Primal-Dual Method based Simultaneous Functional Unit and Register Binding

Jianmo Ni*, Cong Hao†, Nan Wang‡, Takeshi Yoshimura†
*Dept. of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
†Graduate School of IPS, Waseda University, Kitakyusyu, 808-0135, Japan
‡School of Information Science and Engineering, East China University of Science and Technology, Shanghai, 200237, China
Email: nijianmo@gmail.com

***Abstract*—Interconnect reduction is one of the key issues in high-level synthesis. In this paper, we propose a primal-dual based method to solve the functional unit (FU) and register binding simultaneously while minimizing the global interconnection. Specifically, the binding problem is formulated as a min-cost network flow based on splitting weighted and order compatibility graphs (SWOCGs). The interconnect sharing among registers and FUs are maximized by binding the the operations or variables on the same path to the same FUs or registers according to the flow. Experimental results show that, compared with the previous greedy method [10], our proposed algorithm achieves an average 4.8% further reduction in global interconnection for a suite of benchmarks.**

I. Introduction

With the advances in nano-scale very large scale integration (VLSI) designs, there have been increasing concerns on area reduction and performance improvement. Among the many factors that influence the area and performance of VLSI designs, global interconnection is one of the dominant issues, especially for Field Programmable Gate Arrays (FPGA) [1]. Multiplexers, as essential data path elements, have significant impact on the design structure and interconnection density. Recent studies have shown that multiplexers account for 26% of the logic element utilization in FPGA designs [2].

High-level synthesis plays an important role in determining the main micro-architecture of designs [3], which directly affects the interconnection structure. There are three main tasks in high-level synthesis which are scheduling, resource allocation and resource binding. Resource binding is the stage to assign operations or variables to FUs or registers. In the binding procedure, multiplexers are introduced before registers when multiple FUs generate variable to be stored in the same register. Similarly, multiplexers are introduced before the input port of FUs when multiple registers feed data to this port. To this end, resource binding is closely related the multiplexer optimization and hence the interconnect reduction.

Many previous works have studied the FU and register binding for interconnect reduction in high-level synthesis. Huang et al. [4] presented a weighted bipartite-matching algorithm to solve the FU and register binding. Edge cost is used to represent the possibility of introducing multiplexers due to sharing of FUs and registers. Chen et al. [5] addressed the problem of register binding aiming at multiplexer optimization and proposed a bipartite-based algorithm. Since there exists a cyclic inter-dependency between FU and register binding, recent works have focused on the simultaneous FU and register binding schemes [8] [10] [12]. Cong et al. [8] formulated the FU and register binding as a network flow. Sequential FU and register binding is performed in an incremental manner until the result converges. However, the iteration might get into local-optimal. Kim et al. [10] developed a compatibility path based scheme to deal with the FU and register binding concurrently. By choosing compatibility path with maximum weight sum one-by-one, operations and variables on the same path are bound to the same FUs and registers so that the interconnect is decreased to a great extent. Dhawan et al. [12] modified the edge weight function in [10] by adding the consideration of inter-operation-type flow dependency. Nevertheless, both [10] and [12] proceeded the binding procedure in a greedy way and therefore lead to optimality loss.

In this paper, we focus on the interconnection optimization by minimizing the inputs of multiplexer. A primal-dual method is developed by formulating the simultaneous binding problem as a min-cost network flow. The min-cost flow is implemented on splitting weighted and ordered compatibility graphs (SWOCGs), which is extended from [10]. In SWOCG, each edge between operations is assigned with a cost which represents the potential resource sharing of two operations. A smaller cost means higher possibility to interconnect reduction. By calculating the flow, the proposed method guarantees to find out the compatibility paths with minimum sum of costs so as to decrease the number of multiplexer inputs.

The rest of the paper is organized as follows. Section II presents an example that motivates our approach and gives the problem definition. Section III explains the proposed method in detail. Section IV provides the experimental results and final conclusions are summarized in section V.

II. Motivating Example and problem definition

A. Motivating example

Figure 1(a) shows a scheduled DFG in which vertices and edges represent operations and data flow dependencies, respectively. The operations are scheduled into different time slots. A weighted and ordered compatibility graph (WOCG) is a directed acyclic graph denoted as $G(V,E)$, where V is the vertex set and E is the edge set. V consists of vertices representing operations in the DFG. A WOCG corresponds to

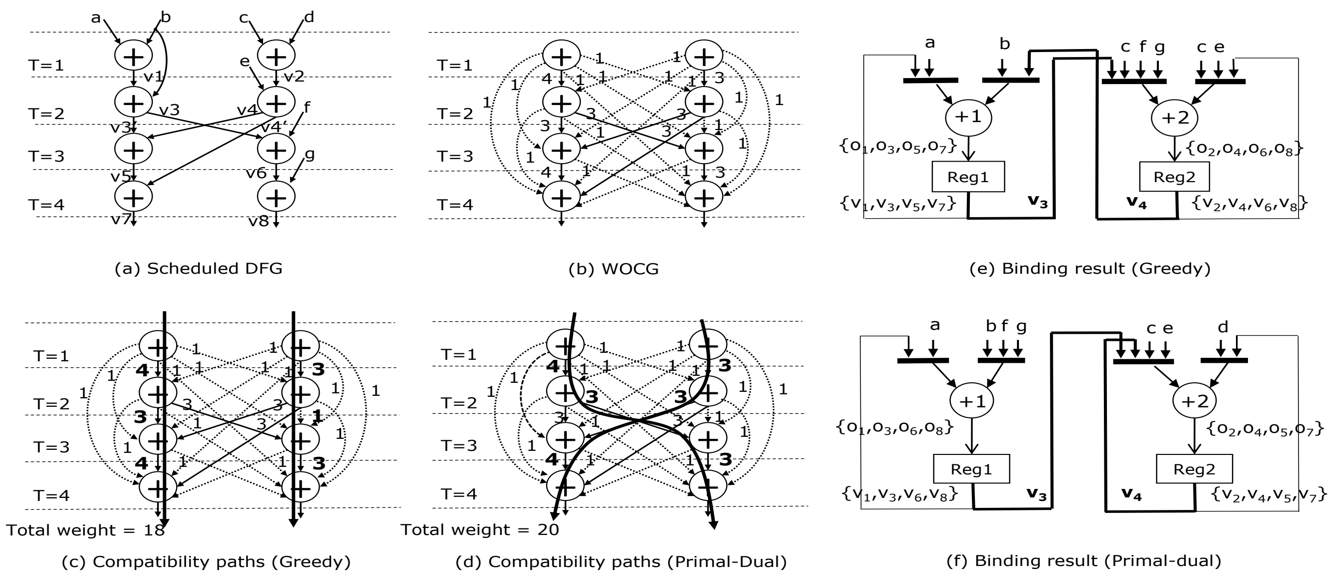


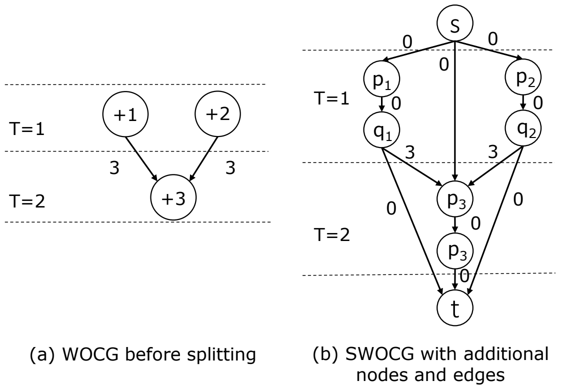
Fig. 1. Motivating example

a certain operation type, therefore all operations in V have the same type. Consider two operations u and v, if u is scheduled at a time slot earlier than v, then u and v are called compatible. A directed edge $u \rightarrow v$ exists if vertices u and v are compatible. Figure 1(b) is the WOCG established from figure 1(a). The calculation of edge weight is explained in detail in the next section.

A compatibility path is a set of compatible operations $v_1, , v_2, ..., v_i, ..., v_n$ in an increasing order of scheduled times. Figure 1(c) depicts the compatibility paths chosen by the greedy algorithm in [10]. According to the graph, the total sum of weights is 18. As figure 1(d) shows, there exists another path set that covers all vertices whose total sum of weights is 20. Therefore, the greedy manner cannot promise the determined path set to have the maximum sum of weights. In fact, the former paths may block the later possible favorable paths in the greedy algorithm resulting in the loss of optimality. Figure 1(e) and (f) are the netlists derived from the binding results based on different compatibility paths. For both two netlists, the numbers of assigned adders and registers are 2 and the number of multiplexer inputs is 11. However, variable v_4 is sent from register *reg*2 to adder +1 in (e) while from *reg*2 to +2 in (f). Due to the characteristic of compatibility path based method, register *reg*2 is closer to adder +2 than register *reg*1, which means the netlist in (f) has a shorter interconnects compared with (e). To overcome the problem in the greedy algorithm, a network flow based algorithm is proposed and is demonstrated in section III.

B. Problem Definition

The simultaneous FU and register binding can be formulated as follows: Given a schedule DFG and primary input variables of each operations, the goal is to find a FU and register binding that minimizes the total number of multiplexer input count. Besides, the numbers of FUs and registers are also minimized.



T=1
T=2
+1
+2
+3
3
3
s
t
p1
q1
p2
q2
p3
0
(a) WOCG before splitting
(b) SWOCG with additional nodes and edges

Fig. 2. An example of splitting WOCG to SWOCG

III. Primal-dual based simultaneous FU and register binding

In this section, we discuss our simultaneous FU and register binding scheme which is based min-cost network flow. To formulate the binding problem as a min-cost flow, a splitting weighted and ordered compatibility graph (SWOCG) is introduced based on the WOCG [10]. Figure 2 gives an example of how to transform the WOCG into SWOCG. As it shows, each operation node i in the WOCG (a) is split into a pair of new nodes (p_i, q_i) in SWOCG (b). Between two split nodes there is a directed edge $p_i \rightarrow q_i$, called splitting edge. To implement network flow on the SWOCG, source node s and sink node t are added. s is connected with p_i node of each operation i and t is linked with each q_i node.

By adding these nodes and edges, the SWOCG is denoted as a network $H = (V_H, E_H)$, where V_H and E_H are node set and edge set, respectively. Specifically, $V_H = V_{split} \bigcup\{s\} \bigcup\{t\}$, where V_{split} is the node set comprising all splitting node pairs

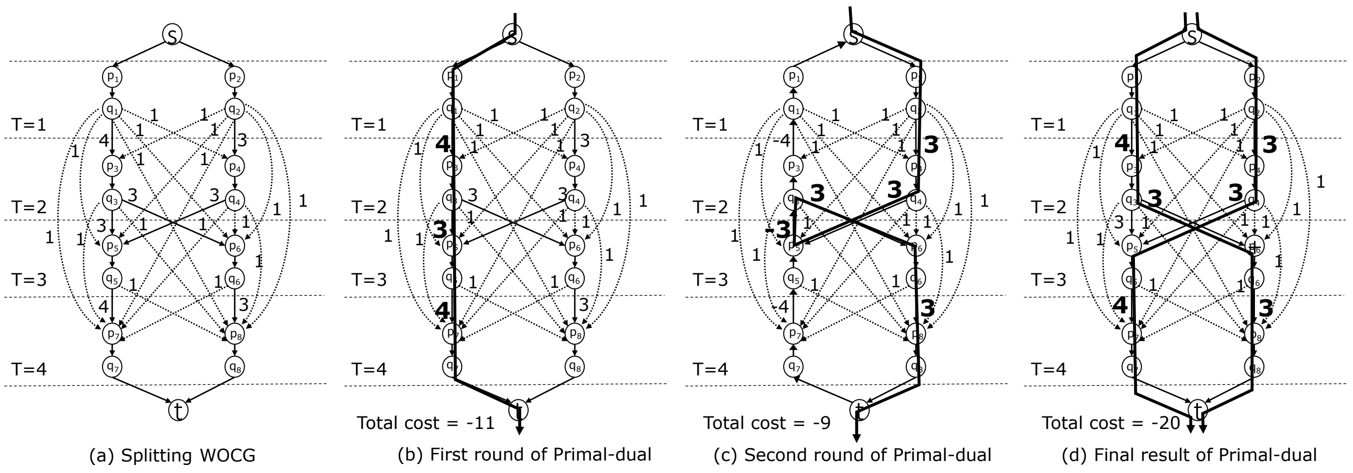

T=1
T=2
T=3
T=4
Total cost = -11
Total cost = -9
Total cost = -20
(a) Splitting WOCG
(b) First round of Primal-dual
(c) Second round of Primal-dual
(d) Final result of Primal-dual

Fig. 3. Splitting Weighted and ordered compatibility graph (SWOCG) and network flow on SWOCG

(p_i, q_i); $E_H = E_w \bigcup E_s \bigcup E_t \bigcup E_{split}$, where E_w is the set of edges representing the compatibility of two nodes as in the initial WOCG, E_s and E_t are edge sets of edges connected to s and t, and E_{split} is the set of splitting edges. It is to be noted that edge $i \rightarrow j$ in the initial WOCG corresponds to the edge $q_i \rightarrow p_j$ in SWOCG. The maximum capacity of each edge is 1. It is because each operation can only be assigned to a FU for one time. The costs of edges in E_s, E_t and E_{split} are all 0 while the edges in E_w are assigned with weights similar to [10] as follows:

$$w_{ij} = \alpha * F_{ij} + NIN_{ij} + R_{ij} + 1 \tag{1}$$

where i and j are indices of the nodes in WOCG, F_{ij} is a boolean variable that represents whether there is flow dependency between operations i and j in WOCG, NIN_{ij} indicates the number of common primary inputs of operations i and j, R_{ij} is the shared input register of operations i and j, α is a constant coefficient and 1 is a constant value. It is to be noted that, when there is no flow dependency or common primary inputs or shared input register, the edge weight is one which encourages more operations to be selected on the same path.

As mentioned in the previous section, this weight function reflects the potential interconnect reduction resulted from the resource sharing of operations. To achieve greater interconnection complexity decrease, paths with larger sum of weights are chosen out and operations on the same path are bound to the same register. In our problem, we consider the minus value of the weight as the edge cost. In this manner, the former simultaneous binding solution by finding out compatibility paths with maximum weight sum is transformed into determining the min-cost flow. The problem formulation is presented as:

$$min: \quad -\sum_{(i,j)\in E_H} w_{ij}x_{ij} \tag{2}$$

$$s.t: \quad \sum_{(i,j)\in E_H} x_{ij} - \sum_{(j,i)\in E_H} x_{ji} = 0, \forall i \in V_{split} \tag{3}$$

$$x_{ij} = 0 \ or \ 1 \tag{4}$$

where eq.(3) is the flow conservation euqation; x_{ij} is a binary variable which represents whether there is flow on the edge (i, j).

Algorithm 1 Primal-dual based simultaneous FU and register algorithm

```
Input: Scheduled DFG, primary inputs
Output: FU and register binding result, number of multiplexer inputs
 1: Construct SWOCG for different operation type, initialize edge capacity, edge cost
 2: while network flow have contained all nodes in SWOCG do
 3:     Calculate the shortest path from the source to the sink, by considering each edge cost to be length;
 4:     Assign the flow equal to the minimum capacity on the shortest path;
 5:     Change the edges on the shortest path into a reversed direction and the edge costs to their minus value;
 6:     Update the edge flow and edge cost;
 7: end while
 8: Merge the compatibility graphs
 9: Find side variables and input variables
10: Final register binding
11: return FU and register binding result, number of multiplexer inputs
```

A primal-dual based method is proposed to solved the the min-cost flow. Based on the network flow, our proposed simultaneous binding method includes 5 steps as algorithm 1 shows. Given a schedule DFG and primary inputs, SWOCG is initially established for each operation type (eg. addition and multiplication) and the min-cost flow method is implemented on each SWOCG separately. For each SWOCG, the compatibility path with minimum cost is first selected by using shortest-path algorithm, where edge cost is considered as distance; flow is then pushed on the chosen path and edges on the path is reversed and their costs are changed to opposite value. The algorithm continues selecting compatibility paths until flow has traversed all nodes in SWOCG. Operations and

TABLE I. Comparison with previous greedy algorithm [9]

DFG	|V|	|E|	Number of multiplexer inputs			Number of FUs						Number of registers		
			Greedy	Primal-dual	Reduction	Greedy		Primal-dual		Overhead		Greedy	Primal-dual	Overhead
						N+	N*	N+	N*	N+	N*			
ar	28	30	34	32	5.9%	2	5	2	5	0%	0%	15	15	0%
ellip	34	33	44	43	2.2%	2	4	2	4	0%	0%	20	21	5%
rand0	91	108	128	123	3.9%	8	6	8	6	0%	0%	46	45	-2.8%
rand1	157	189	226	207	8.4%	14	6	14	6	0%	0%	77	78	1.3%
rand2	129	240	936	901	3.8%	34	17	37	19	8.8%	11.7%	297	305	2.7%
AVG					4.8%					1.8%	2.3%			1.4%

variables on the same path to are bound to the same FU and register. An example of the proposed method is illustrated in figure 3. Figure 3(a) depicts the SWOCG transformed from WOCG in figure 1(a). As shown in figure 3(b), the path with a total cost of -11 is selected in the first round. Then the edges on this path is reversed and assigned with an opposite edge cost. In the second round, the path with a total cost of -9 is selected and the algorithm terminates since flow have passed on all nodes as 3(c). Figure 3(d) is the final result of the selected compatibility paths and it has the total weight sum of -20, which identifies with our motivation example.

After the simultaneous FU and register binding step is completed by solving the min-cost network, the following three stages are performed as in [10]: The compatibility paths in different SWOCGs are merged if two paths p_1 and p_2 have no time overlap, which means the last operation v_1 of the path p_1 is located at a time slot earlier than the first operation v_2 of the path p_2. Then side variables and primary variables are assigned to additional registers. Finally, the registers are merged by using left edge algorithm [11].

IV. Experimental Results

In this work, our proposed primal-dual based simultaneous FU and register binding algorithm and the previous greedy algorithm [10] have been implemented in C and run on a Linux Workstation with ARM Opteron 2.6GHz CPU and 4GB memory. A set of benchmarks from [14, 15] has been tested: ar, ellip, rand0, rand1 and rand2. The scheduled DFGs were generated by performing the force-directed scheduling algorithm [13].

Table I summaries the comparisons of our primal-dual algorithm with the greedy algorithm. Column 1 lists the DFG names and column 2-3 show the number of nodes and number of edges in the DFG, respectively. Column 4-5 give the numbers of multiplexer input counts and column 6 shows the corresponding reduction rates. According to the table, our proposed primal-dual method leads to a further 4.8% multiplexer input count reduction compared with the greedy algorithm. Column 7-10 show the numbers of FUs ($N+$ and $N*$ represent adder and multiplier, respectively) while column 11-12 give the the FU count overheads of our algorithm. As Table I shows, the average overhead of adder and multipler are 1.8% and 2.3%, respectively. Column 13-14 show the numbers of registers and the last column shows the increase of register count in percentage. On average, our algorithm leads to a 1.4% increase of register counts. Compared with the average multiplexer input count reduction of 4.8%, the FU and register overheads are acceptable.

V. Conclusion

In this paper, we have addressed a simultaneous FU and register binding problem for interconnect reduction. A primal-dual based method is proposed to determine the binding result by calculating min-cost flow on splitting weighted and ordered compatibility graphs. Compared with the previous work based on greedy algorithm, our method guarantees the chosen path set with maximized sum of weight and therefore leads to smaller interconnect consumption. Experimental results have shown a further 4.8% reduction in number of multiplexer inputs with only 2% FU and 1.4% register overhead, respectively.

References

[1] F. Li, D. Chen, L. He and J. Cong, *Architecture Evaluation for Power-efficient FPGAs*, ACM International Symposium on FPGA, Feb. 2003

[2] J. Stephenson and P. Metzgen, *Logic Optimization Techniques for Multiplexers*, Altera Literature, 2004

[3] M. McFarland, A. Parker, and R. Camposano, *The High-Level Synthesis of Digital Systems*, Proccedings of the IEEE, vol. 78, no. 2, pp. 301-318, Feb 1990

[4] C.-Y. Huang, Y.-S. Chen, Y.-L. Lin, and Y.-C. Hsu, *Data path allocation based on bipartite weighted matching*, in Proc. Design Autom. Conf., pp. 499ÍC504, Jun. 1990

[5] D. Chen and J. Cong, *Register Binding and Port Assignment for Multiplexer Optimization*, in Proc. ASPDAC, Jan. 2004

[6] D. Chen, J. Cong and Y. Fan, *Lower-power High-Level Synthesis for FPGA Architectures*, in Proc. ISLPED, Aug. 2003

[7] J. Cong, Y. Fan and W. Jiang, *Platform-based Resource Binding using a Distributed Register-file Microarchitecture*, in Proc. Int. Conf. Comput.-Aided Design, pp. 709-715, Nov. 2006

[8] J. Cong and J. Xu, *Simultaneous FU and register binding based on network flow method*, in Proc. Design Autom. Test Eur., pp. 1057ÍC1062, Mar. 2008

[9] T. Kim and C. L. Liu, *An Intergrated Data Path Synthesis Algorithm based on Network Flow Method*, in Proc. IEEE Custom Integr. Circuits Conf., pp.615-618, May 1995

[10] T. Kim and X. Liu, *A Funtional Unit and Register Binding Algorithm for Interconnect Reduction*, IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems, vol. 29, no. 4, pp. 641-646, Apr. 2010

[11] F. J. Kurdahi and A. C. Parker, *REAL: A program for register allocation*, in Proc. Design. Autom. Conf., pp. 210ÍC215, Jun. 1987.

[12] U. Dhawan, S. Sinha, S. Lam and T. Srikanthan. *Extended compatibility path based hardware binding algorithm for area-time efficient designs*, in Proc. 2nd Asia Symp. of Quality Electronics Design, Penang, pp. 151-156, August 2010

[13] P. G. Paulin, J. P. Knight, and E. F. Girczyc, *HAL: A Multi-Paradigm Approach to Automatic Data Path Synthesis*, Design Automation Conference, pp. 263-270, Jul. 1986

[14] David Rhodes, Robert Dick, http://ziyang.eecs.umich.edu/dickrp/tgff/

[15] C.Lee, M. Potkonjak, and W. Mandione-Smith, *Mediabench: a tool for evaluating and synthesizing multimedia and communications systems in Microarchitecture*, Thirtieth Annual IEEE/ACM International Symposium, pp. 330-335, 1997