# SAS: Source Asynchronous Signaling Protocol for Asynchronous Handshake Communication Free From Wire Delay Overhead

Shomit Das Vikas Vij Kenneth S. Stevens
University of Utah

***Abstract*— Asynchronous handshake protocol communication is accomplished by sending data down a communication link coupled with data validity information. Flow control is established by acknowledging the receipt of data, thereby enabling transmission of new data down the link. Handshake protocols operate at target cycle times based on system operational requirements. When the communication delay down wires increases beyond a certain point, the latency in sending the request and acknowledge signals across the link becomes longer than the target cycle time. This reduces the communication bandwidth below the desired value. This deleterious effect is particularly conspicuous on long links and network-on-chip communication. A method of enabling full communication bandwidth on wires with arbitrary delay when employing handshake communication is provided. This method supports end-to-end communication across links with arbitrarily large but finite latency without limiting the bandwidth, so long as line variation can be reliably controlled. This paper introduces the new SAS protocol, provides an efficient implementation, and reports the resultant significant energy and bandwidth improvements over conventional handshaking methods.**

## I. Introduction

Latency down a communication link on an integrated circuit is dependent upon the resistance, capacitance, and current carrying capabilities of the wires. Each new process generation scales the technology down by reducing the cross sections of the wires while simultaneously placing them closer together. This increases the number of communication links in a fixed millimeter squared area of an integrated circuit. However it also modifies the signal carrying properties of the wire. In a scaled wire, capacitance remains about the same but resistance substantially increases due to the reduced cross sectional area. The increased resistance produces a relative increase in the communication latency down a fixed length of wire [1]. Increasing delay and energy properties on interconnect pose design challenges, particularly with global wires or network-on-chip interconnect.

Communication employing asynchronous handshake protocols is accomplished by sending data accompanied with validity information down a communication channel. Flow control is established by acknowledging the receipt of the data, thereby enabling the transmission of new data. The handshake signals are generated using pipelined control logic that implements the protocol and synchronize two adjacent channels. The frequency of communication is determined by the delay of the control logic plus the latency down the communication wires. Maximum operating frequency for any specific controller design is established when the pipelined controllers are physically adjacent to each other, as in a first-in first-out buffer (FIFO). Increasing distance between pipeline stages increases wire delay, and decreases the operating frequency of the system.

The direct consequence of wire latency is easily observed when employing asynchronous request acknowledge handshake based communication. Every additional picosecond of wire delay due to controllers being placed farther apart directly results in at least two picoseconds of degradation in the cycle time (1ps each for request and acknowledge). As the communication distance between control elements increases, the communication overhead increases, with a commensurate decrease in operating frequency and communication bandwidth. At some wire length, the communication delay eventually exceeds the desired performance target.

Traditional solutions to this problem reduce the impact of wire overhead. This is accomplished by employing two-cycle communication, which reduces the number of transient communications down the channel in half compared to four cycle protocols, and placing pipeline stages closer together, which reduces communication latency [2], [3], [4].

A new protocol called source asynchronous signaling (SAS) is provided for asynchronous handshake based communication. Rather than mitigate the wire overhead, this protocol is completely independent of wire delays. Thus the SAS link can achieve the same bandwidth as a traditional pipelined link shown in Fig. 1. The SAS protocol therefore allows for high throughput asynchronous communication for channels with large wire latency. A method of implementing SAS is provided, and results are evaluated and compared against traditional protocols.

The SAS protocol decouples the request and acknowledge handshaking signals in such a way that multiple request operations may occur without an acknowledgment operation, and multiple acknowledgment operations may occur without an intervening request operation. Such communication by definition can not be delay insensitive as handshake events are not directly acknowledged. This results in a necessary set of relative timing constraints that must hold for the circuit to be functional as well as to perform at the desired bandwidth without stalling [5].

## II. Background

### A. Related Work

This work is similar to clocked "source synchronous" signaling [6]. The authors are aware of unpublished work as far back as the mid 1980s when Hewlett Packard began transmitting the clock along with data between memory chips and microprocessor chips. This solved the problem of distributing low skew clocks in systems where data is transmitted over

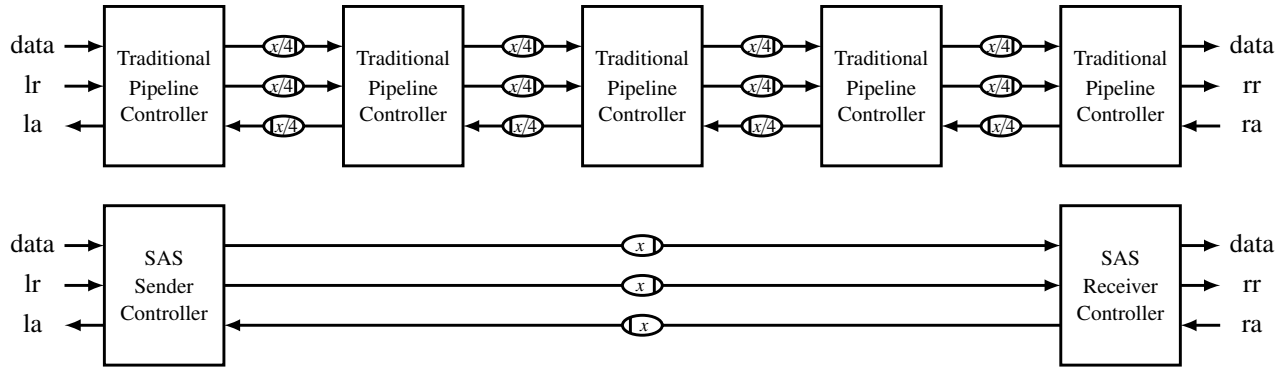


Fig. 1: Traditional and SAS communication pipelines with large wire delay of $x$.

long distances. Similar properties exist today on a single die due to increasing wire delays and transistor counts. Thus source synchronous signaling is relevant for today's system-on-chip and network-on-chip designs [7], [8], [9].

Clocked source synchronous signaling is challenging for two reasons: it creates timing relationships between the sender and receiver nodes, and it necessitates detailed evaluation of the signal carrying properties of the wires. Applying source synchronous concepts to asynchronous handshake protocols is no different; timing relationships and signal integrity become central issues. This work defines all timing constraints necessary for the SAS protocol as applied to the design presented in this work.

The topic of signal integrity is largely left as related work. However, due to its fundamental importance we briefly touch on key aspects related to SAS. A Robust method of repeater deployment on long wires must be used to maintain linear wire delay, reduce variation, and maintain signal fidelity. The critical repeater distance is the maximum distance between line drivers that must be maintained to ensure linear delay and signal fidelity. In several 65nm technologies this distance is approximately 555 $\mu$m. First order equations for communication latency and variation have been developed and validated against SPICE showing approximately 15% maximum error [4]. These models can be used to quickly estimate long communication link robustness to variation, energy, and latencies for any particular physical design configuration.

Since the SAS protocol is independent of wire delay, one can theoretically set channel frequencies to arbitrarily high values. Wave pipelining can result for very long wires and/or very high frequencies. In such cases multiple signals are concurrently in flight down a communication channel. Wave pipelining requires more care in physical design of the communication channel. However, multiple researchers have shown that high speed wave pipelined signaling is possible with proper engineering [10], [11], [12].

In order to put wave pipelining and signal integrity issues in perspective, consider our highest frequency network-on-chip design operating at 2.6 GHz in a 65nm process [13]. The delay in picoseconds of well managed interconnect for this process is modeled with the linear regression equation $len/10+16$, were $len$ is the wire length in microns [14], [15]. This equates in the the network-on-chip router to the time-of-flight down a 3.69mm link. At that distance a skew of 150ps (a 39% variation) in either a delay-insensitive or bundled data protocol on a SAS channel still provides a very robust margin of 85ps.

Previous work defining a method for asynchronous handshake based source synchronous signaling has been performed [16]. That work contains two channels, one for forward communication and one for backward communication, each with a FIFO. The design from [16] requires interface logic in order to connect traditional handshake protocols to the dual-channel internal protocol. The specifics of the internal channel protocol are not defined. In contrast, this work only contains a single channel, and no interface logic is required. The SAS channel protocol is formally defined. One FIFO is placed at each end of the channel. FIFO sizes can be reduced to zero when cycle times are sufficiently large or wire delays sufficiently small. In such a condition the SAS protocol and provided implementation becomes equivalent to a traditional handshake channel. This base case is directly represented in our timing models when no SAS buffers are required.

### *B. Channel Properties*

A channel consists of data, validity information identifying when the data are stable and may be sampled, backward flow control, and the controllers that implement the communication. Traditional handshake communication performance depends on the wire delays in the communication channel and the performance of the pipelined controllers. Data validity is provided by a separate req signal for bundled data channels or encoded in the data for delay insensitive channels. Backward flow control is normally provided with an ack signal. Handshake communication channels are characterized by (i) the channel protocol, (ii) the data encoding employed by the channel, (iii) how and when data are stored, (iv) the amount of concurrency implemented between adjacent channels, (v) the design of the controller that implements concurrency between channels and clocking of the local storage elements.

Asynchronous channel protocols are very simple. Delay-insensitive channels forever repeat the sequence of a valid data encoding, ack, null data encoding, ack. Bundled data channels consist of a forever repeating req and ack events

with the associated data relationship [17]. Channel protocols can take the form of a "return-to-zero" (RZ) or "four-phase" communication, or reduce the number of transitions by using "non-return-to-zero" (NRZ) or "two-phase" communication. There are many examples of different pipelined communication channels and their associated controllers in the literature. Any such controller that can be used to build FIFOs is a candidate and thus can be used in this design.

The SAS protocol introduces a new channel protocol, but otherwise supports all data encodings, data and control relationships, and concurrency between channels. The general design of the controller which interfaces between SAS channels and traditional handshake channels is provided herein. The design is based on FIFO structures with specific signal integrity and timing relationships that must hold. This paper demonstrates the SAS protocol using bundled data encoding. This encoding requires $w+1$ wires to encode $w$ bits of data as well as a communication wire labeled ack to engineer the acknowledge handshaking. The SAS channel is described in this section using a two phase protocol.

### C. Channel bandwidth

Eqn. 1 to 3 represent the maximum communication bandwidth for clocked and traditional asynchronous handshake protocols. Variable $w$ is the width of the data-path in terms of the number of data bits it carries, and delays are converted into frequencies by putting them in the denominator. The maximum bandwidth of a clocked design $B_{clk}$ is proportional to the wire latency down a communication channel ($L_c$) plus the setup time to the flip-flop ($SU$). $B_2$ is the maximum bandwidth of a two phase communication protocol. It is proportional to the sum of $C_{Or}$, the request to acknowledgment responds time of the output channel, plus twice $L_c$. This is because in the two phase protocol, there is one transition each on the req and ack handshake signals. The maximum bandwidth of the four phase protocol $B_4$ is proportional to twice the response time of the output channel $C_{Or}$ plus four times the channel latency $L_c$ due to the four required transitions on the handshake control signals.

$$B_{clk} = w/(L_c + SU) \tag{1}$$

$$B_2 = w/(2 \times L_c + C_{Or}) \tag{2}$$

$$B_4 = w/(4 \times L_c + 2 \times C_{Or}) \tag{3}$$

$$B_{SAS} = w/C_I \tag{4}$$

Assume delays $L_c \gg (SU \approx C_{Or})$, so that $L_c$ is the dominant delay in the above equations, and that the clocked and asynchronous overheads ($SU$ and $C_{Or}$) are effectively identical. In such a case, the clocked design has approximately twice the maximum bandwidth of a two-cycle asynchronous design, and the two-cycle asynchronous design has approximately twice the bandwidth of a four-cycle asynchronous design. This two to four penalty factor for handshake communication can seriously hamper competitiveness in terms of performance, area, or power. SAS bandwidth (Eqn. 4) is only dependent on the input channel frequency and number of data bits. Now if $C_I \approx L_c$ then the SAS channel operates at the same frequency as the clocked design. Thus the SAS protocol overcomes the communication disadvantage, but comes at the cost of adding timing constraints to the design.

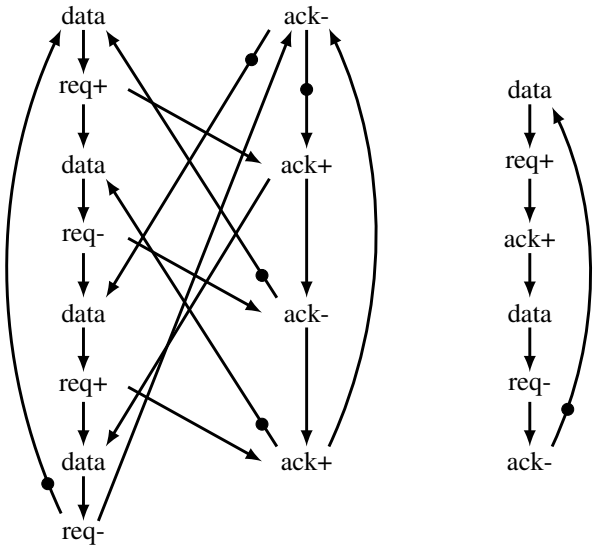


Fig. 2: NRZ SAS with $n = 2$, and traditional channel protocol

## III. Source Asynchronous Signaling

### A. Protocol Specification

Fig. 2 shows the formal representations of a bundled data NRZ SAS channel protocol with a buffer depth of two ($n = 2$) and a traditional bundled data NRZ handshake channel protocol. Each protocol is represented as a petri-net and can be used as a specification for the traditional and SAS communication channels.

Fig. 3 shows an example simulation of a two phase NRZ SAS communication channel where $n = 2$. Rather than requiring that acknowledgment signal arrives before the next request signal, data transfer stalling is delayed by three ack transactions. This allows up to $n+1$ request transactions to occur before an acknowledgment transaction. As a result, the third data value becomes valid and the third req signal asserts indicating data validity, but flow control of the SAS Channel Protocol requires that the data remain valid and the third data transfer transaction does not complete until after the first acknowledge transaction (ack) occurs. Thus for the SAS channel communication protocol, the acknowledge flow control is shifted based on the number of data items $n$ that can be buffered in the SAS sender and SAS receiver elements.

### B. Implementation of SAS Channel and Interfaces

SAS sender controllers interface a traditional channel protocol to the SAS Channel Protocol; SAS receiver controllers perform the dual operation. One can write a petri-net specification to synchronize a SAS channel and a traditional channel protocol in order to build SAS sender and receiver controllers. Our experiments synthesizing such specifications resulted in complex and slow designs. We then realized that

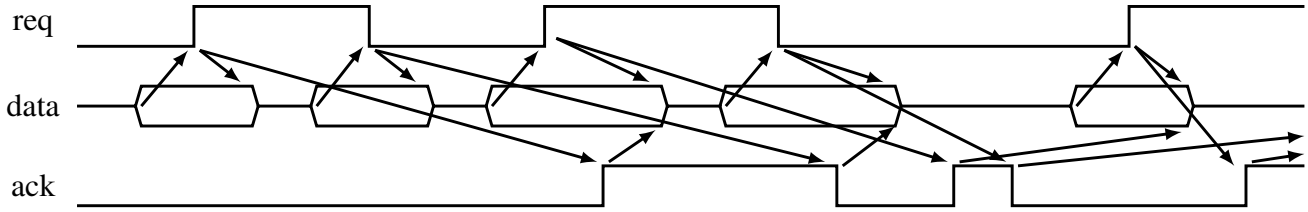


Fig. 3: SAS two-phase handshaking protocol with $n = 2$

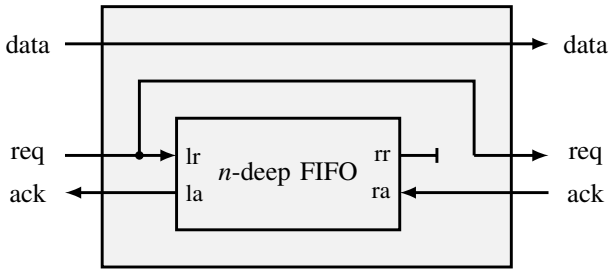


Fig. 4: SAS Sender Block

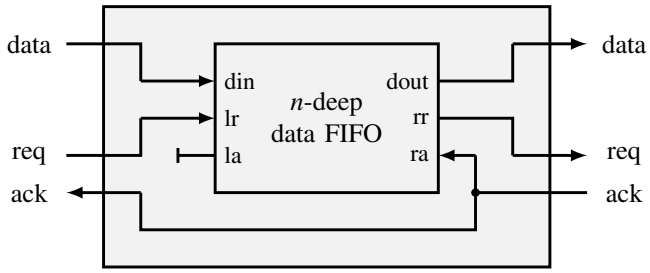


Fig. 5: SAS Receiver Block

the concurrency between the channels very closely mimics the behavior of a FIFO.

Fig. 4 shows the block diagram representation of the bundled data SAS sender controller. The controller consists of an $n$-deep first-in first out (FIFO) that interfaces between a traditional handshake communication channel as an input and a SAS communication channel as the output. SAS controllers may use bundled data or delay-insensitive channels; however this design does not work for GasP or single track communication.

The function of the FIFO is to record how many data transfers have occurred, and to not complete any transactions on the input channel that would exceed $n$ transactions on the SAS communication channel. This is accomplished by implementing an $n$-deep FIFO to communicate between the traditional communication channel and the SAS communication channel. If $n = 2$, then two complete request acknowledge transactions can occur unimpeded on the input channel, and the corresponding data are sent down the SAS output channel. A third transaction may begin and the data transfer down the SAS communication channel would be initiated. However, the transaction would not complete until an acknowledge transaction occurs on the SAS communication channel. Since the depth of the FIFO is two, and two tokens have been placed in the FIFO, the third request transaction on the input port will not be acknowledged until an acknowledgment is received on the SAS channel.

Note that the data and the data validity information (req) on the traditional input channel are the same as the data and data validity information on the SAS communication channel. Other logic may be placed on these signals for various reasons such as to improve signal fidelity, delay the timing reference, change the data and timing encoding, or other standard modifications. The output request (rr) from the FIFO is left unconnected. This results in timing assumptions that must hold for the circuit to operate correctly. All transitions on the rr output of FIFO must occur before an associated response occurs on the acknowledge signal on the SAS channel that connects to the ra signal on the FIFO. This requires that the delay between edges on the ack signal is greater than the response time of FIFO.

The SAS sender element is not limited to bundled data protocols. Dual-rail, m-of-n, LEDR, or any other data encoding may be used in the channels and FIFOs. Likewise any type of asynchronous handshake controller that can implement a FIFO may be used so long as the design meets the SAS timing requirements imposed upon the design.

The SAS receiver controller is represented in Fig. 5. This circuit is in many ways the dual of the SAS sender controller. This element contains a data FIFO that interfaces an input SAS communication channel with a traditional output asynchronous handshake channel. The input SAS channel and traditional output channel may use any data encoding, data transfer protocol, or protocol concurrency.

The function of the FIFO is to buffer and output the data received on the input SAS channel to the output channel. The size of the FIFO will be $n$, and this will be the same depth as the sender FIFO.

The acknowledge signal (ack) on the output handshake channel and the SAS communication channel are the same signal. The input acknowledgment (la) from the FIFO is left unconnected. This results in timing assumptions that must hold for the circuit to operate correctly. All transitions on the acknowledge signal la on the FIFO input must occur before the next transition on the req signal from the SAS communication channel. This requires that the delay between edges on the SAS Channel req signal is greater than the response time of the FIFO.

Fig. 6 shows the block diagram of a complete SAS communication system with interfaces to traditional communication channels. The wires across the SAS Channel may be very long with substantial delay. Both the request

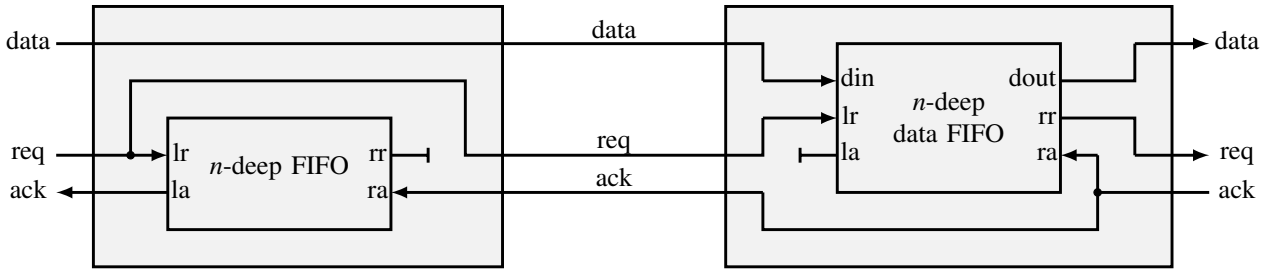


Fig. 6: Complete SAS channel with sender and receiver blocks

and acknowledge timing signals across this channel are not directly acknowledged. Therefore care must be taken to ensure that the fidelity of the signals and the relationship between the data and the timing signals hold across this channel. Repeaters will be inserted on the wires if they are longer than the critical repeater distance to ensure signal fidelity. Wave pipelining may occur on the SAS channel.

## IV. SAS Models

For a correctly designed SAS communication system, the maximum bandwidth $B_{SAS}$ is expressed in Eqn. 4 where $C_I$ is the cycle time of the channel. Note that this equation is strictly dependent upon the target data frequency to be sent down the channel, not upon wire delays as is the case with traditional communication channels as defined by Eqn. 2 and 3. Thus the SAS channel provided can achieve wire overhead free communication down a long communication channel. However, for the SAS communication system to operate properly, a number of constraints must hold, which are now defined.

| | | |
|---|---|---|
| $L_{SAS}$ | Latency of a repeated wire down SAS Channel | (5) |
| $n$ | Depth of sender/receiver FIFOs | (6) |
| $L_{f_S}(n)$ | Forward latency of sender FIFO as a func. of $n$ | (7) |
| $L_{b_S}(n)$ | Backward latency of sender FIFO as a func. of $n$ | (8) |
| $L_{f_R}(n)$ | Forward latency of receiver FIFO as a func. of $n$ | (9) |
| $L_{b_R}(n)$ | Backward latency of receiver FIFO as a func. of $n$ | (10) |
| $C_I$ | Minimum cycle time of the input channel | (11) |
| $C_O$ | Minimum cycle time of the output channel | (12) |
| $C_{Or}$ | Output channel req-to-ack response time | (13) |
| $C_{Sf}$ | Maximum cycle time of SAS sender FIFO | (14) |
| $C_{Rf}$ | Maximum cycle time of SAS receiver FIFO | (15) |

The forward and backward latency (Eqn. 7–10) of all FIFO designs are a function of the depth $n$ of the FIFO. For linear FIFOs, the latency is $n$ times the latency of each stage in the FIFO. Many designs exist that reduce the forward and backward latencies, such as parallel, tree and square configurations [18]. In these designs the latency is not calculated as $n$ times latency per stage; rather there is a more complicated function to determine latency. For example, in an asynchronous tree FIFO, the latency is to the first order $\log_2 n$ times the latency per stage [19]. For the latency values used in this document, the forward and backward latency values $L_{b_S}(n)$, $L_{f_R}(n)$, and $L_{b_R}(n)$ for each FIFO calculate the first order approximation of the latency based on the FIFO design and structure which is a function of $n$.

Overhead free communication only occurs when the channel operates at frequency $1/C_I$ without stalling. A few fundamental conditions must hold for this to be the case based on the above variables.

$$C_I \geq C_O \geq C_{Sf}, C_{Rf} \tag{16}$$

The cycle time of the input and output channels must be greater than or equal to the FIFO cycle times. Otherwise the FIFO will stall the input or output channel(s). Likewise, the cycle time of the input channel may not be less than the cycle time of the output channel, otherwise the input channel will eventually stall under full bandwidth traffic. Since the input channel request drives the SAS receiver FIFO, and the request transaction is not acknowledged by the FIFO, this FIFO must operate faster than the input channel. Likewise, the SAS sender FIFO handshake with the output channel is not acknowledged and the similar condition holds.

The cycle time of input channel $C_I$ is the base factor all equations depend upon, as it dictates the frequency of operation of the SAS communication system. It represents the desired operational frequency for the design. This cycle time is based on the input stage controller and SAS receiver FIFO delays, input channel wire delays, and the rate at which other pipeline stages limit data to that channel. Every other component must operate at least that fast.

The following two values are necessary to define the remaining constraints of a SAS communication system. They are based on the fundamental delays and properties of the links and FIFOs defined above.

| | | |
|---|---|---|
| $T_{S_n}$ | Time to fill the sender FIFO with $n$ tokens | (17) |
| $T_{R_{SAS}}$ | Response time of the SAS system | (18) |

The response time $T_{R_{SAS}}$ is the time from when one data transaction is asserted on the input channel until the entire SAS communication system becomes idle and the sender FIFO is empty when no new tokens are added to the system and none of the channels stall.

Variables $T_{S_n}$ and $T_{R_{SAS}}$ can be represented using values from Eqn. 5 to 15. The SAS receiver FIFO will stall the input channel on transaction $n+1$ if no acknowledge transaction occurs on the SAS channel. The response time across a SAS communication system is approximated by the latency down the SAS channel $L_{SAS}$, the forward latency of the receiver FIFO $L_{f_R}(n)$, the response time of the output channel $C_{Or}$, the latency back down the SAS channel, and the backward latency down the sender FIFO $L_{b_S}(n)$.

$$T_{S_n} = (n+1) \times C_I \tag{19}$$

$$T_{R_{SAS}} = 2 \times L_{SAS} + L_{f_R}(n) + L_{b_S}(n) + C_{Or} \tag{20}$$

If $T_{S_n} < T_{R_{SAS}}$ the input channel will stall because FIFO in the SAS sender element is full and cannot accept more transactions until the response time $T_{R_{SAS}}$ occurs allowing a new transaction to be stored in FIFO. Thus the following equation must hold for SAS system to operate without stalling.

$$T_{S_n} \geq T_{R_{SAS}} \tag{21}$$

Substituting in the values for the above equations results in the following fundamental inequality for SAS systems. The minimum buffer size $n$ for any SAS system design can be calculated from Eqn. 22 when a valid solution exists. A valid SAS solution will exist if the left side of the inequality increases as $n$ grows. This occurs when the cycle time of the input channel is greater than the sum of the forward latency of the receiver FIFO and the backward latency of the sender FIFO when one more stage is added to each.

$$(n+1) \times C_I - (L_{f_R}(n) + L_{b_S}(n)) \geq 2 \times L_{SAS} + C_{Or} \tag{22}$$

This inequality shows that the depth of the FIFOs $n$ is critically dependent on channel cycle time and FIFO latencies. The most efficient designs will select a FIFO in the SAS sender element that has a small backward latency $L_{b_S}$ and a FIFO in the SAS receiver element that has a small forward latency $L_{f_R}$. Another key parameter for SAS communication system designs is the input channel cycle time $C_I$. As the data frequency increases (cycle time $C_I$ decreases) the depth $n$ of the FIFOs increase substantially. The challenge of building a robust system also increases as the input channel frequency increases due to issues of signal fidelity down the SAS channel as well as the tighter constraints on the FIFOs. Area also increases due to the larger FIFO depths $n$.

As the input channel cycle time $C_I$ increases, the inequality demands fewer stages in the FIFOs. When the cycle time $C_I$ becomes sufficiently large in relation to the channel delay $L_{SAS}$, the solution for $n$ becomes zero in SAS Eqn. 22. In this case, there are no SAS FIFOs in the design and Eqn. 2 and 22 are equivalent if $C_I = B_2$. The SAS circuit also becomes identical to a traditional 2-phase bundled data pipeline. This makes perfect sense, because the SAS timing constraints ensure that the unconnected FIFO signals obey a proper traditional handshake protocol, but they are no longer necessary once the FIFOs are no longer needed.

Based on Eqn. 22, some SAS communication system designs will not produce valid solutions. In practice this has been shown to be the case for SAS systems that use simple linear FIFOs. Assume simple linear FIFOs are used in the SAS sender and receiver blocks. These FIFOs have the property where the latency grows linearly with the number of stages and where the sum of the forward and backward latencies equals the FIFO cycle time. Therefore one can let $L_{f_R}(n) = n \times L_f$ and $L_{b_S}(n) = n \times L_b$ and $C_{lf} = L_f + L_b$ where $L_f$ and $L_b$ are the latencies of a single pipeline stage respectively and $C_{lf}$ is the cycle time of the FIFOs. Assume input channel frequency $C_I$ is equal to the FIFO cycle times and $C_{lf} = C_{Sf} = C_{Rf}$, which is valid according to Eqn. 16. Substituting these values into Eqn. 22 yields $C_I \geq 2 \times L_{SAS} + C_{Or}$. Thus when using linear FIFOs, correct operation can be independent of the FIFO size and the design will not correctly implement the SAS channel protocol. We have validated this with design and simulation examples.

$$L_{f_R} + 2 \times L_{SAS} \geq L_{f_S} \tag{23}$$

$$L_{b_S} + 2 \times L_{SAS} \geq L_{b_R} \tag{24}$$

Two additional timing constraints for SAS communication system are shown in Eqn. 23 and 24. This ensures that on initiation the forward latency when coming out of a stalled state, the empty token can propagate to the input of the sender FIFO by the time the receiver observes the acknowledgment and a new token arrives at the sender FIFO input.

## V. Evaluation

Performance of the SAS communication system is evaluated and compared against a traditional channel employing a common burst-mode four-phase handshaking protocol. The analysis is done for 5,000$\mu$m and 10,000$\mu$m channels in a 65nm process with a critical repeater distance of 555$\mu$m. This corresponds to 18 repeaters over the length of the wire. Data width of both links is 64 bits. The same four-phase linear controller is used for the SAS FIFOs and in the traditional communication channel. The circuits were designed in behavioral Verilog, synthesized using Synopsys Design Compiler, and placed and routed using SoC Encounter. Circuits were simulated for timing and functional correctness using Modelsim with postlayout parasitics back-annotated. Testing was performed using pre-defined input vectors. Various performance parameters including forward latency, cycle time, and throughput were also generated from the simulation along with VCD (Value Change Dump). The simulation VCD file along with the parasitics of the place and routed design was used to calculate the power numbers for each design by PrimeTime PX.

### A. Bandwidth

According to simulation results the unpipelined wire of length 10,000$\mu$m can support a bandwidth of 222 Mbps. This

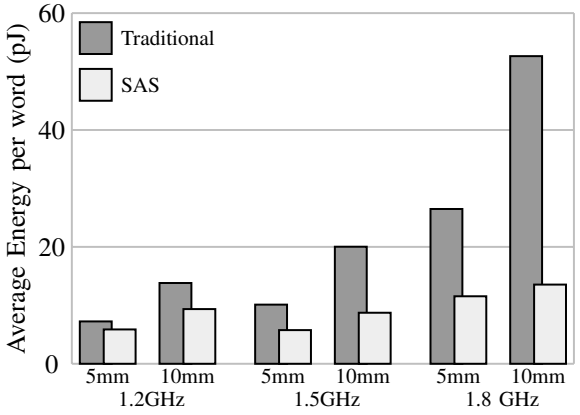


Fig. 7: Energy comparison: traditional vs SAS links

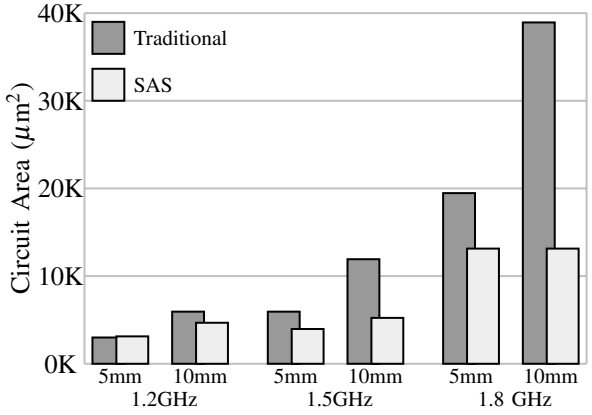


Fig. 8: Area comparison

TABLE I: Bandwidth and energy comparison

| Channel | $n$ | Energy (pJ) | Bandwidth (Gbps) |
|---|---|---|---|
| Traditional | 18 | 16.66 | 88.03 |
| SAS | 10 | 13.51 | 116.36 |

bandwidth is far too low to be practical for the evaluation of the conventional handshake channel in actual communication link of that distance. Pipelining has to be included to obtain a fair appraisal of handshake communication in a real world system-on-chip. One example is to consider the case where all repeaters have been replaced with pipeline latches giving a pipeline link length of 555 $\mu$m. The corresponding bandwidth of a 64-bit link is provided in Tab. I where $n$ refers to the number of pipeline latches along the link and depth of the SAS FIFOs. Note that the bandwidth can be improved by changing the pipeline granularity. However, this comes at a cost of energy and area. We shall investigate these scenarios later in the section.

The bandwidth of the SAS communication channel is independent of the wire latency, and therefore the link length. The rate of data transmission across the channel is determined only by how fast the sender and receiver FIFOs operate. Tree FIFOs at the sending and receiving ends with a capacity $n = 10$ are utilized to obtain a successfully operating SAS channel at high bandwidths. Care was taken to ensure that the design met with the constraints outlined in Sec. IV. The bandwidth of the bus is calculated based on the cycle time of these FIFOs and represented in Table I. The energy for transmitting data across each wire in the traditional and SAS cases is also compared. We observe that the SAS communication channel allows data to be transmitted over 30% faster at nearly 20% less energy.

### B. Energy

As discussed earlier, conventional asynchronous handshaking can achieve higher bandwidths if a higher degree of pipelining is applied. Higher bandwidth targets results in a finer pipeline granularity which ultimately leads to a higher energy cost. We compare the energy expended in transferring data to meet different bandwidth targets over wires of length 5,000$\mu$m and 10,000$\mu$m. The wire latency per pipeline stage that meets the bandwidth target is calculated for a four phase asynchronous handshaking channel. The corresponding link length per pipeline stage is derived from that value, which eventually provides the total number of latches required for the entire link. This number is multiplied by the energy consumed by a single pipeline latch, and the product is added to the wire energy across the entire link. Note that at higher bandwidths, the pipeline controllers need to be spaced closer than the repeaters.

In case of the SAS channel, the choice of appropriate FIFO structures were determined using Eqn. 22. The rule of thumb in this case was to choose a FIFO with cycle times less than the target frequency. The sender FIFO was chosen to have a low backward latency, and the receiver FIFO had lower forward latency. In most cases, these criteria were satisfied by a linear FIFO at the sending end and a parallel FIFO at the receiving end. The capacity of the FIFOs, $n$, was calculated based on the SAS equations. The total energy of the entire setup, including the wire energy is depicted in the figures below.

It can be seen that the energy benefit of using a SAS channel is more prominent in longer wires at higher bandwidths. For a 10,000$\mu$m wire, operating at a bandwidth of 1.8 Gbps, the SAS channel consumes 74% less energy per data transfer when compared to the traditional channel.

### C. Area

The same experiment as the one used for energy analysis is used to determine the area benefits of employing the SAS protocol. The added silicon used for extra pipeline stages manifests itself in the added area numbers. On the other hand, SAS has a lumped distribution of silicon along the length of the wire, with major logic cells only at the sending and receiving ends and repeaters spaced at regular intervals. Note that the wire area remains the same; the added benefit is in the silicon area metric. The SAS channel will have fewer pipeline controllers than the traditional channel, but will require more wire repeaters.

Fig. 8 shows the comparison of area cost of the traditional

and SAS communication systems. The 1.8GHz SAS solution for a 5mm and 10mm communication channel required the same number of SAS buffers, and hence these solutions have identical area. It can be seen that the SAS channel implementation results in a reduction of 66% of the silicon area as compared to a traditional four phase protocol for a 10,000$\mu$m wire operating at 1.8 GHz.

### D. Latency

Latency down a communication channel is the sum of the wire latency and the control logic overhead. The use of a large number of pipeline controllers adds a significant delay to data communication over a long wire, especially for high bandwidth requirements. The SAS channel only has a FIFO at the receiving end in the forward path, along with repeaters at regular intervals. This allows for a very low latency transfer of data over the channel. Fig. 9 shows the comparison of forward latency values for the traditional and SAS communication links.

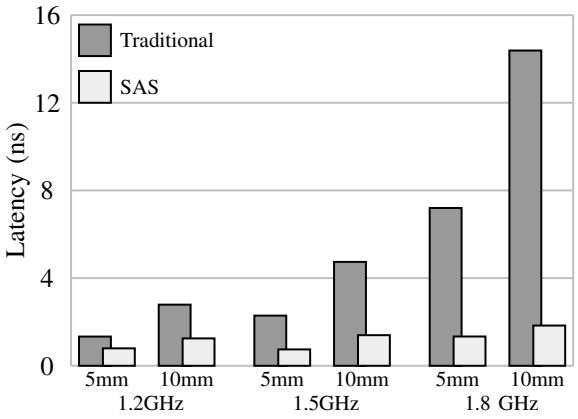


Fig. 9: Latency comparison

## VI. Summary

A novel high throughput asynchronous handshake signaling protocol that is unhindered by wire latency is introduced. Some simple relative timing constraints must hold for the design to operate correctly. The new technique also results in lower latency and area as well as energy savings when compared to traditional handshaking methods that achieve the same bandwidth. For the examples provided in the paper, energy, area, and latency are reduced by 75%, 66%, 77% respectively for a 1.8GHz channel down a 10mm line. The key feature of this protocol is that it allows the bandwidth of data transfer to be independent of the link length. The added latency of longer wires is simply compensated by increasing the capacity of the sender and receiver FIFOs. The SAS channel protocol and design elements provided in this paper apply equally well to two phase or four phase signaling. A SAS communication system also works equally well for bundled data or delay insensitive protocols. Likewise, the design can be automated due to the ability to define timing requirements and communication requirements and to characterize asynchronous FIFOs based on these requirements. Also, any channel protocol, be it more or less concurrent, may be used to implement the FIFOs and SAS communication channels and traditional channels. When channel delays become sufficiently large and signal fidelity can be sufficiently controlled, wave pipelining can be implemented with the SAS channel protocol by allowing multiple transactions to be concurrently in flight down the SAS channel.

## References


[1] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, April 2001.

[2] K. S. Stevens, "Energy and performance models for clocked and asynchronous communication," in *International Symposium on Asynchronous Circuits and Systems*. IEEE, May 2003, pp. 56–66.

[3] R. Ho, J. Gainsley, and R. Drost, "Long wires and asynchronous control," in *International Symposium on Asynchronous Circuits and Systems*, Apr 2004, pp. 240–249.

[4] 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.

[5] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 11, pp. 129–140, Feb. 2003.

[6] R. E. Nikel, "Low cost 400 MHz source synchronous data links," in *Electrical Performance of Electronic Packaging*, 1995, pp. 146–148.

[7] A. T. Tran, D. N. Truong, and B. M. Baas, "A Reconfigurable Source-Synchronous On-Chip Network for GALS Many-Core Platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 897–910, 2010.

[8] T. N. K. Jain, M. Ramakrishna, P. V. Gratz, A. Sprintson, and G. Choi, "Asynchronous Bypass Channels for Multi-Synchronous NoCs: A Router Microarchitecture, Topology, and Routing Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1663–1676, 2011.

[9] A. Mandal, S. P. Khatri, and R. N. Mahapatra, "A fast, source-synchronous ring-based network-on-chip design," in *Design, Automation & Test in Europe (DATE)*, 2012, pp. 1489–1494.

[10] A. J. Joshi, G. G. Lopez, and J. A. Davis, "Design and Optimization of On-Chip Interconnects Using Wave-Pipelined Multiplexed Routing," *IEEE Transactions on Very Large Scale Integration*, vol. 15, no. 9, pp. 990–1002, 2007.

[11] P. Teehan, G. G. F. Lemieux, and M. R. Greenstreet, "Estimating reliability and throughput of source-synchronous wave-pipelined interconnect," in *International Symposiun on Networks-on-Chip*. ACM/IEEE, 2009, pp. 234–243.

[12] R. Dobkin, A. Morgenshtein, A. Kolodny, and R. Ginosar, "Parallel vs. serial on-chip communication," in *International Workshop on System Level Interconnect Prediction*. ACM, 2008, pp. 43–50.

[13] D. Gebhardt, J. You, and K. S. Stevens, "Link Pipelining Strategies for an Application-Specific Asynchronous NoC," in *International Symposium on Network-on-Chip (NOCS)*. IEEE/ACM, May 2011, pp. 185–192.

[14] L. P. Carloni, A. B. Kang, S. V. Muddu, A. Pinto, K. Samadi, and P. Sharma, "Accurate Predictive Interconnect Modeling for System-Level Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 679–684, April 2010.

[15] J. You, "Asynchronous Network-on-Chip Design and Evaluation," Ph.D. dissertation, University of Utah, Salt Lake City, UT, USA, December 2011.

[16] J. C. Ebergen, I. E. Sutherland, and R. J. Drost, "Apparatus and method for high-throughput asynchronous communication," US Patent 7 417 993, Aug 26, 2008.

[17] A. Peeters and K. van Berkel, "Single-Rail Handshake Circuits," in *Asynchronous Design Methodologies*, 1995, pp. 53–62.

[18] E. Brunvand, "Low Latency Self-Timed Flow Through FIFOs," in *16th Conference on Advanced Research in VLSI*, UC Santa Cruz, March 1995, pp. 76–90.

[19] H. Han and K. S. Stevens, "Clocked and Asynchronous FIFO Characterization and Comparision," in *17th International Conference on Very Large Scale Integration*. IFIP/IEEE, Oct. 2009.