On the Complexity of the Port Assignment Problem for Binary Commutative Operators in High-Level Synthesis

Philip Brisk Paolo Ienne
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, CH-1015, Switzerland,

ABSTRACT

The present formulation of the port assignment problem for binary commutative operators tries to minimize the number of wires connected to both the left and right inputs of the operator; intuitively, this minimizes the total number of inputs connected to both inputs, which reduces the size of the multiplexers that are instantiated. This paper revises the formulation to attempt to balance the difference between the number of wires connected to both inputs; minimizing the size of the larger multiplexer place on the input minimizes the delay through the operator.

I. INTRODUCTION

The *Port Assignment Problem (PAP)* for *Binary Commutative Operators (PAP-BCO)* is a problem in high-level synthesis associated with *connectivity binding*. The goal of the PAP-BCO is to minimize the number of wires that are connected to both input ports of the operator. If $k > 1$ wires connect to the input port, then a *k:1 multiplexor (mux)* is instantiated on the input. Thus, the PAP-BCO attempts to minimize the aggregate area of the two muxes.

Pangrle [1] proved the PAP-BCO to be NP-complete for *each* binary commutative operator. The proof used a reduction from the *Dual-Clique Problem (DCP)*: an undirected graph represents all of the wires that must be connected to at least one of the inputs of the commutative operator. The two cliques correspond to wires that must be connected to *only* the left and right input ports of the operator respectively; all remaining wires are connected to both input ports. The DCP formulation of the PAP-BCO tries to minimize the number of wires that connect to *both* muxes, i.e., to minimize their aggregate area. Chen and Cong [2] used this approach for register binding and port assignment in high-level synthesis in 2005. Several papers have also used this approach to mux insertion as a post-processing pass following datapath merging [3-5].

This paper advocates a new strategy that tries to minimize the *difference* in the number of inputs between the two muxes as a secondary optimization criterion, i.e., to reduce the critical path delay. If the operator is non-critical, the extra flexibility by reducing its delay increases the freedom of lower-level optimizations such as slack budgeting [6].

Section II outlines the underlying assumptions that form the foundation of this work. Section III introduces theoretical preliminaries. Section IV formalizes the PAP-BCO, revisits its complexity, and provides two detailed examples showing that the DCP formulation does not always find the best solution. Section V summarizes prior work in high-level synthesis that relates to this work. Section VI concludes the paper.

II. ASSUMPTIONS

This section lists several assumptions used by the theoretical analysis in the remainder of the paper. Practical implementations should tailor these assumptions to the appropriate context (e.g., ASIC vs. FPGA).

Assumption 1.) We use a point-to-point interconnection architecture, similar to theoretical work by Pangrle [1] and a recent high-level synthesis flow published by Chen and Cong [2].

Assumption 2.) We assume the existence of a library of *k:1* muxes for any integer $k > 1$. Mitra and McCluskey [7] described a method to construct a *k:1* mux from smaller ones.

Assumption 3.) We assume that the critical path delay of a commutative operator is the same for both input ports and that the critical path delay of a mux is uniform for all inputs; this simplifies delay modeling. This may not always be the case, as exemplified by a *3:1* mux in Fig. 1(a). It may also be possible to reorganize a mux to reduce the delay of a critical wire, as in Fig. 1(b). In general, even if the delay of a particular wire is non-critical at the high-level synthesis stage, reducing its delay is still beneficial as later stages of synthesis can exploit the slack [6], for example, to reduce area.

III. PRELIMINARIES

The Induced Bipartite Subgraph Problem (IBSP)

Throughout the remainder of the paper, we use the *Induced Bipartite Subgraph Problem (IBSP)* in lieu of the DCP. The IBSP and DCP are *complementary*, i.e.: to reduce one to the other, it suffices to complement the graph.

Let $G = (V, E)$ be an undirected graph. G is *bipartite* if there is a partition of V into two sets, V_1 and V_2, such that for every edge $e = (v_1, v_2) \in E$, $v_1 \in V_1$ and $v_2 \in V_2$, or vice-versa; in other words, V_1 and V_2 are *independent sets*.

Let $V' \subseteq V$; the subgraph of G induced by V' is the graph $G' = (V', E')$, where $E' = \{(v_1, v_2) \in E \mid v_1, v_2 \in V'\}$.

Decision Problem 1 (IBSP). Given an undirected graph G = (V, E) and integer $t \geq 0$, determine if V has a subset V' such that $|V'| \geq t$ and V' induces a bipartite subgraph of G?

The IBSP is an NP-complete *decision problem*. A high-level synthesis tool will solve its analogue as an *optimization* problem: the *Maximum IBSP (MIBSP)*, i.e.: maximize $|V'|$.

The Number Partitioning Problem (NPP)

Let X, Y be sets of integers, $\Sigma(X)$, $\Sigma(Y)$ be the sums of the integers in X and Y respectively, and $\Delta(X, Y) = |\Sigma(X) - \Sigma(Y)|$.

Decision Problem 2 (NPP). Given a set T of positive integers and integer $t \geq 0$, is it possible to partition T into sets X and Y such that $\Delta(X, Y) \leq t$?

The *optimization* analogue of the NPP is the *Minimum NPP (MNPP)*, i.e.: minimize $\Delta(X, Y)$. The MNPP can be solved optimally in pseudo-polynomial-time using dynamic programming [8]. The worst-case time complexity of the algorithm is $O(|T|\Sigma(T))$. In general, $\Sigma(T)$ is unbounded because T may contain a polynomial number of large values. Thus, the pseudo-polynomial-time algorithm does *not* prove that $P = NP$.

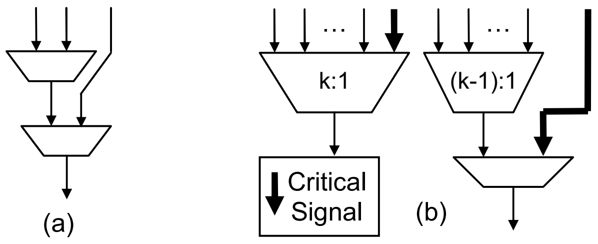

FIGURE 1. (a) Example of a mux with non-uniform delays; (b) reorganization of a mux to reduce the delay of a critical signal.

IV. THE PORT ASSIGNMENT PROBLEM

The *Port Assignment Problem (PAP)* occurs in high-level synthesis [1, 2] or datapath merging [3-5]. At this point, resource allocation, scheduling, and operation binding have completed; slack allocation [6], resource selection (e.g., the choice of a ripple-carry or carry-select adder) and the controller synthesis are performed after port assignment.

The input to the PAP is a schedule S of time-steps; the order of steps i is irrelevant. Let OPS be the set of allocated resources. Port assignment is performed for each operation $op \in OPS$ in isolation. Let $E \subseteq S$ be the subset of steps where op is used, and $e_i \in E$ be a specific time-step where op is used. Let $V = \{v_1, v_2, ..., v_k\}$ be the set of *sources* of op, i.e., the set of resources that provide operands to op over all time steps in E.

First, suppose that op is unary; in this case, only one source will provide data to op each time that op is activated. The solution is trivial: a $k{:}1$ mux is placed on the input port of op. Controller synthesis will ensure that the correct source is selected by the mux at each time step. This is illustrated in Fig. 2(a).

Now, suppose that op is non-commutative. This case is also trivial, as each input port can be treated as a distinct unary operator. This is illustrated in Fig. 2(b).

If op is commutative, however, the problem becomes much more complicated; for the remainder of this paper, we restrict our discussion to the case where op is binary.

Each operation $e_i \in E$ that uses op has the form ... $\leftarrow v_1\ op\ v_2$. Clearly, v_1 and v_2 must connect to opposing input ports in order to perform the computation. Without loss of generality, if v_1 connects to both input ports, then v_2 can connect to either; however, if v_1 connects only to the left input port, then v_2 must have a connection to the right input port. Three possibilities exist for each source $v_i \in V$: (1) v_i connects only to the left input port of op; (2) v_i connects only to the right input port; and (3) v_i connects to both input ports.

Let V_L be the set of sources that connect only the to left input port, V_R be the set of sources that connect only to the right input port, and V_{LR} be the set of sources connected to both input ports. As all sources must connect to an input port, $V = (V_L, V_R, V_{LR})$ is a tri-partition of V.

Decision Problem 3 (PAP-BCO). Given a set V of sources, a set E of time steps, and a positive integer k, is there a tri-partition $V = (V_L, V_R, V_{LR})$ of V such that $|V_L| + |V_R| \geq k$ and for each pair of sources v_1 and v_2 used at each time step $t_i \in E$, one of the three following conditions holds:

(1) $v_1 \in V_L$ and $v_2 \in V_R$, or
(2) $v_1 \in V_R$ and $v_2 \in V_L$, or
(3) $v_1 \in V_{LR}$ or $v_2 \in V_{LR}$?

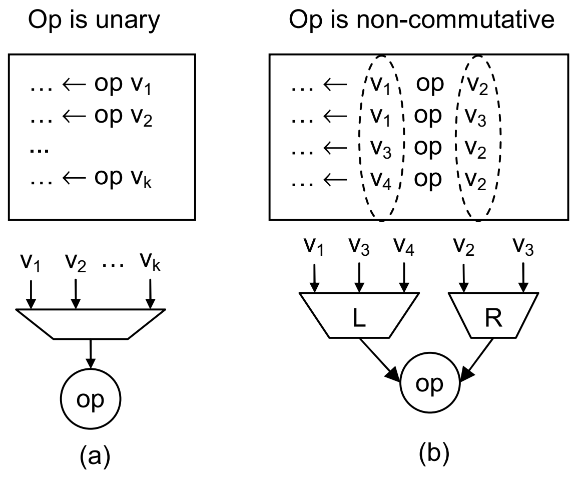

FIGURE 2. The port assignment problem is trivial for (a) unary and (b) binary non-commutative operations.

In conditions (1) and (2), the sources are connected to opposite inputs of the operator. In case (3), one of the source is connected to both inputs; thus, the assignment of the other source to either (or both) input(s) suffices for legality.

The optimization analogue of the PAP-BCO asks for a solution that maximizes $|V_L| + |V_R|$, i.e., that minimizes $|V_{LR}|$. This strategy drives the overall area of the two muxes down.

Pangrle [1] reduced the DCP (MIBSP) to the PAP-BCO. Let $G = (V, E)$ be an undirected graph: (V_L, V_R, V_{LR}) is a solution to the PAP-BCO if and only if $V' = V_L \cup V_R$ induces a bipartite subgraph $G' = (V', E')$ of G. For example, assume there is an edge $e = (v_1, v_2) \in E$ such that $v_1, v_2 \in V_L$; this is not a legal solution, since e mandates that v_1 and v_2 have connections to opposite input ports, while in fact, they are only connected to the left input port.

Fig. 3(a) shows a schedule of time steps where op is used; the corresponding graph G and optimal solution to the MIBSP are shown in Fig. 3(b) and (c) respectively. The largest induced bipartite subgraph of G contains six of the eight vertices; at most six sources can be connected to a single mux.

Fig. 3(d) and (e) show two solutions to the PAP-BCO, both of which correspond to the same induced bipartite subgraph. In the first solution $|V_L| = 4$ and $|V_R| = 2$; in the second solution, $|V_L| = |V_R| = 3$. Fig. 3(f) and (g) show the instantiated muxes corresponding respectively to Fig. 3(d) and (e). In Fig. 3(f), the two muxes respectively have *4* and *6* inputs; in Fig. 3(g), they both have *5* inputs. Since the delay of a *5:1* mux is assumed to be less than that of a *6:1* mux, the solution in Fig. 3(g) has a lower critical path delay while consuming comparable area.

From the perspective of the PAP-BCO as formulated by Pangrle [1] the solutions in Fig. 3(f) and (g) are both optimal, as both have a total of 10 connections to the operator inputs. The position taken by this paper is that the solution in Fig. 3(g) is superior, due to the fact that both muxes have the same size. This motivates the need for a new formulation of the PAP-BCO to account for this discrepancy. Let *PAP-BCO** denote PAP-BCO when using the modified objective.

Let $\delta(V_L, V_R) = max\{|V_L|, |V_R|\} - min\{|V_L|, |V_R|\}$.

Decision Problem 4 (PAP-BCO*). Given sets V of sources and E of time steps, and positive integers k, l, is there a solution (V_L, V_R, V_{LR}) to the PAP-BCO* with $|V_L| + |V_R| \geq k$ and $\delta(V_L, V_R) \leq l$.

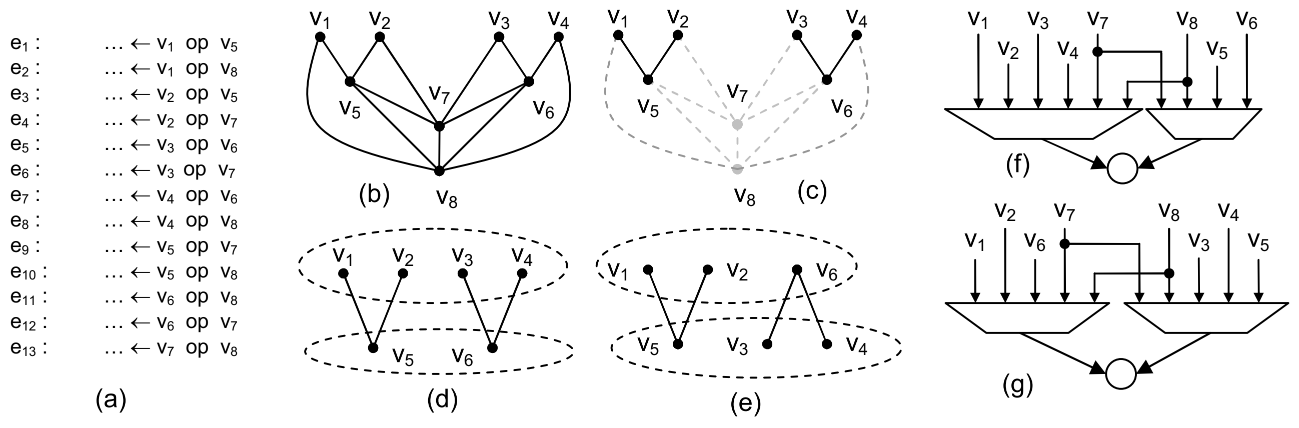

FIGURE 3. Example instance of the PAP-BCO: (a) a schedule of time steps using binary commutative operator op; (b) corresponding MIBSP instance; (c) optimal solution to the MIBSP; (d)/(f) and (e)/(g) two different solutions to the PAP-BCO corresponding to the MIBSP instance in (c); The solution to the PAP-BCO in (e)/(g) is preferable to (d)/(f) because the difference in the size of the two multiplexors is minimized.

The optimization analogue of the PAP-BCO is updated as well: if there are multiple tri-partitions where $|V_L| + |V_R|$ is maximum, the optimal one minimizes $\delta(V_L, V_R)$.

In Fig. 3(c)-(e) the bipartite subgraph has two connected components;, the solution in Fig. 3(e) is the same Fig. 3(d), but with the orientation of the second component, $\{v_3, v_4, v_6\}$, swapped with respect to the two independent sets, balancing them.

Next, we show that the NPP can model the problem of improving a solution to the PAP-BCO/MIBSP by reducing $||V_L| - |V_R||$. Let $G = (V, E)$ be a bipartite graph, with n connected components: $G^{(i)} = (V^{(i)}, E^{(i)})$, $1 \leq i \leq n$.

For connected component $G^{(i)}$, let $V_1^{(i)}$ and $V_2^{(i)}$ be the two independent sets. Without loss of generality, assume that $V_1^{(i)}$ and $V_2^{(i)}$ are chosen such that $|V_1^{(i)}| \geq |V_2^{(i)}|$. Now, for each connected component, we can assign $V_1^{(i)}$ to the left mux (V_L) and $V_2^{(i)}$ to the right mux (V_R), or vice-versa.

Assume that $V_L = V_1^{(i)}$, i.e., $V_1^{(i)}$ is assigned to the left mux; then the left mux will receive $|V_1^{(i)}| - |V_2^{(i)}| \geq 0$ inputs in excess of the right mux. For one connected component, this is unavoidable; when there are multiple connected components, the goal is to balance the assignment such that $\delta(V_L, V_R)$ is minimized; the NPP models this problem precisely.

In Fig. 3(c)-(e), each connected component has 2 vertices in one independent set and 1 vertex in the other; since $2\text{-}1=1$, the corresponding instance of the NPP is $T = \{1, 1\}$. The sub-optimal solution to the PAP-BCO* shown in Fig. 3(d)/(f) corresponds to sub-optimal solution $X = \{1, 1\}$, $Y = \{\}$ of the NPP, where $\Delta(X, Y) = 2$; the optimal solution to the PAP-BCO* shown in Fig. 3(e)/(g) corresponds to solution $X = \{1\}$, $Y = \{1\}$ of the NPP, which is optimal since $\Delta(X, Y) = 0$.

Lemma 1. The PAP-BCO* is NP-Hard.
Proof. Let $T = \{x_0, ..., x_{n-1}\}$ and $t \geq 0$ be an instance of the NPP. For $x_i = j \in T$, create a bipartite graph $G^{(i)} = (V^{(i)}, E^{(i)})$, where $V^{(i)} = \{v_{i,0}, ..., v_{i,j+1}\}$ and $E^{(i)} = \{(v_{i,0}, v_{i,r}) | 1 \leq r \leq j+1\}$.

The orientation of this connected component contributes x_i vertices to one independent set (or the other).

Now, construct a larger graph $G = (V, E)$, where each $G^{(i)}$ is a distinct connected component, i.e., $V = V_0 \cup V_1 \cup ... \cup V_{n-1}$ and $E = E_0 \cup E_1 \cup ... \cup E_{n-1}$. Let G be an instance of the PAP-BCO* with $k = |V|$ and $l = t$. By construction, G is bipartite so V_{LR} is empty. Any solution to the PAP-BCO* has the form $V = (V_L, V_R, \phi)$, which justifies the choice for k.

For each component $G^{(i)} = (V^{(i)}, E^{(i)})$, either $v_{i,0} \in V_L$ and $v_{i,k} \in V_R$, $1 \leq k \leq j+1$, or vice-versa. Then respectively, $G^{(i)}$ contributes x_i more vertices to V_R than to V_L, or vice-versa. In the former case, we insert x_i into Y; in the latter case, we insert x_i into X. Then $|V_L| = \Sigma(X) + n$ and $|V_R| = \Sigma(Y) + n$. Therefore $\Delta(X, Y) = |(\Sigma(X) - n) - (\Sigma(Y) - n)| = |\Sigma(X) - \Sigma(Y)|$, from which it follows that $\delta(V_L, V_R) \leq l$ if and only if $\Delta(X, Y) \leq l$. □

Theorem 1. The PAP-BCO* is NP-complete.
Proof. The PAP-BCO* is NP-Hard by Lemma 1. To verify that an instance of the PAP-BCO* in polynomial time, first verify that it is a legal solution to the PAP-BCO [1], and then test whether or not $\delta(V_L, V_R) \leq l$. □

One approach to solve the PAP-BCO* is to first solve the PAP-BCO and then improve the solution via MNPP, i.e., apply the reverse of the reduction in Lemma 1.

Let $G = (V, E)$ be the original graph, and let $G' = (V', E')$ be the bipartite subgraph, i.e., $V' = V_L \cup V_R$. Now, let us divide G' into connected components; since G' is bipartite, each connected component is bipartite as well. For example, in Fig. 3(d) and (e), G' has two bipartite connected components.

Let U be the set of vertices in a connected component, and let U_L and U_R be the vertices in U belonging to V_L and V_R respectively. Since G' is bipartite, we can swap the vertices between U_L and U_R. If $|U_L| > |U_R|$, swapping the vertices increases $|V_R|$ and decreases $|V_L|$ by $|U_L| - |U_R|$.

For example, consider Fig. 3(d). Without loss of generality, assume that $V_L = \{v_1, v_2, v_3, v_4\}$, and $V_R = \{v_5, v_6\}$. Between the two connected components, let $U = \{v_3, v_4, v_6\}$. Then $U_L = \{v_3, v_4\}$, $U_R = \{v_6\}$, and $|U_L| - |U_R| = 1$. Swapping the vertices in U_L and U_R moves v_3 and v_4 into V_R and v_6 into V_L, a net change of -1 to $|V_L|$ and $+1$ to $|V_R|$. This yields the updated result shown in Fig. 3(e).

More generally, this process of "reorienting" each connected component to minimize $||V_L| - |V_R||$ can be modeled as an instance of the MNPP. For each connected component having k_L vertices in V_L and k_R vertices in V_R, such that $k_L \geq k_R$, create an integer $T_i = k_L - k_R$, which is added to T. Without loss of generality, if the solution to the MNPP places T_i in X, then the vertices corresponding to this component are swapped between V_L and V_R; otherwise, the assignment is not perturbed. Processing each component this way yields an instance of the MNPP for which $\Sigma(T)$ is bounded by $|V|$; this instance can be solved optimally by dynamic programming in $O(|V|^2)$ time [8].

For example, the instance of the MNPP corresponding to Fig. 3(d) is *{1, 1}*. Both connected components begin with two vertices in V_L and one vertex in V_R. The initial partition, corresponding to the solution in Fig. 3(d), is $X = \{1, 1\}$ and $Y = \{\}$, and $\Delta(X, Y) = 2$. Reorienting either of the two connected components yields a solution $X = \{1\}$, $Y = \{1\}$, $\Delta(X, Y) = 0$ to the MNPP, which is an optimal solution, and corresponds to Fig. 3(e).

If we begin with an optimal solution to the PAP-BCO, this approach can not, in general, guarantee an optimal solution to the PAP-BCO*: some optimal solutions to the PAP-BCO simply have greater capacity for improvement via MNPP than others do. Fig. 4 shows an example where this occurs. An undirected graph is shown in Fig. 4(a), along with two single-component MIBSPs in Figs. 4(b) and (c). The MNPP cannot improve either solution since both contain only one connected component. Both of these components correspond to optimal solutions to the PAP-BCO, but only the one in Fig. 4(c) is an optimal solution to the PAP-BCO*.

Fig. 4 establishes that MNPP cannot transform an optimal solution to the PAP-BCO into an optimal solution to the PAP-BCO* in the general case. This suggests that new algorithmic techniques that solve the PAP-BCO* directly are needed in future high-level synthesis systems.

V. RELATED WORK

Prior high-level synthesis systems solve the PAP-BCO in conjunction with other binding problems [9-14]. Raje and Bergamaschi [12], for example, bind operations to resources one-by-one: when a new operation is bound to a resource, the port assignment is permuted to reduce the cost; estimates of the future connectivity cost drive the binding decisions. We agree that binding methods should include estimates of the costs of the muxes inserted during port assignment; however, it always possible to improve the solution after binding [1, 2].

Register and function unit binding methods should include estimates of the costs of the muxes inserted during port assignment when making their decisions, and they can even construct default port assignment solutions to help guide them; however, as discussed here, and also by Chen and Cong [2], ports can be re-assigned after the fact. The primary contribution of this paper is that the PAP-BCO* should be used in place of the PAP-BCO formulation [1].

Port assignment can also occur in the context of datapath merging [3-5]: rather than synthesizing a single datapath, multiple pre-synthesized datapaths are merged into a single multi-operational datapath that implements all of their functionality, but with reduced area. The PAP emerges when multiple operations originating from different datapaths are bound to the same resource. The same PAP-BCO* formulation described here can be used.

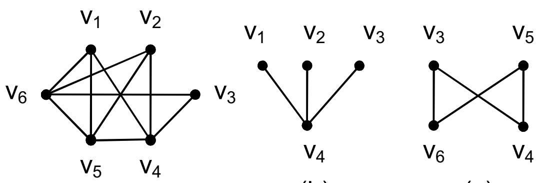

FIGURE 4. MNPP cannot always transform an optimal solution to the PAP-BCO into an optimal solution to the PAP-BCO*. (a) An example graph; (b) one optimal solution to the PAP-BCO that is a sub-optimal solution of the PAP-BCO*; (c) another optimal solution to both the PAP-BCO and PAP-BCO*. MNPP cannot improve either of these two solutions.

VI. CONCLUSION

Pangrle's [1] formulation of the PAP-BCO has been updated to account for the secondary goal of evenly distributing input signals among two muxes placed on the input of a binary commutative operator. Although the MNPP can improve a solution to the PAP-BCO*, it cannot guarantee optimality. Better results could be obtained through heuristic methods that solve the PAP-BCO* directly.

REFERENCES

[1] B. Pangrle, *On the complexity of connectivity binding*. IEEE Trans. CAD, vol. 10, no. 11, Nov., 1991, pp. 1460-1465.

[2] D. Chen and J. Cong, *Register binding and port assignment for multiplexor optimization*. Asia and South Pacific Design Automation Conf., Yokohama, Japan, Jan., 2004, pp. 68-73.

[3] P. Brisk, A. Kaplan, and M. Sarrafzadeh, *Area-efficient instruction set synthesis for reconfigurable system-chip designs*. 41st Design Automation Conf., San Diego, CA, USA, June, 2004, pp. 395-400.

[4] N. Moreano, E. Borin, C. C. de Souza, and G. Araujo, *Efficient datapath merging for partially reconfigurable architectures*. IEEE Trans. CAD, vol. 24, no. 7, July, 2005, pp. 969-980.

[5] M. Zuluaga, and N. Topham, *Resource sharing in custom instruction set extensions*. IEEE Symposium on Application-Specific Processors, Anaheim, CA, USA, June 8-9, 2008, pp. 7-13.

[6] S. Ghiasi, E. Bozorgzadeh, P-K. Huang, R. Jafari, and M. Sarrafzadeh, *A unified theory of timing budget management*. IEEE Trans. CAD, vol. 25, no. 11, November, 2006, pp. 2364-2375.

[7] S. Mitra, L. J. Avra, and E. J. McCluskey, *Efficient multiplexor synthesis techniques*. IEEE Design and Test of Computers, vol. 17, no. 4, October-December, 2000, pp. 90-97.

[8] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. New York: W.H. Freeman & Co., 1979.

[9] C. Y. Huang, Y. S. Chen, Y. L. Lin, and Y. C. Hsu, *Data path allocation based on a bipartite weighted matching*. Design Automation Conf., Orlando, FL, USA, June, 1990, pp. 499-504.

[10] M. Rim, R. Jain, and R. De Leone, *Optimal allocation and binding in high-level synthesis*. Design Automation Conf., Anaheim, CA, USA, 1992, pp. 120-123.

[11] T. Kim and C. L. Liu, *An integrated data path synthesis algorithm based on network flow method*. IEEE Custom Integrated Circuits Conf., May, 1995, pp. 615-618.

[12] S. Raje, and R. A. Bergamaschi, *Generalized resource sharing*. Int. Conf. Computer-Aided Design, San Jose, CA, USA, November, 1997, pp. 326-332.

[13] H. W. Zhu and C. C. Jong, *Interconnection optimization in data path allocation using minimal cost maximal flow algorithm*. Microelectronics, vol. 33, no. 9, September, 2002, pp. 749-759.

[14] T. Kim and X. Liu, *Compatibility path based binding algorithm for interconnect reduction in high level synthesis*. Int. Conf. Computer-Aided Design, San Jose, CA, USA, November, 2007, pp. 435-441.