Low Power Gated Bus Synthesis using Shortest-Path Steiner Graph for System-on-Chip Communications

Renshen Wang[†], Nan-Chi Chou[‡], Bill Salefski[‡] and Chung-Kuan Cheng[†]
[†]University of California, San Diego, La Jolla, CA 92093-0404
[‡]Mentor Graphics Corporation, 1001 Ridder Park Drive, San Jose, CA 95131
{rewang, ckcheng}@cs.ucsd.edu, {nanchi_chou, bill_salefski}@mentor.com

ABSTRACT

Power consumption of system-level on-chip communications is becoming more significant in the overall system-on-chip (SoC) power as technology scales down. In this paper, we propose a low power design technique of gated bus which can greatly reduce power consumption on state-of-the-art bus architectures. By adding demultiplxers and adopting a novel shortest-path Steiner graph, we achieve a flexible tradeoff between large power reduction versus small wire-length increment. According to our experiments, using the gated bus we can reduce on average 93.2% of wire capacitance per transaction, nearly half of bus dynamic power and on a scale of 5%∼10% of total system power.

Categories and Subject Descriptors

J.6 [**Computer Applications**]: Computer-aided design

General Terms

Algorithms, design, performance

Keywords

Gated bus, Steiner graph, power efficiency

1. INTRODUCTION

System-on-chips (SoC) are nowadays being developed with increasing complexity and on-chip communication demand. However global on-chip wires which are to meet this demand do not scale well towards 35nm feature size [9]. As a result, global interconnect is becoming a bottleneck of improving system performance and power consumption [12]. Bus architectures are therefore regarded as an important aspect in low power SoC design. Current state-of-the-art bus architectures including AMBA [18], CoreConnect [19], Avalon [20], AMBA AHB bus matrix [13], *etc*, provide solutions for SoC on-chip communications. Research in [12] shows that these bus circuits may consume as much power as other major components such as processor, memory controller and cache. Therefore, reducing power on buses makes significant contribution to the whole system's power consumption.

Techniques of clock gating [6] and power gating [14] have been widely and effectively used to reduce power consumption of electronic systems, among which clock gating reduces dynamic switching power, and power gating reduces static leakage power. Both of the techniques save power by masking off signal/power when/where it is not needed. Since a clock distribution network consumes more than 40% of the total power budget of a CMOS circuit, clock gating has become a necessity in most digital circuit designs. We find that bus connections in current communication architectures are facing a similar (if not the same) situation as clock networks.

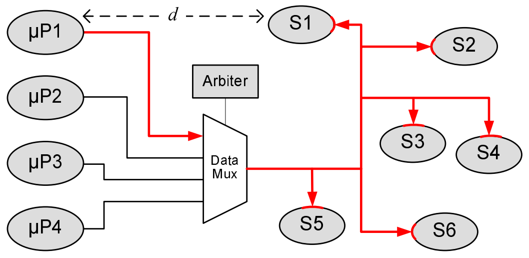

Figure 1: A signal switching on AMBA data bus

Figure 1 shows the wires and fanouts involved in the signal switchings of a data transaction (highlighted arrows) in AMBA bus architecture. Microprocessor 1 sends data to slave 1 through the data multiplexer and a wire net which connects all the slaves. While only the input on slave 1 needs the signals, the entire net and the input stage of other slaves are all driven, resulting in a low power efficiency. Considering the distance d between processor 1 and slave 1, ideally the wire used for the transaction needs only length of d, so most of the switching power on the wire net and fanouts are wasted. If we can mask off the wires not leading to slave 1, like clock gating, the power can then be saved.

In this paper, we propose a physical level on-chip bus gating scheme which can greatly reduce power consumption of current bus architectures. We focus on minimizing the total capacitance of the driven circuit in each transaction which determines its dynamic power. Distributed multiplexers and demultiplexers are used on a shortest-path Steiner graph to ensure minimum wire capacitance, with some increased total wirelength that can be cheaply traded off.

There is a body of work on bus power analysis on system-level and RTL-level such as [3], [12] and [13], which provide valuable data and orientations. Physical-level approaches like bus segmentation [4] and bus splitting [11] reduce bus power by masking off certain bus segments, which has a similar effect as bus gating. However the bus structures in [4] and [11] are limited to tree structures, and the bus type is bi-directional shared bus which is not suitable for on-chip communications. To the knowledge of the authors, this paper is the first attempt to optimize state-of-the-art on-chip bus power consumption by gating techniques applied on a non-tree structure.

2. BACKGROUND

2.1 Current on-chip bus architectures

The AMBA bus architecture [18] is one of the most popular commercial on-chip communication architectures. Figure 2 is a sketch of the components and interconnections in AMBA AHB (Advanced High-performance Bus) protocol. AHB supports up to 16 masters and 16 slaves. A master initiates a bus transaction by providing address to the decoder and control information to the arbiter. Arbiter ensures only one master at a time is allowed to use the bus. Decoder finds the slave corresponding to the address. A slave responds passively. AHB can connect to an APB (Advanced Peripheral Bus) bus on a lower hierarchy via the AHB-APB bridge, which is a slave of AHB and the only master of APB. Another bus architecture CoreConnect [19] contains an OPB (On-chip Peripheral Bus) protocol which has a structure similar to that of AMBA AHB.

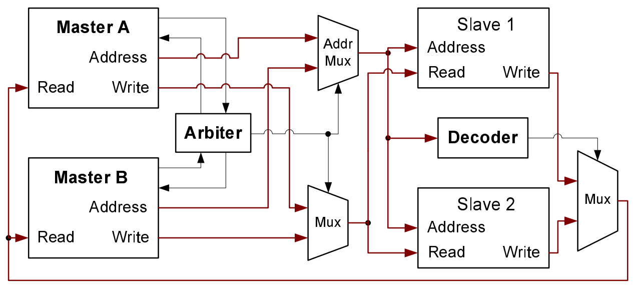

Figure 2: AMBA AHB on-chip bus

In these on-chip buses, the signals coming from a master go though a selection multiplexer, and then sent to all the slave inputs. This applies on the address bus, data bus, and reversely on the slave-to-master data bus. It may be the simplest way of implementing the bus architecture, but as mentioned before (figure 1), the power efficiency of this implementation is quite low. The entire wire net plus all the input stages of connected components are all driven by the multiplexer, while only one component needs the signals. Due to poor scaling of global wires and increasing number of components in SoCs, the limitations of such connections may seriously deteriorate system power consumption.

Gated bus is a solution to eliminate the wasted dynamic power. Like clock gating, bus gating can mask off unnecessary load capacitance on the net, which is especially large on AMBA bus wire nets. Besides, the physical routing of wire nets can change from minimizing total wire length to minimizing individual path lengths, which can further reduce wire capacitance induced by data transactions. Meanwhile, the main arbitration and control mechanism in the original bus architecture does not need to be changed.

2.2 Design flows

Physical-level design of electronic systems starts from gate level netlists and goes through placement, routing, timing/power analysis, *etc.* Since the buses are included in the system as components, bus gating will result in updates on the original netlists of logic design. Figure 3 shows the stage of bus gating fitted in the physical design flow.

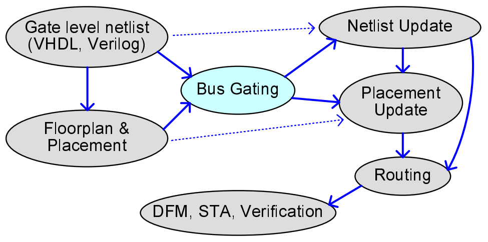

Figure 3: Physical design flow with bus gating

Like clock gating, the added gating stage is dependent on circuit placement, while the placement may also be updated by the gating. This may bring extra complexity on the flow, but the update by bus gating is not large and can be automated with appropriate algorithms.

3. MINIMIZING DYNAMIC POWER

3.1 Distributed multiplexer and demultiplexer

A simple bus gating technique is to add a demultiplexer after the multiplexer, so that only a single path of bus wires is driven at a time. The selection signals are from the arbiter at masters' side and from the decoder at slaves' side. In this way, not only the wire branches to other components are masked off, the connection can also take the shortest path to minimize wire capacitance. However, there are also two problems: single demultiplexer requires more routing resources, and the path length of different data transactions cannot be all minimized.

By distributing the big multiplexer and demultiplexer in to small ones over the nets, the wires can be shared by paths and therefore reduced. The CoreConnect OPB [19] bus supports distributed multiplexers for design flexibility, but no demultiplexers are used. If we add both to AMBA AHB or CoreConnect OPB, as shown in figure 4, the effect is both reduced wire length and masked off wire capacitance. There is an overhead of distributing the control signals, but the number of such wires is small compared to the bus width (at least 32-bit address lines and 32-bit data lines). And

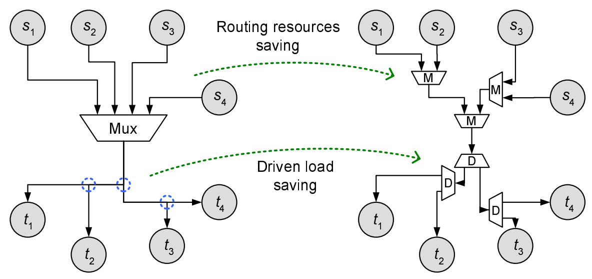

Figure 4: Using distributed multiplexers and demultiplexers on AMBA AHB

since the control signals remain still during a transaction, the dynamic power overhead is negligible.

In current bus architectures, the wire nets connecting all the master (or slave) inputs are best routed on a minimal Steiner tree. With distributed demultiplexers, the best result is a shortest-path Steiner tree/arborescence [15], [5], because on each transaction, signals are delivered on only one path. From the conclusion of [1], switching from RMST (rectilinear minimal Steiner tree) to MRSA (minimal rectilinear Steiner arborescence) only induces 2%∼4% of additional wirelength on average. So by the distributed demultiplexer alone, the simple bus gating can effectively eliminate most of the dynamic power on bus wires and input stages of bus components.

3.2 Shortest-path Steiner graph

The modified bus in figure 4 has a tree structure such that every path from its root (set between the last multiplexer and the first demultiplexer) to a leaf is the shortest path, *i.e.* an arborescence structure. However, in figure 4, the path length from source s_1 to terminal t_1 is larger than the Manhattan distance between them. The tree structure has limitations in that there is only one root placed at a certain point, which may not be on the shortest path of every connection. We use Steiner graph replacing the Steiner arborescence to enable overall path optimizations.

As defined in [2], for an unweighted graph $G = (V, E)$, a (possibly weighted) graph $G' = (V', E', \omega)$ is a Steiner graph of G if $V \subseteq V'$, and for any pair of vertices $u, w \in V$, the distance between them in G' (denoted $d_{G'}(u, w)$) is at least the distance between them in G ($d_G(u, w)$). In our bus communication, the original graph G has full connections between a set of masters $s_1, s_2, \cdots, s_m$ and a set of slaves $t_1, t_2, ..., t_n$. Additional vertices are added into G' as Steiner points. If for any s_i, t_j, $d_{G'}(s_i, t_j)$ equals their rectilinear distance, G' is a shortest-path Steiner graph.

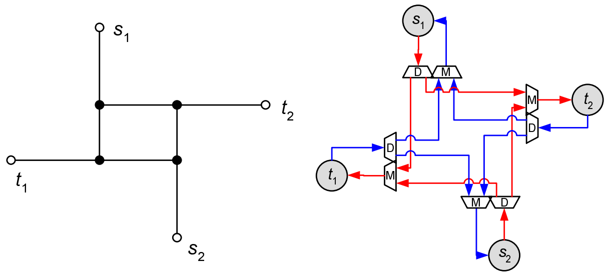

Figure 5: Shortest-path Steiner graph connections

Figure 5 shows a shortest-path Steiner graph of the connections between $\{s_1, s_2\}$ and $\{t_1, t_2\}$, and also its bus implementation on both master-to-slave and slave-to-master directions. This Steiner graph is minimal in terms of total wirelength. The loop in the center is necessary for including all the shortest paths of connections.

A bus architecture consumes minimal dynamic power using wire tracks of a shortest-path Steiner graph, combined with corresponding controls. A possible problem is that the loops require additional routing resources as well as control overhead. But sometimes the wirelength can also be reduced by sharing master-to-slave and slave-to-master connections. As illustrated in figure 6(a), the data bus in current bus architectures needs over $4l$ of length, because on each direction the multiplexer needs at least length l to connect s_1, s_2 and length l to connect t_1, t_2. We can reduce it to just over $2l$ by combining the nets on the two directions, since only one master device can access the bus at a time, double wires are enough to support all-direction transactions. Plus the address bus, the total wirelength is $4l$ vs $6l$. The signal switch in figure 6(b) acts as a mixed multiplexer and demultiplexer, controlled by 3 or less extra control signals. In average cases with well optimized Steiner graph, the wire length increment is small compared to the power reduction it brings (detailed results in section 5).

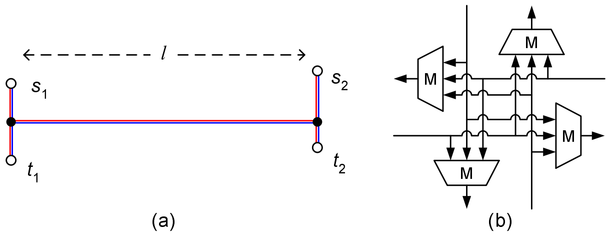

Figure 6: Sharing wires by controlled switches

4. HEURISTICS FOR GENERATING SHORTEST-PATH STEINER GRAPHS

In our problem of minimal shortest-path Steiner graph, the locations of a set of sources $s_1, s_2, \cdots, s_m$ and terminals $t_1, t_2, ..., t_n$ are given, and the objective is to find a rectilinear routing solution containing all the source-to-terminal shortest paths, with total wirelength as small as possible.

In single source cases, this is a rectilinear Steiner arborescence (shortest-path tree) problem which has been studied by [15], [5], *etc.* Finding the exact solution of a mininum rectilinear Steiner tree (MRST) is NP-complete [7], and finding a minimum rectilinear Steiner arborescence (MRSA) is believed to be hard [15] although without hardness proof. For practical use, several efficient heuristic algorithms are introduced and compared in [5], among which the 2-IDeA/G algorithm has the best average performance over runtime. In general cases containing multiple sources, the problem has not been studied before. We adopt the the 2-IDeA/G heuristic as basis and add more heuristics to construct the shortest-path Steiner graph.

4.1 k-IDeA/G heuristic for MRSA

The k-IDeA/G (iterated k-deletion for arborescence) algorithm [5] is based on the RSA heuristic (denoted RSA/G), which is proved to be 2-approximate in [15].

The basic flow of RSA/G is to start with n terminals as n subtrees and iteratively merge a pair of subtree roots v and v' such that the merging point is as far from the source as possible, so that the wires can be shared as much as possible. It terminates when only one subtree remains. For efficient implementation, the RSA/G first sorts all the nodes on the Hanan grid [17] in decreasing distance to the source s, and visits each node maintaining a peer set P of subtree roots. Details are in the pseudo code in table 1. We denote the rectilinear distance from s to v as $\Delta_s(v)$ or $\Delta(s, v)$. Two basic operations are used in RSA/G at: terminal merger opportunity (TMO), when a terminal is added into P as a subtree; and Steiner merger opportunity (SMO), when $|X| \geq 2$ and the subtrees in X are merged.

The heuristic of iterated k-deletion proposed in [5] is to

Table 1: The RSA/G algorithm

```
Given a source $s$ and $n$ terminals $t_1, \cdots, t_n$,
$v_1, \cdots, v_N$ are the Hanan grid nodes sorted by
$\Delta_s(v_1) > \cdots > \Delta_s(v_N)$;
---
$P \leftarrow \phi$;
for $i = 1$ to $N$ do
    if there is $t_j$ at $v_i$, then                    (TMO)
        $P \leftarrow P \bigcup \{v_i\}$;
    $X \leftarrow P \bigcap \{v_j | \Delta_s(v_j) = \Delta_s(v_i) + \Delta(v_i, v_j)\}$;
    if ($|X| \geq 2$) then                              (SMO)
        merge the nodes in $X$ rooted at $v_i$
        $P \leftarrow (P \bigcap \overline{X}) \bigcup \{v_i\}$;
return the arborescence rooted at $s$;
```

remove up to k nodes from $v_1, \cdots, v_N$ when running the RSA/G algorithm. By removing some nodes, the SMO merges are skipped at those points, which in some cases can result in better overall solution. In each iteration of the k-IDeA algorithm, all the combinations of skipping k or less nodes are tried in the RSA/G and the best set of skipped nodes are marked as permanently deleted. The iterations are then repeated until no further improvement is obtained.

4.2 Multiple MRSA construction with shared wires

With multiple sources $s_1, \cdots, s_m$, our algorithm needs to construct a Steinter graph G which contains all the MRSAs starting form every source. While each MRSA is minimized by k-IDeA, the m arborescences should share as much wire as possible to minimize total wirelength on G. We devise additional heuristics based on k-IDeA to construct multiple MRSAs one by one, explained as follows.

First, on each MRSA (rooted at s_i on ith iteration) construction, the terminals requiring connections can move towards the source s_i along existing edges of G, so that the wires can be reused and shared. As shown in figure 7, with the wires of previous aborescences, we only need to connect 8 nodes instead of the original 16 terminals to form the MRSA rooted at s_2, because all the other terminals can be reached from one of these nodes with shortest path from s_2. This set of nodes (denoted T') can be obtained by checking each terminal t_j, which necessitates a shortest-path connection from s_i. Starting from t_i, we move towards s_i as much as possible along existing wire paths until reaching a vertex v (a terminal or a Steiner node) in G where no vertex closer to s_i can be reached, then add it to T'. When there are multiple paths in the graph, we pick the final vertex closest to s_i so the rest part of the path is short and likely to need less wires. Details are in routine 'Necessitate(v)' in table 2.

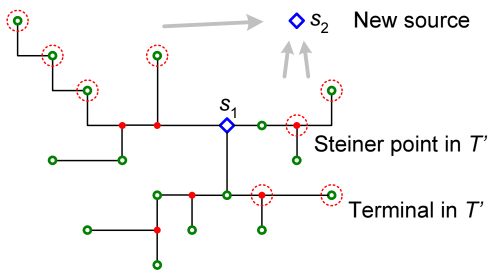

Figure 7: Nodes requiring connections

Second, we construct the MRSA on the set of nodes T' using existing wires. The TMO condition is then changed to $v_i \in T'$. The SMO condition is changed, also for the purpose of wire reusing, from $|X| \geq 2$ to $|X| \geq 2$ or ($|X| = 1$ and $v_i \in G$). Because when v_i is already in the graph, it can share wires with the node in X like the case when $|X| \geq 2$ in RSA/G. As figure 8 shows, when X contains only one node $\{t_2\}$, it should be connected into G when v_i comes to t_3, and half of the connection length can be saved using the existing horizontal wire. The detailed algorithm is described in table 2, where routine 'connect(u, v)' uses existing wires if applicable on shortest connections.

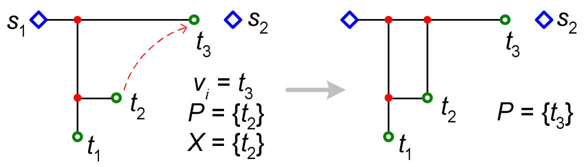

Figure 8: Connecting a node into the Steiner graph

Table 2: Revised RSA/G' algorithm

```
Given existing Steiner graph $G$, source $s_k$, terminals
$t_1, \cdots, t_n$, and $v_1, \cdots, v_N$ are same as in RSA/G;
---
Routine Necessitate(vertex v);
    $U \leftarrow \{u \in G$ and exists a wire path from
        $v$ to $u$ of length $\Delta_{s_k}(v) - \Delta_{s_k}(u)\}$;
    $T' \leftarrow T' \bigcup \{u_m \in U$ with minimum $\Delta_{s_k}(u)\}$;
---
$T' \leftarrow \phi$;
for $i = 1$ to $n$ do Necessitate($t_i$);
$P \leftarrow \phi$;
for $i = 1$ to $N$ do
    if $v_i \in T'$ then $P \leftarrow P \bigcup \{v_i\}$;          (TMO)
    $X \leftarrow P \bigcap \{v_j | \Delta_{s_k}(v_j) = \Delta_{s_k}(v_i) + \Delta(v_i, v_j)\}$;
    if ($|X| \geq 1$ and $v_i \in G$) then                   (SMO)
        for each ($u \in X$) connect($v_i, u$);
        $P \leftarrow P \bigcap \overline{X}$;
        Necessitate($v_i$);
    else if ($|X| \geq 2$) then                            (SMO)
        merge the nodes in $X$ rooted at $v_i$
        $P \leftarrow (P \bigcap \overline{X}) \bigcup \{v_i\}$;
return;        (the MRSA rooted at $s_k$ is added to $G$)
```

The k-IDeA iterations remain unchanged. After the shortest path Steiner graph is constructed by applying k-IDeA on the m sources, there are possibly some redundant edges that can be removed. So the final step is to check each edge $(v_i, v_j) \in G$, if G still contains all the master-to-slave shortest paths without (v_i, v_j), remove edge (v_i, v_j).

4.3 Switch control signals and wires

Besides the main part of bus wires, we have control signals for the switches on each Steiner node or terminal of G. When master s_i obtains the bus access from the arbiter and calls slave t_j by giving its address to the decoder, the connection is established through a shortest path in G, and the switches will guide the signals along this path. With pre-computed shortest paths, the control signals of each switch can be saved in a lookup table $C_{dat}(v, s_i, t_j)$, $v \in G$.

The control on data bus is relatively simple. The degree of a Steiner node can be 3 or 4 (s_i or t_j on a 4-degree node can be connected beside the node through a 3-way switch).

For a full 4-way switch as in figure 6(b), there are $6 = \binom{4}{2}$ configurations to connect 2 ports out of 4, so 3 wires are enough to send control signals (2^3 configurations) for this switch. And for a 3-way switch with $3 = \binom{3}{2}$ configurations, which is most common on the Steiner graph on random test cases, 2 wires are needed. Note that the connection is always between a master and a slave, so when some ports of a switch lead to only masters (or only slaves), the number of configurations may be reduced. For instance, every Steiner nodes in the tree of figure 7 has degree 3, but only 1 edge leads to the source (tree root). Thus, each switch needs only one control signal to guide the master-to-slave connection.

For address bus, the master-to-slave connection control can be shared with the data bus control. But there is also a master-to-decoder connection for each s_i, so the address signals go through another path to the decoder. All these paths form an MRSA, which is a subgraph of G. Each switch needs 1 or 2 control signals from a separate lookup table $C_{add}(v, s_i)$.

5. EXPERIMENTAL RESULTS

We compare three types of buses in our experiments: a) the original AMBA AHB, b) Steiner arborescence bus (as figure 4), and c) shortest-path Steiner graph bus. The three bus architectures are constructed on a set of cases with fixed placement of components. We list the average wire capacitance driven by data transactions in table 3 and the total wirelength in table 4.

For the original AMBA AHB, the wire net connecting the $s_1, \cdots, s_m$ (or $t_1, \cdots, t_n$) is routed as a Steiner tree. There are heuristics such as BI1S [8] and BOI [10] producing near optimal RMSTs (rectilinear minimum Steiner tree). We implement BOI to minimize the Steiner tree, and the multiplexer (MUX) is placed at the Steiner node where the total wirelength from $t_1, \cdots, t_n$ (or $s_1, \cdots, s_m$) is minimal. The bus control units (arbiter and decoder) in all cases are placed at center of the chip so that the overall wirelength is relatively small for control signals.

Table 3: Average data transaction wire capacitance

Case(m, n)	AHB	MRSA	SPSG	P$_{save}$
T0 (1,16)	5934	1638	469	72.4%∼92.1%
T0 (2,16)	7584	2138	613	71.8%∼91.9%
T0 (3,16)	8418	2038	635	75.8%∼92.5%
T1 (3,16)	14550	4067	1754	72.0%∼87.9%
T2 (2,30)	12305	2074	669	83.1%∼94.6%
T3 (3,16)	8230	2019	691	75.5%∼91.6%
T4 (5,15)	9674	1862	689	80.8%∼92.9%
T5 (6,16)	11210	2012	694	82.1%∼93.8%
T6 (8,8)	8952	1945	689	78.3%∼92.3%
T7 (12,6)	12107	1933	657	84.0%∼94.6%
T8 (16,10)	14377	1905	683	86.7%∼95.2%
T9 (8,16)	11652	1883	618	83.8%∼94.7%
T10 (8,16)	12899	2103	749	83.7%∼94.2%
T11 (6,12)	8970	1897	668	78.9%∼92.6%
T12 (12,12)	13242	1936	669	85.4%∼94.9%

Table 3 shows the large power saving on wires by bus gating. For each transaction, bus gating can reduce the driven wires from a large net to a single path, by which the wire capacitance on data bus is reduced by over 90% for SPSG bus. Here we only count shortest paths on data bus, because large, power intensive transactions can be transmitted by burst operation [18] which sends only one value on address bus. By the data in [12], the power on wires presents 14% of the bus power. Another 35% power consumed by bus interfaces (input stage of masters and slaves) is also reduced by about $\frac{m-1}{m}$ or $\frac{n-1}{n}$, because the unselected devices consume power by merely observing the AHB traffic in current AMBA AHB, and by [16] the wasted dynamic power dominates under the 0.15μm technology in [12]. In conclusion, nearly half of the power consumed by the AMBA AHB bus can be saved with bus gating. The percentage may vary as leakage power scales up with technology, but the bus wires are also scaling up and becoming a more important factor [9]. Considering that the bus consumes about 15% of the total system power in [12], bus gating may result in 5%∼10% of system power reduction.

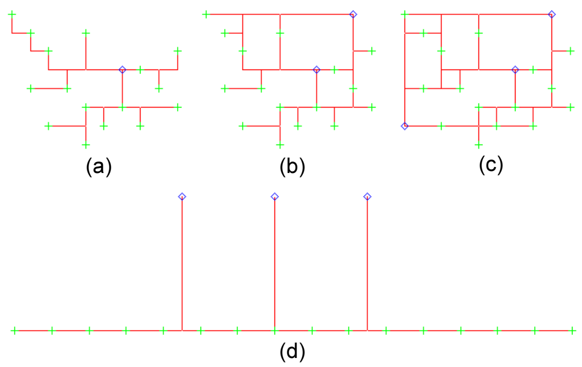

Figure 9: Shortest-path Steiner graphs of T0 & T1

Table 4: Total wirelength

Case(m, n)	AHB	MRSA	SPSG	L$_{increment}$
T0 (1,16)	11200	10316	9656	-7.9%∼ −13.8%
T0 (2,16)	14000	11850	14078	−15.4%∼0.6%
T0 (3,16)	16000	13350	21697	−16.6%∼35.6%
T1 (3,16)	27000	15131	18238	-44.0%∼−32.5%
T2 (2,30)	23257	16386	24535	−29.5%∼5.5%
T3 (3,16)	15248	11554	14274	-24.2%∼−6.4%
T4 (5,15)	17840	13729	22792	−23.0%∼27.8%
T5 (6,16)	20988	15778	35287	−24.8%∼68.1%
T6 (8,8)	16623	15800	25297	−5.0%∼52.2%
T7 (12,6)	22640	13227	27113	−41.6%∼19.8%
T8 (16,10)	27316	17018	40844	−37.7%∼49.5%
T9 (8,16)	22183	16055	35267	−27.6%∼59.0%
T10 (8,16)	24173	15906	34337	−34.2%∼42.0%
T11 (6,12)	16920	12566	22070	−25.7%∼30.4%
T12(12,12)	25369	17256	38305	−32.0%∼51.0%

The total wirelength (including control wires) of the bus may increase if we ensure all the master-to-slave paths are shortest. With regular placement of components, such as in figure 6 or the artificial case T1 in figure 9(d), the wirelength of SPSG bus can be smaller than AMBA AHB. In random cases (T2∼T12), the wirelength is increased because more loops are produced in order to include all the shortest paths, especially when n and m are both large like the cases T5 and T12 in figure 10. T5 has the Steiner graph with most wirelength increment, where we can see some long but thin loops. If we squeeze the loop and combine the two long edges, we reduce wirelength by just slightly increasing the length of some master-to-slave paths. When the routing resource is not abundant, we can reduce wirelength by this operation. Ultimately if we use the MRSA bus which has on average 30% less wirelength than the original AMBA AHB, we still achieve around 80% power reduction on wires and the same mask-off effect on unselected bus interfaces.

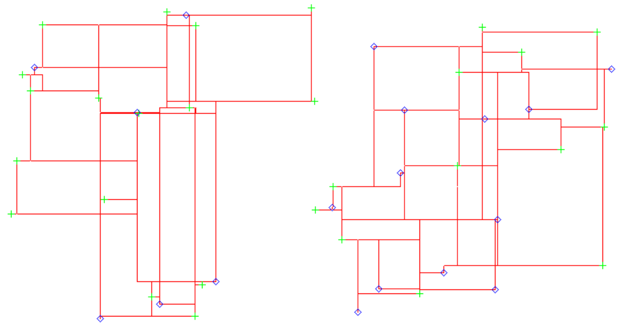

Figure 10: Shortest-path Steiner graphs of T5 & T12

We see a tradeoff between power consumption and wirelength. Figure 11 is drawn by data in the tables above. We have a flexible choice depending on routing resources, while in any case bus gating can always save a large percentage of power consumption on wires. The overhead power on additional controls and switches is low, considering the total number and activity rate of control signals, and that repeaters are usually inserted in bus wires to reduce delay.

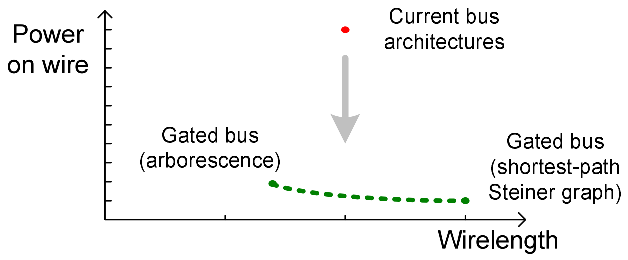

Figure 11: Tradeoff between power and wirelength

6. CONCLUSIONS AND FUTURE WORKS

Bus gating is a technique of reducing power consumption using the same effect as clock gating. Due to that only one pair of master-slave is involved at a time, bus gating can be more effective in terms of the percentage of power reduction. This percentage may become even more significant as technology scales down and the size of global wires relatively scales up.

Moreover, the Steiner graph approach also provides a possibility to allow multiple access as a bus matrix. In fact, if we are not restricted on resources, a full bus matrix in [13] added with demultiplexers can be used as a min-power bus, because the connection between each pair of master-slave is routed independently. In practice where the resources are usually limited, the Steiner graphs in figure 10 with appropriate controls also enable some pairs of connections to be active simultaneously, similar to independent train routes running on streets. Optimization on these multiple connections is a more challenging problem. Based on our shortest-path Steiner graph, power-bandwidth co-optimization on bus matrix will be explored in future works for more efficient on-chip communications.

7. ACKNOWLEDGEMENTS

This research is under the support of NSF CCF-0811794 and California MICRO Program.

8. REFERENCES

[1] C. J. Alpertt, A. B. Kahng, C. N. Szet, and Q. Wang. Timing-driven Steiner trees are (practically) free. *ACM/IEEE Design Automation Conf.*, pages 389–392, 2006.

[2] B. Bollobás, D. Coppersmith, and M. Elkin. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Math.*, pages 1029–1055, 2005.

[3] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi, and C. Turchetti. System-level power analysis methodology applied to the AMBA AHB bus. *Design, Automation and Test in Europe: Designers' Forum*, 2:20032, 2003.

[4] J. Y. Chen, W. B. Jone, J. S. Wang, H. I. Lu, and T. F. Chen. Segmented bus design for low power systems. *IEEE Trans. VLSI Systems*, 7(1):25–29, 1999.

[5] J. Cong, A. B. Kahng, and K.-S. Leung. Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design. *IEEE Trans. Computer-Aided Design*, 17(1):24–39, Jan. 1998.

[6] M. Donno, A. Ivaldi, L. Benini, and E. Macii. Clock-tree power optimization based on RTL clock-gating. *ACM/IEEE Design Automation Conf.*, pages 622–627, 2003.

[7] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Math.*, 32:826–834, 1977.

[8] J. Griffith, G. Robins, J. Salowe, and T. Zhang. Closing the gap: Near-optimal Steiner trees in polynomial time. *IEEE Trans. Computer-Aided Design*, 13:1351–1365, 1994.

[9] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings IEEE*, 89:490–504, 2001.

[10] C.-T. Hsieh and M. Pedram. An edge-based heuristic for Steiner routing. *IEEE Trans. Computer-Aided Design*, 13(12):1563–1568, Dec. 1994.

[11] C.-T. Hsieh and M. Pedram. Architectural power optimization by bus splitting. *Design, Automation and Test in Europe*, pages 612–616, 2000.

[12] K. Lahiri and A. Raghunathan. Power analysis of system-level on-chip communication architectures. *Int'l Conf. Hardware-Software Codesign and System Synthesis*, pages 236–241, 2004.

[13] S. Pasricha, Y.-H. Park, F. J. Kurdahi, and N. Dutt. System-level power-performance trade-offs in bus matrix communication architecture synthesis. *Int'l Conf. Hardware-Software Codesign and System Synthesis*, pages 300–305, 2006.

[14] M. Powell, S. H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories. *Int'l Symp. Low Power Electronics and Design*, pages 90–95, 2000.

[15] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor. The rectilinear Steiner arborescence problem. *Algorithmica*, 7:277–288, 1992.

[16] S. Rodriguez and B. Jacob. Energy/power breakdown of pipelined nanometer caches (90nm/65nm/45nm/32nm). *Int'l Symp. Low Power Electronics and Design*, pages 25–30, 2006.

[17] M. Zachariasen. A catalog of Hanan grid problems. *Networks*, 38:200–1, 2000.

[18] AMBA 2.0 specification. *http://www.arm.com/products/solutions/AMBA_Spec.html*, 1999.

[19] Coreconnect bus architecture. *IBM White Paper*, 1999.

[20] Avalon interface specifications. *http://www.altera.com/literature*, 2008.