Introducing Real Constraints in Partitioned ILP-Based Binding in High-Level Synthesis

Nagisa Ishiura and Yuuki Oosako

School of Science and Technology, Kwansei Gakuin University, Sanda, Hyogo, Japan

Abstract— This paper presents an efficient ILP based method of binding in high-level synthesis. The binding problem can be broken into subproblems based on partitioning of the set of control steps. To incorporate a global view in solving each subproblem by ILP, constraints for other unsolved subproblems without integer restrictions are added. Experiments on some designs shows that this produces better solutions within less computation time.

I. Introduction

High-level synthesis [1] now plays an indispensable role in VLSI design. Among the primary tasks in high-level synthesis, binding, which assigns operations and values in a CDFG (control dataflow graph) to functional units and registers in a datapath, is the most CPU intensive and has the largest impact on the quality of the synthesized circuits in terms of the size and the critical path delay.

While graph or flow based polynomial time algorithms [2, 3] are typically employed to find reasonable binding solutions within practical time, ILP (integer linear programming) is also used to get high-quality solutions for small to medium designs [4, 5]. Although ILP formulation is also convenient in incorporating various new factors into binding, computation time for large models often gets prohibitive. To curve the CPU time, the models may be partitioned into smaller models and solved piece by piece, but this drops the quality of solutions, for optimization is attempted based only on local information.

To address this issue, this paper proposes to incorporate constraints of other subproblems without integer restriction into those of a subproblem in focus. Experiments on some designs yielded better solutions within less computation time than the simple partitioning method.

II. ILP formulation of Binding

Given a scheduled CDFG, we try to find an assignment of operations and values to functional units and registers which minimizes the total cost of functional units and multiplexers. As formulated in [6], we force all the paths including false paths (produced by chaining) not to exceed the clock period. In case we encounter a contradiction, we resolve it by allocating extra functional units (as in [5]) instead of altering the scheduling (as in [6]).

Let F and R be the sets of the available functional units and the registers in the datapath, respectively. Let $U = F \cup R$. We call $u \in U$ a unit. Let W_u and T_u be the sets of the output terminals and the input terminals of unit u. Let W and T be the sets of all the output terminals and all the input terminals, respectively. For $t \in W \cup T$, let u_t denote the unit that t belongs to. Let $d_u \in \mathcal{Z}^+$ and $c_u \in \mathcal{Z}^+$ be the delay and the cost of unit u, respectively. Let p_{max} be the maximum permissible critical path delay of the datapath. Let the cost of the k-input multiplexer be $(k-1) \cdot c_{mux}$. Let O and V be the sets of the operations and the values of a given scheduled DFG, respectively, and let $N = O \cup V$. We call $n \in N$ a node. Let P_n and Q_n be the sets of the output ports and the input ports of node n. Let P and Q be the sets of all the output ports and all the input ports, respectively. For $p \in P \cup Q$, let n_p denote the node that p belongs to. $E \subseteq P \times Q$ represents the data dependency of the DFG. Let S be the set of the steps. For $n \in N$, U_n be the set of the units which can execute n, and s_n be the step where n is executed. Let $t_{p,u}$ be the terminal corresponding to port p when the node that p belongs to is bound to unit u. Let $N_{s,u} = \{n \in N \mid s_n = s, u \in U_n\}$.

The base variables for our ILP formulation is:

- $x_{n,u}$ ($n \in N$, $u \in U$): 0-1 variable where $x_{n,u} = 1$ iff node n is bound to unit u.

In addition, the following auxiliary variables are used:

- $c_{w,t}$ ($w \in W$, $t \in T$): 0-1 variable where $c_{w,t} = 1$ iff there is a connection from w to t.
- m_t ($t \in T$): integer variable representing the multiplexer cost on t.
- u_u ($u \in U$): 0-1 variable where $u_u = 1$ iff unit u is used.
- p_u ($u \in U$): integer variable representing the maximum path delay from registers to the outputs of u.

We have the following six constraints.

1. Each node is bound to a unit:
$$\forall n \in N : \sum_{u \in U_n} x_{n,u} = 1$$
2. Each unit is used at most once during a step:
$$\forall s \in S \, \forall u \in U : \sum_{u \in N_{s,u}} x_{n,u} \le 1$$
3. For each data dependency, there is a corresponding connection:
$$\forall (p,q) \in E \, \forall u \in U_{n_p} \, \forall v \in U_{n_q} : \\ x_{n_p,u} + x_{n_q,v} - 1 \le c_{t_{u,p},t_{q,v}}.$$
4. Multiplexer cost:
$$\forall t \in T : m_t = \sum_{w \in W} c_{w,t} - 1.$$
5. Path delay:
   (a) $\forall u \in R : p_u = 0.$
   (b) $\forall (w,t) \in (W \times T) : p_{u_w} + (d_{u_t} - D) \le p_{u_t}$, where D is a constant greater than p_{max}.
   (c) $\forall u \in F : p_u \le p_{max}.$

The objective in our formulation is to minimize the sum of the costs of the used units and the multiplexers.

$$\text{Minimize} \quad \sum_{u \in U} c_u u_u + \sum_{t \in T} c_{mux} m_t .$$

III. Partitioned and Partially Real Constrained Approach

When the ILP model for binding is too large to solve, it may be partitioned. One simple way is to determine the binding for k steps at a time. To lessen the degradation of solution quality, this paper proposes to incorporate constraints regarding other steps without integer restriction, into those of the steps that are being solved.

At each iteration, we partition the set of steps S into four disjoint sets S_S, S_I, S_R, and S_N. S_S is the set of the already solved steps and S_I is the set of the steps to be solved at the current iteration. S_R is the set of the steps in which constraints are relaxed. All the constraints regarding S_N are ignored in the current iteration. In the initial iteration, S_S is empty and binding for S_I is searched using the integer constraints of S_I and the real constraints of S_R. Then, in the next iteration, S_S is set to $S_S \cup S_I$ and new S_I and S_R is chosen. It is preferable to set $S_R = S - (S_S \cup S_I)$, but S_R are chosen so that the subproblem is solved within feasible time.

In this formulation, all the auxiliary variables $c_{w,t}$, m_t, u_u, p_u are now real variables. $x_{n,u}$ are handled as follows:

- $x_{n,u}$ where $n \in S_S$ is replaced by its solution.
- $x_{n,u}$ where $n \in S_I$ is unchanged.
- $x_{n,u}$ where $n \in S_R$ is a real variable ($0 \leq x_{n,u} \leq 1$).
- All the constraints 1–3 regarding $x_{n,u}$ where $n \in S_S$ are removed.

IV. Experimental results

A binding program based on the proposed method has been implemented. CPLEX 12.6.1.0 is used as an ILP solver. The main program itself is written in Perl5, and ILP models and results are passed via intermediate files. Steps for S_I and S_R are simply chosen in the order of appearance in given CDFG files.

TABLE I summarizes the result of the experiment on three models. "ellip" has 75 nodes which are scheduled into 17 steps where chaining up to three operations has been performed. In the first run, $|S_I|$ and $|S_R|$ are set to 1 and 0, respectively, which is equivalent to the bipartite matching based method [2]. The cost is in terms of the total cost function in ILP where the costs of an ALU, a multiplier, an MUX, and a register are 32, 128, 32, and 32, respectively. The CPU time was on 1.7GHz Core i7 with 8GB memory for "ellip" and "s2m", and 3.4GHz Core i7 with 16GB memory for "RSA." By solving 5 steps at a time, the cost is reduced significantly. However, our method with $|S_I| = 1$ and S_R being the set of all the unsolved steps found better solution within less computation time. Similar result is obtained on the second model "s2m" with almost the same size. The third model is an RSA codec whose CDFG is generated from a C program by a high-level synthesizer. Chaining is not performed. Since the number of the nodes in a step varies largely, we partitioned the step set by the number of nodes. $s(n)$ is the minimum number of the next steps that include n nodes; when the next 3 and 4 steps have 25 and 35 nodes, respectively, then $s(30) = 3$. By solving multiple steps at a time, costs become significantly less than the bipartite matching method[1], but the proposed method found a better solution with the same amount of CPU time.

Note that the result is highly dependent on the model, due to the complex nature of the biding problem. There were also cases where the normal partitioning method with $S_R = \phi$ found better solution than our method.

TABLE I
Experiments on small scale examples.

	$\lvert N\rvert$	$\lvert S\rvert$	$\lvert S_I\rvert$	$\lvert S_R\rvert$	cost	CPU [s]
ellip	75	17	1	0	1,456	1.07 *
			5	0	1,296	357.99 *
			1	**all**	**1,264**	**20.68** *
s2m	81	14	1	0	1,264	0.59 *
			6	0	1,168	1.89 *
			7	0	1,136	5.98 *
			1	**all**	**1,136**	**3.27** *
RSA	5,417	854	1	0	12,032	2950.61 **
			s(30)	0	9,728	206.05 **
			s(30)	**s(870)**	**8,992**	**8271.24** **
			s(350)	0	9,408	8913.13 **
			s(400)	0	9,248	7555.20 **

N: set of nodes, S: set of steps
S_I: set steps solved by ILP, S_R: set steps solved by LP
cost: ALU=32, multiplier=128, MUX=32(#in−1), register=32
all: all the unsolved steps (namely, $S_R = S \cup \overline{S_S \cup S_I}$ and $S_N = \phi$)
$s(n)$: the minimum number of the next steps that include n nodes
* 1.7GHz Core i7 (8GB memory), ** 3.4GHz Core i7 (16GB memory)

V. Conclusion

This paper has presented a method of incorporating a global view in partitioned ILP based binding. We are now working on further experiments to refine the method.

Acknowledgement

Authors would like to thank all the members of Ishiura Lab. of Kwansei Gakuin University. This work was partly supported by JSPS KAKENHI Grant Number 16K00088 .

References

[1] Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, and Steve Y-L Lin: *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic (1992).

[2] C.-Y. Huang, Y.-S. Chen, Y.-L. Lin, and Y.-C. Hsu: "Data path allocation based on bipartite weighted matching," in *Proc. 27th DAC*, pp. 499-504 (June 1990).

[3] J. Cong and J. Xu: "Simultaneous FU and register binding based on network flow method," in *Proc. DATE 2008*, pp. 1057–1062 (Mar. 2008)

[4] W.-T. Shiue and C. Chakrabarti: "ILP-based scheme for low power scheduling and resource binding," in *Proc. ISCAS 2000*, vol. 3, pp. 279–282 (May 2000).

[5] Y. Hara-Azumi and H. Tomiyama: "Clock-constrained simultaneous allocation and binding for multiplexer optimization in high-level synthesis," in *Proc. ASP-DAC 2012*, pp. 251–256 (Jan.–Feb. 2012).

[6] A. Kondratyev, L. Lavagno, M. Meyer, Y. Watanabe: "Share with care: A quantitative evaluation of sharing approaches in high-level synthesis," in *Proc. DATE 2013*, pp. 1547–1552 (Mar. 2013).

[1]The CPU time for $|S_I| = 1$ is unexpectedly large because of many file I/Os for repetitive invocations the ILP solver. Proper implementation will reduce it to far smaller time.