Simultaneous Functional-Unit Binding and Floorplanning*

Yung-Ming Fang † and D. F. Wong ‡

Department of Electrical and Computer Engineering †
Department of Computer Science ‡
University of Texas at Austin, TX 78712

Abstract

As device feature size decreases, interconnection delay becomes the dominating factor of system performance. Thus it is important that accurate physical information is used during high level synthesis. In this paper, we consider the problem of simultaneously performing functional-unit binding and floorplanning. Experimental results indicate that our approach to combine binding and floorplanning is superior to the traditional approach of separating the two tasks.

1 Introduction

Existing CAD systems treat the tasks of high level synthesis (e.g. scheduling, allocation, and binding) and the tasks of physical design (e.g. floorplanning, placement, and routing) independently. As device feature size decreases, interconnection delay becomes the dominating factor of system performance. It is thus important that accurate physical information is used during high level synthesis (HLS) [7, 6, 2, 10, 8]. In this paper, we consider simultaneous functional-unit binding and floorplanning for performance optimization. Note that GB [5] and BITNET [9] also consider binding with physical information. However, GB applies only to one dimension bit-slice design and BITNET does not consider interconnection delay.

Fig. 1 illustrates the effect of different binding solutions on the performance of register-transfer level (RTL) design when interconnection delay is considered. We are given a scheduled data flow graphs (SDFG) in which there are three generic types of operators: `a`, `b`, and `c`, and their corresponding functional units are A, B, and C, respectively. We also assume that delay(A) > delay(B) > delay(C), and that interconnection delay is less than functional-unit delay, just to simplify the example. Fig. 1a shows a binding of the SDFG and a floorplan of the functional units. There are three type-C functional units (C1, C2, C3), two type-B functional units (B1 and B2), and one type-A functional unit (A). Nodes sharing the same functional unit are grouped together and labeled with the name of the functional unit. It is clear that the critical path is given by operations `b1`, `a1` and `c1` chained together in Step 1, with critical path delay = $5.5 + T_{FU}$,

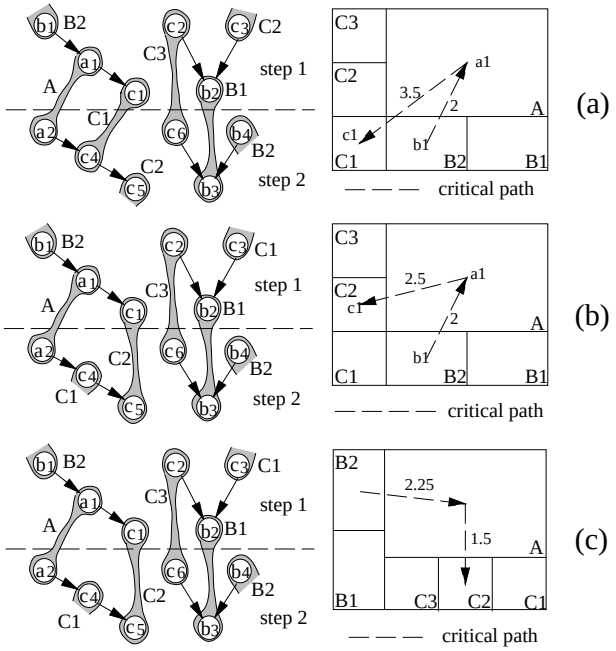

Figure 1: (a)Original binding (b)New binding (c)New binding and new floorplan

where T_{FU} = delay(A) + delay(B) + delay(C) and interconnection delay between two modules is given by the Manhattan distance between their centers in the floorplan. Fig. 1b shows a different binding solution (re-binded `c1` to functional unit C2) with the same floorplan of functional units. Again, `b1`, `a1` and `c1` in Step 1 is the critical path but the critical path delay is reduced to $4.5 + T_{FU}$ due to the shorter interconnection distance from A to C2. This clearly shows the importance of considering interconnection delay during the binding step. Finally, Fig. 1c shows further improvement by changing the floorplan. In this case, the critical path delay is reduced to $3.75 + T_{FU}$. This example shows the interrelation between binding and floorplanning. In the rest of this paper, we consider two problems on functional-unit binding using physical information, assuming a scheduled data flow graph is given. Problem 1 addresses the binding problem for cycle time optimization with respect to a given fixed floorplan. This problem will be referred to

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459.

as performance-driven binding and will be addressed in Section 2. Section 3 considers Problem 2 which is the problem of simultaneous binding and floorplanning. As we will see in the later sections, Problem 1 is the key to the solution of Problem 2.

2 Performance-Driven Binding

Assume that we are given a SDFG and a set of allocated modules (functional units and registers) with specification of area, delay, and geometric information. Geometric information specifies the minimum and maximum aspect ratio of a module, representing the flexibility of its implementation during the floorplanning stage. (Note that if the minimum and maximum aspect ratios are equal, the module has been completely designed.) Each functional unit is modeled as a 2-input 1-output combinational circuits and each register is modeled as a 1-input 1-output circuits. Interconnection topology is multiplexor-based point-to-point interconnection.

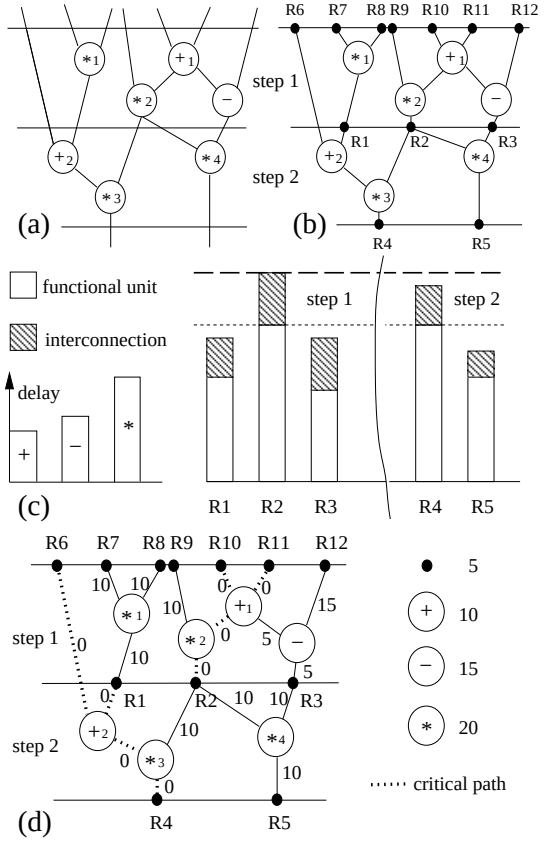

Figure 2: (a)SDFG. (b)Extended SDFG. (c)Path delay of extended SDFG. (d)Extended SDFG annotated by total slack.

To capture the physical meaning of data transfer, a SDFG is transformed into an *extended* SDFG such that each node of the extended SDFG is either a functional unit (*internal node*) or a register (*external node*). In a SDFG, we define *crossing edge* as an edge joining two nodes scheduled at two different control steps and *boundary edge* as an edge incident with either the beginning of the first control step or the end of the last control step. A SDFG can be transformed into an extended SDFG by inserting new nodes representing registers at the crossing edges and boundary edges. Crossing edges originating from the same node share the same register node. If an edge crosses more than one control steps, a new node is added only at the smallest control step. See Fig. 2a and Fig. 2b for an example. In the rest of this paper, SDFG will be referred to as extended SDFG unless mentioned otherwise.

The cycle time of a SDFG is the maximum path delay among all control steps where each path delay is given by the total functional and interconnection delay along the path. For example, the cycle time of the SDFG in Fig. 2b is shown in Fig. 2c with and without counting interconnection delay. The horizontal dotted line gives the cycle time if we only consider functional unit delay. The horizontal dashed line gives the cycle time if both interconnection and functional delay are counted. Note that the height of the shaded area between the two lines can be reduced by using shorter interconnections and hence improving cycle time. This shows that with the schedule fixed and modules allocated under a given technology, the performance can be further improved by shortening the interconnection in the critical paths.

2.1 Critical Path Analysis

Slack [4] calculation is used to analyze the critical paths of SDFG. It assigns each edge of SDFG a number called *total slack* (TS), and the smaller the TS is, the more critical the edge becomes. Especially, for the edges with zero TS, they are in the critical paths. Thus, the edges of SDFG can be partitioned into groups with different demand on interconnection resource. By binding the edges with smaller TS to closer modules first, the interconnection delay in the critical paths can be minimized. For a given SDFG, the delay of a node is determined by the module allocated for the node, and the delay of an edge is determined by the locations of module assigned to the nodes. With no binding information available, the edge delay is set to zero. Let us define ES_{ij}, EF_{ij}, LS_{ij}, LF_{ij} as *Estimated Start time*, *Estimated Finished time*, *Latest Start time*, and *Latest Finished time* of *signal propagation* p_{ij} from node i to node j. The following equations are used to compute the slack of internal nodes and the slack of external nodes. For internal node i with delay d_i, and its immediate predecessors and successors indexed by k and j,

$ES_{ij} = \max\{EF_{ki}\} + d_i$.
$EF_{ij} = ES_{ij} + p_{ij}$.
$LS_{ij} = LF_{ij} - p_{ij}$.
$LF_{ki} = \min\{LS_{ij}\} - d_i$.
$TS_{ij} = LF_{ij} - EF_{ij}$.

For external node i with delay d_i,

$ES_{ij} = d_i$ at *forward pass*.
$LF_{ki} = T_{cycle} - d_i$ at *backward pass*.

where T_{cycle} is the cycle time of SDFG, forward pass is the signal propagation from primary input to primary output, and the backward pass is the propa-

SG	TS	Edges
1	0	(R4,$*_3$) ($*_3$,$+_2$) ($+_2$,R6) ($+_2$,R1) (R2,$*_2$) ($*_2$,$+_1$) ($+_1$,R10) ($+_1$,R11)
2	5	(R3,$-$) ($-$,$+_1$)
3	10	(R1,$*_1$) ($*_1$,R7) ($*_1$,R8) ($*_2$,R9) ($*_3$,R2) (R5,$*_4$) ($*_4$,R2) ($*_4$,R3)
4	15	($-$,R12)

Figure 3: An SDFG partitioned by total slack.

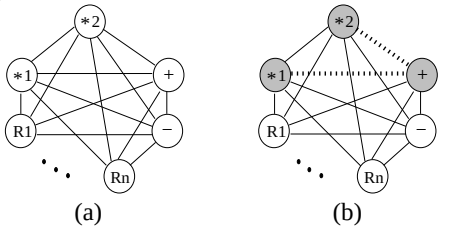

Figure 4: (a) G. (b) Q_x for edge x of type (*,+).

gation from primary output to primary input. The SDFG in Fig. 2b is used as an example to illustrate the slack computation. It is re-drawn in Fig. 2d with nodes annotated with delay and edges labeled with TS. The critical path is shown in dash line. Sorted by the TS assigned, the edges can be partitioned into four groups with TS values: 0, 5, 10 and 15 (see Fig. 3).

2.2 Functional Unit Binding

We first construct a weighted complete graph G to capture all the interconnection resources in the given floorplan. Each vertex in G corresponds to a module and the weight of an edge represents the interconnection delay between the two modules. For each edge x in the SDFG, we can identify the set Q_x of all edges y in G that are of the same "type" (in the sense that the two nodes of x in the SDFG and the two end vertices of y in G represent the same types of operations/registers). We pre-compute Q_x for all x and they will be used later during the binding stage. Note that if two edges x and z in the SDFG are of the same "type", Q_x and Q_z are the same and we only need to store one of them. Q_x represents the set of all possible interconnection resources for the edge x. During binding, the actual set of interconnection resources for an edge x is only a subset of Q_x (see Fig. 4)

Because it is possible to have two links created between the output of a source module and both the input ports of a destination module, the concept of port assignment is introduced to model the hardware cost at the destination module. In Fig. 5, a 4-cycle 2-input port assignment at a destination module is shown as a 4 by 2 table. Each entry of the table specifies a connection made by a source module at a specific cycle. By counting the number of different entries at each column as the number of communication links, and thus the required 2-to-1 MUXs, we can measure the hardware cost of this port assignment table. For example, we need four links and three 2-to-1 MUXs at the left input, and three links and two 2-to-1 MUXs at the right input. However, we can see entry a and entry c are duplicated at both columns. For a commutative functional unit, cost can be reduced by swapping if

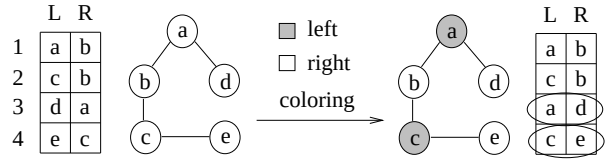

Figure 5: Port assignment by graph coloring.

they only appear once in both columns. A graph is constructed for such purpose: a node is created for each distinct entry in the table and an edge is created between the entries of the same row in the table. Then we color the graph with *left* and *right* to indicate the port assignment for each distinct entry. For nodes colored as *left* but located at the right port, they are interchanged with their left-port siblings. Using this scheme, we are able to reduce the cost of our example from seven links and five MUXs to five links and two MUXs by swapping entry a with entry d at the third cycle and entry c with entry e at the fourth cycle.

We can now describe our algorithm for functional-unit binding.

Step 1: Select an unbound edge with the smallest TS in the sorted list.

Step 2: Construct the interconnection resource of the edge. Weight each interconnection by its potential increment of links over the current RTL design. Select the shortest interconnection (a pair of modules) from the group with the lowest communication cost.

Step 3: Select a set of edges from the sorted list to be bound together with the selected edge and modules. All of them are scheduled in different control steps and are executable by the selected modules.

Step 4: Bind the selected edges to the selected source and destination functional units: 1) assign the node of the edges to its associated functional unit at its scheduled cycle; 2) assign the source functional unit to the left or right input port of the destination functional unit; 3) adjust the assignment of ports at the destination functional unit by graph coloring [3] to reduce the communication cost.

Step 5: Repeat the above steps until all edges are bound.

3 Binding and Floorplanning

Our approach to combine binding with floorplanning is to repeatedly use the performance-driven binding algorithm presented in Section 2 as part of the cost function evaluation procedure in a well known simulated annealing based floorplanner by Wong and Liu [11]. So that the solution space can be searched effectively , their algorithm embodies two key ideas: a special solution representation and a novel neighborhood structure defined by a set of three moves that bring a solution to its neighboring solutions. The algorithm in [11] only considers slicing floorplans. A slicing floorplan for n modules is a floorplan that can be obtained by recursively cutting a rectangle by either a vertical line or a horizontal line into n smaller rectangular regions. A slicing floorplan can be repre-

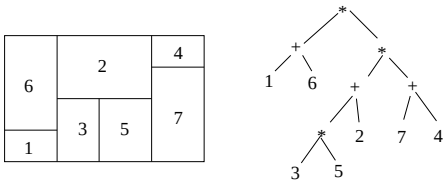

Figure 6: Slicing tree representation of a slicing floorplan.

sented by an oriented rooted binary tree, called slicing tree. Each internal node of the tree is labeled either * or +, corresponding to either a vertical cut or a horizontal cut. A postorder traversal of a slicing tree, gives a Polish expression which is a string representation of the slicing floorplan. Essentially, [11] uses Polish expressions to represent floorplans. Fig. 6 shows shows a slicing floorplan and its slicing tree. The Polish expression representation of the floorplan is 16+35*2+74+**. Floorplan transformation is achieved by using three types of moves (M1, M2, and M3) to transform a Polish expression into another one. See [11] for the definitions of the moves. Fig. 7 shows a series of floorplan transformations. Each floorplan considered by the algorithm in [11] during the simulated annealing process is evaluated based on area (A) and wire length (W) using a cost function of the form $\mathbf{A}+\lambda\mathbf{W}$. We modify the cost function evaluation procedure as follows: First we apply our performance-driven binding algorithm to obtain a binding which is suitable for the current floorplan. Then, in addition to computing the A and W, we also compute the critical path interconnection delay D and use a new cost function of the form $\mathbf{A}+\lambda\mathbf{W}+\gamma\mathbf{D}$.

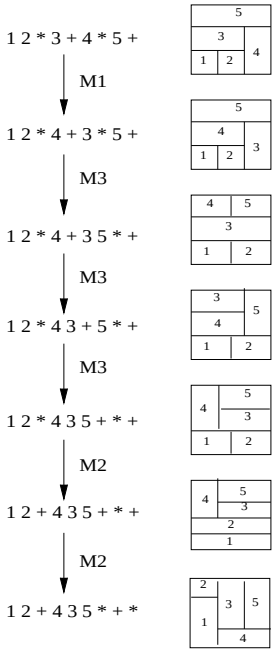

Figure 7: Floorplan Transformation.

4 Experiments and Results

Computation of interconnection delay depends on the electrical parameters derived from a given process technology. The following equations [1] are used to calculate electrical parameters and interconnection delay.

$$R_{int} = \rho\frac{l_{int}}{W_{int}H_{int}}$$

$$C_{int} = \epsilon_{ox}\left\{1.15\left(\frac{W_{int}}{t_{ox}}\right) + 2.8\left(\frac{H_{int}}{t_{ox}}\right)^{0.222}\right\}$$

$$T = R_{int}C_{int} + 2.3(R_{tr}C_{int} + R_{tr}C_L + R_{int}C_L)$$

where R_{int} and C_{int} are the total resistance and total capacitance of a wire segment; R_{tr} is the equivalent resistance of transistor at the driving end of the wire, and C_L is the load capacitance at the receiving end; ρ, resistivity of the wire material, ϵ_{ox}, dielectric constant of silicon dioxide, t_{ox}, thickness of oxide, l_{int}, wire length, W_{int}, wire width, and H_{int}, wire height, are all process-related parameters.

For simplicity, we have the following assumptions: First, Aluminum Al is used as the major inter-module wiring material, and the values of R_{tr} and C_L are both fixed at corresponding nominal values [12]. Second, for different technologies, the wiring geometry is fixed as the following: $H_{int} = 1/3W_{int}$ and $t_{ox} = 1/2W_{int}$. Under these assumptions, the corresponding parameters of process 1.6 μm, 1.2 μm, and 1.0 μm are computed accordingly. Functional unit libraries used in the experiments are listed in Table 1, where adder and multiplier are from 3D [10] and register is estimated relative to adder. Note that in Table 1 the unit of delay and area are shown in nano second and square micron, respectively. Two SDFGs are used in our

Library 1						
Fabrication	16-bit Adder		16-bit Multiplier		16-bit Register	
Technology	area	delay	area	delay	area	delay
$1.6\mu m$	746875	18	8711250	200	597500	3
$1.2\mu m$	420000	13	4900000	150	336000	3
Library 2						
Fabrication	8-bit Adder		8-bit Multiplier		8-bit Register	
Technology	area	delay	area	delay	area	delay
$1.2\mu m$	46875	13	564375	32	37500	2
$1.0\mu m$	21250	12	261875	30	17000	1.5

Table 1: Libraries.

experiments: FIR2 (Fig. 8a) is a 2-step 16-point FIR, and FIR3 (Fig. 8b) is a 3-step 16-point FIR. Information of the corresponding extended SDFGs are shown in Table 2.

SDFG	nodes	edges	cycles	modules	function units	slack group
FIR2	54	53	2	13	36	8
FIR3	59	58	3	9	32	8

Table 2: States of two benchmarks.

Three experiments are conducted. In Experiment 1, we test the effectiveness of our performance-driven binding algorithm. We first obtain a binding solution for the SDFG using a classical binding algorithm. We then generate a large number of floorplans and apply our performance-driven binder to each of them and compare the new binding solution with the original one (in terms of performance and cost). In Experiment 2, we apply our performance-driven binder to improve solutions obtained by the traditional design approach of separating HLS and physical design. First we use standard HLS algorithms and the floorplanner in [11]. Then we apply our performance-driven binder

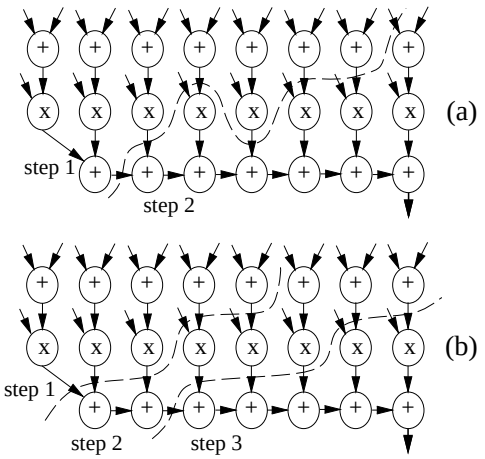

Figure 8: (a) FIR2: 2-step 16-point FIR filter (b) FIR3: 3-step 16-point FIR filter.

to obtain a binding solution which is more suitable to the final floorplan. Again, comparisons are made based on performance and cost. Finally, in Experiment 3, we test our approach of simultaneous binding and floorplanning. We apply the algorithm in Section 3 to the same set of benchmarks used in Experiment 2 and compare the results with those obtained in Experiment 2. The performance is measured as the

test	tech	link	mux	crit_path	wlength	topologies
FIR2	1.6μm	+4.17%	+5.85%	-40.64%	+6.75%	5070
LIB1	1.2μm	+4.04%	+5.47%	-44.39%	+3.44%	4940
FIR2	1.2μm	+2.62%	+1.19%	-40.68%	+4.47%	5200
LIB2	1.0μm	+2.46%	+0.72%	-46.22%	+5.34%	4940
FIR3	1.6μm	+1.89%	+1.51%	-23.81%	+5.39%	3420
LIB1	1.2μm	+1.77%	+2.55%	-30.10%	+2.26%	3420
FIR3	1.2μm	+0.92%	+4.42%	-15.36%	+7.21%	3330
LIB2	1.0μm	+1.18%	+4.87%	-16.57%	+6.94%	3330

Table 3: Experiment 1.

test	tech	link			mux			wlength(mm)			crit_path(ns)		
	μm	t	p1	d	t	p1	d	t	p1	%	t	p1	%
FIR2	1.6	45	47	2	15	16	1	185.6	204.5	+10.21	38.74	24.93	-35.65
LIB1	1.2	45	47	2	15	16	1	172.4	192.7	+11.78	31.21	24.22	-22.39
FIR2	1.2	45	47	2	15	16	1	48.4	58.3	+20.25	12.44	6.36	-48.88
LIB2	1.0	45	46	1	15	15	0	33.9	35.4	+4.57	11.17	5.76	-48.38
FIR3	1.6	45	45	0	22	21	-1	179.7	196.3	+9.25	26.78	21.4	-20.10
LIB1	1.2	45	44	-1	22	21	-1	142.5	158.2	+11.01	24.95	16.84	-32.51
FIR3	1.2	45	45	0	22	22	0	46.1	53.9	+16.94	10.20	6.84	-32.99
LIB2	1.0	45	46	1	22	23	1	33.0	34.3	+3.97	4.16	3.53	-15.27

Table 4: Experiment 2.

interconnection delay along the critical path, denoted as 'crit_path'. The cost has two major categories: the *structure cost* to measure the quality of RTL design and the *physical cost* to measure the quality of floorplan. For structure cost, the number of communication links and the number of 2-to-1 MUXs are used as the quality indication of RTL, denoted as 'link' and 'mux'. For physical cost, the area of floorplan and the total wire length of the final circuits are used as the quality indication of floorplan, denoted as 'area' and 'wlength'. 'test' and 'tech' indicate the experiment configuration. 'topologies' is the number of randomly generated floorplans. 't' , 'p1', and 'p2' represent the traditional approach, the approach to Problem 1, and the approach to Problem 2.

test	tech	link		mux		wlength(mm)			area(mm^2)			crit_path(ns)		
	μm	p1	p2	p1	p2	p1	p2	%	p1	p2	%	p1	p2	%
FIR2	1.6	47	47	16	16	204	214	+4.6	55.3	55.8	+0.9	24.9	22.4	-10
LIB1	1.2	47	46	16	15	192	145	-24.7	32.2	31.7	-1.6	24.2	19.5	-19.5
FIR2	1.2	47	47	16	16	58	58	-0.4	3.6	3.8	+7.4	6.4	6.3	-1.7
LIB2	1.0	46	46	15	15	35	29	-16	1.7	1.7	+4.2	5.8	4.0	-30.6
FIR3	1.6	45	44	21	20	196	224	+14.4	46.5	44.7	-3.7	21.4	15.9	-25.9
LIB1	1.2	44	46	21	24	158	154	-2.6	25.9	25.1	-2.8	16.8	12.0	-28.5
FIR3	1.2	45	44	22	21	54	52	-3.9	2.9	2.9	+0.1	6.8	5.5	-20
LIB2	1.0	46	44	23	21	34	30	-12.5	1.3	1.3	-0.7	3.5	3.4	-3

Table 5: Experiment 3.

Results of Experiment 1 (Table 3) show that on average the proposed approach improves the performance by up to 32% while only increasing the cost (structure and physical) by less than 6%. Results of Experiment 2 (Table 4) show that on average the proposed performance-driven binding improves the performance by up to 32% with only 10% increase of wiring. Results of Experiment 3 (Table 5) show that on average the simultaneous functional-unit binding and floorplanning further improves the performance by up to 17% with almost no increase of area and even 5% decrease of wiring.

References

[1] H. B. Bakoglu, "Circuits, Interconnections and Packaging for VLSI," pp. 194-225, 1990.

[2] F. Brewer and D. Gajski, "Chippe: A System for Constraint Driven Behavioral Synthesis," *IEEE Trans. on CAD,* Vol. 9, No. 7, pp. 681-695, 1990.

[3] Michael R. Garey and David S. Johnson, "Graph K-Colorability," Computers and Intractability, A Guide to the Theory of NP-Completeness, 1979.

[4] M. S. Hecht, "Flow Analysis of Computer Programs," North-Holland, 1977.

[5] Hyuk-Jae Jang and Barry M. Pangrle, "A Grid-Based Approach for Connectivity Binding with Geometric Costs," *Proc. of ICCAD,* pp. 94-99, 1993.

[6] David W. Knapp, "Fasolt: A Program for Feedback-Driven Data-Path Optimization," *IEEE Trans. on CAD,* Vol. 11, No. 6, pp. 677-695, 1992.

[7] Michael C. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions," *Proc. of 23rd DAC,* pp. 474-480, 1986.

[8] Vasily G. Moshnyaga, Hiroshi Mori, Hidetoshi Onodera, Keikichi Tamary, "Layout-Driven Module Selection for Register-Transfer Synthesis of Sub-micron ASIC's," *Proc. of ICCAD,* pp. 100-103, 1993.

[9] Ashutosh Mujumdar, Minjoong Rim, Rajiv Jain, and Renato De Leone "BITNET: An Algorithm for Solving The Binding Problem," *7th International Conference on VLSI Design,* pp. 163-168, 1994.

[10] Jen-Pin Weng and Alice C. Parker, "3D Scheduling: High-Level Synthesis with Floorplanning," *Proc. of 28th DAC,* pp. 668-673, 1991.

[11] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," *Proc. of 23rd DAC,* pp. 101-107, 1986.

[12] Dian Zhou, Franco P. Preparata and S. M. Kang, "Interconnection Delay in Very High-Speed VLSI," *IEEE Trans. on Circuits and Systems,* Vol 38, No. 7, pp. 779-790, 1991.