Bus-Aware Microarchitectural Floorplanning

Dae Hyun Kim
School of Electrical and Computer Engineering
Georgia Institute of Technology
daehyun@ece.gatech.edu

Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology
limsk@ece.gatech.edu

***Abstract*— In this paper we present the first bus-aware microarchitectural floorplanning. Our goal is to study the impact of bus routability on other important floorplanning objectives including area, performance, power, and thermal. We developed a fast performance-aware bus routing algorithm, which is integrated into the floorplanning engine to ensure routability while optimizing other conflicting objectives. Our related experiments performed on high performance processors show that we obtain 100% routability at the cost of minimal increase on area, performance, and power objectives under thermal constraint.**

I. Introduction

Microarchitectural floorplanning bridges the gap between computer architecture and physical design to effectively address performance, power, and thermal problems. The geometric information available from floorplanning allows architects to efficiently tackle issues such as interconnect delay, thermal coupling, etc. On the other hand, the architectural simulation results can be exploited during the floorplanning stage for more effective optimization on these metrics.

All existing works on microarchitectural floorplanning [1], [2], [3], [4], however, fail to address the impact of buses that connect the functional units. These buses typically are very wide and long, thereby consuming significant area, delay, and power. In our architecture, we have 51 buses with the widest one covering a functional module almost completely. Since these buses obstruct other buses, routability becomes an important factor to consider. Even a floorplan optimized for other objectives may not be routable because of wide and long buses. Moreover, the actual bus length has significant impact on IPC (instructions per cycle) and power consumption of the underlying architecture. In addition, if the buses are routed over hotspots, the temperature-dependent resistance will increase the delay and reduce performance.

Recent works on bus-aware floorplanning are targeting circuit designs, not architectural designs. Xiang *et al.* [5] presented bus-aware floorplanning so that all modules in a single bus are aligned vertically or horizontally. This work does not allow any bend in the buses. Law and Young [6] extended [5] by allowing bends in the buses, but this is not suitable for microarchitecture design as it forces the functional units to be aligned and compromise performance. Rafiq *et al.* [7] integrated bus routing into floorplanning, but this algorithm considers only two-pin buses. In addition to the restrictions on the bus topology, these works fail to address the impact of bus routability on performance/power/thermal issues.

In this paper, we present a simple but powerful bus-aware microarchitectural floorplanning. Our goal is to study the impact of bus routability on other important floorplanning objectives including area, performance and power under thermal constraint. Our thermal analyzer models the heat dissipated by the buses as well as the functional modules. Our related experiments performed on high performance processors show that we obtain 100% routability at the cost of minimal increase on area, performance and power objectives under thermal constraint.

*This material is based upon work supported by the National Science Foundation under CAREER Grant No. CCF-0546382 and Focus Center for Circuit & System Solutions (C2S2), Semiconductor Research Corporation.

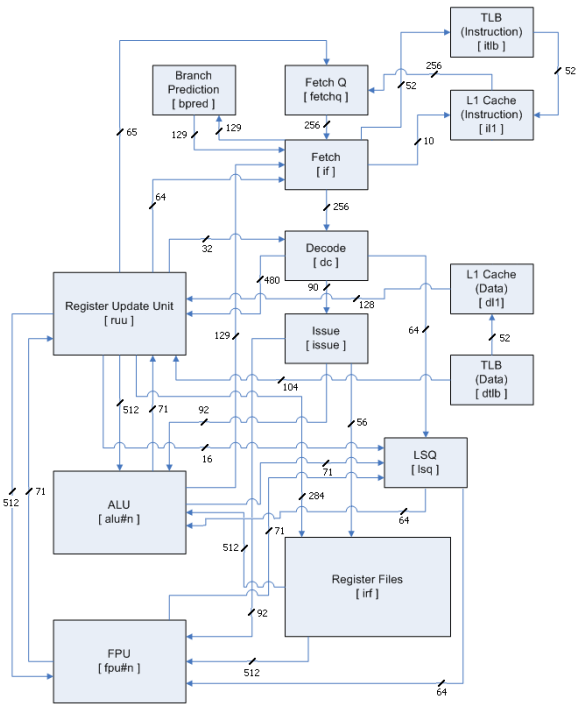

Fig. 1. Overview of our microarchitectural bus model

II. Architectural Design and Simulation

Figure 1 shows the architectural modules and buses used in our experiments. We assume that the issue width is four, and the microarchitecture is a 64-bit machine. We have four ALUs and four FPUs, so the total number of functional units is 20.

A. Integrated Design Flow

Our design flow consists of three steps. During the first simulation step, we use SimpleScalar [8] to collect the access pattern for each module and bus for the given application. We also use Wattch [9] to collect power-related data so that the power consumption for each functional module and bus are computed. During the second floorplanning step, we optimize Sequence Pair [10] representation of the floorplan using Simulated Annealing. For a given candidate floorplanning solution, we quickly measure its routability, IPC, area, power, and thermal metrics for evaluation (details are presented in Section V). We use these quick estimates of the metrics since an accurate computation would be computationally prohibitive if repeated for many candidate floorplans. During our last validation step, we perform architectural simulation to accurately compute the

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

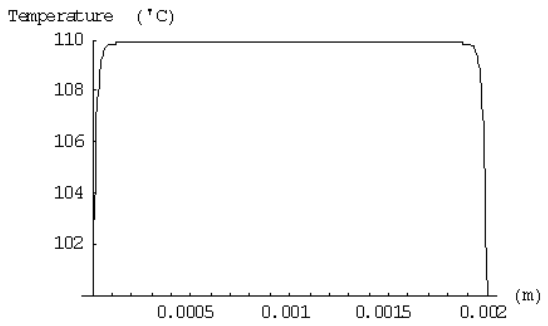

Fig. 2. Thermal profile of 1-bit wire (length:2000um, T_R = $T_{line}(0)$ = $T_{line}(2000um)$ = 100^oC)

new IPC value and use HotSpot [11] for thermal analysis on the final floorplan.

III. Bus Thermal Modeling

A. Bus Modeling for Thermal Profile

Heat diffusion equation in single dimension bus is stated in [12] and [13] as follows:

$$\frac{d^2T_{line}(x)}{dx^2} = \lambda^2 T_{line}(x) - \lambda^2 T_{ref}(x) - \theta \quad (1)$$

where

$$\lambda^2 = \frac{1}{k_m}\left(\frac{k_{ins}}{t_m \cdot t_{ins}} - \frac{{I_{rms}}^2 \cdot \rho \cdot \beta}{w^2 \cdot {t_m}^2}\right) \quad (2)$$

$$\theta = \frac{{I_{rms}}^2 \cdot \rho}{w^2 \cdot {t_m}^2 \cdot k_m} \quad (3)$$

In the above equations, k_m is the thermal conductivity of the interconnect material, k_{ins} is the thermal conductivity of an insulator of thickness t_{ins}, and t_m and w are thickness and width of the interconnect, respectively. ρ is the metal electrical resistivity and β is the temperature coefficient of resistance.

Assuming $T_{ref}(x)$ is constant in short distance L, the solution of the equation (1) becomes:

$$T_{line}(x) = A \cdot e^{\lambda L} + B \cdot e^{-\lambda L} + (T_R + \frac{\theta}{\lambda^2}) \quad (4)$$

where

$$A = \frac{T_L \cdot e^{\lambda L} - T_0 - (T_R + \frac{\theta}{\lambda^2})(e^{\lambda L} - 1)}{e^{2\lambda L} - 1} \quad (5)$$

$$B = \frac{-T_L \cdot e^{\lambda L} + (T_R + \frac{\theta}{\lambda^2})(e^{\lambda L} - e^{2\lambda L}) + T_0 \cdot e^{2\lambda L}}{e^{2\lambda L} - 1} \quad (6)$$

with boundary conditions $T_0 = T_{line}(0)$, $T_L = T_{line}(L)$ and $T_R = T_{ref}(x)$.

Figure 2 shows an example. The $T_{line}(x)$ is almost constant between $x = 0$ and $x = L$, and rapidly changes near the end-points due to the exponential terms in the equation (4). Therefore we approximate the temperature to the constant in the $T_{line}(x)$ if the L is sufficiently small:

$$T_{line}(x) = T_{constant} = T_R + \frac{\theta}{\lambda^2} \quad (7)$$

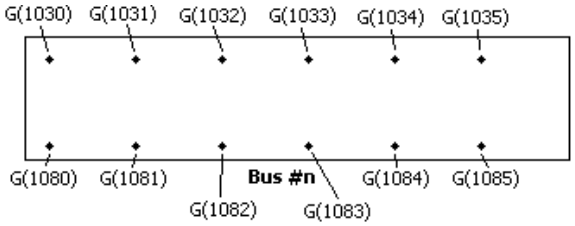

Fig. 3. Illustration of bus temperature computation in our extended HotSpot

B. HotSpot Simulation

We integrate our bus thermal modeling equations into HotSpot [11]. HotSpot partitions the floorplan into $m \times n$ grid and computes the temperatures of each grid point. The following is our algorithm to compute the final temperature while considering the buses:

Algorithm Bus Thermal Analyzer
for (all grid points)
  if (the grid is overlapping with any bus)
    for (four directions left, right, up, and down)
      compute the constant $T_R + \theta/\lambda^2$;
    pick up the maximum value for the grid;
  save the new temperature;
restore the new temperatures;

For instance, let the current grid be G(1081) in Figure 3. Since there are three adjacent grid points G(1031), G(1080) and G(1082), we compute the thermal constants (= Equation (7)) between G(1081) and these neighboring points. Then we pick up the maximum constant value and use it for the temperature of G(1081). Notice that maximum constants are used to estimate the worst cases. We save these constant values and reuse them for computing the temperature of other grid points. The temperature continues to increase once the new constant values are used for the computation of other grid points. Section V-A presents experimental results on our bus thermal modeling.

IV. Bus-Aware Floorplanning Algorithm

Given a set of microarchitectural modules and buses, our bus-aware microarchitectural floorplanning problem seeks to determine the location of each module such that (i) there is no overlap among modules, and (ii) all buses are routed. We assume that one pair of horizontal and vertical layers is given for bus routing. Our objective is to maximize IPC (instructions per cycle) while minimizing the number of bends on each bus, footprint area of the floorplan, and power consumption to drive buses under the user-specified thermal constraint.[1]

A major difference between circuit-level vs microarchitecture-level bus routing is on how to optimize performance: circuit-level bus routing is focused on clock period minimization, whereas the microarchitecture-level routing is targeting IPC improvement. At circuit-level, more details on critical paths such as types and number of gates as well as their placement are available. This makes the delay computation and optimization more accurate compared with microarchitecture-level. On the other hand, microarchitecture-level routing can exploit the statistics of the application execution available from architectural simulation. This allows the router to target runtime behavior of the chip more directly, resulting in higher IPC.

[1]We assume the interconnects are pipelined so that the clock period remains fixed. In this case, longer interconnects incur higher latency.

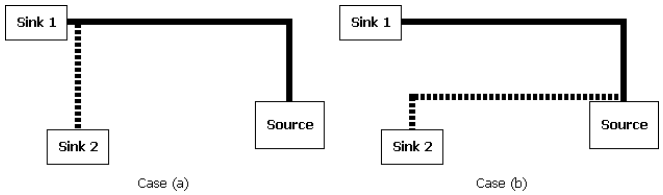

Fig. 4. Comparison between wirelength minimization and SSL(Source-to-Sink Length) minimization

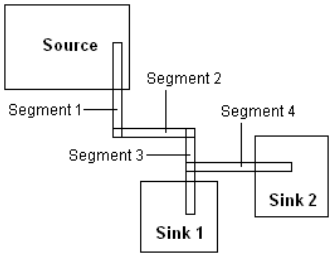

Fig. 5. Illustration of bus segments

A. Factors to Consider

One of the most important factors in microarchitectural bus routing is the length of the source-to-sink connection (= SSL). Existing works are mostly focused on the overall bus length, but we see that SSL has greater impact on IPC than the overall length. In Figure 4, the source-to-sink1 bus is already routed, and we want to connect sink2 to the bus. If we route as in case (a), the overall bus length is shorter than case (b). However, SSL of sink2 in case (a) is longer than case (b). This may increase the latency on source-to-sink2 bus and negatively affect IPC.

In Figure 4, case (b) is better than case (a) with respect to SSL. However, the number of clock cycles needed to transfer signals from the source to the sink2 may be same in both cases. For instance, assume that the SSL of sink2 is $80\mu m$ in case (a) and $50\mu m$ in case (b). If the clock period is equivalent to the delay of $100\mu m$-long bus, case (a) is better as it takes the same number of clock cycle but the total wirelength is shorter. However, if the clock period is equivalent to the delay of $60\mu m$-long bus, case (a) needs one more clock cycle than case (b) so the IPC goes down. Thus, this latency-aware timing slack can be traded in microarchitectural bus routing for other objectives.

The overall length of the bus is directly proportional to the power consumption. In addition, the switching activity of the source module is another important factor in terms of power consumption. Thus, power-aware bus routing needs to consider these factors. On the other hand, if the bus routing is done on top of hotspot, the temperature-dependent resistance will increase the delay of the bus and affect the clock period. Thus, bus routing needs to avoid hotspots as much as possible if it does not degrade the objectives.

B. Floorplanning Algorithm

One of the most important aspects of floorplan optimization using Simulated Annealing is on how to evaluate a candidate floorplan. Since the annealing process generates many candidate solutions, this evaluation process becomes not only runtime bottleneck but also crucial factor in determining the quality of the final solution. For a given candidate floorplanning solution, we quickly measure its routability, IPC, power, and thermal metrics for evaluation as follows:

- Routability: we perform bus routing to compute the routability accurately. Our bus routing considers source-to-sink length and thermal hotspot simultaneously while minimizing the bus area and latency.
- IPC: we need a timing-consuming architectural simulation such as SimpleScalar [8] if we desire an accurate IPC value. Instead, we use the weighted wirelength as suggested in [2], where the weights are based on the access frequency statistics collected during the simulation step.
- Thermal: an accurate computation of temperature requires a full-blown thermal analysis. Instead, we use power density of each module so that the modules with high power density values are separated apart.
- Power: we need a timing-consuming architectural simulation such as Wattch [9] if we desire accurate power values. Instead, we use another kind of weighted wirelength, where the weight this time is based on switching activity of the source module.

Our cost function used during floorplan is defined as follows:

$$F_{cost} = \alpha_1 \cdot A + \gamma \cdot R + \eta_1 \cdot F + \mu_1 \cdot P \quad (8)$$

where

$$F = \sum_{buses} (access\_freq \times latency) \quad (9)$$

$$P = \sum_{buses} (access\_freq \times bus\_area) \quad (10)$$

In the above equations, α_1, γ, η_1 and μ_1 are weighting constants. A is the final floorplan area, R is the number of unrouted buses for routability objective, F is the performance metric, and P is the power consumption of the buses. We also calculate the thermal overlap for each floorplan and discard it if the thermal overlap is greater than the constraint. Thermal overlap is computed as

$$TO = \sum_{buses} (access\_freq \times \sum_{modules} (P_{dense} \times A_{overlap})) \quad (11)$$

P_{dense} is the power density of a module, and $A_{overlap}$ is the amount of overlap between a pair of module and bus. TO corresponds to the constraint of bus area overlapping with hotspots.

C. Bus Routing Algorithm

Our bus routing algorithm is sequential, where we route each bus one-by-one. For a given bus to be routed, we select the nearest sink to the existing partial topology and connect it. In this case, we explore several different topologies and choose the one that optimizes the cost function that combines bus bend count, area, performance and overlap with thermal hotspots. Once all sink modules are connected to the bus, we route the next bus.

Our bus routing is based on the concept of *bus segments*, where each bus segment is either horizontal or vertical segment in a bus topology. An illustration is shown in Figure 5. Each segment has its own area, coordinates, length, and the direction relative to the source. Given a sink module to connect to the existing bus, there exist three possible topologies. Case (a) and case (b) use one segment to connect the sink module to the routed bus. Case (c) uses two segments (one for horizontal and another for vertical). For a given sink module to connect to the existing bus, we enumerate all possible connections from the module to *all* segments existing in the bus and choose the best one based on the following cost function:

TABLE I
MICROARCHITECTURE CONFIGURATIONS

module	size	module	size
ALU	4	IF	-
FPU	4	BPRED	2048
L1 I-$	32K	DEC	-
L1 D-$	32K	LSQ	64
ITLB	16	IRF	128
DTLB	32	RUU	128
FetchQ	16	ISSUE	4 (issue width)

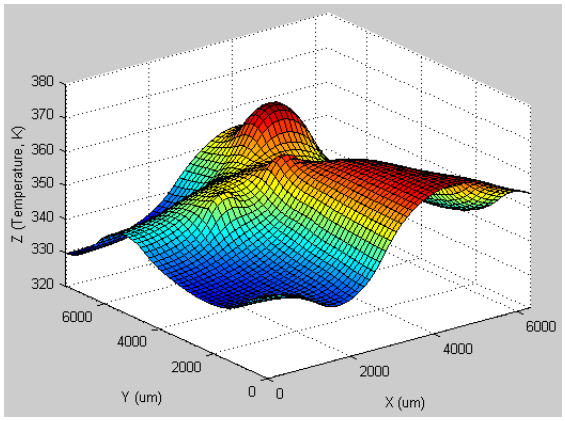

Fig. 6. Before including the bus effect on the temperature

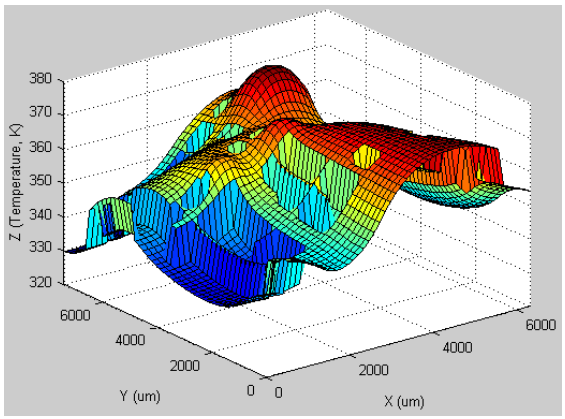

Fig. 7. After including the bus effect on the temperature

TABLE II
COMPARISON WITH [7]. R DENOTES ROUTABILITY IN PERCENTAGE, AND A DENOTES THE FLOORPLAN AREA IN mm^2

			[7]		Ours	
	Blocks	Buses	R	A	R	A
apte	9	36	80.9	55.46	100.00	48.77
xerox	10	98	79.8	22.2	98.78	21.54
hp	11	37	76	12.57	98.60	14.05
ami33	33	53	78.6	1.42	96.22	1.31
ami49	49	156	81.6	39.7	98.50	43.72

$$R_{cost} = \alpha_2 \cdot B + \mu_2 \cdot A_{bus} + \eta_2 \cdot L + \zeta_2 \cdot T \quad (12)$$

where

$$T = \sum_{modules} (P_{dense} \times A_{overlap}) \quad (13)$$

In the above function, α_2, μ_2, η_2 and ζ_2 are weighting constants. B is the total number of bends in the bus, A_{bus} is the area of the bus for power consumption minimization, L is the latency of the bus, and T is the temperature metric. P_{dense} is the power density of a module, and $A_{overlap}$ is the amount of overlap between a pair of module and bus. The intuition behind T is that we can avoid the hotspot by multiplying the power density of each module by the overlapped area.

Notice that the bus ordering affects the objectives since we route buses sequentially. Therefore we can use a specific ordering for each objective. For example, sorting by bus widths increases routability and decreases bus area, sorting by access frequencies increases performance, and sorting by access frequency times bus width decreases power consumption.

V. EXPERIMENTAL RESULTS

Our algorithms are implemented in C++ and experimented on Solaris 9 installed on SUN UltraSPARC-II 400MHz machine with 4GB main memory. We used MCNC benchmark to compare our algorithm with [7] and [6]. These are not architectural designs but circuit modules. Thus, we only report runtime, routability and floorplan area since other metrics require architectural simulations. The second benchmark is SPEC 2000 tested on our microarchitecture model shown in Section III. Table I shows our microarchitecture configurations used in SimpleScalar. In addition, we used $720nm$ pitch (wire width+spacing) global routing layers based on $65nm$ technology. The bus latency value is based on 3GHz clock frequency.

A. Bus Thermal Modeling

Figure 6 and 7 show the impact of bus on the temperature of the underlying floorplan (shown in Figure 8). One can see that the buses add non-negligible amount of heat onto the underlying floorplan (= about $10°C$ on average).

B. Comparison with Existing Works

We compare our result with [7] in Table II. Runtime is not shown because different machines were used in the experiments. The results show that we consistently outperform in terms of routability. The floorplan area is comparable. Table III shows the comparison with [6]. Benchmarks named with suffix N7 were also generated to evaluate routing algorithms for higher connectivity, and uarch is our own microarchitecture model. We use the same machine for this comparison with [6]. We observe that our algorithm outperforms [6] in terms of runtime with comparable routability and area results. We obtained better routability even if the average and the maximum numbers of blocks a bus connects are large. This is due to the flexibility of our algorithm since our buses consist of multiple segments.

C. Tradeoff Study

Table IV shows the IPC values based on various floorplanning/routing objectives. These values were obtained from SimpleScalar simulation, which we hacked to include bus-delay effects. We observe that performance-driven or power-driven algorithms obtain better IPC results than area-driven algorithm. The reason is that performance or power objectives tend to make the bus length shorter. Area objective has much lower IPCs, which shows that minimizing area only is not enough for IPC optimization. The reason that IPC of power-driven for swim and lucas is higher than that of performance-driven might be that bottlenecks of the benchmarks are different from others. In all of these floorplans, we obtain 100% bus routability.

TABLE III

COMPARISON WITH [6]. T DENOTES RUNTIME IN SECONDS, R DENOTES ROUTABILITY IN TERMS OF % (=# OF ROUTED BUSES × 100 / # OF BUSES), AND A DENOTES THE FLOORPLAN AREA IN mm^2

	[6]			Ours		
	T	A	R	T	A	R
apte	77	49.73	100	13	48.14	100
apteN7	92	82.90	60	141	48.06	100
xerox	90	20.42	100	15	21.30	100
xeroxN7	143	26.47	100	145	21.36	100
hp	213	9.38	100	38	11.83	100
hpN7	264	13.18	100	95	14.68	100
ami33-1	353	1.29	100	289	1.31	100
ami33-2	529	1.33	100	220	1.28	100
ami33N7	84	2.30	83.33	1552	1.38	100
ami49-1	560	38.87	100	412	40.08	100
ami49-2	1664	38.10	100	493	40.72	100
ami49-3	1564	41.12	100	614	41.29	100
ami49N7	548	37.84	40.00	2709	47.87	93.33
uarch	6942	45.66	86.27	1583	45.05	100

TABLE IV

IPC RESULTS BASED ON VARIOUS FLOORPLANNING/ROUTING OBJECTIVES. THE BOLDFACES SHOW THE BEST VALUE.

	Performance-driven	Area-driven	Power-driven
gzip	**1.1612**	0.4357	0.9087
swim	1.2879	0.4437	**1.6507**
vpr	**1.2292**	0.4291	0.9946
art	**1.1453**	0.4281	1.0740
mcf	**0.6001**	0.3339	0.5101
equake	**1.3683**	0.4422	1.3556
lucas	1.2081	0.4439	**1.4619**
bzip2	**1.3406**	0.4344	1.2688
twolf	**1.0568**	0.4240	0.8233

TABLE V

FLOORPLANNING & ROUTING RESULTS BASED ON VARIOUS OBJECTIVES (SCALED TO BOLDFACE)

	Performance-driven	Area-driven	Power-driven
Success rate	93.33%	80.00%	83.33%
Area	1.2202	**1.0000**	1.1342
Power	1.5952	1.5173	**1.0000**
Thermal overlap	1.3474	1.7747	**1.0000**

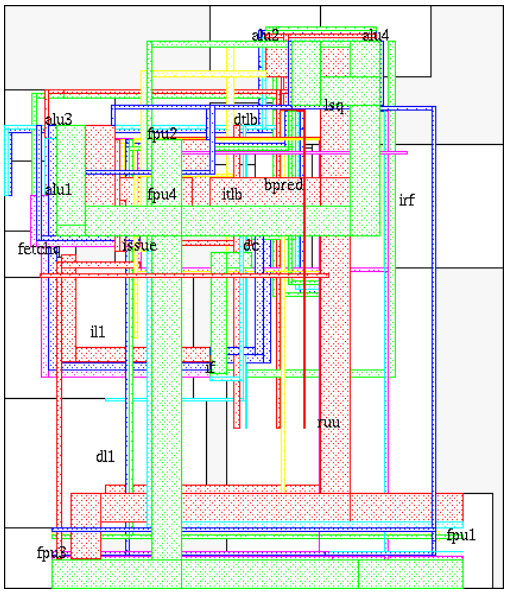

Fig. 8. Bus-aware floorplanning and routing example

Table V shows the tradeoff among performance, area, and power objectives under thermal overlap constraint (=2.0). Under the "success rate" metric, we report how many times we obtain 100% routability out of 20 runs of floorplanning. We first note that performance objective tends to increase the success rate than area/power objectives. This is because the area/power objectives tend to minimize area and wirelength, thereby further lowering the success rate. The area is the best in our area-driven algorithm and the worst in our performance-driven algorithm. This is expected since performance objective tends to make certain modules close to each other, thereby making it hard to compact overall floorplan. The power metric is the best in our power-driven algorithm, which is primarily due to the bus length/area decrease. The thermal overlap is the worst in area-driven algorithm since it blindly packs hot modules close together. Figure 8 show a snapshot of our floorplanning and bus routing results.

VI. CONCLUSION

In this paper, we presented the first bus-aware microarchitectural floorplanning. We developed a fast performance-aware bus routing algorithm, which is integrated into the floorplanning engine to ensure routability while optimizing other conflicting objectives. Our related experiments showed that we obtain 100% routability at the cost of minimal increase on area, performance, and power objectives under thermal constraint.

REFERENCES

[1] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, "Microarchitecture evaluation with physical planning," in *Proc. ACM Design Automation Conf.*, 2003.

[2] M. Ekpanyapong, J. Minz, T. Watewai, H.-H. Lee, and S. K. Lim, "Profile-guided microarchitectural floorplanning for deep submicron processor design," in *Proc. ACM Design Automation Conf.*, 2004.

[3] V. Nookala, Y. Chen, D. Lilja, and S. Sapatnekar, "Microarchitecture-Aware Floorplanning Using a Statistical Design of Experiments Approach," in *Proc. ACM Design Automation Conf.*, 2005.

[4] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H. S. Lee, and G. Loh, "Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2007.

[5] H. Xiang, X. Tang, and M. Wong, "Bus-Driven Floorplanning," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2004.

[6] J. Law and E. Young, "Multi-Bend Bus Driven Floorplanning," in *Proc. Int. Symp. on Physical Design*, 2005.

[7] F. Rafiq, M. Chrzanowska-Jeske, H. Yang, M. Jeske, and N. Sherwani, "Integrated Floorplanning With Buffer/Channel Insertion for Bus-Based Designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2003.

[8] T. M. Austin, "Simplescalar tool suite," http:/www.simplescalar.com.

[9] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. IEEE Int. Symp. on Computer Architecture*, 2000.

[10] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle packing based module placement," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1995, pp. 472–479.

[11] HotSpot, http://lava.cs.virginia.edu/HotSpot.

[12] H. Schafft, "Thermal Analysis of Electromigration Test Structures," *IEEE Trans. on Electron Devices*, 1987.

[13] M. P. K. Banerjee and A. Ajami, "Analysis and Optimization of Thermal Issues in High-Performance VLSI," in *Proc. Int. Symp. on Physical Design*, 2001.