfloor T_c/D_c \rfloor$. For all modes of communication except synchronous latch based, N_P is calculated as $\lceil L_w/D_c \rceil/(N_{\text{rep}} + 1)$, where $\lceil L_w/D_c \rceil$ represents the total number of repeaters required for a 10 000 μm interconnect. There are two latches acting as repeaters per pipeline stage for latch based communication, hence N_P in that case is equal to $\lceil L_w/D_c \rceil/(N_{\text{rep}} + 2)$.

The parameters in Table II include the interconnect design information of our example process. To identify the minimum delay we performed a 2-D sweep on the repeater size and number of repeaters required for a 10 000 μm long interconnect [8]. The optimum design required the number of repeaters to be 18 and the size of the repeater to be 96× a minimum size inverter. The delay across each repeater stage D_c is approximately 1.6 times a nominal FO4 delay with the energy required to drive a critical repeater distance as E_{rep}. Here we assume that repeaters are placed optimally at the distance of critical length. The energy required for data transfer in various protocols will be calculated relative to E_{rep}.

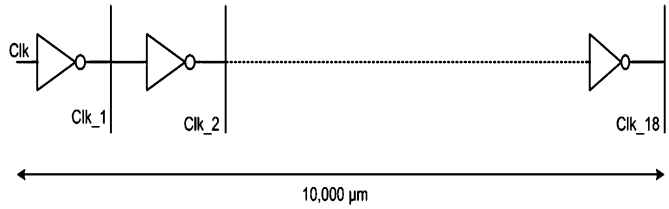


Fig. 5. Clock tree to distribute clock over 10 000 μm interconnect.

TABLE III
ASYNCHRONOUS CONTROL PARAMETERS FROM 65-nm SPICE SIMULATIONS OF THE DESIGNS

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
DI 1-of-4	6.04	1.2	0.60
STFB 1-of-2	2.46	1.04	0.87
STFB 1-of-4	2.55	1.16	0.60

The E_{clktree} parameter in Table II represents the energy required by the clock tree to distribute clock to a 10 000 μm interconnect. For the sake of simplicity we model the clock distribution network as a single wire running in parallel with data having repeaters at the critical distance as shown Fig. 5. Parameter P_{dc} defines the percentage of the energy of the repeated clock wire E_{clktree} that is used for data communication. In particular, P_{dc} models the fact that the clock may be used not only for distributing the clock to our data communication setup but also to other logic blocks in the vicinity. With the values set in Table II, for a 32-bit data path the clock distribution contributes 10% of the energy to the synchronous extreme pipelined design. More complicated models of clock trees are also possible and may be mapped to this simple model.

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, two-phase NRZ protocols generally require more complicated control with larger delays compared to four-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. The E_{ctl} parameter represents the average energy per phase per bit for the control logic for the asynchronous protocols. 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.

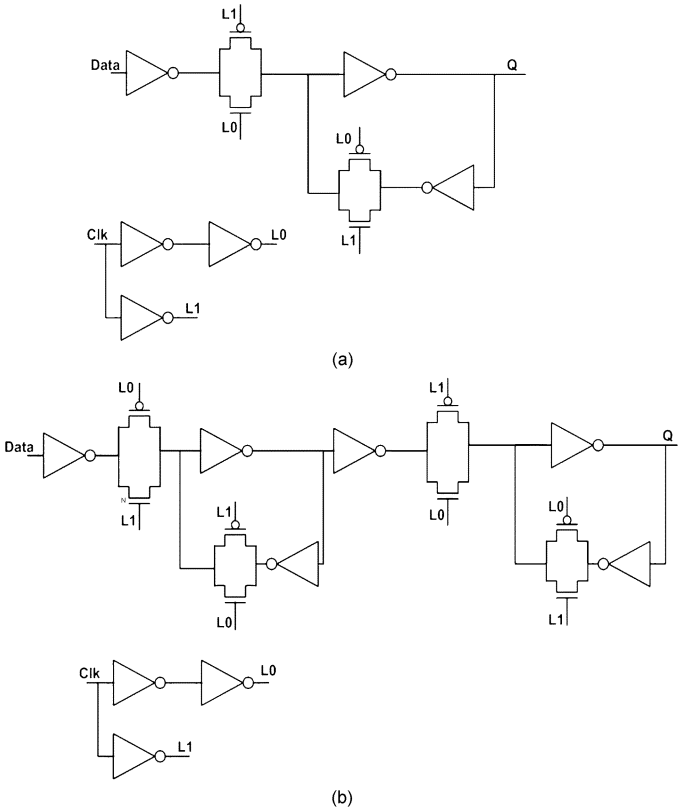


Fig. 6. Latch and flop transistor level design. (a) Latch. (b) Flop.

V. MODELS

A. Synchronous Communication—Latch and Flop

Fig. 6 shows the transistor level diagram of the latch and flop used in this work. The sizes of devices in this paper are calculated using logical effort [20]. Data must pass through one flop per clock cycle for flop based communication, while in the case of latch-based communication data has to go through two latches, one being driven by clock and the other one being driven by inverted clock as shown in Fig. 7.

The minimum cycle time of a flop represented by equation C_{ff+} presented in Table IV will be bounded by one of two conditions. First, the clock cycle cannot be shorter than twice the width of the minimum sized pulse $2T_{pw}$ that can be safely engineered and propagated. Otherwise, the delay through a stage will be the sum of the maximum logic and communication delay from the source flop to the destination flop $T_{lc+} + T_{cc+}$. Synchronous systems have additional overheads of setup time, clock skew and jitter, and other CAD related inaccuracies $T_{su}+T_{skj}+T_{\text{cad}}$. Upon arrival of the clock edge, the data must also propagate through the flop T_{cq+}. Setup, skew, and flop delays create the overhead for clocking.

Equation C_{l+} in Table IV is the minimum cycle time for a synchronous latch protocol. Similar to the flop protocol the cycle time is bounded by one of the following two conditions, twice the width of the minimum sized pulse $2T_{pw}$ and the delay through a stage which is equal to the communication delay from

TABLE IV
CYCLE TIME EQUATIONS. CYCLE TIME MEASURED IN FO4 DELAYS

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

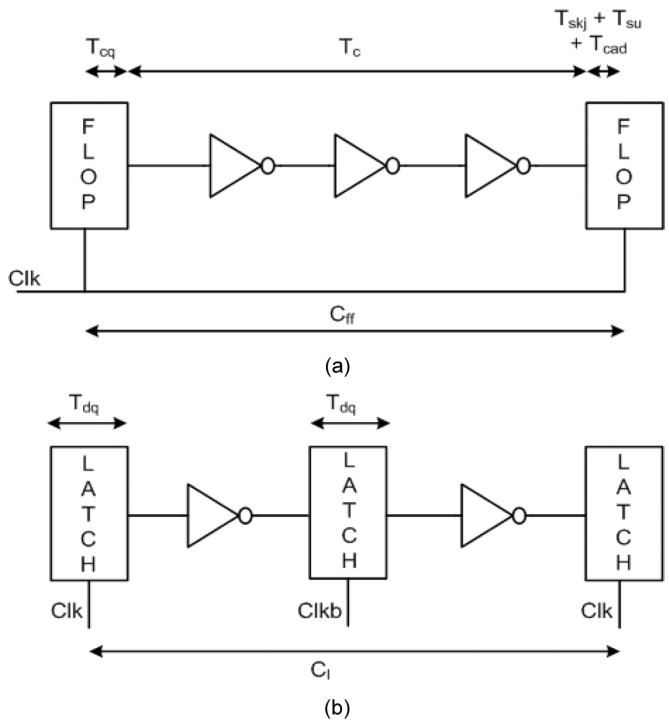


Fig. 7. Latch and flop pipeline stage configurations for distance of 4 critical repeater distances. (a) Flop based communication. (b) Latch-based communication.

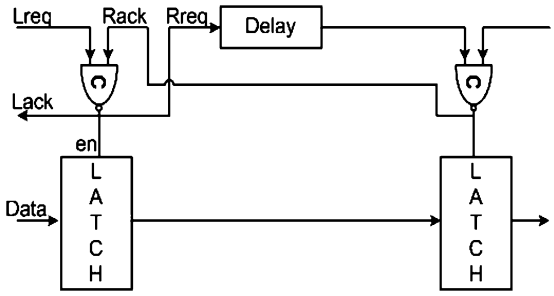


Fig. 8. Simulated implementation for bundled data four-phase protocol.

the source latch to the destination latch $T_{lc+} + T_{cc+} + 2T_{dql}$. Use of latches instead of flops helps in reducing clock skew and jitter overheads because of the time borrowing allowed between stages. However the data must propagate through two latches $2T_{dql}$ to account for the alternate phase control of the latches.

The equations that calculate the energy consumption for latches E_{le} and flops E_{fe} in Fig. 6 are shown in Table VI. The energy consumed by the flop/latch to transfer data the critical distance of a communication link is represented as the parameter $E_{\text{dataf}}/E_{\text{datal}}$. The average energy consumed per transition by the flop/latch due to switching of the clock is represented as parameter $E_{\text{clkf}}/E_{\text{clkl}}$. The average energy per transaction for clock gating of a flop E_{gf}/latch E_{gl} is calculated as a single transition on the clock driver, which is one-fourth the clock load of the flop plus energy into the gating NAND for the average number of idle bus cycles per transaction. The average energy per transaction consumed by the clock distribution network is given as $(1/A_b)2P_{dc}E_{\text{clktree}}$, where $1/A_b$ accounts for the amount of energy consumed by the clock distribution network when the data bus is idle.

The energy consumed per transaction of a single flop/latch pipeline stage can be attributed to two factors: the energy consumed by the memory element (flop/latch), and energy of the repeaters in that pipeline stage. The energy consumed by a flop is $W_b(2(E_{\text{clkf}}+E_{gf})+A_wE_{\text{dataf}})$ and the energy consumed by the repeaters $W_bA_wE_{\text{rep}}N_{\text{rep}}$ to drive the data. For the case of latch based communication the energy consumed by the latches is $W_b(4(E_{\text{clkl}} + E_{gl}) + A_w(2E_{\text{datal}}))$ due to two latches per pipeline stage.

B. Asynchronous Communication

In this section, we present the cycle time and energy models for some representative protocols used for asynchronous communication.

1) Bundled Data Protocol: The datapath in a bundled data asynchronous communication is similar to the datapath in synchronous communication. Controllers generate the local clock signals for the latches instead of the global clock, as shown in Fig. 8. The bundled data protocol can be implemented in both four-phase and two-phase handshaking protocols. The key timing assumption (also known as *bundling data constraint*) in a bundled data protocol is that the data should be valid before there is an associated transition on the request line. To satisfy this constraint the request signal needs to be delayed using a delay line such that this delay is greater than the worst case delay of the data plus the setup time of the latch.

Equation C_{bd2+} and C_{bd4+} in Table IV gives the minimum cycle time for a two phase and four phase bundled data protocol, respectively. The same controller is used for both the

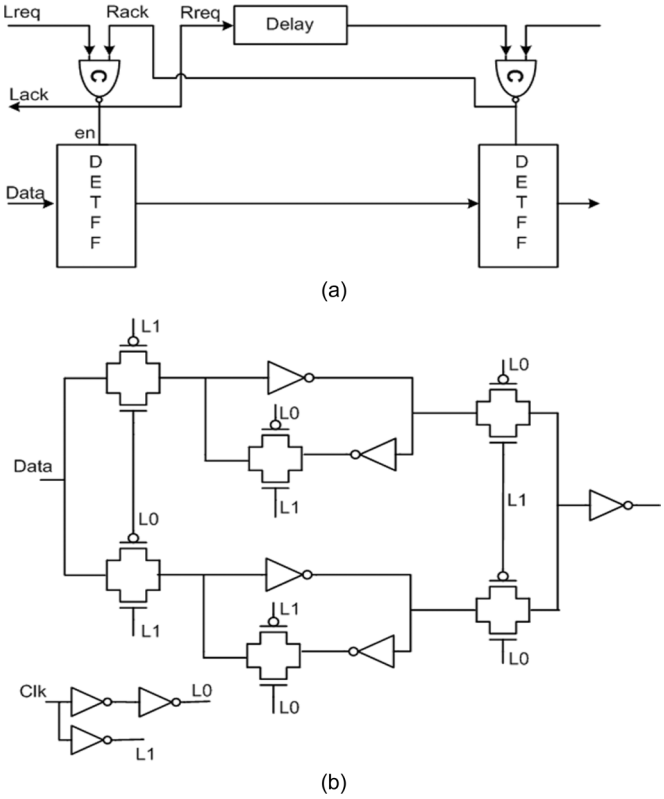


Fig. 9. Simulated implementation for bundled data two-phase protocol. (a) Bundled data two-phase design. (b) DETFF transistor level design.

four-phase and two-phase designs shown in Figs. 8 and 9. The key difference in these designs is the use of double edge triggered flip-flops [14] rather than simple latches for two-phase design. Parameter T_{fsm+} represents the cycle time of the controller divided by the number of phases. It is equal to 4.25FO4 for a four-phase protocol and 4.78FO4 for two-phase protocol. The parameter T_{ps} in Table IV is the worst case path separation or margin in terms of FO4 delays between the control and data paths. T_{cdel} is the number of gate delays that must be added to the control path to guarantee that we satisfy the bundling data constraint. The worst-case values must be taken assuming that, as pipe stages are composed, it is possible to have worst-case skew between data and control in all stages. This creates a safe margin for the control and data race at the expense of throughput.

Equations E_{4e} and E_{2e} in Table VI calculates the energy per pipeline stages for four-phase and two-phase bundled data protocols, respectively. The energy required by the asynchronous controller per phase is characterized as E_{ctl}. This parameter represents the energy consumed in transmitting the control signals (req and ack) a critical distance along the communication link. The parameter E_{clkdef} in Table I represent the average energy consumed per transition by a double edge triggered flip-flop shown in Fig. 9.

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 four-phase protocol is equal to $4*(E_{\mathrm{ctl}}+E_{\mathrm{rep}}N_{\mathrm{rep}})$ and $2*(E_{\mathrm{ctl}}+E_{\mathrm{rep}}N_{\mathrm{rep}})$ for the two-phase protocol. The energy consumed by the latch in a four-phase protocol is calculated as $W_b(2E_{\mathrm{clkl}}+A_w(E_{\mathrm{datal}}+E_{\mathrm{rep}}N_{\mathrm{rep}}))$ and for a two-phase protocol energy is calculated as $W_b(E_{\mathrm{clkdef}}+A_w(E_{\mathrm{datal}}+E_{\mathrm{rep}}N_{\mathrm{rep}}))$.

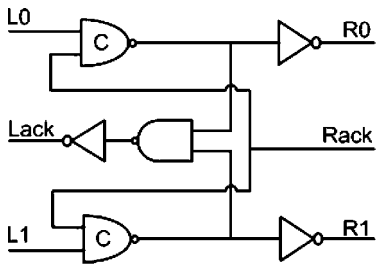


Fig. 10. Simulated DI 1-of-2 protocol implementation.

2) Delay Insensitive Protocol: The data path in a DI protocol is comprised of data encoded as 1-of-N rails, where N wires are used to represent $\log_2 N$ bits of data and an additional wire which acts as the acknowledgment signal. The most well known of this class of protocols is dual rail (1-of-2) which uses two data wires per data bit, and 1-of-4 which uses four data wires to represent two bits of information.

Delay insensitive protocols are typically implemented using four-phase handshake protocols. Hence there are four iterations through the data acknowledge detection and propagation logic $4T_{\mathrm{fsm+}}$. Fig. 10 shows a 1 bit WCHB [13] controller implementing a 1-of-2 DI protocol handshake. The acknowledgment wire can be made wider to reduce delay on its transitions $2O_{wd}T_{c+}$.

The equations that calculate the energy for communication using DI protocols are shown in Table VI. The asynchronous controller is comprised of domino logic which is responsible for forward data propagation along with a completion detection logic. The completion detection logic is responsible for generating the acknowledgment signal such that `ack` becomes valid only when all the input bits have arrived. The overhead of the generation of the completion signal becomes a limiting factor for the cycle time of the design, particularly for wide data buses of 32 bits and above. The average value of the controller delay per phase T_{fsm} for this particular example of DI protocol is 6.65FO4. However this performance can be improved by using 2-D [13] pipelining methodology which motivates a tradeoff between performance and area. The main idea behind 2-D pipelining is to reduce the cycle time overhead by using multiple completion detection logic modules, each module working on a subset of the data path at the cost of more aggregate area. For the sake of simplicity in this paper we will limit our discussion to only 1-D pipelines; however faster values will be obtained for large buses with 2-D pipelines.

Equation C_{di2+} and C_{di4+} in Table IV gives the minimum cycle time for 1-of-2 and 1-of-4 DI protocols, respectively. The data transitions have less variation because the encoding result in the wires being half shielded $2T_{\mathrm{csh+}}$.

The energy required by the asynchronous controller per phase is characterized as E_{ctl}. Unlike bundled data protocols, DI protocols encode the `req` control signal within the data itself. The

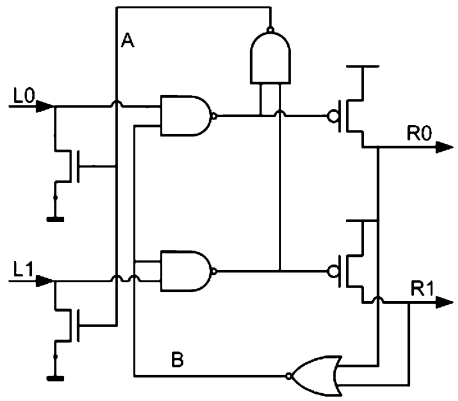


Fig. 11. Simulated single-track 1-of-2 protocol design.

controller of Fig. 10 transmits 1 bit of data and the ack control signal. The parameter $E_{\rm ctl}$ represents the energy of the controller per phase for transmitting 1 bit of data and the ack control signal. The energy consumed per transaction per pipeline stage for a DI protocol using 1-of-2 encoding is the sum of the controller energy $4W_bE_{\rm ctl}$ and the energy of the repeaters in that pipeline stage to drive the data $2W_bE_{\rm rep}N_{\rm rep}$ and the energy required to drive the acknowledgment wire $2E_{\rm rep}N_{\rm rep}$. For 1-of-4 encoding, each wire transition represents 2 bits of data, so the energy required by the repeaters to drive the data reduces by half to $W_bE_{\rm rep}N_{\rm rep}$.

3) Single-Track Protocol: The single-track protocol is a two-phase asynchronous protocol which uses 1-of-N data encoding similar to DI protocol with the key difference being that there is no separate acknowledgment wire [9]. The sender drives one of the N wires high thereby sending the data and, after receiving this data, the receiver sends an acknowledgment by driving the same 1-of-N wires low. After driving the wire to its desired state, the sender and receiver(s) must tristate the wire to ensure that they do not try to drive the wire in opposite directions at the same time. One popular single-track asynchronous controller is single-track full buffer (STFB) [9]. Fig. 11 shows a typical 1 bit 1-of-2 STFB buffer.

The two-phase single-track protocol reduces the overheads of the reset handshake. This yields substantially higher performance than four-phase DI protocols and fewer transitions per bit which substantially lowers power. Compared to bundled-data protocols there are no timing assumptions that require margins on the forward latency, yielding additional performance improvement. However, the number of transitions per bit is often larger, producing higher average power. One limitation of STFB is that every repeater must act as a pipeline stage. Hence STFB designs are restricted to fine-grained pipelining. Although STFB can be used in 1-D pipelined designs similar to the DI protocol, this paper reports simulation results that are bit level 2-D pipelined; i.e., each completion detection logic detects the validity of only a single bit.

Equation C_{st+} in Table IV gives the cycle time for 1-of-2 and 1-of-4 STFB, respectively. The average value of the controller delay per phase $T_{\rm fsm}$ for 1-of-2 STFB is 4.80FO4 and 5.10FO4 for 1-of-4 STFB. Equations E_{st2} and E_{st4} in Table VI represent the energy required by an STFB controller to drive data through a critical distance communication link. $E_{\rm ctl}$ represents the energy per phase consumed by the controller. In case of 1-of-2 STFB designs the energy per pipeline stage is equal to the sum of controller energy and the energy of the repeaters in that pipeline stage $2W_bE_{\rm ctl}$ and in case of 1-of-4 STFB designs this energy is equal to $W_bE_{\rm ctl}$.

C. Source Synchronous Communication

Another mode of data communication is wavepipelining [4] where several "waves", i.e., data signals, can concurrently propagate through a pipeline with proper timing separation between two consecutive waves. This timing separation is created by a "constructive clock skew" in conventional wavepipelining, by using a separate timing reference in source synchronous protocols and a "fast" signal in an asynchronous "surfing" protocols [22], [23]. The timing reference signal is routed in parallel with the data. A novel characteristic of asynchronous surfing is that when logic blocks receive the fast reference pulse their computation delay drops, thus creating a surfing effect wherein events are bound in close proximity with the pulse on fast reverence signal.

The key to the performance of any wavepipelining method is the time separation of waves which defines their cycle time. In this paper, we have explicitly modeled the cycle time of source-synchronous communication with equation C_{ss+} in Table IV. In particular, the cycle time is lower bounded by three terms: the minimum pulse width that can propagate through a critical distance; the communication delay of the data through a repeated wire; and the propagation delay of the clock signal plus the delay through controller which generates the enable signal for the latch. Equation E_{ss} in Table VI gives the energy model for source synchronous protocol. In the source synchronous models used here, multiple values are not propagated between flop stages.

VI. Comparisons

In this section, the various modes of communication are compared in terms of maximum throughput, energy consumption and bandwidth they can provide. These results are based on the first-order models developed in this paper using parameter values directly derived from SPICE simulations of the designs in the 65-nm process.

A. Throughput

Fig. 12 shows a pipeline with an ideal logic delay of 4. Ideal pipelines would have no variation, and the flops would have zero latency and no overhead. This would allow a new data item to be propagated through these pipe stages every four gate delays. We compare the actual throughput and overheads of the protocols as calculated by the equations in Tables IV against the ideal pipeline delays. The equations calculate extra delays that are added due to device variation, latch delays, etc.

The throughput models applied to our parameter values are plotted in Figs. 13–16. The x-axis plots delays based on the pipeline granularity in terms of the number of gate delays between flops or latches. The left-most side contains pipelines with a logic pipeline depth of 27.8 ideal gate delays per stage which is necessary to propagate a signal down a 10 000 μm wire. The

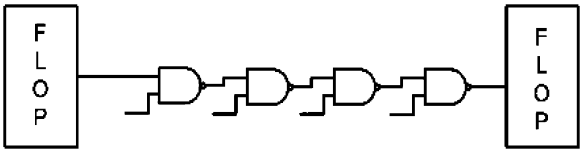


Fig. 12. Logic pipeline with delay of 4.

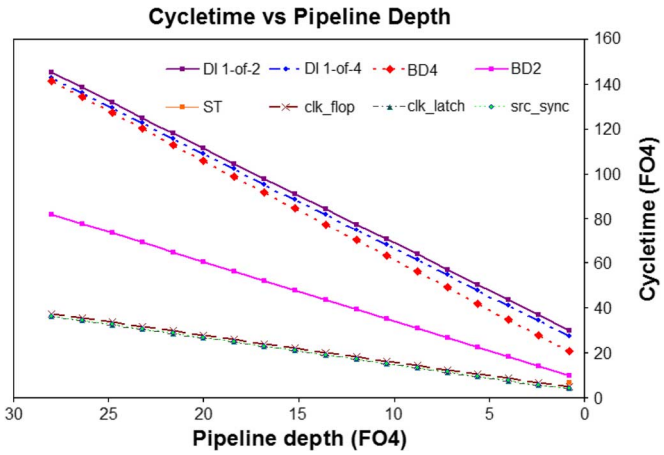


Fig. 13. Worst-case protocol throughput.

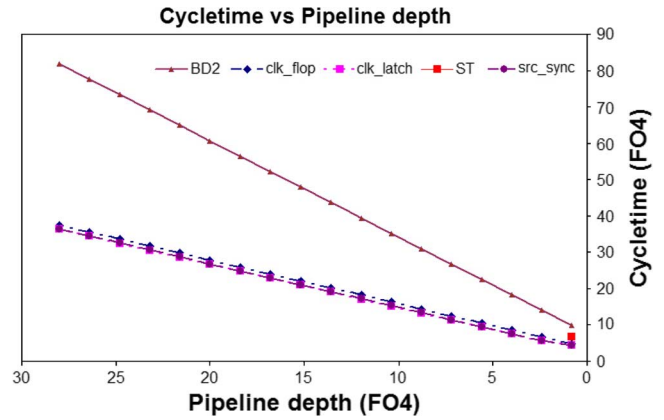


Fig. 14. Throughput of efficient protocols.

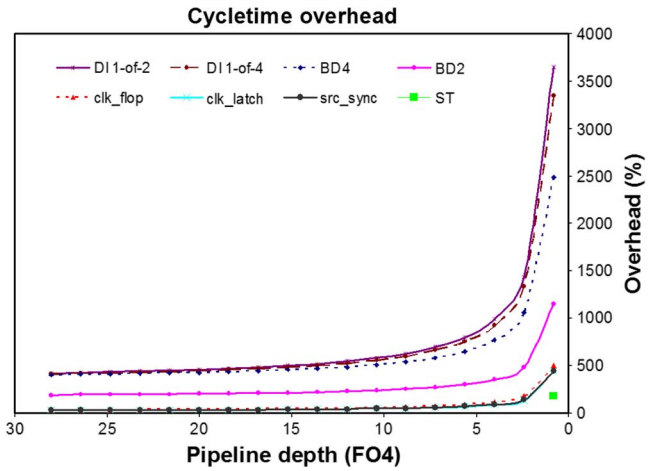


Fig. 15. Throughput overhead against ideal.

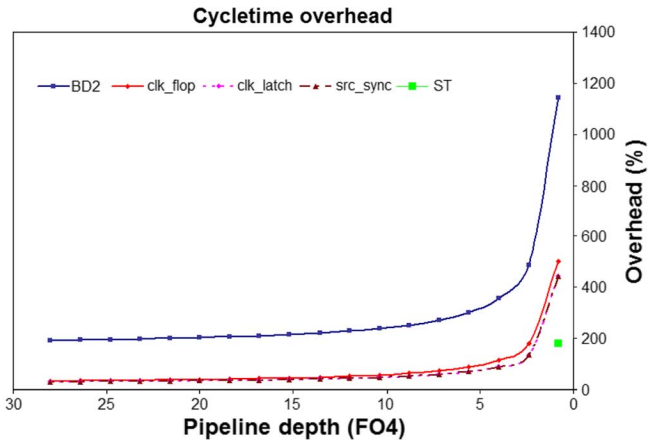


Fig. 16. Overhead of efficient protocols.

right-most point contains a single ideal gate delay per stage. The y-axis plots FO4 delays or the overhead calculated by comparing the modeled worst-case delay to the ideal delay.

These plots allocate all of the delay to communication. As expected, the asynchronous protocols with acknowledgment are the least efficient for communication, with the four-phase protocols being the worst. The most efficient protocols are the clocked and source-synchronous protocols. The two-phase asynchronous protocol shows significantly better performance due to the reduction in control transitions propagated between sender and receiver. The source synchronous protocol is the best asynchronous protocol at low frequencies because its request is propagated as a pulse and no acknowledgment is explicitly included. This protocol is marginally slower compared to synchronous protocols largely due to the conservative margin in the delay elements. In a real design, one would expect the source-synchronous circuit to outperform the synchronous design due to its adaptive nature.

One of the limitations of single-track protocol is that every repeater must be a single-track pipeline stage, limiting single-track to fine-grained pipelines. This results in a single point in the graphs. However, it is noteworthy that at high frequencies two-phase single-track protocol is the most efficient asynchronous protocol because this two-phase protocol avoids the delay margin needed in bundled-data protocols.

The cost of decreasing the amount of logic in each pipe stage to increase frequency can be inferred from these graphs. Fig. 17 shows the historical trend in logic gates per pipe stage for Intel's recent microprocessors. As the amount of logic per stage is reduced, there is a considerable increase in power and performance sapping overhead. Figs. 15 and 16 show that shortening the pipe depth has come at little cost in the past, but the overheads start to dramatically increase as pipelines reach a certain granularity. In a synchronous latch design there is a 22% loss in efficiency as the pipe depth is decreased from 15 to 10 ideal gate delays. This balloons to a 50% loss if one moves from 10 to five ideal gate pipelines. Hence, future frequency increases for synchronous designs will result in diminishing performance

TABLE V
LATENCY PER STAGE. DELAY MEASURED IN FO4 DELAYS

Name		equation	description
L_{ff+}	$\geq$	$\max(2T_{pw}, (T_{lc+} + T_{cc+} + T_{su} + T_{skj} + T_{cq+} + T_{cad}))$	Clk_flop latency
L_{l+}	$\geq$	$\max(2T_{pw}, T_{lc+} + T_{cc+} + 2T_{dql})$	Clk_latch latency
L_{di+}	$\leqslant$	$T_{lat+} + T_{csh+} + T_{l+}$	2-rail and 1-of-4 DI latency
L_{bd2+}	$\leqslant$	$T_{lat+} + T_{cc+} + T_{cdel}$	2-phase bundled data latency
L_{bd4+}	$\leqslant$	$T_{lat+} + T_{cc+} + T_{cdel}$	4-phase bundled data latency
L_{ss+}	$\leqslant$	$\max(2T_{pw}, (T_{cq+} + T_{l+} + T_{cc+} + T_{su}), T_{c+} + T_{pdel} + T_{fsm+})$	src-sync latency
L_{st+}	$\leqslant$	$T_{lat+} + T_{csh+}$	STFB 1-of-2 and 1-of-4 latency

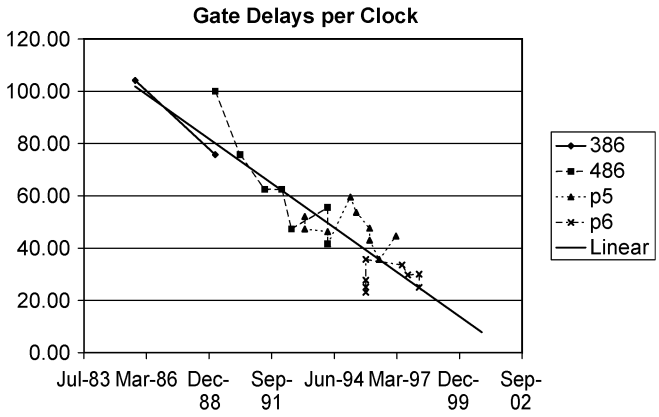


Fig. 17. Gate delay scaling of products [11].

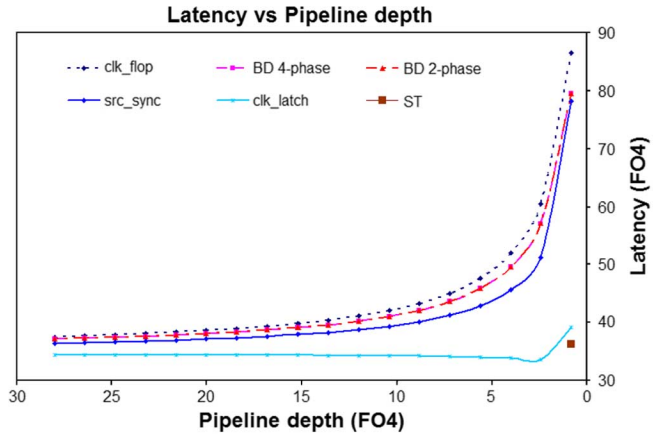


Fig. 18. Latency versus pipeline depth.

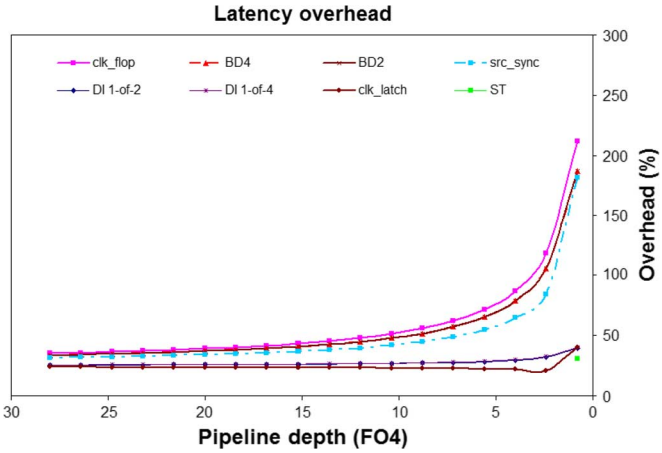


Fig. 19. Latency overhead versus pipeline depth.

gains. This also comes at an increased energy consumption cost as is show in Section VI-B.

B. Latency

Figs. 18 and 19 compare the latency and overhead per pipeline based on the equations given in Table V. The x-axis plots delays based on the pipeline granularity in terms of the number of gate delays between flops or latches. The overhead is calculated by comparing the latency of the design to the wire latency given by parameter L_w in Table I. These plots allocate all of the data path delay to communication.

In case of ideal pipelines there are no overheads associated with the memory elements, so the latency is the same as wire delay. However in case of synchronous designs, especially synchronous flop communication, each stage has to suffer an overhead to account for clock to output delay, setup delay, and clock skew margin. As the pipeline granularity is increased the overhead increases to as much as 200% as shown in Fig. 19. Margins for setup time and clock skew are not required for synchronous latch communication due to time borrowing between stages. However this protocol suffers the overhead of two data-to-output delays $2T_{dql}$ through the latches.

The overhead associated with bundled data asynchronous designs comes from two sources. The first source is the delay associated with the controller used to generate local clock signals. The second is the extra delay margin inserted in the request line as shown in Fig. 8 to satisfy bundling data constraint. The DI protocols and single-track protocol provides the lowest latency of all the protocols. This can be attributed to two properties of these circuits. First, since the data coding is delay-insensitive in nature no extra margins are required for setup to the latches. Second, the use of fast domino logic used in the data path leads to faster propagation of data compared to latch/flop designs.

C. Energy

Fig. 20 shows the energy dissipated to transmit data across a 10 000 μm bus with varying amounts of pipelining. Each equation in Table VI calculates the energy for a single pipeline stage. This number is multiplied by the number of pipeline stages N_P which scales the results for the same 10 000 μm distance. The x-axis shows the number of repeaters between the flops or control elements in the pipeline, the right-most value when every inverter is replaced with a flop.

The asynchronous DI protocol based on 1-of-N codes have a much poorer power profile due to their activity factors. The rest of the protocols have fairly similar energy profiles. The curves are very flat except for ultra-fine grain pipelining, where the control overhead becomes a significant power drain on the

TABLE VI
MODELS FOR AVERAGE ENERGY PER TRANSACTION PER PIPE STAGE

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}	$\frac{1}{A_b}2P_{dc}E_{clktree} + W_b(2(E_{gf} + E_{clkf}) + A_w(E_{datf} + E_{rep}N_{rep}))$	Clk_flop energy
E_{le}	$\frac{1}{A_b}2P_{dc}E_{clktree} + W_b(4(E_{gl} + E_{clkl}) + A_w(2E_{datl} + E_{rep}N_{rep}))$	Clk_latch energy
E_{4e}	$4(E_{ctl} + E_{rep}N_{rep}) + W_b(2E_{clkl} + A_w(E_{datl} + E_{rep}N_{rep}))$	4-phase bundled data energy
E_{2e}	$2(E_{ctl} + E_{rep}N_{rep}) + W_b(E_{clkdef} + A_w(E_{datl} + E_{rep}N_{rep}))$	2-phase bundled data energy
E_{ss}	$2(E_{ctl} + E_{rep}N_{rep}) + W_b(2E_{clkf} + A_w(E_{datl} + E_{rep}N_{rep}))$	src_sync energy
E_{di2}	$W_b(4E_{ctl} + 2E_{rep}N_{rep}) + 2E_{rep}N_{rep}$	DI 1-of-2 energy
E_{di4}	$W_b(4E_{ctl} + E_{rep}N_{rep}) + 2E_{rep}N_{rep}$	DI 1-of-4 energy
E_{st2}	$2W_bE_{ctl}$	STFB 1-of-2 energy
E_{st4}	W_bE_{ctl}	STFB 1-of-4 energy

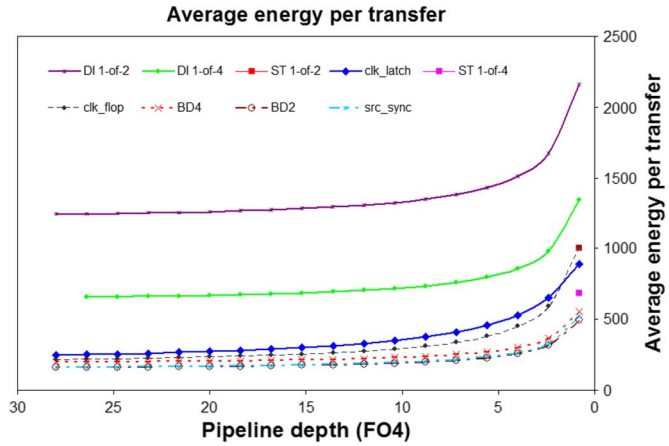


Fig. 20. Average energy per transaction.

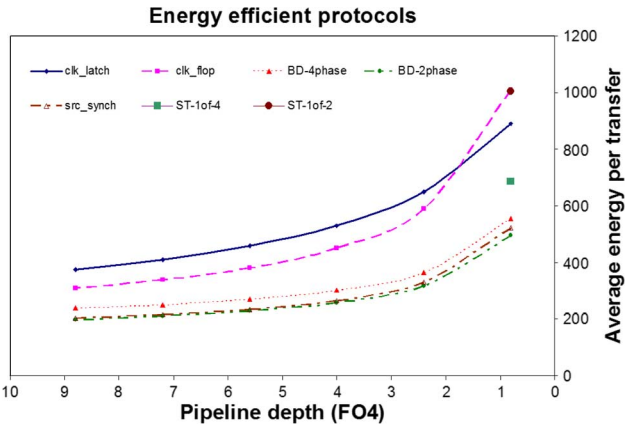


Fig. 21. Highly pipelined transfer energy.

circuits. Fig. 21 shows the most efficient protocols at high frequency where bus activity factor is set at 5% and the activity factor of the data wires is set to be 18%. This shows that at high frequencies when some idle cycles are present the bundled data asynchronous protocols have a distinct advantage over synchronous protocols. This can be explained because of clock switching and energy required in the gating logic even during idle phases, while asynchronous protocols provides ideal clock gating at no extra cost. Observe that while the plots show asynchronous two-phase bundled data protocols to be more energy efficient than four-phase bundled data protocols, this may not always be true.

D. Bandwidth

In order to effectively scale a chip, additional metal layers are added to support the increased bandwidth of the design. The effective utilization of wires in a process therefore determine the cost and bandwidth of the design. This work defines bandwidth to be proportional to the wire area. Table VII shows how throughput is scaled based on the wire area to calculate the bandwidth of a design. The delay insensitive and single-track protocol modeled in this paper require two wires per data bit. Therefore, for the same wire area the bandwidth of these asynchronous designs would have roughly half the bandwidth. The penalty for control signal and valid bit overhead are also calculated for the studied designs as shown.

TABLE VII
AVERAGE WIRES PER DATA BIT

Name	equation	description
B_v	$1 + 1/W_b$	synchronous valid bit
B_{da}	$2 + O_{wa}/W_b$	DI data encoding & ack
B_{st}	2	Single-Track data encoding
B_{ra}	$1 + 2O_{wa}/W_b$	req and ack signals
B_r	$1 + O_{wa}/W_b$	req signal

The final bandwidth models graphed in this paper are shown in Table VIII. The value is the bandwidth-scaled number of data words that can be pipelined in the 10 000 μm wire. This allows us to compare bandwidth and area for various levels of pipelining across all the designs. The bandwidth values from this table are used to calculate the x-axis point for each protocol and the energy numbers are used for the y-axis values of Figs. 2, 3, and 25.

VII. ANALYTICAL MODELS VERSUS SIMULATION

The accuracy of the analytical models were evaluated by comparing the results from the first-order models against

TABLE VIII
BANDWIDTH EQUATIONS FOR THE EIGHT PROTOCOLS

Name	equation	description
B_{cf}	$L_w/(B_v C_{ff+})$	Clk_flop bandwidth
B_{cl}	$L_w/(B_v C_{l+})$	Clk_latch bandwidth
B_{di}	$L_w/(B_{da} A_{di+})$	1-of-2-rail and 1-of-4 DI b/w
B_2	$L_w/(B_{ra} A_{2+})$	2-phase bandwidth
B_4	$L_w/(B_{ra} A_{4+})$	4-phase bandwidth
B_{ss}	$L_w/(B_r SS_+)$	Src_sync bandwidth
B_{st2}	$L_w/(B_{st} A_{st2+})$	STFB 1-of-2 bandwidth
B_{st4}	$L_w/(B_{st} A_{st4+})$	STFB 1-of-4 bandwidth

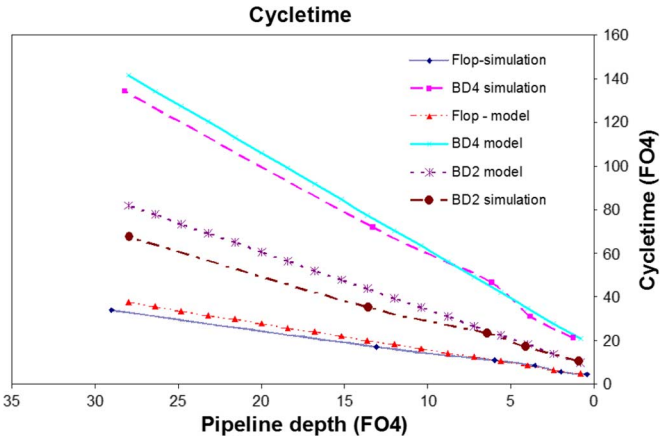


Fig. 22. Cycle time comparison of analytical models and SPICE simulation.

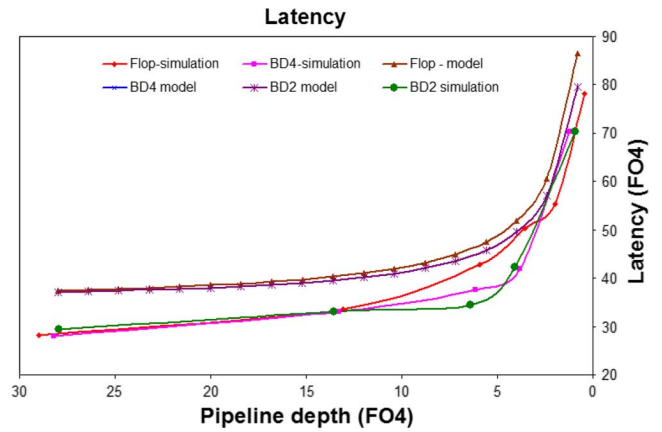


Fig. 23. Latency comparison between models and SPICE simulations.

SPICE simulations of layout of the complete 65-nm designs. Cycle time, latency, and energy required to transfer data on a 10 000 μm long interconnect by varying the pipeline granularity were compared against SPICE simulations for three target designs. Figs. 22–24 compare the cycle time, latency, and energy per transfer for asynchronous four-phase bundled data, asynchronous two-phase bundled data and the synchronous flop modes of communication. The x-axis plots delay based on the pipeline granularity in terms of the number of gate delays in a single pipeline stage.

Fig. 22 compares the cycle time while Fig. 23 compares the latency. The results used in the paper have accounted for worst case process variations, such as clock skew and delay margin to be added in case of bundled data protocols. These are not modeled in the SPICE simulations, so the parameters T_{cdel}, T_{skj} and

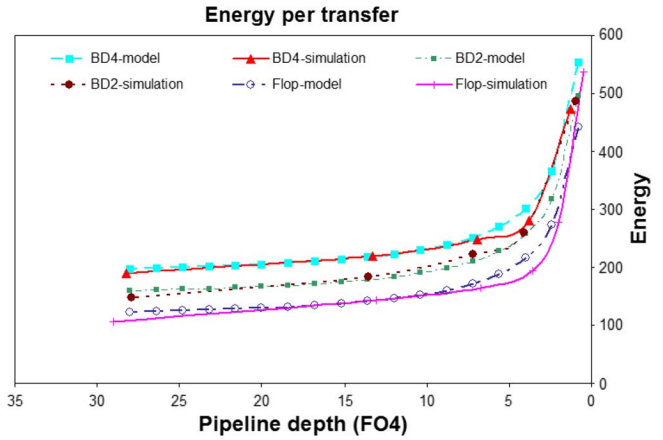


Fig. 24. Energy per transfer comparison between models and simulation.

T_{cad} are set to 0 for this comparison. Fig. 24 compares the energy required per transaction. No clock gating is assumed in the spice simulations for the case of synchronous flop communication. Thus for fair comparisons the parameter E_{gf} in our models is set to zero.

From the above comparisons, we can see that the proposed models are fairly accurate. The errors in the cycle time comparison are less than 15%, less than 20% for latency comparisons, and less than 10% for energy comparisons. These errors can largely be attributed to the models assuming a worst case environment which is difficult to include in our spice simulations. For example, even though our SPICE simulation set up captures the loading effects of coupling capacitance, it does not model the worst case crosstalk noise. The shapes of the curves are similar, validating the proposed models.

VIII. OBSERVATION AND CAVEATS

Accurate apples-to-apples comparisons are always challenging. These equations can model the first-order effects of most implementation styles. The magnitude of many of these effects, such as first-droop, vary based on the design and implementation styles employed. Hence, the models are highly parameterizable to match the argued effect by applying correct parameter values used by the equations. This also allows one to study the results of a particular effect that is trending up or down as processes are scaled.

Simulating all of the various architectures is outside the scope of this work, so these models are validated by performing SPICE simulations on a subset of the protocols. Further, there are some effects which are difficult to simulate in a simple simulation environment such as worst case crosstalk noise, the clock tree, and clock gating. These have not been considered during the SPICE simulations. Therefore the accuracy bounds reported here may not apply to all scenarios.

There are other effects that are also difficult to compare. The specific design of the flop and latch, for instance, will have impact on power and performance. This work selected a single simple flop and latch style and applied it to all protocols. This made the work relatively accurate between models, but the absolute value will be different based on the design used. However, the largest difference in most flop designs is the energy required

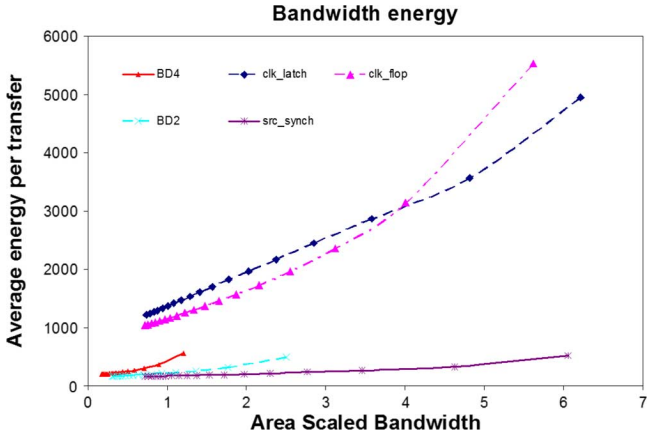


Fig. 25. Bandwidth/energy for lower bus utilization.

in the clocking. Since this can be determined separately from data energy and delays the variation across flop styles should be rather simple to estimate.

This work can also be applied directly to pipeline protocols for logic blocks, and some of the protocols include this modeling. However, the logic family, design and protocol style, and implementation aspects of the logic blocks have a much greater variability than a repeated wire, so the accuracy of such a model is much more difficult to compare and arguably less accurate. Min-delay issues become very important, but they can largely be ignored in communication since max and min signal propagation delays will be similar.

Fixed bandwidth can be achieved through combinations of throughput and parallelism. For instance, doubling the throughput and halving the wires will achieve the same bandwidth, at a smaller area. Likewise, creating double data rate designs from the four-phase bundled data asynchronous protocols is a relatively straightforward transformation that nearly doubles the area scaled throughput of these designs. The effect of such transformations can be observed using simple modifications to these models.

The parameter values used in this paper represent a single design point. For example, increasing the activity factor of the bus by a factor of 10 makes no significant change to Figs. 2 and 3. However, decreasing the activity factor by the same amount gives a significant advantage to the asynchronous protocols as shown in Fig. 25. Here the four-phase, two-phase, and source synchronous protocols show significant energy advantages at all bandwidths over synchronous designs. Thus, the parameters need to be configured based on your target application.

Optimization based on these models has not been done. For instance, there is a tradeoff between faster control signal propagation and its deleterious effect on bandwidth.

IX. Conclusion

Parameterized first-order analytical models are presented for many communication models. SPICE simulation of three of the protocols show the analytical models to be accurate within 15% for cycle time, 20% for latency, and 10% for energy across the full range of pipeline depths.

The parameters can be modified to quickly compare various communication protocols operating at different design targets, such as low frequency and power designs to performance constrained targets. Comparisons between various protocols have been demonstrated based on varying the parameters of the models.

The efficiency of asynchronous communication is very dependent on the protocol. Communication is one of the worst-case scenarios for asynchronous design due to the latency of the handshake signals. These latencies require most asynchronous protocols to be pipelined deeper than the clocked protocols to achieve similar bandwidths. Yet the efficient asynchronous protocols are shown to have similar results to the synchronous protocols when measuring average transaction energy for a target bandwidth using parameters targeted for a microprocessor bus.

Many design parameters favor either a synchronous or asynchronous style. Wider buses and high bit-level activity factors favor asynchronous communication. Higher bus utilization factors favor synchronous designs. Changing the operating environment reflected in these parameters can result in asynchronous communication being far superior to synchronous protocols.

The results in this paper demonstrate that energy per transaction as well as latency and cycle time overheads remains relatively flat across most pipelined designs until pipelining becomes aggressive. At that point there begins to be a considerable penalty for increasing the pipelining. Thus there is a broad range of pipeline frequencies that can be implemented with relatively small energy and performance penalty.

Asynchronous designs can exploit this flat region of the graphs since pipelining frequency can be dynamically chosen. This is not the case with clocked protocols where distances and pipelining are fixed relative to the clock frequency. This implies that scalability and the ability to optimize for a particular power/performance point is enhanced in asynchronous designs. Asynchronous designs also provide substantial latency advantages over synchronous designs which are very important in high performance network-on-chip designs [10]. Furthermore, asynchronous designs can be repeated and pipelined at the optimal critical distance for the target topology without requiring margin for future process scaling.

This study only compares the physical data transmission efficiencies. Communication effects on the overall processor performance and power are an important extension [12]. The benefits of implementing asynchronous communication in an otherwise globally synchronous processor must override the cost of synchronization with the destination frequency. Future designs—such as those with multiple on-die cores or designs with power islands—will decompose the chip into different clock domains for power, thermal, and performance reasons. For such architectures, asynchronous communication exhibits significantly lower latency and has been shown, through this study, to have similar or better physical transport efficiencies when compared to synchronous methodologies.

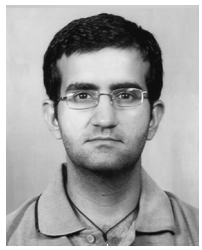
References

[1] "ASU predictive spice models," 2009. [Online]. Available: http://www.eas.asu.edu/~ptm/

[2] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, 2002.
[3] W. J. Bainbridge and S. B. Furber, "Delay-insensitive, point-to-point interconnect using M-of-N codes," in *Proc. Int. Symp. Asynch. Circuits Syst.*, May 2003, pp. 132–140.
[4] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 464–474, Sep. 1998.
[5] Y. Cao, T. Sato, M. Orshansky, D. Silvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," in *Proc. Custom Integr. Circuits Conf.*, May 2000, pp. 201–204.
[6] R. Dobkin, A. Morgenshtein, A. Kolodony, and R. Ginosar, "Parallel vs serial on-chip communication," in *Proc. Int. Workshop Syst. Level Interconnect Prediction*, 2008, pp. 43–50.
[7] R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, and A. Kolodny, "High rate wave-pipelined asynchronous on-chip bit-serial data link," in *Proc. Int. Symp. Asynch. Circuits Syst.*, Mar. 2007, pp. 3–14.
[8] M. A. El-Moursy and E. G. Friedman, "Optimal wiring of RLC interconnect with repeaters," in *Proc. 13th ACM Great Lakes Symp. VLSI*, 2003, pp. 27–32.
[9] M. Ferretti and P. A. Beerel, "Single-track asynchronous pipeline templates using 1-of-N encoding," in *Proc. Des., Autom. Test Eur. (DATE)*, Mar. 2002, pp. 1008–1015.
[10] D. Gebhardt and K. S. Stevens, "Elastic flow in an application specific network-on-chip," *Electron. Notes Theoretical Comput. Sci.*, vol. 200, pp. 13–15, Feb. 2008.
[11] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.
[12] A. Iyer and D. Marculescu, "Power-performance evaluation of globally asynchronous, locally synchronous processors," in *Proc. Int. Symp. Comput. Arch. (ISCA)*, Anchorage, AK, May 2002, pp. 158–168.
[13] A. M. Lines, "Pipelined asynchronous circuits," M.S. thesis, Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, 1998.
[14] M. Pedram, Q. Wu, and X. Wu, "A new design of double edge triggered flip-flops," in *Proc. Des. Autom. Conf.*, Feb. 1998, pp. 417–421.
[15] S. J. Piestrak, "Membership test logic for delay-insensitive codes," in *Proc. Int. Symp. Adv. Res. Asynch. Circuits Syst.*, Mar. 1998, pp. 194–204.
[16] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*. Boston, MA: Addison Wesley, 1980, ch. 7.
[17] M. Singh and S. M. Nowick, "High-throughput asynchronous pipelines for fine-grain dynamic datapaths," in *Proc. 6th Int. Symp. Adv. Res. Asynch. Circuits Syst.*, Apr. 2000, pp. 198–209.
[18] M. Singh and S. M. Nowick, "Mousetrap: High speed transition-signaling asynchronous pipelines," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 6, pp. 684–698, Jun. 2007.
[19] K. S. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. J. Myers, K. Y. Yun, R. Kol, C. Dike, and M. Roncken, "An asynchronous instruction length decoder," *IEEE J. Solid-State Circuits*, vol. 36, no. 2, pp. 217–228, Feb. 2001.
[20] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan Kaufmann, 1999.
[21] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.
[22] B. D. Winters and M. R. Greenstreet, "A negative-overhead, self-timed pipeline," in *Proc. Int. Symp. Asynch. Circuits Syst.*, Apr. 2002, pp. 37–46.
[23] B. D. Winters and M. R. Greenstreet, "Suring: A robust form of wave pipelining using self-timed circuit techniques," *Microprocess. Microsyst.*, vol. 27, no. 9, pp. 409–419, Oct. 2003.

Kenneth S. Stevens (S'83–M'84–SM'99) received the the B.A. degree in biology and the B.S. and M.S. degrees in computer sciencefrom the University of Utah, Salt Lake City, in 1982, 1982, and 1984, respectively, and the Ph.D. degree in computer science from the University of Calgary, Calgary, Calgary, AB, Canada, in 1994.

He is an Associate Professor with the University of Utah. Prior to this position, he worked with the Strategic CAD Lab, Intel, Hillsboro, OR, an Assistant Professor with the Air Force Institute of Technology, with Hewlett Packard Labs, and with Fairchild Labs, AI Research. His research interests include asynchronous circuits, VLSI, architecture and design, hardware synthesis and verification, and timing analysis.



Pankaj Golani received the B.S. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2003 and the M.S. and Ph.D. degrees from University of Southern California, Los Angeles, in 2005 and 2009, respectively.

His research interests include high-performance and low-power asynchronous VLSI design.

Peter A. Beerel received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, in 1989, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1991 and 1994, respectively.

He joined the Department of Electrical Engineering-Systems, Viterbi School of Engineering, University of Southern California (USC), in 1994, where he is currently an Associate Professor. He was on leave of absence from USC between June 2002 to September 2004, during which time he served as Vice-President of CAD and Verification at Fulcrum Microsystems. He was also the Faculty Director of Innovation Studies at the USC Stevens Institute for Innovation from 2006 to 2008. He is currently on a second leave of absence to start his own company, TimeLess Design Automation, commercializing asynchronous VLSI tools and libraries. His research interests include a variety of topics in CAD and asynchronous VLSI design.

Dr. Beerel has been a member of the technical program committee for the International Symposium on Advanced Research in Asynchronous Circuits and Systems since 1997, was program cochair for ASYNC'98, and was general cochair for ASYNC'07. Dr. Beerel was recipient of a VSoE Outstanding Teaching Award in 1997 and the VSoE Junior Research Award in 1998. He was a recipient of a National Science Foundation CAREER Award and a 1995 Zumberge Fellowship. He was also co-recipient of the Charles E. Molnar Award for two papers published in ASYNC'97 that best bridged theory and practice of asynchronous system design, and was a co-recipient of the Best Paper Award in ASYNC'99. He was the 2008 recipient of the IEEE Region 6 Outstanding Engineer Award for significantly advancing the application of asynchronous circuits to modern VLSI chips.