On-chip Bidirectional Wiring for Heavily Pipelined Systems using Network Coding

Kalyana C. Bollapalli*, Rajesh Garg‡, Kanupriya Gulati*, Sunil P. Khatri*
* Department of ECE, Texas A&M University, College Station TX 77843
‡ Intel Corporation, Hillsboro, OR 97124

***Abstract*— In this paper, we describe a low-area, reduced-power on-chip point-to-point bidirectional communication scheme for heavily pipelined systems. When data needs to be transmitted bidirectionally between two on-chip locations, the traditional approach resorts to either using two unidirectional wires, or to using a single wire (with a unidirectional transfer at any given time instant). In contrast, our bidirectional communication scheme allows data to be transmitted *simultaneously* between two on-chip locations, with a *single wire* performing the bidirectional data transfer. Our approach borrows ideas from the emerging area of network coding (in the field of communication). By utilizing coding units (which also serve the purpose of buffering the signals) along the wire between the two endpoints, we are able to achieve the same throughput as a traditional approach, while reducing the total area utilization by about 49.8% (thereby reducing routing congestion), and the total power consumption by about 11.5%. The area and power results include the contribution of routing wires, coding units, drivers, the clock distribution network and the required reset wire. Our bidirectional communication approach is ideally suited for heavily pipelined data intensive systems.**

I. Introduction

Recently, the demand for high performance on-chip implementations of data intensive applications has been on the rise. This has been largely in response to the dramatic increase in the demand from the gaming, DSP and multimedia sectors. Circuit blocks such as filters, streaming media, encryption and decryption blocks are increasingly being integrated on the same large IC die or SoC. Such applications exhibit a high tolerance to pipeline latencies, while demanding extremely high performance in terms of computational throughputs.

In traditional implementations of such systems, it is often the case that two blocks need to conduct a bidirectional data transfer between them. In other words, block A needs to transfer data a to block B, while B needs to transfer data b to A. Both of these are instances of unicast communication. In general, a and b can both be n bits wide, where n is the width of the data bus between A and B. Without loss of generality, the discussion in this paper will assume that $n = 1$, although our results are provided for $n = 64$. There are two traditional approaches that have been used to achieve the communication between A and B.

- Two wires can be routed between A and B, one for the transfer of a from A to B, and the other for transferring b from B to A. This method requires a higher wiring area, but allows 2 bits of transfer per clock period.
- A single wire could be routed from A to B. Assuming that the transfer rate desired in both directions is equal, we would transfer a in one clock cycle and b in the next clock cycle. This would require tristateable drivers at both ends of the wire. This method has a lower wiring area compared to the previously described technique, but allows only one bit of transfer per clock period. However the inability to insert buffers restricts this approach to short wires, which is a serious restriction.

The first approach achieves twice the throughput of the second, and can be employed for long wires. Hence it is the method which we compare our approach to, since a high throughput over long wires (typically buses) is desired in data-intensive applications. However, the first approach requires two wires to be routed between A and B. In the sequel, we refer to the first of the above listed approaches as the *traditional approach*.

This paper is motivated by the *goal of achieving 2 bits of transfer per clock period (one bit from A to B and the other bit from B to A), while using a single wire*, thereby helping alleviate routing congestion in the IC. We borrow ideas developed in the area of network coding in the field of communication theory. By using intermittent *coding units* in the wire between A and B, we achieve this goal. Each coding unit is a synchronous element which reads incoming data from both directions, nondestructively encodes the data, and broadcasts the coded data in both directions.

Our approach has the following advantages. First, the buffering of the wire between A and B (as required in deep sub-micron technologies) is performed in the coding unit. Thus the area overhead due to the coding logic is quite small in practice. Second, our approach is not limited to short wires since the wire is intermittently buffered by the coding units. Thirdly, twice the amount of data is transferred simultaneously between A and B, while using a *single* wire for the transfer. Fourth, our approach, when compared to the traditional approach (with optimal buffer insertion), achieves $\sim 49\%$ lower total area utilization (hence reducing routing congestion) and $\sim 11\%$ less total power consumption while matching the throughput of the traditional approach. Note that the area and power results include the contribution of routing wires, coding units, drivers, the clock distribution network and the reset wire, used in our approach.

The transmission of data between A and B is pipelined in our approach. Hence, our approach is particularly suitable for data intensive bidirectional communication instances, where the transfers are latency tolerant, while requiring extreme high data throughputs and a low area and power utilization. With the increasing complexity and die sizes of DSP and multimedia ICs, such designs need to be heavily pipelined, making our technique more suitable with each process generation. Recent Network-on-Chip (NOC) topologies such as the flattened butterfly [1] and Multidrop Excess Channels [2] (MECS) could benefit from our approach as well.

The key contributions of this paper are:

- This is the first paper to apply the concept of unicast network coding to the VLSI context.
- For heavily pipelined VLSI systems, for the same latency, our approach improves on the traditional wiring approach

978-1-4244-5028-2/09/$25.00 ©2009 IEEE

in terms of area (by about 49.8%) as well as total power (by about 11%).

The remainder of this paper is organized as follows. Previous work is described in Section II, while Section III provides the details of our bidirectional communication approach. In Section IV we present results from experiments which we conducted to implement our approach, along with comparisons with the traditional (optimal buffer insertion based unidirectional communication) approach. We conclude in Section V.

II. Previous Work

When data needs to be multicast in a communication network, network nodes are typically used to simply duplicate and forward incoming messages. Recently, there has been significant interest in techniques in which network nodes do more than simple duplication and forwarding. In other words, these nodes are used for encoding, and are allowed to generate new symbols by combining incoming data symbols. This paradigm is referred to as *network coding*. Network coding has been studied in the context of networking and communication [3]. It has been shown that the capacity of a multicast network can be increased by network coding [4], [5].

Several recent research efforts have shown significant improvements by applying network coding in the communication domain [6], [7], [8], [9], [10]. These papers have addressed important network coding related topics such as network capacity, algorithms for network coding, network information flow, coding advantage, etc. Establishing efficient multicast connections is one of the central problems in network coding. In the *multicast network coding problem* a source s needs to deliver h symbols to a set K of k terminals over the underlying communication graph G. It was shown in [3] and [11] that the capacity of the network (i.e. the maximum number of packets that can be sent between s and K per time unit) is equal to the minimum size of a cut that separates the source s from a terminal $t \in K$. In other words, a source s can send h symbols to all terminals in K if and only if the total capacity of all edges in any cut that separates s and $t \in K$ is at least h. In [11], it was proved that *linear network codes* are sufficient for achieving the capacity of the network. In a subsequent work, [12] developed an algebraic framework for network coding. This framework was used by [13] to show that linear network codes can be efficiently constructed through a randomized algorithm. The authors of [14] proposed a deterministic polynomial-time algorithm for finding feasible network codes for multicast networks.

The concept of multicast network coding has been applied to improve the routability of VLSI designs in [15]. In a VLSI IC setting, any signal with a fanout of 2 or more is an example of a multicast signal. Among such signals, a restricted subset can benefit from network coding. By employing network coding in such situations, the authors of [15] demonstrated that an improvement could be achieved in routability, power (about 7.7%), wirelength (about 8%) and active driver area as well (2.2%). There is a modest delay increase (2.5%) that is incurred.

In [16], the authors applied the idea of multicast network coding to the problem of FPGA routing. The technique consisted of enumerating the multicast routing scenarios, filtering out those that would benefit from network coding, and then implementing the coding using free LUTs in the design. In about 37% of the examples, the maximum number of routing tracks was reduced, with a small decrease (3%) in total wirelength, and about 15% decrease in worst case routing delay. In [17], the authors use transition based encoding for an on-chip bus, to reduce energy. However, this approach does not achieve bidirectional data transfer in a single wire.

Recently, it was shown that network coding can also be applied in a unicast wireless communication scenario [18]. The authors demonstrated this approach on a 20-node wireless node testbed, showing that unicast network coding can achieve improved network throughput. The basis of [18] is illustrated in Figure 2. Suppose A needs to communicate data packet a to B, and B needs to communicate b to A, over a wireless channel, using an available relay unit. The conventional wireless communication scenario is shown in Figure 1. The entire transfer requires 4 transmissions, as shown in the sequence in Phases 1 through 4. However, in the unicast network coded example of Figure 2, the transfer requires 3 transmissions. First a and b are transmitted to the relay unit. The relay unit in this case does more than store-and-forward, and is therefore referred to as a *coding unit*. In particular it broadcasts the XOR of a and b. When A receives this broadcast data, it can recover the data transmitted by B by XORing the received data ($a \oplus b$) with what it had originally transmitted (a). B similarly recovers the data intended for it as well.

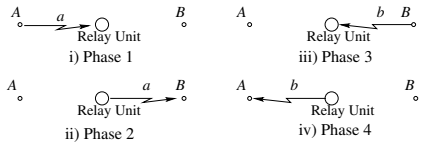

Fig. 1. Conventional Wireless Communication (without Network Coding)

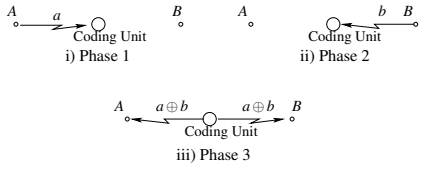

Fig. 2. Network Coding in Communication Systems [18]

In this paper, we apply the above unicast network coding idea to the VLSI domain. When two blocks A and B need to conduct a bidirectional data transfer, we have an instance of unicast communication. We show that in such a situation, we can achieve 2 bits of transfer per clock period (one bit from A to B and the other bit from B to A), while *using a single wire*. Similar to the coding unit of Figure 2, we utilize intermittent coding units in the wire between A and B. Each coding unit is a synchronous element which reads incoming data from both directions, encodes (XORs) the received data, and broadcasts the coded data in both directions. In general, a and b can both be n bits wide, where n is the size of the data transfer between A and B.

A prior approach [19] which attempts to achieve this goal (of bidirectional data transfer using a single wire) uses circuit level techniques. By using current mode signaling, bidirectional communication was achieved. Both the locations A and B have current mode drivers and receivers, allowing for a bidirectional signal transfer. However, the receivers need to perform high precision voltage sensing, making the method complex, and sensitive to process variations. Additionally, the power consumption of [19] is about $100\times$ that of our approach, since it uses current mode signalling. Further, for long wires in Deep Sub-micron (DSM) technologies, buffer insertion is a crucial requirement to reduce wiring delays. The approach of [19] does not address this. Modifying

the approach to incorporate buffer insertion would (at best) require a drastic re-design of the scheme. In contrast, our scheme does not require complex voltage sensing. Also, it incorporates buffer insertion naturally in the formulation.

Another prior approach [20] addresses bidirectional signal communication along a wire by using *boosters*. The insertion of a booster does not break a wire into smaller sections as is the case with buffer insertion. The wire has intermittent boosters, and each booster has an early edge detection circuit, along with a driver which aids the signal transition. However, unlike our approach, the booster method does not allow data to be *simultaneously* transmitted between A and B. Also, the early edge detection circuits in the booster approach are highly susceptible to noise, making it impractical for DSM technologies.

III. Our Approach

In recent VLSI designs, wiring delays have been dominating logic delays [21]. As a consequence, signals on long wires, if buffered, lead to a reduced wiring delay (since the buffered wire segment has a lowered capacitance and resistance) while increasing the logic delay (due to the use of buffers). Thus there is an optimal number (and sizing) of buffers such that the total signal delay is minimized. This is referred to as the buffer insertion problem, and has been widely studied. An excellent survey of this problem can be found in [22].

Suppose the clock period of an IC is T. Even with optimal buffer insertion, the maximum distance d_L that a signal can be transmitted on metal layer L in one clock period T is limited, and does not cover the entire die area. This distance is longer for wires on higher metal layers.

The starting point of our work was to quantify d_L. Using a series of HSPICE [23] sweeps, we found this distance for metal layers of interest. For this experiment, we used 45-nm PTM [24] model cards, and also used wiring parasitics from [24] for a wiring configuration having minimum pitch wires on the metal layer of interest, with ground planes on the metal layers above and below the layer of interest. The maximum distance a signal can be transmitted on metal layers 3 and 4 (assuming a clock speed of 3 GHz) is 7.07 mm. This distance is achieved by optimal buffer insertion (with an optimal number and sizing of buffers) obtained via HSPICE sweeps.

Now we can make two observations. If the total length of the wire on metal layer 3 or 4 is greater than 7.07 mm, then for 3 GHz circuit operation, the data transfer will need to be pipelined. Further, for bidirectional data transfers, 2 wires would be needed in order to achieve maximal throughput using the traditional approach, thereby increasing wiring congestion.

With our network coding based bidirectional data transfer scheme, we need just one wire to achieve the same throughput, thereby reducing routing congestion. Just as in the case of buffer insertion, we would require the transfer to be pipelined.

A. Overview

Our network coding based approach is described next. When high-throughput bidirectional data transfer is required between two locations on the die, we use a single wire, with a number of coding units between these two locations. Figure 3 (i) illustrates 3 successive coding units, with their intervening wiring segments shown. Note that each coding unit is a synchronous circuit element, and adjacent coding units utilize complementary clock phases. Figures 3 (ii) and (iii) illustrate the mechanism of data transfer. When *CLK* is

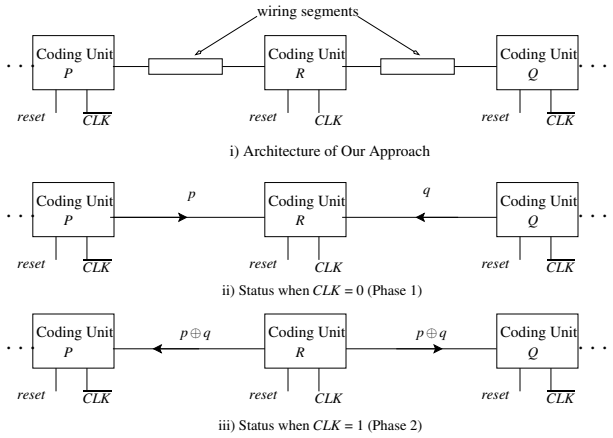

Fig. 3. Network Coding for Bidirectional Data Transfer in VLSI

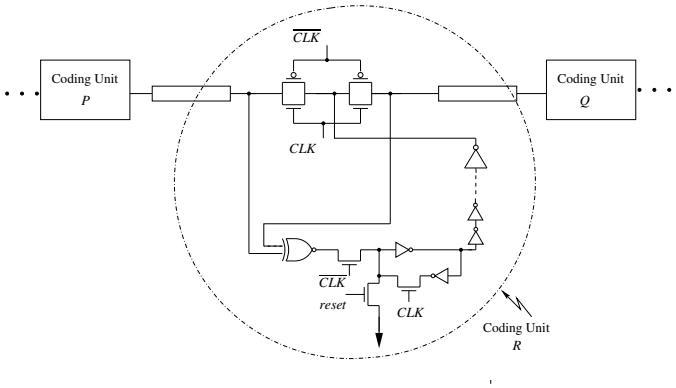

Fig. 4. Circuit Structure of Coding Unit

low (Figure 3 (ii)) coding units P and Q are transmitting, while coding unit R receives. Coding unit P transmits data p and coding unit Q transmits data q. Coding unit R receives both p and q. When *CLK* is high, coding unit R transmits $p \oplus q$, while P and Q receive, as shown in Figure 3 (iii).

Figure 4 illustrates the circuit level diagram of a coding unit. Figure 4 shows the same block level arrangement of coding units as in Figure 3, with the coding unit R shown (circled) at the circuit level. During the low phase of *CLK*, the values from coding units P and Q are XORed, and stored in a latch. When *CLK* is high, the latched value is buffered and driven out along the wiring segments towards coding units P and Q. There are always an even number of buffers in the buffer chain in the coding unit. Note that the latch is resettable, a feature which will be discussed subsequently. Also, the total area of the buffer chain dominates the area of the coding unit. Finally, we observe that coding units P and Q have an opposite *CLK* polarity compared to coding unit R.

In this manner, data is transferred bidirectionally (using a single wire) between the two die locations in a pipelined fashion, with each coding block alternating between a receiving and a transmitting functionality during alternating phases of the *CLK* signal.

B. Bidirectional Communication Protocol

We next discuss the implications of our transfer protocol on the endpoints of the wire. Consider the bidirectional data transfer between endpoints A and B whose topology is shown in Figure 5. Table I enumerates the data on the wires during the first 3 clock cycles of communication. The first row indicates the reset state of the communication line.

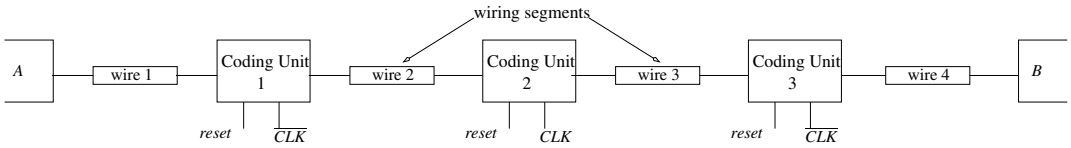

Fig. 5. Bidirectional Data Transfer Protocol Topology

Clock Cycle	Clock Phase	Received at A	Wire 1	Wire 2	Wire 3	Wire 4	Received at B
0	reset	-	0	0	0	0	-
1	high	-	$\overrightarrow{a_1}$	$\overleftarrow{0}$	$\overrightarrow{0}$	$\overleftarrow{b_1}$	-
1	low	0	$\overleftarrow{a_1}$	$\overrightarrow{a_1}$	$\overleftarrow{b_1}$	$\overrightarrow{b_1}$	0
2	high	-	$\overrightarrow{a_2}$	$\overleftarrow{a_1 \oplus b_1}$	$\overrightarrow{a_1 \oplus b_1}$	$\overleftarrow{b_2}$	-
2	low	b_1	$\overleftarrow{a_1 \oplus b_1 \oplus a_2}$	$\overrightarrow{a_1 \oplus b_1 \oplus a_2}$	$\overleftarrow{a_1 \oplus b_1 \oplus b_2}$	$\overrightarrow{a_1 \oplus b_1 \oplus b_2}$	a_1
3	high	-	$\overrightarrow{a_3}$	$\overleftarrow{a_2 \oplus b_2}$	$\overrightarrow{a_2 \oplus b_2}$	$\overleftarrow{b_3}$	-
3	low	b_2	$\overleftarrow{a_2 \oplus b_2 \oplus a_3}$	$\overrightarrow{a_2 \oplus b_2 \oplus a_3}$	$\overleftarrow{a_2 \oplus b_2 \oplus b_3}$	$\overrightarrow{a_2 \oplus b_2 \oplus b_3}$	a_2

TABLE I

BIDIRECTIONAL DATA TRANSFER PROTOCOL

The subsequent rows indicate the value on different wire segments that are part of the communication line between A and B. Each entry in columns 4 through 7 describe the value on the wire segments symbolically with the direction they are being driven as an arrow on the top. During chip reset, all coding units are reset to a 0 value. Now, when CLK = 1, A and B transfer a_1 and b_1 towards coding units 1 and 3 respectively, while coding unit 2 drives out a 0 (the reset value) towards coding units 1 and 3. When CLK = 0, coding units 1 and 3 drive out $(a_1 \oplus 0)$ and $(b_1 \oplus 0)$ respectively to their neighbors. Again when CLK = 1, A and B drive new data a_2 and b_2 towards coding units 1 and 3 respectively, while coding unit 2 drives out $a_1 \oplus b_1$ towards coding units 1 and 3. Finally, when CLK = 0, coding units 1 and 3 drive out $a_1 \oplus b_1 \oplus a_2$ and $a_1 \oplus b_1 \oplus b_2$ respectively. From this received data, A can recover the message sent by B two cycles ago, by XORing the received message $(a_1 \oplus b_1 \oplus a_2)$ with the running XOR of the data it sent out over the last two cycles ($a_1 \oplus a_2$ in this case). Similarly, B can recover the message sent by A two cycles ago, by XORing the received message $(a_1 \oplus b_1 \oplus b_2)$ with the running XOR of the data it sent out over the last two cycles $(b_1 \oplus b_2)$. During this phase of the clock, coding unit 2 receives $(a_1 \oplus b_1 \oplus a_2)$ and $(a_1 \oplus b_1 \oplus b_2)$ and would hence drive out $(a_2 \oplus b_2)$ when CLK becomes 1. It can be observed from Figure I that in subsequent cycles, A (B) can retrieve the data sent by B (A) two cycles previously by XORing their received data with the running XOR of the data transmitted for the last 2 cycles. In this manner, we achieve 2 bits of transfer per clock period (one bit in both directions), while using a single wire and thus reducing routing congestion.

In the above example, the number of coding units m is odd. In this case, the transfer latency is $(m+1)/2$ clocks, and each of A and B need to store a running XOR of the last $(m+1)/2$ values that they transmitted. Note that m can be even as well, in which case the above analysis still holds, with the exception that A and B transmit data on opposite phases of the CLK signal.

The decoding of received messages at A and B requires the computation of an XOR of the received signal with the running XOR of the last $(m+1)/2$ transmitted signals. The latency of the XOR computations at A and B can be hidden (by adding an additional pipeline stage, for instance). Since the communication is heavily pipelined, this does not affect the transfer throughput.

C. Circuit Level Considerations

Note that the overhead of routing the *reset* and *CLK* signals is not high, since the high-throughput on-chip data transfers that commonly occur in high performance SoCs are parallel in nature (typically 32, 64 or 128 bits wide). The *CLK* and *reset* signals are shared between all coding units as shown in Figure 6. Since our scheme reduces the number of data bus wires by a factor of 2, the addition of 2 additional control wires does not appreciably degrade the improvement in the bus wiring area achieved by our method.

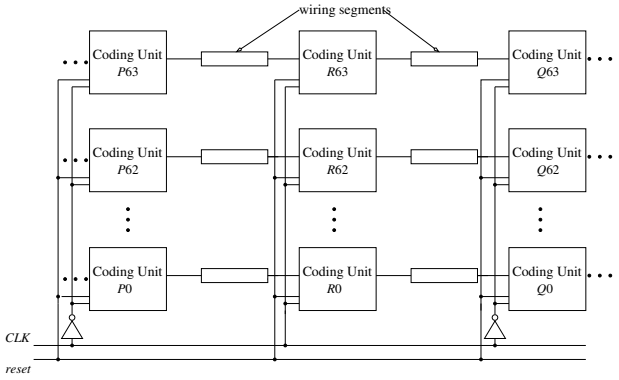

Fig. 6. Network Coding based 64-bit Data Transfer

The number of pipeline stages for our approach is greater than that required by a buffer insertion solution due to two reasons.

- In case of traditional buffer insertion, if the length l between the endpoints is greater than the maximum distance d_L that a signal can be transmitted on metal layer L, then $\lfloor l/d_L \rfloor$ pipeline flip-flops must be inserted. The length of each resulting wiring segment is the distance that can be driven in a clock period T of the design. However, in our approach, the coding units are latch based. Therefore, the length of a wiring segment is the distance that can be driven by a buffer in *half the clock period* $(T/2)$. For this reason the number of wiring segments in our approach is typically larger. Note that this does not reduce the bidirectional transfer throughput.
- We cannot insert traditional buffers or inverters between coding units since a bidirectional transfer is desired on

the wire. As a consequence, the length of the wiring segment between two successive coding units is equal to the distance that can be driven by the driver in the coding unit in half the clock period ($T/2$).

Another observation is that there is never a situation where any coding unit has to charge both its wiring segments when it is driving a logic high value. This is because it drives a logic high value if the incoming signals (p and q in Figure 3) are complements of each other. In such a case, one of the wires is already driven high. Thus a high value need only be driven on one wire. This allows us to have a smaller PMOS device for the last driver stage of the buffer chain in the coding unit.

The coding unit R in Figure 4 drives its output during the high phase of CLK. The output of the final inverter of the buffer chain in the coding unit R reaches its driving value before CLK goes high[1]. As a result, when CLK goes high, the output capacitance of the final inverter of the buffer chain initially undergoes a charge sharing with the capacitance of the wiring segment. This helps the signal of the wiring segment switch faster.

Finally, we note that the buffer chain in the coding unit is largely insensitive to the driver sizing ratio. Traditionally, in an inverter chain, optimal delay is achieved if the output capacitance of any inverter in the chain is about $3\times$ its input capacitance. In our coding unit however, a much larger capacitance ratio can be tolerated, since the output of the final inverter of the buffer chain reaches its driving value before CLK goes high. As a result, delay optimality of the wiring chain is not important, and hence a larger output to input capacitance ratio is tolerable in the inverter chain. The increase in power incurred by such a choice (due to longer signal transition times) is a small fraction of the power consumed in switching the large wiring capacitances.

IV. Experiments

In this section, we compare the area and power characteristics of our approach in comparison with the traditional approach for achieving bidirectional communication. We distinguish between the active area (area for coding units as well as drivers) and the wiring area utilization of both the methods. The total area and total power results in this paper include the contribution of routing wires, coding units, drivers, the clock distribution network and the reset wire.

We conducted many experiments for different total wire lengths (i.e. the distance between endpoints A and B). We varied the number and drive strength of the buffers (for the traditional design) and the coding units (for our approach) and present the results in this section. We also varied the size of the data bus under consideration. Specifically, we conducted experiments for 32-bit, 64-bit and 128-bit data buses. In all our experimental results, we are able to achieve a healthy reduction in total area and power consumption.

Experiments were conducted using a 45-nm PTM [24] process, assuming that wires are driven on Metal4. The clock frequency was assumed to be $3GHz$ for all simulations in this paper. Simulations were performed using HSPICE [23]. Actual process feature dimensions and parasitic values are reported in Table II. In future processing generations, the increasing complexity and die sizes of DSP and multimedia ICs would require the wires on higher metal layer wires to be heavily pipelined as well, making our technique more useful in the future.

[1]This occurs because we require that the final inverter of the buffer chain be stable before CLK goes high, in order to make the design variation tolerant.

Parameter	Value
Metal4 width	0.14 μm
Metal4 space	0.14 μm
Metal4 thickness	0.28 μm
Metal4 R/l	0.5634 $\Omega/\mu m$
Metal4 C/l	$1.41e^{-16} F/\mu m$

TABLE II
Parameters and Their Values

We require that each driven signal, at the inputs of its adjacent receiving coding units, achieve a 90% signal swing within a guardband adjusted timing constraint. This guardband adjusted time is 90% of the time allocated (which is $T/2$ in our case, where T is the clock period utilized). This provides a modest 10% guardband to account for process variations. If this guardband is increased or decreased, the results do not exhibit any qualitative change (in terms of the power and area gains achieved by our approach). Also, we assume that a worst-case capacitive cross-talk exists between neighboring wires in the bus being simulated. Finally, our results account for the delay incurred due to an extra inversion in every alternate coding unit.

Note that the solution to the traditional buffer insertion problem finds the optimal number and optimal size of buffers to minimize the delay of a signal from signal source to destination. However, the problem being solved here is – *given a guardband adjusted timing constraint* we need to find the optimal number and optimal driver size of coding units to minimize a cost function (either area or power).

Table III reports the results comparing total area (wire area plus active area), dynamic and leakage power for both our approach and the traditional buffer insertion approach, for a 64-bit bus. Table III also reports the ratios of our data to that of the traditional method. This ratio is presented as a percentage in braces next to the data of coding buffers. All absolute area numbers are in μm^2, while power numbers are in mW. All *results are iso-throughput*. By this we mean that for our approach as well as the traditional approach, the transfer throughput is same which is $6 \times 64 = 384$ Gb/sec for each entry of Table III. All tables present both the area optimal as well as the power optimal solutions. In Table III , a '-' indicates that a solution does not exist for our approach.

In Table III, we present solutions from both approaches, for varying latencies. Note that the latency of our approach includes the delay of coding units. For example, the 4^{th} row of any table compares the optimal solution for both traditional buffer insertion and our approach, with a latency of 4 clock cycles. The 5^{th} row compares solutions with a latency of 5 clock cycles and so on. These results have been gathered from HSPICE simulations, in which we swept the number and size of the drivers. Note that as mentioned earlier, the number of coding units required by our approach is typically larger than the number of pipeline stages required for buffer insertion. This is observed in the table. Also, for each value of latency where a solution exists for both our approach as well as the traditional approach, we notice that our approach has a systematically lower power consumption (by as much as 45%) for the 64-bit bus. The leakage of our approach is slightly higher when the latency is low, but is comparable to the traditional approach for high latencies (heavily pipelined systems). When the latency is low, the drivers in our coding units are larger, hence resulting in increased leakage.

The total area numbers in Tables III represent the active plus the wiring area. The wiring area for our approach was 2100 μm^2, while that of the traditional buffer insertion approach was 4200 μm^2. From this information, and from the area numbers in Table III, the active area numbers can be

0.75 cm	Optimal Area Solutions						Optimal Power Solution					
	Standard Buffers			Coding Buffers			Standard Buffers			Coding Buffers		
Stages	Area (μm^2)	Leakage (mW)	Dyn. Power (mW)	Area (μm^2)	Leakage (mW)	Dyn. Power (mW)	Area (μm^2)	Leakage (mW)	Dyn. Power (mW)	Area (μm^2)	Leakage (mW)	Dyn. Power (mW)
1	-	-	-	-	-	-	-	-	-	-	-	-
2	4201.85	0.06	2.89	-	-	-	4201.85	0.06	2.89	-	-	-
3	4201.73	0.08	3.36	-	-	-	4201.8	0.08	2.82	-	-	-
4	4202	0.1	3.18	2109.62 (50.21%)	0.13 (130%)	2.52 (79.25%)	4202	0.1	3.18	2109.94 (50.21%)	0.1 (100%)	2.28 (71.7%)
5	4201.99	0.13	3.59	2108.47 (50.18%)	0.15 (115.38%)	2.61 (72.7%)	4201.99	0.13	3.59	2109.36 (50.2%)	0.19 (146.15%)	2.29 (63.79%)
6	4202.39	0.16	3.93	2108.11 (50.16%)	0.16 (100%)	2.5 (63.61%)	4202.39	0.16	3.93	2108.81 (50.18%)	0.18 (112.5%)	2.31 (58.78%)
7	4202.79	0.18	4.12	2107.99 (50.16%)	0.18 (100%)	2.46 (59.71%)	4202.79	0.18	4.12	2108.7 (50.17%)	0.2 (111.11%)	2.33 (56.55%)
8	4203.19	0.2	4.25	2108.08 (50.15%)	0.22 (110%)	2.46 (57.88%)	4203.19	0.2	4.25	2108.99 (50.18%)	0.21 (105%)	2.4 (56.47%)
9	4203.59	0.2	4.35	2108.12 (50.15%)	0.22 (110%)	2.46 (56.55%)	4203.59	0.2	4.35	2109.02 (50.17%)	0.23 (115%)	2.43 (55.86%)
10	4202.77	0.25	3.2	2108.49 (50.17%)	0.23 (92%)	2.5 (78.13%)	4202.77	0.25	3.2	2109.27 (50.19%)	0.25 (100%)	2.49 (77.81%)
11	4203.04	0.27	3.34	2108.75 (50.17%)	0.24 (88.89%)	2.55 (76.35%)	4203.04	0.27	3.34	2109.72 (50.2%)	0.29 (107.41%)	2.54 (76.05%)
12	4203.32	0.29	3.48	2108.88 (50.17%)	0.27 (93.1%)	2.58 (74.14%)	4203.32	0.29	3.48	2108.88 (50.17%)	0.27 (93.1%)	2.58 (74.14%)
13	4203.54	0.32	3.64	2109.27 (50.18%)	0.32 (100%)	2.61 (71.7%)	4203.54	0.32	3.64	2109.27 (50.18%)	0.32 (100%)	2.61 (71.7%)
14	4203.07	0.34	3.71	2109.6 (50.19%)	0.36 (105.88%)	2.66 (71.7%)	4203.07	0.34	3.71	2109.6 (50.19%)	0.36 (105.88%)	2.66 (71.7%)
15	4203.29	0.36	4.05	2109.87 (50.2%)	0.38 (105.56%)	2.7 (66.67%)	4203.29	0.36	4.05	2109.87 (50.2%)	0.38 (105.56%)	2.7 (66.67%)
16	4203.51	0.35	4.38	2110.08 (50.2%)	0.39 (111.43%)	2.74 (62.56%)	4203.51	0.35	4.38	2110.08 (50.2%)	0.39 (111.43%)	2.74 (62.56%)
17	4203.73	0.43	4.6	2110.23 (50.2%)	0.41 (95.35%)	2.78 (60.43%)	4203.73	0.43	4.6	2110.23 (50.2%)	0.41 (95.35%)	2.78 (60.43%)
18	4203.95	0.47	4.79	2109.84 (50.19%)	0.44 (93.62%)	2.81 (58.66%)	4203.95	0.47	4.79	2109.84 (50.19%)	0.44 (93.62%)	2.81 (58.66%)
19	4204.17	0.48	4.94	2110.4 (50.2%)	0.46 (95.83%)	2.86 (57.89%)	4204.17	0.48	4.94	2110.4 (50.2%)	0.46 (95.83%)	2.86 (57.89%)
20	4204.39	0.48	5.1	2110.42 (50.2%)	0.47 (97.92%)	2.9 (56.86%)	4204.39	0.48	5.1	2110.42 (50.2%)	0.47 (97.92%)	2.9 (56.86%)

TABLE III

OPTIMAL BUFFER INSERTION COMPARISON FOR 0.75*cm* WIRE

inferred. Although our approach requires a greater active area in general, the total area savings overshadow this increase in active area, and also helps reduce wiring congestion. The area numbers exhibit a minimum value for 7 stages in Table III. This is because when there are few stages, the drivers in the coding units are large, yielding a larger total area. When there are a large number of stages, the drivers of the coding units are small, but large in number, again yielding a larger total area. Hence there is an optimum value of the number of stages (7 for Table III). A similar behavior is observed for dynamic as well as total power, with the same explanation.

Note that our approach typically exhibits a $\sim 49\%$ reduced total area utilization for the 64-bit bus. We also performed experiments for a 32-bit and a 128-bit data bus as well. We also conducted experiments for 1.0 cm and 1.5 cm busses. We found that these results (the area and power ratio of the solutions from both approaches) are very similar to the results obtained for the 64-bit data bus for any number of stages (or for any row of Table III). Therefore, the results for the 32-bit and 128-bit data buses are not reported for brevity. Note that our approach performs bidirectional data transfer on wires between coding units therefore, there are no traditional buffers on these wires.

V. CONCLUSIONS

In this paper, we describe a low-area reduced-power on-chip point-to-point bidirectional communication scheme for heavily pipelined systems. Our communication scheme uses a single wire for bidirectional data transfer, borrowing from the emerging area of network coding (in the field of communication). Utilizing coding units, which also serve the purpose of buffering the signals along the wire between the two endpoints, we are able to achieve the same throughput as a traditional approach. Thus reduces the total area utilization by about 49%, and the power consumption by about 11%.

REFERENCES

[1] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *ISCA'07 Proceedings of the 34th annual international symposium on Computer architecture*, pp. 126–137, June 2007.

[2] B. Grot, J. Hestness, S. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *Proceedings of the 15th International Symposium on High-Performance Computer Architecture (HPCA)*, Feb 2009.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] E. S. C. Chekuri, C. Fragouli, "On Average Throughput Benefits and Alphabet Size in Network Coding," *To appear in IEEE Trans. Inform. Theory*.

[5] M. Charikar and A. Agarwal, "On the Advantage of Network Coding for Improving Network Throughput," in *Proceedings of IEEE Information Theory Workshop, San Antonio*, 2004.

[6] N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic Network Coding by Matrix Completion," in *Proceedings of the IACM-SIAM Symposium on Discrete Algorithms*, 2005.

[7] M. Langberg, A. Sprintson, and J. Bruck, "The Encoding Complexity of Network Coding," *To appear in the joint special issue of the IEEE Transactions on Information Theory and the IEEE/ACM Transactions on Networking on Networking and Information Theory*, 2006.

[8] D. Lun, M. Médard, T. Ho., and R. Koetter, "Network Coding with a Cost Criterion," in *Proceedings of International Symposium on Information Theory and its Applications (ISITA 2004)*, October 2004.

[9] D. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding," in *Proceedings of INFOCOM*, March 2005.

[10] T. Ho, D. Karger, M. Medard, and R. Koetter, "Network Coding from a Network Flow Perspective," in *Proceedings of ISIT*, 2003.

[11] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371 – 381, 2003.

[12] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782 – 795, 2003.

[13] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[14] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction," *IEEE Transactions on Information Theory*, vol. 51, pp. 1973–1982, June 2005.

[15] N. Jayakumar, K. Gulati, S. Khatri, and A. Sprintson, "Network coding for routability improvement in VLSI," in *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, (New York, NY, USA), pp. 820–823, 2006.

[16] K. Gulati and S. P. Khatri, "Improving FPGA Routability Using Network Coding," in *GLSVLSI '08: Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pp. 147–150, 2008.

[17] M. Anders, N. Rai, R. Krishnamurthy, and S. Borkar, "A transition-encoded dynamic bus technique for high-performance interconnects," in *Symposium on VLSI Circuits*, pp. 16–17, 2002.

[18] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 243–254, 2006.

[19] H.-Y. Huang, C.-C. Wu, and S.-L. Chen, "Simultaneous Current-mode Bidirectional Signaling for On-chip Interconnection," in *Proceedings, IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, pp. 380 – 383, Aug 2004.

[20] A. Nalamalpu, S. Srinivasan, and W. Burleson, "Boosters for Driving Long Onchip Interconnects - Design Issues, Interconnect Synthesis, and Comparison with Repeaters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 50–62, Jan 2002.

[21] "The International Technology Roadmap for Semiconductors." `http://www.itrs.net/`, 2007.

[22] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout," *Integration, the VLSI Journal*, vol. 21, no. 1-2, pp. 1–94, 1996.

[23] HSPICE http://www.synopsys.com/products/mixedsignal/hspice/hspice.html.

[24] PTM http://www.eas.asu.edu/∼ptm.