International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.4, July-Aug. 2012 pp-2735-2741 ISSN: 2249-6645

Power Efficient and Reuse of Memory with Steiner Graphs

Aditya Putta[1], B. Chinna rao[2], P. M. Francis[3]
[1]*M. tech (PG student) Gokul Eng College*
[2]*Prof. &Head,Dept.of ECE, Gokul Eng College*
[3]*Asst.Prof. in Dept. of ECE, Gokul Eng College*

Abstract: *Rapid demand on system-on-chip(SoCs) and communication increases with the help of very-large-scale integration (VLSI) circuits, even though prime factor is to reduce the Power consumption and the thermal wall have become the major factors limiting the speed of while interconnect is becoming a primary power consumer. High bandwidth is desired to enhance parallelism for better performance, and the power efficiency on this bandwidth is critical to the overall SoC power consumption. Current bus architectures such as AMBA, Core connect, and Avalon are convenient for designers but not efficient on power. This paper proposes a physical synthesis scheme for on-chip buses and bus matrices to minimize the power consumption, without changing the interface or arbitration protocols. By using a bus gating technique, data transactions can take shortest paths on chip, reducing the power consumption of bus wires to minimal. Routing resource and bandwidth capacity are also optimized by the construction of a shortest-path Steiner graph, wire sharing among multiple data transactions, and wire reduction heuristics on the Steiner graph. Experiments indicate that the gated bus from our synthesis flow can save more than 90% dynamic power on average data transactions in current AMBA bus systems, which is about 5–10% of total SoC power consumption, based on comparable amount of chip area and routing resources.*

Index Terms: *Algorithm, communication graph, data throughput, physical synthesis, power efficiency, Steiner graph.*

I. Introduction

AS the feature size of process technology scales down, system-on-chips (SoCs) are capable of integrating more components and gaining higher complexity. Since clock frequency on single components is reaching a limit due to power and thermal limitations, better performance will be mostly exploited through parallelism [1], [3]. As a result, two factors determine that on-chip communication architectures are becoming a critical aspect in future systems. First, the communication latency and bandwidth among system components may become a bottleneck of performance. Second, the percentage of power consumed on inter-component communications in the whole system power has scaled up to a significant level [9], [13], [15]. Industrial on-chip bus standards include AMBA [29], [31], CoreConnect [30], Avalon [32], and so on. These existing standards can provide an interface for IP developers and a communication solution for system designers. Compared to the network-on-chip [10] type of communications, buses are small on silicon footprint, fast in terms of latency, and easy to implement. Moreover, the implementations can be reconfigured according to specific applications, enabling designers to apply various optimizations for best performance on available resources. The advantages of simplicity make buses popular in industrial SoC designs. However, current bus architectures are not power efficient on transferring data through bus lines. And since this part of power is scaling up as technology advances [13], it becomes a necessity to introduce physical level optimization on bus synthesis to minimize the power consumed by inter-component communication on bus lines. When high bandwidth is required on these buses, wire efficiency may also become low, which ultimately limits the system bandwidth capacity and performance. We propose a physical synthesis scheme for on-chip buses to eliminate the disadvantages in existing bus architectures, but not to change the existing protocols and component interfaces. Based on shortest-path Steiner graphs, efficiency on bus lines is maximized without the need to redesign system components and IP modules. Routing resource is also reduced without compromising low power. The cost on our new scheme is the additional silicon resource consumed by distributed controls and switches, which is scaling down by Moore’s law. Under technology trends, this physical synthesis scheme is capable of bringing a large improvement on power and performance based on current state-of-the-art on-chip buses and bus matrices.

A. Related Work

An elaborate power analysis on AMBA on-chip bus is performed in [15]-[18] where the detailed decomposition of power consumed by system components is obtained by simulation on NEC’s gate-level power estimator. Power saving techniques have been explored and applied extensively to break through the “power wall” of VLSI circuit performance. Clock gating [5] is nowadays widely used to reduce dynamic power, and power gating [20] is used to avoid unnecessary static power. In bus communications, a large part of the power is consumed on the wires of bus lines [15], which is relatively scaling up with technology and applications [9], [13]. Techniques of clock gating can be used on bus lines to achieve a similar goal, which is to mask off signals wherever they are not needed. Bus segmentation in [6] has such effect to help reduce dynamic power, but the effect is largely limited by tree structures topologies. Also in [18], a power performance tradeoff is analyzed on bus matrices, where a bus matrix is composed of a set of tree structured buses. We extend the structures from trees to graphs, using Steiner graph connections for a thorough optimization of “bus gating” to minimize the communication power. Topologies have been mostly discussed in bus optimizations, while the physical/geometrical information is not being emphasized.

B. Paper Overview

In this paper, to obtain the bus getting architecture and optimize bus communications, and get minimal tradeoff power maximal bandwidth, and minimal total wire length. the protocols are of AMBA AHB [29] and AXI [31], since they are most popular in industrial designs. And apply optimizations which is biased toward minimal power, but also favors bandwidth and routing resource.To construct a minimal shortest-path Steiner graph, and to reduce its scale with a minimal increment on path lengths. The overall optimization flow can be viewed as three major steps:
Step 1: generating the shortest-path Steiner graph *H* (for minimal power);
Step 2: deciding edge weights on *H*

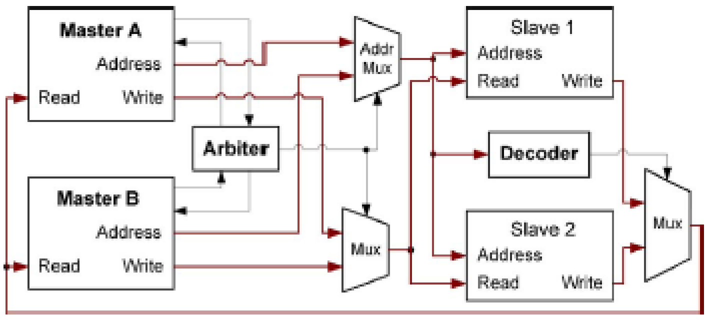


Fig 1 AMBA AHB bus

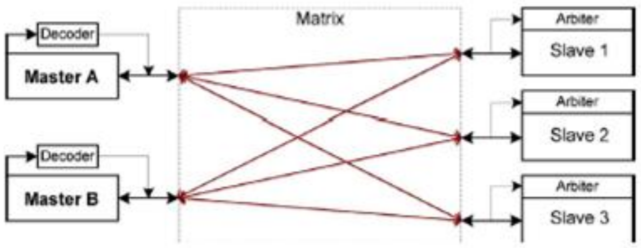


Fig 2 AMBA AXI full bus matrix (drawn)

Step 3: applying incremental modifications on *H* (for minimal wire length).
The rest of this paper is organized as follows. Section II introduces some background information on bus architecture and bus gating. Section III shows the heuristics for minimizing power and Section IV for minimizing wire length. Experiments are illustrated in Section V. Finally, Section VI gives our conclusions on bus matrix, with comparisons to network-on-chips and analogies to city traffic planning.

II. Bus Architectures and Bus Gating Background

Standard on-chip buses like AMBA were designed to enable fast and convenient integration of system components into the SoC, where simplicity is one of the major objectives. When the bus power consumption comes to a significant level that we cannot afford to ignore [15], power optimization will be desirable. We introduce a “bus gating” technique [23] to minimize the power on bus lines with a small compromise on design simplicity.

A. AMBA On-Chip Bus and Bus Matrix Architectures

The AMBA AHB on-chip bus [29] and bus matrix [31] are drawn in Figs. 1 and 2. The components connected by these buses can be classified into masters and slaves. Masters are typically microprocessors, each can start a transaction with one slave device at a time, where the slave is selected by giving an address to the decoder. Slave devices respond to masters passively. When conflicting requests come from multiple masters, arbiters will decide the order of services. The main difference between the bus and bus matrix is on multiple access from masters. The basic bus allows one master access at a time, while the bus matrix may allow multiple accesses. In a full bus matrix like Fig. 2, the masters and slaves are connected like a bi-clique, and each slave has an arbiter. Full bus matrices have largest bandwidth capacity, typically applied for maximum performance.

B. Power and Wire Efficiency of Gated Bus Using Steiner Graphs

The power efficiency of a bus architecture like Fig. 1 is low because the bus lines from masters to slaves are connecting all the slave devices by a single large wire net. The same is on slave-to-master connections. While the communication is one-to-one, the signals are sent to all the receivers regardless of whether they are needed, which results in wasted dynamic power on bus wires and component interfaces. Moreover, this low power efficiency is still being worsened by the technical scaling of global wires [13] and the increasing number of components integrated into SoCs. Gated bus is a solution to save the wasted dynamic power. The simplest way is to add a de-multiplexer after each multiplexer in Fig. 1, and add a de-multiplexer after each master device in Fig. 2, so that the signals only propagate to where they are needed. This method works in a similar way as clock gating [5], [11], and can be even more effective because the signal receivers here have much less complex behaviors than in a clock tree. For tree structured buses, distributing the multiplexer and de-multiplexer into the wire net [Fig. 3(a)] helps to save both power and wires. For wire length, while the single multiplexer needs independent lines from every sender, the lines can be shared with distributed multiplexers and form a Steiner arborescence [7], [21], [22]. An arborescence is a directed tree such that every root-to-leaf path is shortest. On the receivers’ side with distributed de-multiplexers, the bus lines change from a rectilinear Steiner minimum tree [12], [14] to a minimum rectilinear Steiner arborescence (MRSA). By the research in [2], this change increases the wire length by only 2–4% on average. So the total bus wire length can be reduced by the distributing the multiplexer/de-multiplexers, while the dynamic power can also be reduced at the same time. There is a small control overhead for sending the signals over the arborescence, but compared to the bus width and data throughput, this dynamic power overhead is negligible. Based on the same tree topology, effective bus gating can be applied by distributing the control over the entire tree (arborescence). On bus matrices, however, simply adding de-multiplexers may increase the total wire length, because when the number of master-to-slave paths becomes large, each path will need its own bus wires [as in Fig. 3(b)]. To reduce wire length in the bus matrix, also to further reduce power on the basic bus, we adopt the structures of Steiner graphs. A Steiner graph is a generalization of Steiner trees, without the limitation of tree structure that there is only one root placed at a certain point, which cannot be on the shortest path of every connection. By removing the constraint of tree topologies, we gain higher freedom to choose shortest paths for reduced power on data

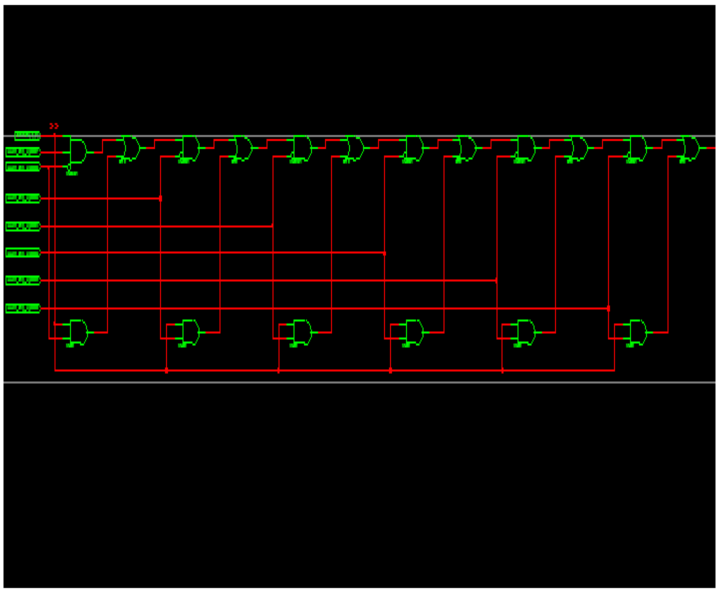


Fig 3 Bus gating using distributed mux and de-mux. (a) On single bus. (b) On bus matrix.

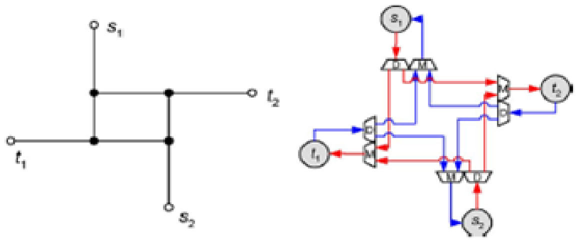


Fig 4 Shortest-path Steiner graph *Gn* and its bus implementation.

transactions, and to let the paths share wires for reduced. Shortest-path Steiner graphs have advantage on power efficiency as shown above. Naturally, graph structures also have advantage on communication bandwidth over trees. Our objective of bus gating and bus matrix synthesis is to perform a balanced optimization on power and bandwidth even when available routing resource is limited.

III. Bus Matrix Graph Construction

The flow we use is to first construct a shortest-path Steiner graph based on the given placement of Vs U Vt and communication graph Gc and then decide the weight ω(e) on each edge. The single-source case is the MRSA problem, which is well studied in previous work such as [7] and [21]. Although it is proved to be NP-complete in [22], heuristic algorithms can provide close-to-optimal solutions of Steiner arborescences.

RSA algo

```
Given a source s and n terminals t1, · · · , tn,
v1, · · · , vN are the Hanan grid nodes of {s, t1, · · · ,
tn}sorted by decreasing distance to s
Q←Ø
For I = 1 to N do
Else if
If there is Tj at Vi then Q←QUVi
X← Q∩{Vj:\\P(s-p(vj) ) =|| P(s ) – P(Vi)||+||P(Vi)-P(Vj)||}
If (|x|> 2) then merge the nodes in X rooted at Vi
Q←(Q∩X')U{Vj} return the arborescence at s'
```

Our shortest-path Steiner graph is constructed by multiple iterations of a revised MRSA construction.

A. k-IDeA/G Heuristic for MRSA

The RSA/G heuristic for the MRSA problem was first introduced in [21], and is proved to be 2-approximate. Given a single source and *n* terminals, the basic flow is to start with *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 [26]

In each iteration, it removes up to *k* nodes from *v*1, · · ·.. , *vn* some nodes are skipped it will reduse the memory and utilize the same location as adress when running the RSA/G algorithm. By removing the nodes, some SMO merges are skipped, which in some cases can result in a better overall solution. All the combinations of the *k* or fewer skipped nodes are tried in an iteration, and the best set of skipped nodes are marked as permanently deleted. The iterations are repeated until no further improvement occurs.

B. Shortest-Path Steiner Graph by Multiple MRSAs

For a shortest-path Steiner graph with multiple sources *s*1, · · · , *sm*, the idea behind single source MRSA is still valid. In fact, our algorithm constructs the Steiner graph *H* just by iteratively constructing the MRSA rooted at every source. While a single arborescence can be optimized by the *k*-IDeA heuristic, the *m* arborescences are individually optimized with the same idea, plus that these arborescences also need to share as much wire as possible to optimize the final Steiner graph. For this purpose, we add additional heuristics based on the RSA/G to construct multiple MRSAs one by one. First, starting from the second MRSA construction, we can reduce terminals by using existing wires. For each MRSA with source *si*, the terminals that need connection from the source can be moved along existing edges of *H* toward *si*. As the example shown in Fig. 5, with the wires of previous aborescences, we only need to connect eight 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 eight nodes by a shortest path from *s*2. This set of nodes (denoted as *T'*) can be obtained by checking each terminal *tj* , move from *tj* toward *si* as much as possible along existing paths until reaching a vertex (can be a terminal or a Steiner node) in *H* where no vertex closer to *si* can be reached, and add this vertex to *T'*. When there are multiple paths in the graph, we pick the final vertex closest to *si*, so the rest part of the path is short and likely to need less wires. Details are in the routine "Necessitate(*v*)" Second, we construct the MRSA based on the set of nodes *T'* using as much existing wires as possible. Compared to the RSA/G heuristic, the TMO condition is changed to *vi',* ΔT' he SMO condition is changed, also for the purpose of wire reusing, from |*X*| ≥ 2 to |*X*| ≥ 2 or (|*X*| = 1 and *vi*Δ *H*). Because when *vi* is already in the graph, it was added into previous MRSAs and can share wires with the node in *X* like the case in RSA/G when $|X| \geq 2$. As the example in

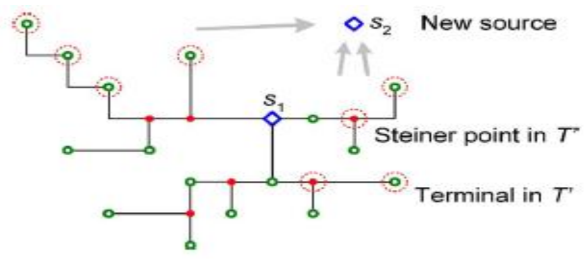


Fig 5 Nodes requiring connections (in dotted circles).

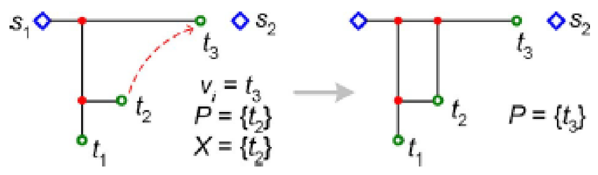
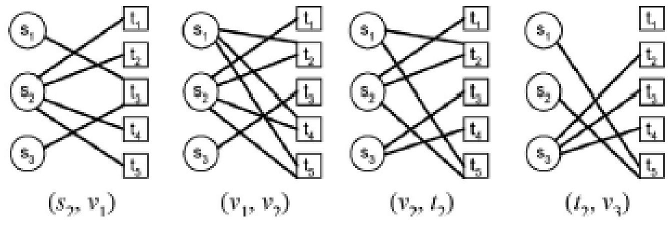


Fig. 6 Connecting a node into the Steiner graph.

shows, when *X* contains only one node {*t*2}, it should be connected into *H* when *vi* comes to *t*3, and half of the connection length can be saved using the existing horizontal wire. where the routine "connect(*u*, *v*)" uses existing wires if applicable on shortest connections. The *k*-IDeA iterations remain unchanged here. And 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 (*vi*, *vj*) Δ *H*, if *H* still contains all the source-to-terminal shortest paths without (*vi*, *vj*), then remove it from *H*.



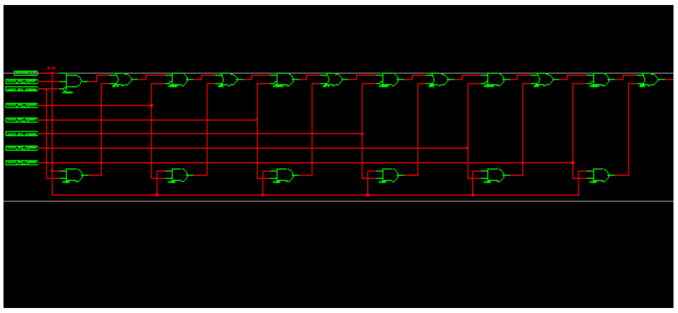


Fig7 Bipartite graphs of four edges in the bus matrix

i.e., all the $|Vs| \times |Vt| = 15$ arcs are present. The resulting bus matrix graph contains five Steiner nodes and 13 edges. Every arc from a master *si* to a slave *tj* has a connection of minimal length, and the 15 shortest paths shown in Fig. 7 are fixed. To assign a weight on each edge, we take *e* = (*s*2, *v*1) as example. Six of the 15 paths go through *e*, so *G'*(*e*) consists of the six corresponding arcs (*s*1, *t*3), (*s*2, *t*1), (*s*2, *t*2), (*s*2, *t*4), (*s*2, *t*5), and (*s*3, *t*3). The maximum matching has two edges, because *t*3 can only connect to one of *s*1 and *s*3. Therefore, $\omega(e) = 2$ is adequate to support all communication patterns. Fig.7 shows the bipartite graphs of four edges on the central horizontal line. Despite the number of connections, most of the edges are weighted 1. Yet this bus matrix graph is adequate for maximum bandwidth capacity, i.e., wires will not be the bottleneck of multiple simultaneous connections. The total weighted wire length in this bus matrix is 108. Compared to the total path length 266 if implemented as a full bus matrix in Fig. 2, the Steiner graph approach saves more than half of the routing resources.

IV. Tradeoffs on Power, Wire, and Bandwidth

A. Steiner Graph Reduction

Since high bandwidth bus matrices will need significantly more wires to support parallel communications across the chip, routing resource may become another limitation as more components are integrated into SoCs and interactions increase. Especially when the components are placed in irregular placement instead of cell arrays, the shortest-path Steiner graph generated by the algorithm in which bring additional wire length. We look for changes in the graph structure which can significantly reduce the wires, while preserving the short paths at the same time.

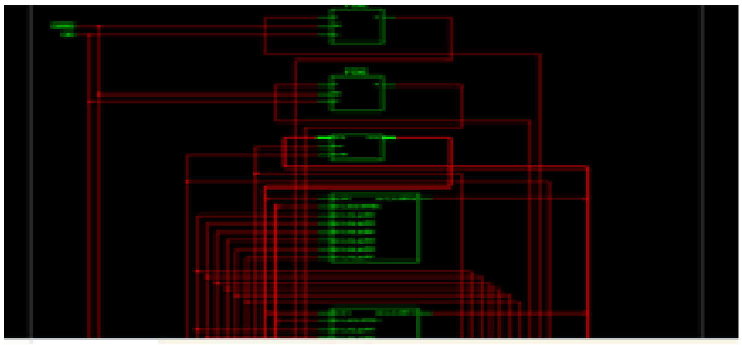


Fig. 8 Searching for mergeable parallel segments (in vertical direction).

when the double edges are geometrically very close to each other, combining them into one edge only slightly increases the length of some connections, while possibly saving much more wire length. Fig. 8shows the effect of merging parallel segments in narrow rectangles. The total edge length is greatly reduced, while the increment on average path length is relatively small. Although fewer edges will generally result in larger edge weight, the total weighted edge length (wire length) can still be reduced by this merging operation due to improved wire sharing among paths. Thus, if we relax the requirement on the path length in definition 4, from the exact Manhattan distance $||P(u)-P(v)||$ to within $(1 + \varepsilon)||P(u) - P(v)||$, we can merge the double parallel edges to save wires. Assume we have a vertical narrow rectangle with dimensions $h \times w$, and we merge the two vertical edges to a single edge placed in middle. The total edge length may be reduced by *h*, while the lengths of some connection paths increase by $w2+ w2 = w$. So if the h/w ratio is high, this operation can be very helpful on relieving routing congestion, while preserving the low power consumption of a bus matrix. In the wire length reduction algorithm, we repeatedly search for pairs of parallel double lines in the bus matrix graph, and for each pair, calculate its potential reduction Δ*l* on edge length and possible increment Δ*p* on path lengths. The pair with highest Δ*l*/Δ*p* ration is merged, and the modified graph will have a new set of connection paths and edge weights. If the added total wire length is really reduced, we keep the merging operation and continue to the next iteration, otherwise discard the operation. Eventually, there will be no positive wire length reduction in the graph, and we have a series of bus matrix graphs with decreasing wire length and increasing path lengths, where a comprise can be chosen. The process of searching for vertical mergeable parallel segments is illustrated in Fig. 8. (Horizontal lines are processed in the same way with *x*-*y* coordinates switched.) First, the vertical line segments in the Steiner graphs are sorted by their *x* coordinates, denoted as *u*1, *u*2, · · · , *uk*. Then for each pair of segments *ui*, *uj* (*i* < *j*) with a common *y* interval [*y*1, *y*2], if between *i* and *j* there is no other vertical segment on [*y*1, *y*2], *ui* and *uj* are a pair of mergeable segments. On the parallel segments *ui* and *uj* , let *cl* denote the count of horizontal lines connected to the left, *cr* denote the count of lines connected to the right, and *cm* the count of lines connecting *ui* and *uj* in the middle. Assume *cl* < *cr*, so the combined vertical segment may not be at the middle but have an offset *δ* to the right of the midpoint. The reduction on total edge length Δ*l* is by combining the vertical segments of length *h* and changing the lengths of related horizontal connections. The two vertical segments are reduced to one, which reduces edge length by *h*. The central *cm* edges of length *w* are totally removed. However, the lengths of *cl* connections on the left are increased by ω/2 +δ and the lengths of *cr* connections on the right are increased by ω/2 –δ To sum up Δ*l* =h+Cmω - Cl(ω/2 +δ) – Cr(ω/2 –δ) On the possible increment on path lengths, since the left vertical segment is pushed rightward by *w*2 +*δ*, a path may need to detour and add Δ*p* = *w* + 2*δ* of distance. So the ratio is

$$\Delta l/\Delta p = h+Cm\omega-Cl(\omega/2+\delta)-Cr(\omega/2-\delta)/\omega+2\delta$$
$$= Cr-Cl/2+h-(Cr-Cm)\omega/\omega+2\delta$$

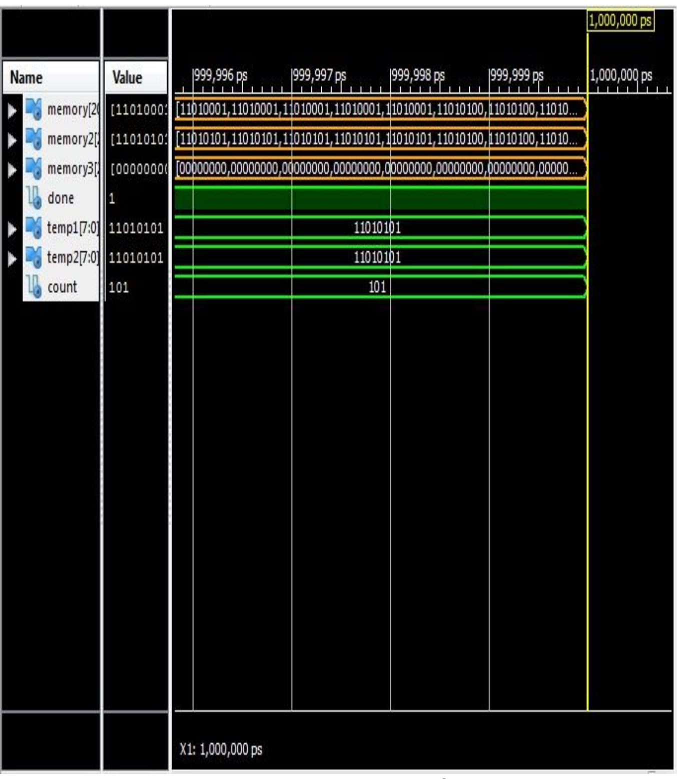


Fig 9 Merging stages of iterations

In Fig. 9, the stages of the merging iterations applied on a Steiner graph. First, the long and narrow rectangles are removed, followed by wider rectangles. still can achieve a significant reduction on total wire length in average cases Notice that the segment merging operation also helps to merge Steiner nodes which are generated very close to each other. In practice, locally congested Steiner nodes can be hard to implement, because each node needs the area for a switch box and its control unit. Our operation does not guarantee to resolve all closely placed nodes, since it prioritizes longer segments, and may leave small square-shaped sub graphs unchanged. Nevertheless, this situation can be easily resolved by a post-processing algorithm, which scans each Steiner node (denoted as *vi*), look at *vi*'s close neighbors within a small *d*×*d* box and compute the density of Steiner nodes in the area. For a box with too many nodes, we can shrink all the nodes in that box into one, and implement it by a single switch. The changes on the bus matrix graph by this operation are limited in the small box areas.

V. Bus Matrix Control Units and Wires

Apart from path lengths and data wire lengths, the control overhead needs to be considered for a complete optimization. Although the data lines consume the major amount of routing resource because they are usually at least 64 bit (32 bit × 2-way) wide, control overhead is increased compared to traditional bus architectures by adopting Steiner graphs. We need a lot of switches at Steiner nodes to guide the on-chip traffic, and each switch needs a certain number of control signals depending on its node degree and edge weights. Each slave device has an arbiter which handles the requests from masters and decides the connection. The result is sent to the central switch control unit, where all the connection paths are stored. Depending on the set of active paths, the central switch control sends control signals to all the switches on each path, which together instantly create the master-to-slave connection requested by the master device.

VI. Experimental Results

In our experiments, we implement all the related algorithms, including the shortest-path Steiner graph generation, Steiner graph reduction by parallel line merging, and the edge weight maximum matching. The programs are tested on Windows Vista platform with a 2.2 GHz Intel Core2 processor. The running time is short on all the test cases, because the algorithms are time/space efficient, and also because most SoC bus matrices will not need to connect too many components (under 32 in our cases). The test cases we use are mostly artificial, hand made (*T*0 and *T*1), or randomly generated (*T*2Δ12). They are the same cases used in [23] and [24]. In each test case, the master and slave devices are distributed over a 10mm × 10 mm square. The power consumption is estimated by the driven capacitance

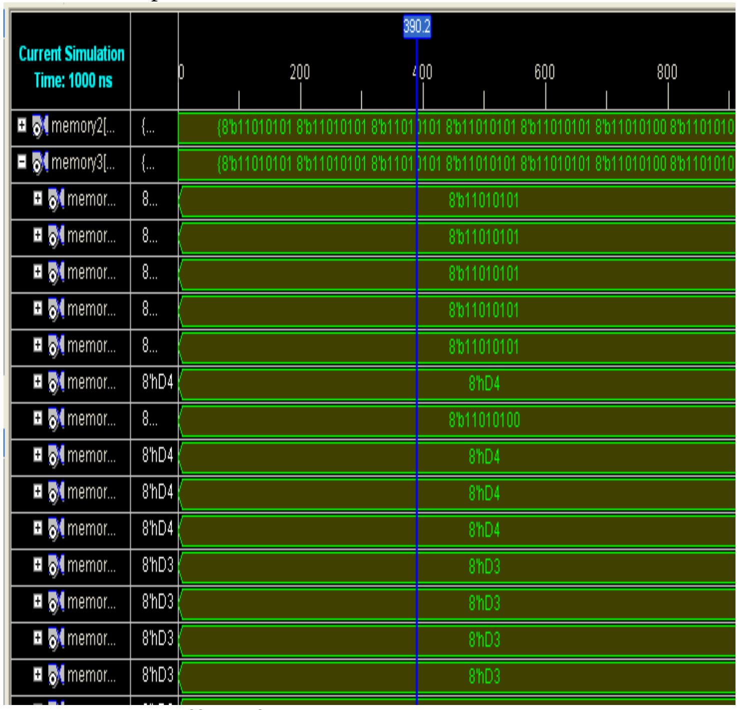


Fig 10 the effect of merging parallel segments with address location

of data transactions, and can be calculated as a linear combination of path length and switches along the path. Path lengths are minimized by the bus matrix graph construction, since wires are the major power consumer. For the purpose of data completeness, we add the power overhead from the switches on Steiner nodes. According to [27] and [28], we estimate that under 90 nm technology, each mux or demux in crossbar switches has about the same capacitance as 25 μm of wires. The total wire length on data wires and control overhead are added straightforwardly. Data wire length is the sum of weighted edge length in the bus matrix graph.

The objective can be minimum power (i.e., average path length), minimum wire length, or a combination of the two.

At the top of each column:

1)$\sum$*Lvs,vt* is the sum of Manhattan distances on all the master-slave pairs;

2)*Ltree* is the average induced path length (major dynamic power) of master-slave connections in tree structured AMBA AHB buses or bus matrices;

3)*Lpath* is the average path length (major dynamic power) of master-slave connections in the bus matrix graph;

4)*Pswitch* is the added percentage of power overhead in data transactions by the switches on Steiner nodes;

5)$\sum$*Lwire* is the total data wire length;

6)$\sum$*Wctrl* is the added percentage of control wire overhead.

In the minimal power section, the average path length is exactly $\sum$*Lvs,vt* /(*mn*), while the total wire length is about one ourth to one third of the total connection length. Compared to traditional bus implementation in [23], the dynamic power saving is mostly over 90% even with the switching overhead added. Overhead on dynamic power increases with the number of components increasing, which requires more bandwidth and larger switches. The percentage is generally under 20% on random cases with under 30 components. So the overall dynamic power here is close to optimal. On the overhead of control wires, the percentage is mostly under 10%, because the number of control signals required is usually very low compared to the 64 bit wide data lines.

In the minimal wire section, the bus matrix graphs are reduced by the parallel line merging heuristic. As a result the wire length on most cases is greatly reduced Compared to the reduced wire length, the increase on average path length is much lower, mostly around 10% and all under 20%. The power overhead percentage is also increased, because although the Steiner nodes are reduced, the switches along each path are not reduced as much in number, but increased in size. Still, these solutions are relatively power efficient, and we have series of intermediate solutions between minimal power and minimal wire are available for choice.

To see how the path lengths reflect communication power in SoCs, we calculate the bus power consumption with a fixed set of parameters. Assuming 1V of power voltage, 0.2 fF/μm

of wire capacitance, 4 Gb/s of transaction bit rate, and 20% of bus matrix activity rate, Table V lists the estimated power on bus matrix in each of our test cases. Again we can see a large reduction on total bus power (*Ppath* + *Pswitch*) compared to *Ptree* by traditional Steiner tree structures between certain master-slave pairs may only happen at some specific conditions. So instead of a set of arcs *A* in the communication graph, we can have a series of arc sets *A*1,*A*2, · · ·,*Ac*, each one smaller than the original set *A*, denoting a set of simultaneous connections.

VII. Conclusion

We optimized on-chip communications referring to the AMBA AHB bus (matrix) architecture. The weaknesses of original bus matrices, such as low power efficiency and low wire efficiency, are resolved by using a Steiner graph structure. Compared to network-on-chip which has better bandwidth flexibility, bus matrix has much less latency Therefore, we believe bus matrix architectures will be widely applied for efficient communications in various future systems. The principle of our work on reducing power is to minimize the data movement on the chip; and that on reducing wires is to maximize wire sharing among different connections. Devised algorithms which can

extensively exploit the on-chip physical design space for a thorough optimization on power and wire efficiency. The results show promising potentials of bus matrices for low power and high performance on-chip communications. More improvements can be explored in future works on formulations, algorithms, and the overall
Optimization flow.

References

[1] S. V. Adve, V. S. Adve, G. Agha, M. I. Frank, M. J. Garzar´an, J. C. Hart, W.-M. W. Hwu, R. E. Johnson, L. V. Kale, R. Kumar, D. Marinov, K. Nahrstedt, D. Padua, M. Parthasarathy, S. J. Patel, G. Rosu, D. Roth, M. Snir, J. Torrellas, and C. Zilles, *Parallel Computing Research at Illinois: The Upcrc Agenda*. Urbana, IL: Univ. Illinois Urbana-Champaign, Nov. 2008.

[2] C. J. Alpertt, A. B. Kahng, C. N. Szet, and Q. Wang, "Timing-driven Steiner trees are (practically) free," in *Proc. ACM/IEEE Des. Autom. Conf.*, Sep. 2006, pp. 389–392.

[3] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The landscape of parallel computing research: A view from Berkeley," Dept. Electric. Eng. Comput. Sci., Univ. California, Berkeley, Tech. Rep. UCB/EECS-2006-183, 2006.

[4] B. Bollob´as, D. Coppersmith, and M. Elkin, "Sparse distance preservers and additive spanners," *SIAM J. Discrete Math.*, vol. 19, no. 4, pp. 1029– 1055, 2005.

[5] L. A. Ca, Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," in *Proc. IEEE Custom Integr. Circuits Conf.*, vol. 47. Mar. 2000, pp. 415–420.

[6] 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. Very Large Scale Integr. Syst.*, vol. 7, no. 1, pp. 25–29, Mar. 1999.

[7] 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. Comput.-Aided Design*, vol. 17, no. 1, pp. 24–39, Jan. 1998.

[8] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Trans. Comput.-Aided Design*, vol. 11, no. 6, pp. 739–752, Jun. 1992.

[9] W. Dally, "Keynote: The end of denial architectre and the rise of throughput computing," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jul. 2009, p. xv.

[10] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection network," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2001, pp. 684–689.

[11] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2003, pp. 622–627.

[12] J. Griffith, G. Robins, J. Salowe, and T. Zhang, "Closing the gap: Nearoptimal Steiner trees in polynomial time," *IEEE Trans. Comput.-Aided Design*, vol. 13, no. 11, pp. 1351–1365, Nov. 1994.

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

[14] C.-T. Hsieh and M. Pedram, "An edge-based heuristic for Steiner routing," *IEEE Trans. Comput.-Aided Design*, vol. 13, no. 12, pp. 1563– 1568, Dec. 1994.

[15] K. Lahiri and A. Raghunathan, "Power analysis of system-level onchip communication architectures," in *Proc. Int. Conf. Hardw.-Softw. Codesign Syst. Synthesis*, 2004, pp. 236–241.

[16] K. Lahiri, A. Raghunathan, and S. Dey, "Efficient exploration of the SoC communication architecture design space," in *Proc. Int. Conf. Comput.- Aided Design*, 2000, pp. 424–430.

[17] S. Pasricha, N. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane, "Floorplan-aware automated synthesis of bus-based communication architectures," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2005, pp. 565–570.

[18] S. Pasricha, Y.-H. Park, F. J. Kurdahi, and N. Dutt, "System-level power performance tradeoffs in bus matrix communication architecture synthesis," in *Proc. Int. Conf. Hardw.-Softw. Codesign Syst. Synthesis*, 2006, pp. 300–305.

[19] A. Pinto, L. Carloni, and A. Sangiovanni-Vincentelli, "Constraint-drive communication synthesis," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2002, pp. 783–788.

[20] M. Powell, S. H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gatedvdd: A circuit technique to reduce leakage in deep-submicron cache memories," in *Proc. Int. Symp. Low Power Electron. Design*, 2000, pp. 90–95.

[21] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, "The rectilinear Steiner arborescence problem," *Algorithmica*, vol. 7, nos. 1–6, pp. 277– 288, 1992.

[22] W. Shi and S. Chen, "The rectilinear Steiner arborescence problem is np-complete," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2000, pp. 780–787.

[23] R. Wang, N.-C. Chou, B. Salefski, and C.-K. Cheng, "Low power gated bus synthesis using shortest-path Steiner graph for system-on-chip communications," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jul. 2009,pp. 166–171.

[24] R. Wang, E. Young, R. Graham, and C.-K. Cheng, "Physical synthesis of bus matrix for high bandwidth low power on-chip communications," in *Proc. ACM Int. Symp. Phys. Des.*, 2010, pp. 91–96.

[25] D. West, *Introduction to Graph Theory*. Englewood Cliffs, NJ: Prentice- Hall, 1999.

[26] M. Zachariasen, "A catalog of Hanan grid problems," *Networks*, vol. 38, no. 2, pp. 200–201, 2000.

[27] L. Zhang, H. Chen, B. Yao, K. Hamilton, and C.-K. Cheng, "Repeated on-chip interconnect analysis and evaluation of delay, power, and bandwidth metrics under different design goals," in *Proc. Int. Symp. Quality Electron. Design*, 2007, pp. 251–256.

[28] Y. Zhang, X. Hu, A. Deutsch, A. E. Engin, and C.-K. C. J. Buckwalter, "Prediction of high-performance on-chip global interconnection," in *Proc. Int. Workshop Syst.-Level Interconnect Prediction*, 2009, pp. 61– 68.

[29] *Amba 2.0 Specification*. (1999) [Online]. Available: http://www. arm.com/products/solutions/AMBA Spec.html

[30] "Coreconnect bus architecture," in *IBM White Paper*. 1999. [31] *Amba 3 Specification*. (2003) [Online]. Available: http://www.arm.com/ products/solutions/axi spec.html