Gaining Predictability and Noise Immunity in Global Interconnects

Yinghua Li
UC Berkeley
yinghua@eecs.berkeley.edu

Alex Kondratyev
Cadence Berkeley Lab
kalex@cadence.com

Robert K. Brayton
UC Berkeley
brayton@eecs.berkeley.edu

ABSTRACT

We present a bundled data communication scheme that is robust to crosstalk effects, and to manufacturing and environmental variations. Unlike a data bus, where each receiver always connects to all data lines from the sender, we consider the case where each receiver can have a subset of all data lines routed to it. Such generalization can be used for a bundled data communication method applicable to both local and global communication. It can be used to make a clock unnecessary in a design. It also leads to a new routing problem for which we present an algorithm based on MRSA tree construction to solve it.

1. Introduction

One of the major problems associated with future system-on-a-chip (SoC) designs arises from non-scalable global wire delays. Global wires carry signals across a chip, but do not scale in length with technology scaling [1]. A problem with global wires is that they are typically implemented in the top-level metal layers, with the routing performed automatically in later stages of the design cycle. Consequently, these wires have parasitic capacitance and resistance that are difficult to predict a priori. Process variation and dynamic effects such as cross-talk contribute further to the wire delay uncertainty.

Several approaches have been proposed to improve the predictability of communication delays using predefined architectures (buses) and "self-shielding" encoding [2, 3]. However, bus architectures often introduce area and performance penalties and are not flexible enough for fine-grain data communication. Sharing buses between different modules causes high wire loads and resistance levels, slowing down signal propagation.

Developing a relatively cheap, reliable, and predictable scheme for point-to-point communication is important for SOC architectures. Such schemes are necessary also for data synchronization (in clocked or clockless) SOC designs. Difficulties with synchronization, stem from two sources: a) with technology scaling, global interconnects may have delays of several clock cycles, and b) SOC designs typically combine dozens of intellectual property blocks (IP-blocks) designed with different clocks.

Implementing a global reference clock for the whole chip becomes increasingly problematic (if not impossible) in SOC technology. The most likely synchronization paradigm for future chips is GALS (globally asynchronous locally synchronous), with many clocks [4, 5, 6]. The basic idea is to partition a system into independently clocked modules that are communicating in a self-timed fashion. In this, the functionality of each subsystem is still described and synthesized along with well-established synchronous design flows, while the communication between locally-synchronous modules requires specialized asynchronous components.

In an asynchronous link, timing information must be transferred together with data. Two general strategies exist for such an implementation. In *delay-insensitive* (DI) communication [7], the completion of the data propagation can be logically inferred by the data content, using redundant encoding. One drawback is a significant increase in the number of wires (usually two-fold) with a subsequent increase in the power consumption for data transfers. Wire and power penalties incurred in implementing asynchronous links can be reduced by using some timing assumptions. This leads to another strategy, the *bundled data* approach [20], which infers the completion of communication by observing transitions on special dedicated outputs (called *request* or *done*). The timing requirement is that *done/request* must change last, relative to other signals. In general, the bundled timing assumption may be difficult to guarantee because of the low predictability of delays in global interconnects.

This paper proposes a *bundled routing* scheme that easily satisfies the bundled data timing assumption, even under dynamic variations such as cross-talk.

In the experimental section, we present some preliminary comparisons (in terms of area, timing, congestion, and wire lengths) between our bundled routing approach and the use of DI-encoded communications., using conventional single-rail routing as a reference (even though it is not reliable).

We see two main applications for the suggested communication scheme:

1. Point-to-point data transfer between different synchronous domains in SOC designs (e.g. for GALS architectures).
2. Block-based design methods, where designers manipulate blocks consisting of hundreds or thousands of cells at a time rather than working with individual cells [8]. A major problem with such a flow is the loss of optimality due to the inherent loss of timing accuracy across block boundaries. This may be alleviated by using bundled data

implementations where every block carries the information about its completion. An example of this use-case scenario is given by implementations based on dynamic PLAs [9].

This paper is organized with Section 2 describing two self-timed schemes for communication. Section 3 covers our bundled data communication in more detail, arguing its immunity to noise and global variations. Also, a bundled routing algorithm is introduced in Section 3. Section 4 compares power consumption of the bundled routing scheme with 2- and 4-rail DI encoding. Section 5 shows some preliminary results comparing bundled data, DI, and conventional single-rail. Section 6 concludes with some final remarks and directions for future work.

2. Communication schemes with self-timing

2.1 Delay-insensitive communication

Due to the absence of timing assumptions about signal propagation, communication between modules in self-timed links usually relies on a two-phase operation; data changes from the spacer phase (*reset*) to a proper data codeword during the *set* phase, and then back to the spacer in the reset phase. The most common approach to DI encoding uses multi-valued one-hot encoding of the communicating signals.

Examples of DI encoding based on one-hot codes are:

1) dual-rail encoding, in which each signal a is represented by two wires *a.0* and *a.1* (i.e. $a = 1$ is encoded as *a.0=0, a.1=1,* and $a = 0$ encoded as *a.0=1, a.1=0*), or
2) *n*-rail encoding, in which a *n*-value signal a is encoded by n wires *a.0,…a.n.*

An attractive property of DI encoding is the capability for a receiver to determine that a proper result has arrived by the codeword itself, without appealing to timing assumptions. For example, if a dual-rail encoded signal a is transmitted as soon as one of the wires (*a.0* or *a.1*) goes high, a valid dual-rail codeword has been received at the inputs of a receiver module. A completion detector in this case is implemented simply by an OR gate.

Once a receiver deduces the completion of data transfer, it needs to acknowledge it to the sender. This can be done explicitly by triggering a transition on a dedicated *ack* line or implicitly by a protocol implying that the sender should stall (not issue new communication) until some indication comes from a receiver or through the expiring of some timing delay.

The main advantages of DI communication are:

a) the simplicity of detecting the completion of data transfer, and
b) its very high tolerance to variability of physical properties of interconnects and environment conditions.

The main shortcoming of this scheme is an excessive penalty in wiring: the most popular dual-rail and 1-out-of-4 DI encodings both require a 2x wire increase in interconnects. Increasing the number and/or the length of interconnects induces power penalties, which may be significant since power consumption related to interconnects can be as high as 70% in modern chips [10]. Although power penalty might be reduced in more elaborated DI encodings through the reduction of the number of switching wires (see [21] e.g.) the latter results in even higher increase in the wiring and presents significant challenges for automatic routing.

2.2 Bundled communication

Bundled data communication relies on observing the behavior of special dedicated signals, *done*, to deduce when data is ready. For that, two conditions must be ensured

a) *done* must be generated by a sender strictly *after* all data lines have settled to their correct values.
b) *done* must propagate though interconnects *not faster* than the data signals.

Formally, bundled data communication must satisfy the following timing constraint at the receiver:

$$Arrival\_time(data) \leq Arrival\_time(done) \quad (2.1)$$

If the proper ordering of *done* and data signals is ensured at the sender then the above constraint can be simplified to

$$Propagation\_time(data) \leq Propagation\_time(done) \quad (2.2)$$

The latter is usually ensured by including an additional delay in the *done* interconnect to provide a safe margin for (2.2).

The main advantages of bundled data communication are its simplicity and low cost. Indeed, in this scheme, the encoding of the data is used as is, with the only penalty coming from adding a single line for the *done* signal.

However, satisfying (2.2) is tricky and is becoming even more challenging with technology scaling. Guarding *done* by an additional delay implies a performance penalty, which increases with the growth of the delay uncertainty in global interconnects.

3. A reliable scheme for bundled communication

In the subsequent discussion, we assume that the proper temporal ordering of *done* and data lines at the sender end is ensured. Then the main task to support bundled

data communication is in satisfying the timing constraint (2.2). Bearing in mind that crosstalk is the main contributor to the possible delay variations of wires in a bundle we need a signaling protocol where the validity signal (done) is always put in the worst noise conditions compared to data signals. We propose a communication scheme with the following characteristics:

1. Use single-rail signals in communication, i.e. one wire for each data signal.
2. Use two-phase communication: one phase is *reset* when all data lines are 0 and *done* is 1; the other is the *data* phase when data lines may remain 0 or rise to 1, while *done* always changes from 1 to 0 (when data is ready). We note (for a comment about power in Section 4) that the reset phase for the data lines is not necessary for correct operation; reset is used on the data lines to make the communication invulnerable to crosstalk effects. Note, that for some technologies (dynamic logic e.g.) two-phase operation is a natural choice.
3. Route each data signal only to the modules where it is needed (its fanouts); route *done* to all its associated data fanouts.
4. Route the *data* and *done* signals that are output from the same module, as a **bundle** of varying width as shown in Figure 1.

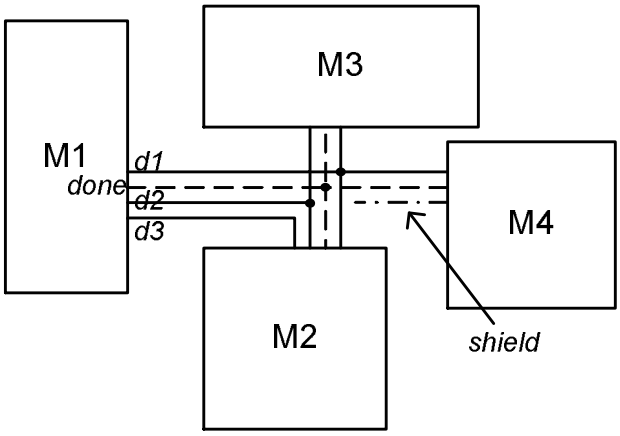

Figure 1. Bundled data communication among modules

An example of a signaling protocol for communication is shown by the Signal Transition Graph (STG) in Figure 2.

Events for $data\_send_i$ and *done_send* denote the transitions at the sender outputs for data and *done* lines, while $data\_rec_i$ and *done_rec* denote the corresponding transitions at the inputs of the receiver. Solid arcs in the STG stand for causal relations while dotted arcs show timing constraints that must be satisfied for the protocol correctness. Data signals are produced at the sender end before the sender releases the *done* signal (see timings arcs between $data\_send_i+$ and *done_send-).* Similar relationship must be maintained at the receiver end. The latter is ensured by two arguments: 1) there is no vulnerability to delay variations due to crosstalk (see the discussion below) and 2) since the same metal layers and routing lengths are shared in the bundle as well as having all vias similar and close, the delay constraint will be invulnerable to global and/or systematic variations.

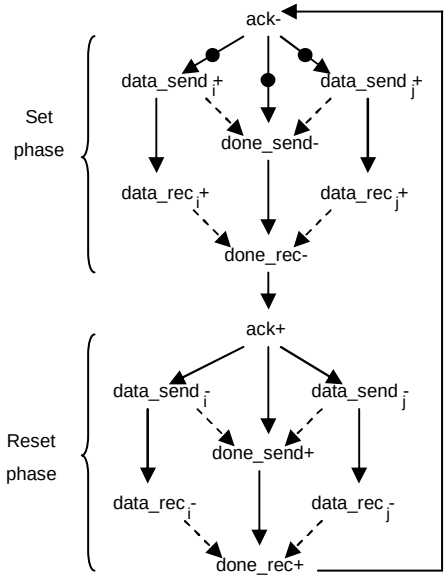

Figure 2. Signaling protocol for communication

The communication proceeds in two phases. In the set phase the sender sets data lines to a proper codeword and releases the *done* signal. The receiver acknowledges a transition at signal *done* by setting the *ack* signal e.g. which tells the sender to proceed with the reset phase. Note, that the use of the dedicated *ack* signal is not always needed because the acknowledgement might also come implicitly through changing the data lines at the sender inputs (with corresponding signal *done*) caused by the receiver actions.

Different variations of the protocol from Figure 2 are possible using timing assumptions (reset phase e.g. might be triggered simultaneously for several blocks and be acknowledged by timeout simply, as in the case of dynamic PLAs e.g.).

The varying width of a bundle is decided as follows. If the number of data lines in a segment is 1 or 2, the width will be 3, (1 for *done*, 1 for *data*, and 1 for an added shield wire or 1 for *done*, and 2 for *data*). The bundle width increases by 1 with each additional *data* signal beyond 2. In addition, the bundle is constructed with *done* being internal, so its neighbors are either its associated data wires or the shield wire.

Now consider what could happen during the data (set) phase. Two cases are possible.

Case 1. Data wires adjacent to *done* are quiescent. In this case *done* is not impacted by crosstalk. Although the line lengths, metal dimensions, and via resistances, may vary as functions of the distance along the bundle, we assume that this variation is close to the same for all wires in the bundle (i.e. local variations across these wires are small.). This is because the same metal layers and routing lengths are shared in the bundle as well as having all vias similar and close.[1] Thus, we have

$$\text{Delay}_{00}(done) \geq \max_i[\text{Delay}_{00}(data_i)],$$

where Delay_{00} is the delay of a line with both neighbors quiescent. Of course $data_i$ can be sped up by a (rising) neighboring $data_j$, which only reinforces the inequality.

Case 2. Data line(s) adjacent to *done* propagate rising transitions. If the slews and the strengths of the drivers for *done* and data signals are similar then by symmetry, *done* is slowed down by approximately the same amount as the maximally-slowed *data* signal (its neighbor). The following inequality is valid:

$$\text{Delay}_{00}(done) + \delta \geq \delta + \max_i[\text{Delay}_{00}(data_i)],$$

where δ is the slowdown caused by crosstalk when adjacent lines propagate opposite transitions.

From the above analysis, it follows that bundled routing guarantees that inequality (2.2) is not vulnerable to crosstalk as well as to global manufacturing or environmental variations. However, local variations within the wires, such as variations in via resistance, if substantial would have to be checked.

3.2 An MRSA-based routing algorithm

The bundled data communication scheme proposed in Section 3.1 poses a new kind of routing problem. Since the data signals in the bundle are routed only to modules where they are needed, a bundle can have different widths along its different segments. One naive idea is to use a constant-width thick wire routing with the width set to accommodate all data lines and *done*. Clearly, this overestimates space needed for most segments and can cause unnecessary rip up and reroute later. Experiments show a large wire length and interconnect delay penalty for this.

We generalize the MRSA (Minimum Rectangular Steiner Arborescence) tree routing algorithm [11] to solve such a variable-width wire routing problem and develop a new router to implement bundled communication. The MRSA algorithm was chosen mainly because it constructs the routing structure from sinks to the source. Thus, at each merging point of the tree, the sinks connected to it are known, so the actual width of the segment starting from this merging point is known and overestimation can be avoided. The algorithm is shown in Figure 2.

The MRSA algorithm [11] is based on branch-and-bound. The nodes in the routing graph are ranked by their distances from the source terminal. Nodes are scanned in a rank decreasing order until the source terminal is reached. At each scan level *i*, terminal nodes above this level have all been connected into subtrees and the roots of all subtrees form a *peer* set *P*. Branching in the algorithm happens when a Steiner node is met. At this point, two cases are considered, whether the Steiner node is chosen as a merging point or not. It has been shown that *P* together with *C*, the total wire length in all subtrees, and *S*, the set of selected Steiner points, can completely characterize the constructed partial arborescence tree. All formal definitions can be found in [11].

Our algorithm is a modification of this and we only discuss differences from the original algorithm. The main differences lie in the characterization of merging points and the bounding conditions. To characterize a merging point, we need not only the location information but also its data signal set. We add a set *SigSet* for each point in the peer set, which contains all data signals appearing in the subtree rooted at this point. Another difference is the bounding condition when a terminal merging point is met. In the original algorithm, since each wire has the same width, all points that are dominated by this terminal point should be merged with it. This dominance does not hold for all cases when the data signal set is considered. As shown in Figure 2, there are two cases:

1. The data signal set of the terminal point contains the signal set of the dominated point: *SigSet* $v_j \subseteq$ *SigSet* v_i. Then only connecting v_j to v_i can lead to the optimal result.
2. Otherwise, both merging at v_i and not merging needs to be considered.

The difference in this bounding condition also leads to a difference in the program to generate arborescence from *S*: when a terminal merging point is met, we need to check if it is contained in *S* to determine if the points dominated by it but whose signal set is not covered should be connected to it.

In Figure 2 the function General_RSA/G(*P*, *N*, deleted) is called to finally generate the MRSA tree. This function is the same as function RSA/G(*P*, *N*, deleted) shown in Table 1 in [11] except that we compute the width for each segment. Note that when no nodes are deleted, this function becomes a heuristic algorithm for constructing an RSA tree. It is much faster than the optimal algorithm

[1] *An interested reader may look into [22] to see the evidence that random variations have negligible (less than 5%) impact on timing.*

and can be used for fast evaluation which we do in our experiments.

4. Power consumption

Since the data signals and the *done* signal in the same bundle are always routed together, the coupling capacitance of internal wires in the bundle is maximized by having their neighbors always occupied. Thus, in bundled routing communication propagating a transition through a wire might consume more power versus wires that are routed freely. However, we argue (informally) that a bundled routing scheme would consume less power than a DI scheme. The arguments are derived from the analysis of the switching activity of communication wires and of the additional circuitry (if needed) for the data encoding/decoding and completion. The impact of coupling capacitance on power consumption for these two schemes can be qualitatively derived from experimental data about total wire lengths (see Section 5).

The comparison is provided for an n-bit data bus. We assume that three probabilities are given for each data signal: 1) *p0 (p1)* is the probability that the signal takes value 0 (1) in the current data pattern and 2) p^{tr} is the probability of the signal to have a data value different from the one taken in the previous communication cycle.

1. **Bundled data communication.**
   Bundled routing requires only single-rail signals. In the reset phase, all data signals that have values "1" are discharged to "0", while in the set phase, those that should take value "1" are charged from "0" to "1". A done signal changes once per phase. Then for a single communication cycle consisting of set and reset phases we have the following power consumption:
   $$P_{bundled} = n*(p1*P_{wire\downarrow} + p1* P_{wire\uparrow}) + P_{done\downarrow} + P_{done\uparrow} \approx 2*(n*p1 +1) * P_{wire},$$
   where $P_{wire\downarrow}$, $P_{wire\uparrow}$ is the power consumption for propagating the falling and the rising transitions respectively through a data line.
2. **Bundled data communication (no data reset)**
   As mentioned in Section 3 (point 2), using two phases is only necessary for the *done* signal. If enough of a timing margin is provided by the module to tolerate crosstalk effects in its bundle, the charging up events for those data lines can be reduced (by not resetting them to 0). In this case the power consumption for a single cycle for the bundled communication is reduced to:
   $$P_{bundled\_noreset} = n*p^{tr}*P_{wire} + P_{done\downarrow} + P_{done\uparrow} \approx (n*p^{tr} +2) * P_{wire}$$
3. **Dual-rail DI communication.**
   A dual-rail DI bus for *n* bits contains *2n* wires. In the set phase half of the wires go up, while in the reset phase they return to "0". In addition, the receiver needs to infer the completion of the communication phases. The complexity of the completion circuitry is linear from the width of the bus (it requires an OR gate for every dual-rail pair and an AND gate to collect all outputs of the OR gates). Hence the power consumption of a single communication cycle in the dual-rail bus is:
   $$P_{2rail} = n*P_{wire\downarrow} + n* P_{wire\uparrow} + P_{complete} \approx 2*n* P_{wire} + P_{complete},$$
4. **1-out-of -4 DI communication.**
   1-out-of-4 encoding is known to be one of the most power efficient among DI encodings [12]. It represents the values for a *pair* of data bits by 4 wires using one-hot encoding. In this way the width of the 1-out-of-4 DI bus is the same as for the 2-rail bus (*2n*) but only one out of 4 wires is transitioning in the set and reset phases. However, each wire requires an additional encoder (at the sender) and decoder (at the receiver). The power consumption of a single communication cycle in the 1-out-of-4 bus is:
   $$P_{1of4} = 2*n*(P_{wire\downarrow}/4 + P_{wire\uparrow}/4 + P_{encode} + P_{decode}) + P_{complete} \approx 2*n* (P_{wire}/2 + 2*P_{enc\_dec}) + P_{complete},$$
   where P_{encode}, P_{decode} and P_{enc_dec} is the power consumption for the encoding, the decoding and the average of the encoding-decoding of a single wire in 1-out-of-4 DI bus.

The above results on the power consumption per a transfer of 1-bit of data are summarized in Table 1.

Table 1. Power consumption for self-timed communications

	Bundled	Bundled (no reset)	2-rail	1-of-4
P_{wire}	$2*p1+2/n$	$p^{tr}+2/n$	2	1
$P_{complete}$	none	none	$\approx P_{Or\_gate}$	$\approx P_{Or\_gate}$
P_{enc_dec}	none	none	none	$\approx 2*P_{Or\_gate}$

Note, that the typical values of p^{tr} are in the range of 0.15-0.2 for modern designs [13], while *p1* is usually less than 0.5 [14]. Based on this, we conclude that bundled data communications are more power efficient than DI schemes.

5. Experimental Results

5.1. Experimental flow

The algorithm shown in Figure 2 is integrated in a block placement program to provide global routing after placement. The block placement program, (Figure 3) uses a simulated annealing (SA) framework with sequence pairs as the layout representation. At each SA step, a new placement is generated, and the heuristic

algorithm for constructing an RSA tree, *General_RSA/G*(*P*, *N*, null), is used to generate a net topology for each bundled net. Then a cost function, which includes area, aspect ratio, routing congestion, and wire delay, is computed to evaluate the placement. After a final placement is obtained, the function *General_RSA/DP/G* is run on each net. After the MRSA tree is obtained for each net, a post processing function is called. This targets reducing routing congestion by repositioning Steiner merging points while maintaining the topology of the Steiner Arborescence tree.

Experiments compared conventional single-rail, bundled data and dual-rail DI communication. For the experiments, we used a set of benchmarks that are commonly used in logic synthesis. Multi-level networks of nodes in these benchmarks were first clustered into multi-output nodes. The outputs of each of these nodes formed a bundle. All data signals as well as *done* signals were assumed to have the same driving buffer. Two groups of examples are representative of the two proposed use-cases for communication schemes:

- Group 1 consists of examples with high capacity of communication resources. For these, additional wiring required by different communication schemes does not impact the areas because the designs are not congested. This case, in our opinion, represents the SOC design scenario when wiring area is negligible with respect to the area of IP modules.
- Group 2 consists of examples with relative low capacities for communication resources. For these, wiring penalties have a higher impact on the design area because the designs are wiring congested. This case represents block-based design flow where both block and wiring areas matter.

Table 2 gives the details of the examples used in the experiments.

The SA_Place algorithm (Figure 3) was applied to each example to obtain a placement with global routing for bundled communication. For the dual-rail DI communication experiments, each data signal becomes two separate wires, and there are no *done* signals. We ran the same SA_Place algorithm on this, but instead of bundled routing, the original RSA/DP/G routing algorithm was run for each net (the same was applied for single-rail communication). For each example, the three placements were compared and if the difference in congestion was beyond some threshold, the one with worst congestion had SA_Place rerun with more weight on the routing congestion parameter. This was continued until the final congestion values (after post processing) in the two placements were similar. For some examples, modification of the cost function could not lead to the requested congestion values; in that case, the lowest congestion values achieved are shown.

Table 2. Examples for experiments

Design	#blocks	#data signals	Ave. Bundle Size
Group 1			
D1	19	18	2.5
D2	25	48	3.0
D3	42	43	3.3
D4	43	61	3.0
Group 2			
D5	25	52	3.5
D6	54	161	4.1
D7	38	102	3.9
D8	114	221	3.9

In all experiments, 0.18um technology was used with routing done on metal layers metal_3 and metal_4. Experimental results are shown in Table 3 and Table 4.

Routing congestion was computed as follows. The entire layout area was separated into grids, each grid containing around 60 routing lines. Since our global routes for each net are made up only of point-to point connections between Steiner points of a tree, and we know for each segment its width, the congestion through each grid is computed using the probability that each segment may pass through it. The percentage of the number of grids, with metal usage greater than 1 in probability, is listed in Table 4. This serves as a measure of the difficulty that detailed routing would face.

5.2. Analysis of experimental results

Table 3 shows area numbers for different communication schemes. As expected for examples from Group 1, the total area does not depend on the choice of communication scheme. In contrast, for the examples from Group 2, wiring overhead for bundled data and DI communications does impact the total area, resulting in 7% (bundled) and 42% (DI) penalties versus single-rail designs. Thus, area penalty for bundled data is significantly lower than for DI designs.

Table 4 shows the comparison of designs by total-length/congestion of communication wires. For Group 1, most of the extra wire length of the bundled data scheme comes from the presence of the additional *done* wires, while DI communication shows approximately 2x wire length penalty because of twice the number of wires. For Group 2 the wire length overhead of bundled data communication increases slightly for Group 1 (from 40% to 41%) while for DI it increases considerably (from 93%

to 136%). The latter increase is due to the area increase in DI designs from Group 1 to Group 2. For Group 2, Table 4 also reports the number of congested grids for each of the designs next to the total wire length numbers. DI designs are the most congested.

Table 3. Total implementation area

	Area_SR um^2	Area_B um^2	Ratio B/SR	Area_DI um^2	Ratio DI/SR
Group 1					
D1	5736	5832	1.02	5864	0.99
D2	15688	15862	1.01	15596	0.99
D3	15116	15128	1.00	15152	1.01
D4	11947	12233	1.02	12201	1.02
Ave			**1.01**		**1.00**
Group 2					
D5	8623	8621	1.00	9225	1.07
D6	27139	28831	1.06	32784	1.21
D7	14601	15501	1.06	23678	1.62
D8	35994	42208	1.17	63780	1.77
Ave			**1.07**		**1.42**

Table 4. Wire-length ratios and congestion percentages

	WL_SR um/%cong	WL_B um/%cong	Ratio B/SR	WL_DI um/%cong	Ratio DI/SR
Group 1					
D1	965/0	1451/0	1.50	1863/0	1.93
D2	4939/0	6513/0	1.32	9033/0	1.83
D3	4431/0	6255/0	1.42	8598/0	1.94
D4	4512/0	6073/0	1.35	9039/0	2.00
Ave			**1.40**		**1.93**
Group 2					
D5	5805/.004	8184/.028	1.41	11614/.026	2.00
D6	24111/.008	27832/.006	1.15	44380/.022	1.84
D7	14461/.010	19797/.018	1.37	38469/.041	2.66
D8	35233/.002	59753/.048	1.70	104055/.055	2.95
Ave			**1.41**		**2.36**

Table 5 shows relative communication delays of bundled data and DI designs versus single-rail communication.

Table 5. Communication delay ratios

	Del_SR	Del_B	Del_DI
Group 1			
D1	1	1.04	1.00
D2	1	1.10	1.06
D3	1	1.00	1.00
D4	1	1.07	1.03
Ave		**1.05**	**1.02**
Group 2			
D5	1	1.27	1.17
D6	1	1.02	1.08
D7	1	1.27	1.61
D8	1	1.28	1.77
Ave		**1.21**	**1.41**

To compare wire delays in the three communication schemes, we used the Elmore delay model. (Note: we are only computing wire delays which do not include module delays). Then, we determined the delay for each signal in the single-rail experiment. For the other communication schemes, we found the corresponding signal and assigned a delay to it. For the bundled scheme, we assigned the delay of each data signal to be the delay of its corresponding *done* signal, since all wires in the bundle have delay less than *done*. For dual-rail DI, we take the delay as the maximum delay of the two wires. We then compute the total wire delays for each scheme and show ratios to the single-rail total in Table 5. For designs in Group 1, the delay penalties of bundled data and DI schemes are low (5% and 2% respectively). This is different for Group 2 where the delay penalties are significant (21% and 41% respectively).

Note that the data of Table 5 is overly optimistic for single-rail communication because it is not robust and delays need to be given some margins. To give a flavor of a more fair comparison under process variations and crosstalk, we assumed the following models of variability:

1. Wire delay variations due to process variations is 30% [15], out of which approximately half is systematic and half is random. For long wires the random component of variations tend to compensate and therefore altogether we assume wire delay variations to be about 20%
2. According to some sources [16], the slowdown due to crosstalk in global interconnects can be as high as 75%. We assume a more moderate penalty of

15% which is typically cited by tools for signal integrity analysis [17].

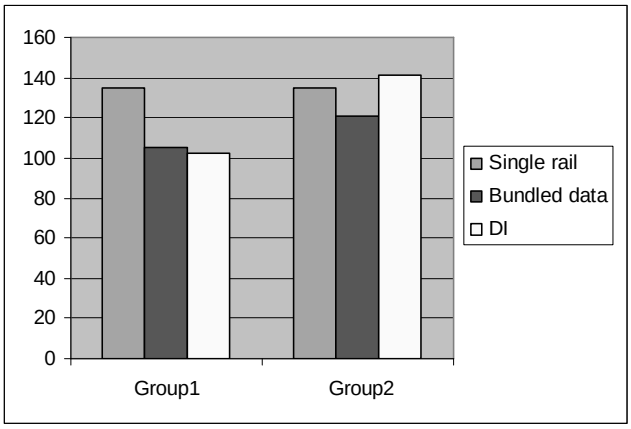

Figure 4. Communication delays with process variations and crosstalk

Both bundled data and DI schemes tolerate process variations and crosstalk noise well because the information about the completion of communication is derived from data content rather than from timing assumptions. Therefore variations do not impose additional delay penalties in these schemes. For single rail communication all the variability margins must be imposed up-front because single-rail designs are implemented for the worst case scenario. Figure 4 summarizes the adjusted delay numbers for all communication schemes.

The chart shows that when using automatic routing for SOC interconnects, the DI scheme might be slightly more timing efficient than bundled, while for the block based design flow the bundled routing scheme is superior.

The qualitative picture for area/power/delay trade-offs for different communication schemes based on the above experimental results is shown in Figure 5. We stretched the positions for DI and bundled data implementations along the power axis because our power analysis is not accurate and is based purely on switching activities. We also stretched the position of single rail implementations along the delay axis because its performance varies with different assumptions about the process variations and crosstalk impacts. Note, that our analysis and conclusion (possibly) are different from some known works for evaluating the efficiency of communication schemes (see [18] [19] e.g.). This is because we based our observations on **automated** methods for implementing global interconnects rather than checking the potential of what the custom based approaches may show.

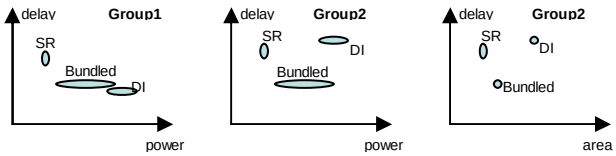

Figure 5. Area/power/delay trade-offs for communication

6. Conclusions and Future Work

We presented a scheme for bundled data communication, argued that it is immune to noise and global variations, and developed a routing algorithm for this. This bundled routing method was compared with dual-rail DI communication (which would also have such immunities) with respect to power consumption, routing congestion, wire length, wire delay and area. Experimental results show that bundled routing communication can result in large reductions in congestion, wire lengths and area as compared to dual-rail DI communication. It also outperforms the conventional single-rail communication when process variations and crosstalk are taken into account

In the future, it is necessary to obtain experimental results on power consumption for the bundled routing scheme to measure its efficiency in reducing power consumption. A DI communication scheme which is a more power-attractive one would use 1-out-of-4 signaling, in which pairs of data wires are combined and encoded with one-hot encoding. This would have the same number of wires as dual-rail DI encoding, but should be superior in power. However, it would also come with some penalty in terms of extra synthesis and clustering causing increased logic area.

Acknowledgements.

This work was supported partially by the C2S2 Marco research center as well as the California Micro program and our industrial sponsors, Fujitsu, Intel, Magma, and Synplicity.

References

[1] M.A. Horowitz, et al., “The Future of Wires”, *Proceedings of the IEEE*, Volume: 89 Issue: 4, April 2001 pp. 490–504.

[2] B.M. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2001.

[3] K.N.Patel and I.L.Markov "Error Correction and Crosstalk Avoidance in DSM Busses," *IEEE Trans. on VLSI*, Volume 12, Number 10, pp 1076-1080, 2004

[4] Daniel M. Chapiro, Globally-Asynchronous Locally-Synchronous Systems, Ph.D. thesis, Stanford University, Oct. 1984.

[5] J. Muttersbach, Globally-Asynchronous Locally-Synchronous Architectures for VLSI Systems, Series in Microelectronics Volume 120, Hartung Gorre Verlag, ISBN-3-89649-724-3, 2001.

[6] S. Moore, G. Taylor, R. Mullins, and P. Robinson, *Point to point GALS interconnect*, in Proc. of Int. Symp. on Asynchronous Circuits and Systems, pp. 769--775, Apr. 2002.

[7] David E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proceedings of an International Symposium on the Theory of Switching*, pages 204-243. Harvard University Press, April 1959.

[8] M. Hunt and J. Rowson. Blocking in a system on a chip. IEEE Spectrum, November 1996

[9] Fan Mo, R. Brayton, "*PLA-Based Regular Structures and Their Synthesis*", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Jun 2003.

[10] G. Chandra, P. Kapur and K. C. Saraswat, Scaling Trends for the On Chip Power Dissipation, Proc. Of the IEEE 2002 International Interconnect Technology Conference, pp 154-156, 2002

[11] Jason Cong, Andrew B. Kahng, Kwok-Shing Leung. Efficient Algorithms for the Minimum Shortest Path Steiner Arborescence Problem with Applications to VLSI Physical Design. In *IEEE Transactions on CAD of IC and Systems*, vol. 17, No. 1, Jan 1998.

[12] John Bainbridge and Steve Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, 22:16-23, 2002.

[13] *Private communication. Cadence customers*

[14] J. L. Hennessy and D. A. Patterson. *Computer Architecture -- A Quantitative Approach*. Morgan Kaufmann Publishers, 3rd edition, 2003.

[15] Sani R. Nassif: Modeling and forecasting of manufacturing variations (embedded tutorial). ASP-DAC 2001: 145-150

[16] R. Ho, K. Mai, and M. Horowitz, "The Future of Wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490-504, April 2001.

[17] User guide. Celtic User Manual, Cadence Design Systems, Inc., 2004.

[18] Kenneth S. Stevens. Energy and Performance Models for Clocked and Asynchronous Communication. In *9th International Symposium on Asynchronous Circuits and Systems*, May 2003, pp. 56-66

[19] C.Svensson. Low-power and low-voltage communications for SOCs", Low-power Electronics design, C. Piguet ed., CRC press, 2003

[20] Victor I. Varshavsky, editor. *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.

[21] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber. Delay-insensitive, point-to-point interconnect using M-of-N codes. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 132-140. IEEE Computer Society Press, May 2003.

[22] C.Amin, N.Menezes, K.Killpack, F.Dartu, Y.Ismail. Statistical static timing analysis: How simple can we get? In Proceedings of the ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU), 2005.

Function General_RSA/DP/G(G, N, *SigSetN*)
Given an SPDAG $G=(V, E)$ with ranked nodes and a set of terminals $N \in V$, the signal set *SigSetN* for each terminal, return the minimum wire length arborescence.

Compute the "$<$" relation;
$H \leftarrow \{(\{v_{\|V\|}\}, \{SigSet\ v_{\|V\|}\}, \|V\|, 0, \varnothing)\}$;
while $H \neq \varnothing$ do
(*) find $T=(P, SigSetsP, i, C, S) \in H$ such that i is maximized;
$H \leftarrow H - T$;
$X \leftarrow \varnothing$;
foreach $v_j \in C_i$ do
$W \leftarrow \{v \mid v_j \leqslant v$ and $v \in P\}$;
if $\|W\|>1$ then
$Cost\_merge = \| SigSet\ v_j \| * \| v_i \rightarrow v_j \| + \Sigma_{v \in W} \| SigSet\ v \| * \| v \rightarrow v_j \|$;
$Cost\_nomerge = \Sigma_{v \in W} \| SigSet\ v \| * \| v \rightarrow v_i \|$;
if $Cost\_merge < Cost\_nomerge$ then goto (*);
$X \leftarrow X + W$;

```
    if v_i ∈ N then
        foreach v_j ∈ X do
            if SigSet v_j ⊆ SigSet v_i then
                P ← P − {v_j} ; X ← X − {v_j} ;
                C ← | SigSet v_j |*| v_i ⇝ v_j |;
        SigSets ← SigSetsP + {SigSet v_i};
        H ← H⊕{(P + {v_i} , SigSets, i-1, C, S)};
        if |X|>0 then
            SigSet v_i ← (∪_{v∈X} SigSet v) U SigSet v_i;
            SigSets ← SigSetsP - {SigSet v| v ∈ X} + {SigSet v_i };
            H ← H⊕{(P + {v_i} , SigSets, i-1, C + Σ_{v∈X} | SigSet v |*| v_i ⇝ v |, S U {v_i} )};
    else if |X|>1 then
        H ← H⊕{(P, SigSetsP, i-1, C, S)};
        SigSet v_i ← (∪_{v∈X} SigSet v);
        SigSets ← SigSetsP - {SigSet v| v ∈ X} + {SigSet v_i }
        H ← H⊕{(P-X + {v_i} , SigSets, i-1, C + Σ_{v∈X} | SigSet v |*| v_i ⇝ v |, S U {v_i} )};
foreach v_i ∈ V do deleted[i] ← (v_i ∈ S)? false:true;
return General_RSA/G(P, N, deleted);
```

Figure 2. Generalized MRSA algorithm for various-width wire routing

```
SA_Place
randomly generate an initial placement
for each scheduled annealing step {
    randomly do one of the following:
        (1) swap a pair in one of the sequence pairs
        (2) flip one of the blocks
    update layout
    for each bundled net,  run General_RSA/G to generate a RSA tree net topology
    evaluate area, aspect ratio, congestion and delay
    evaluate cost
    accept or reject
}
for each bundled net, run General_RSA/DP/G
postprocess for reducing congestion
```

Figure 3. Simulated annealing based placement with RSA/DP/G for global routing