Functional area lower bound and upper bound on multicomponent selection for interval scheduling

Shen, Zhao Xuan; Jong, Ching Chuen

2000

Shen, Z. X., & Jong, C. C. (2000). Functional area lower bound and upper bound on multicomponent selection for interval scheduling. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*. 19(7), 745-759.

<https://hdl.handle.net/10356/91566>

<https://doi.org/10.1109/43.851990>

© 2000 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. <http://www.ieee.org/portal/site> This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Functional Area Lower Bound and Upper Bound on Multicomponent Selection for Interval Scheduling

Zhao-Xuan Shen, *Member, IEEE*, and Ching Chuen Jong

Abstract—In a realistic register-transfer-level component library, there usually exist several different hardware implementations for one generic function. This gives rise to a large design space of component selection which is interleaved with the scheduling of operations. Previous methods ignored the presence of multicomponent selection in the process of lower/upper bound estimation of scheduling, and produced the local lower/upper bounds which would cause the suboptimum designs.

Opposite to the previous methods, we compute, in this paper, the lower/upper bounds which consider scheduling and component selection simultaneously. A new problem of multicomponent selection integrated with interval scheduling is studied. We present a very interesting and important result that both the lower bound and upper bound of multicomponent selection are obtained on the most cost-effective components which have the minimum area-delay products. This property leads to that the lower bound and upper bound of multicomponent selection can be calculated efficiently. An integer linear programming model and a surrogate relaxation technique are proposed to derive an optimum surrogate lower bound which has the asymptotic performance ratio less than two for a single type of function. An upper bound with the same asymptotic performance ratio is also obtained which turns out to be the optimum solution value of the traditional unicomponent selection with the most cost-effective components. Both the theoretical analysis and the experimental results show that the performance of our bounds are very promising.

Index Terms—Component selection, high-level synthesis, integer linear programming, interval scheduling, lower/upper bound estimation, surrogate relaxation.

I. INTRODUCTION

GENERALLY speaking, given as input a behavioral description, a component library and a set of design requirements, the task of high-level hardware synthesis is to find the optimal register-transfer-level (RTL) architecture from all the possibilities that implements the behavior and satisfies the requirements. Large design space which is resulted from the freedom in input behavior, i.e., the scheduling of data flow graph (DFG), and the freedom in component library, i.e., the selection of components, has to be explored in order to obtain the optimal designs. Scheduling determines the assignment of behavioral operations to control steps while component selection determines from the component library the types and the numbers of components required. Scheduling and component selection

are closely interdependent as different operation scheduling requires different component selection while different component selection gives rise to different operation scheduling. To achieve the optimal design, they must be considered simultaneously.

Both scheduling and component selection problems are NP hard in general. Tackling them simultaneously is inevitably a computationally intensive problem. To obtain the optimal solution, use of lower bound and upper bound estimation is an efficient approach. It not only can efficiently prune a large amount of inferior designs without performing the actual synthesis which is very time consuming, but also can guide high-level synthesis systems by providing the optimal delay-area tradeoff trajectory. Furthermore, the tight lower/upper bounds can be used for evaluating the results produced by various heuristic synthesis techniques.

Previous work on lower/upper bound estimation for high-level synthesis can be found in [1]–[11]. Since scheduling with precedence constraint is NP hard, precedence relaxation techniques are often adopted. There are two types of precedence relaxation used in high-level synthesis for lower bound estimation. The first one is the complete precedence relaxation which simply removes the precedence constraints of the DFG, while the second one is the partial precedence relaxation which uses the as-soon-as-possible (ASAP) and the as-late-as-possible (ALAP) intervals to replace the precedence constraints of the DFG. The later is referred to as the interval scheduling in the paper, and is detailed in Section II.

The work of Jain *et al.* [1], [2] who gave the first lower bound in high-level synthesis literature falls in the first category. They proposed a mathematical model for predicting the area-delay curve. Their bound was very fast to compute, however, as they did not consider the precedence constraints and adopted a unicycle scheme where all operations were assumed to take the same clock cycle, their bound was often criticized to be over-lose and trivial.

Comparing to the complete precedence relaxation, use of ASAP and ALAP intervals to replace the precedence constraints of the DFG can produce better bounds. The reason is that the ASAP and ALAP intervals are the partial relaxation of the precedence constraints and easy to compute (with linear computation time). Due to this advantage, almost all the lower/upper bound estimation methods use the ASAP and ALAP interval scheduling to relax the precedence constrained scheduling. Two kinds of bounds, namely, the minimum load bound, and the integer linear programming (ILP) bound, were proposed in literature. The minimum load bound is based on the following Principle: the minimum $\lceil n/m \rceil$ resources or cycles, (called the load), are required to schedule n unit-delay

Manuscript received November 4, 1998; revised October 4, 1999. This paper was recommended by Associate Editor R. Camposano.

Z.-X. Shen is with Arcadia Design Systems, Inc., San Jose, CA 95134 USA.
C. C. Jong is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore.

Publisher Item Identifier S 0278-0070(00)05855-3.

operations of the same type if m cycles or resources are provided. The method is to enumerate all the combinations of the ASAP and ALAP intervals and compute the load on each interval. The maximum load over all the intervals yields the lower bound on resource or performance. Exemplified work includes Sharma and Jain's [3], Rabaey's [4], Hu's [5], [6] and Ohm's [7], [8] bounds. The ILP bound is derived from the various forms of relaxations of the integer linear programming (ILP) formulations of the scheduling problem. For example, Rim and Jain [9] obtained a greedy algorithm to compute the lower bound on performance from an ILP formulation of the resource constrained scheduling. Their bound was further improved by Langevin and Cerny [10] who adopted so-called ASAPUC (ALAPUC) values instead of ASAP (ALAP) values to refine the resource constraints.

The only exception work that did not use interval scheduling to relax the precedence constrained scheduling was done by Chaudhuri and Walker [11]. They derived the lower bound on functional units from a distinct relaxation of the ILP formulation. They first used Lagrangian relaxation to move the precedence constraints into the objective function, and then transformed the Lagrangian Dual problem into an alternative extended problem which they proved to be able to solve in polynomial time.

However, there is a common weakness in these previous approaches. All of them only considered the design space of scheduling and ignored the freedom in component selection. A simplified unicomponent selection assumption that only one component type exists for each functionality was presumed in the estimation. Under this assumption, the design space of component selection was not fully explored and resulted in local lower bounds being produced which would cause the suboptimal designs.

This point can be illustrated with the following simple example. Given a DFG as shown in Fig. 1(a), and a component library as shown in Fig. 1(b), assume the time constraint is two cycles. Under the unicomponent selection assumption, as shown in Fig. 1(c), the minimum number of components required is two type-A adders which produces the functional area lower bound of $2 * 50 = 100$. However, as shown in Fig. 1(d), if different implementations of the addition operations are allowed, the optimal architecture requires only one type-A adder and one type-B adder, giving the functional area cost of $50 + 15 = 65 < 100$, which means that the lower bound of the unicomponent selection is invalid.

This example shows that when there exist multicomponent implementations for a type of function, the traditional unicomponent selection methods produce the invalid lower bounds. If the bound is used for pruning the design space such as in branch and bound and/or heuristic algorithms, it will result in the suboptimal results.

In contrast to the previous unicomponent selection methods, we compute, in this paper, the functional area lower bound and upper bound for multicomponent selection. We adopt the traditional interval scheduling model in the estimation. A new problem of multicomponent selection integrated with interval scheduling is studied. We show that the problem remains NP hard even after the problem decomposition and surrogate relax-

Fig. 1. A simple example.

ation. We present a very interesting and important result that both the lower bound and the upper bound of multicomponent selection are obtained on the most cost-effective components which have the minimum area-delay products. This property greatly narrows down the large design space of the multicomponent selection so that the bounds of multicomponent selection can be calculated efficiently. An optimum surrogate lower bound with asymptotic performance ratio less than two is derived from the proposed ILP model and surrogate relaxation technique. An efficient upper bound with the same asymptotic performance ratio is also obtained which turns out to be the optimum solution value of the traditional unicomponent selection with the most cost-effective components. Theoretical analysis of the performance factors of the bounds, such as the absolute error, the performance ratio and the asymptotic performance ratio, which were rarely discussed in the high-level synthesis literature, are also presented.

Two assumptions are taken in the paper.

- 1) The area costs of the storage and interconnection components are assumed to be ignorable comparing with those of the functional components.
- 2) Each component in the library is assumed to be able to perform a unique function. The complex multifunctional components are not included.

The remainder of the paper is organized as follows. Section II formulates the problem. Section III computes the optimum area cost of the traditional unicomponent selection while Section IV uses it to produce a brute force upper bound for the multicomponent selection. In Section V, we derive an optimum surrogate lower bound, and as a by-product, we obtain in Section VI a more efficient algorithm for the upper bound of the multicom-

ponent selection. Theoretical estimation of the performance of the bounds is presented in Section VII. Experimental results, further discussions and conclusions are given in Sections VIII, IX, and X, respectively.

II. PROBLEM FORMULATION

A. Preliminary Definitions

Define *function set* F be the set of operation functions such as addition, multiplication, comparison, etc. Let $f(v)$ be the operation function of an operation v , and $F(v)$ be the set of all the operation functions.

We use $A \in Z^{+1}$ to denote the area set, $D \in Z^{+}$ to denote the delay set, and $S \in Z^{+}$ to denote the control step set.

Define a *component type* τ be a tuple $\langle a, f, d \rangle$, where $a \in A$ is the area, $f \in F$ is the function it can perform and $d \in D$ is the corresponding delay. An instance of a component type is called a *component*. The type of a component α is denoted by $\tau(\alpha)$. Let $a(\alpha) \in A$ be the area of a component (type) α , $f(\alpha) \in F$ be the function that can be performed by α , and $d(\alpha) \in D$ be the delay of α .

Define a *component type implementation set* of a function $f \in F$, denoted by $\Psi(f) \in \Gamma$, be the set of component types that can perform f .

Define a *component library* M be an unlimited multiset of component types, i.e., a set of component types where each component type can have unlimited number of components. Let Γ denote the set of component types.

Define a *component subset* $\Lambda \subset \Gamma \times Z^{+}$ be a limited subset of component library M , i.e., $\Lambda \subset M$. We use $\Gamma(\Lambda)$ to denote the set of component types in a component subset Λ , $Z(\Lambda, \tau) \in Z^{+}$ to denote the number of components of type τ in Λ . The component subset can be expressed as $\Lambda = \Gamma \times Z = \{\alpha_1^1, \alpha_1^2, \dots, \alpha_1^{Z(\Lambda, \tau_1)}, \dots, \alpha_i^j, \dots, \alpha_n^1, \alpha_n^2, \dots, \alpha_n^{Z(\Lambda, \tau_n)}\}$ where $\tau(\alpha_i^j) = \tau_i$ for $j = 1, \dots, Z(\Lambda, \tau_i)$, $i = 1, \dots, n$.

Define $a(\Lambda) = \sum_{\tau \in \Gamma(\Lambda)} Z(\Lambda, \tau) * a(\tau)$ be the *area cost* of Λ .

A function f or an operation v is said to be able to be performed by a component (type) α if and only if $f = f(\alpha)$ or $f(v) = f(\alpha)$.

Given a set of operation V and a set of component subset $\Lambda = \Gamma \times Z$, a *component allocation* is to find a mapping $\mu : V \rightarrow \Gamma \times Z$ such that for $\forall v \in V, \exists$ a component $\alpha \in \Gamma \times Z$ that $f(v) = f(\alpha)$, $\mu(v) = \alpha$, i.e., v is performed by α .

A component allocation μ is said to be *1-1 mapping* (read as 1 to 1 mapping) if only one component type exists for each function, i.e., $\tau(\mu(v_i)) = \tau(\mu(v_j))$ for $\forall v_i, v_j \in V$ with $f(v_i) = f(v_j)$.

A component allocation μ is said to be *1-n mapping* (read as 1 to n mapping) if there exist multiple component types to perform a function, i.e., $\exists v_i, v_j \in V$ with $f(v_i) = f(v_j)$, that $\tau(\mu(v_i)) \neq \tau(\mu(v_j))$.

B. Component Selection for Interval Scheduling

Let $e(v_i)$ be the earliest time step that the operation v_i could be started, $l(v_i)$ be the latest time step that the operation v_i must

be completed. Then the time interval $[e(v_i), l(v_i)]$ gives the *active range* of the operation v_i , i.e., the set of time steps within which the operation v_i must be started and completed.

Given a set of operations V where each operation is associated with an interval $I(v_i) = [e(v_i), l(v_i)]$, and a component subset $\Lambda = \Gamma \times Z$, the *integrated component allocation and interval scheduling*, denoted by $\text{ICAIS}(V, I, \Lambda)$, is to find a component allocation $\mu : V \rightarrow \Gamma \times Z$ and an interval scheduling $\Theta : V \rightarrow S$, satisfying the following:

- 1) *Interval Constraint*: for $\forall v_i \in V, e(v_i) \leq \Theta(v_i) < \Theta(v_i) + d(\mu(v_i)) \leq l(v_i)$, i.e., each operation must be scheduled within its interval;
- 2) *Resource Sharing Constraint*: for $\forall v_i, v_j \in V$, if $f(v_i) = f(v_j)$, $\mu(v_i) = \mu(v_j)$, then $S(v_i) \cap S(v_j) = \emptyset$ where $S(v_i) = \{s \in S \mid \Theta(v_i) \leq s \leq \Theta(v_i) + d(\mu(v_i)) - 1\}$, i.e., a component cannot perform any other operation before it completes the executing operation;
- 3) *Component Constraint*: $\forall \tau \in \Gamma, Z(\Lambda, \tau) \geq \text{Max}_{0 \leq s \leq T} |\{v_i \in V \mid \Theta(v_i) \leq s \leq \Theta(v_i) + d(\mu(v_i)) - 1, \tau(\mu(v_i)) = \tau\}|$, i.e., for each time step, the number of operations that can be performed cannot exceed the number of components which can perform the types of the operations.

Given a set of operations V where each operation is associated with an interval $I(v_i) = [e(v_i), l(v_i)]$, and a component library M , the *component selection for interval scheduling*, denoted by $\omega(V, I, M)$, is to find a component subset $\Lambda = \Gamma \times Z$, such that $\exists$ an integrated component allocation and interval scheduling $\text{ICAIS}(V, I, \Lambda)$. The area cost of Λ is called the *area cost of the component selection*, denoted by $a(\omega(V, I, M))$. The component selection with the minimal area cost $a(\omega(V, I, M))$ is called the *optimal component selection*, denoted by $\omega^*(V, I, M)$.

According to the types of component allocation μ , there are two types of component selection for interval scheduling:

If μ is 1-1 mapping, we call it *unicomponent selection*, denoted as $1 - 1\omega(V, I, M)$.

If μ is 1-n mapping, we call it *multicomponent selection*, denoted as $1 - n\omega(V, I, M)$.

Obviously, $1 - 1\omega(V, I, M)$ is a special case of $1 - n\omega(V, I, M)$.

III. THE OPTIMUM AREA COST OF $1 - 1\omega(V, I, M)$

In this section, we compute the optimum area cost of the traditional unicomponent selection $1 - 1\omega(V, I, M)$ based on the well-known pigeonhole principle [14] stated as below.

Pigeonhole Principle: n pigeons go into m pigeonholes, at least $\lceil n/m \rceil$ pigeons are in the same pigeonholes.

Corollary 1: If at most m of n pigeons are allowed to enter the same pigeonhole, the minimum and feasible² number of pigeonholes is $\lceil n/m \rceil$. ■

Lemma 1: Assuming that there are n operations and each operation takes delay d to complete, the minimum and feasible number of components needed to complete all the operations within interval $[t_1, t_2]$, $(t_2 - t_1 \geq d)$ is $\lceil n / \lceil (t_2 - t_1) / d \rceil \rceil$.

¹ Z^{+} is the set of positive integer.

²“Minimum and feasibly” means the minimum value which is also feasible, i.e., there exists a feasible solution for the minimum value.

Proof: There are at most $\lfloor (t_2 - t_1)/d \rfloor$ time units (resp. pigeonholes) to perform all these n operations (resp. pigeons). By corollary 1, the minimum and feasible number of components needed to complete all the operations within the interval $[t_1, t_2]$, $(t_2 - t_1 \geq d)$ is $\lceil n / \lfloor (t_2 - t_1)/d \rfloor \rceil$. ■

Theorem 1: Assuming that there are n operations with the same function, each operation has an active range $[e_i, l_i]$, and takes time d ($l_i - e_i \geq d$ for all i) to complete, the minimum and feasible number of components needed to complete all the operations within $[0, T]$ ($T \geq \text{Max}_{v_i} l_i$) is

$$P = \text{Max}_{v[t_1, t_2] \in [0, T]} \left\lceil \frac{\sum_{i=1}^n \eta_i(t_1, t_2)}{\lfloor (t_2 - t_1)/d \rfloor} \right\rceil$$

where

$$\eta_i(t_1, t_2) = \begin{cases} 1, & t_1 \leq e_i < l_i \leq t_2 \\ 0, & \text{otherwise.} \end{cases}$$

Proof: Obviously, $\eta_i(t_1, t_2) = 1$ represents that operation v_i must be started and completed within $[t_1, t_2]$. So, $\sum_{i=1}^n \eta_i(t_1, t_2)$ gives the number of operations that must be started and completed within $[t_1, t_2]$. By Lemma 1, the minimum and feasible number of components to complete all the operations within $[t_1, t_2]$ is $\lceil (\sum_{i=1}^n \eta_i(t_1, t_2) / \lfloor (t_2 - t_1)/d \rfloor) \rceil$. Therefore, P gives the minimum and feasible number of components required for all the intervals within $[0, T]$, i.e., the minimum and feasible number of components needed to complete all the operations. ■

Based on Theorem 1, we obtain the algorithm of computing the optimum area cost of $1 - \text{nw}(V, I, M)$ as follows:

Algorithm 1: Computing the optimum area cost of

$1 - \text{nw}(V, I, M)$

For each function $f \in f(V)$

{ $E \leftarrow \{e(v_i) \text{ sorted in ascending order } | f(v_i) = f, e(v_i) \notin E\}$

$L \leftarrow \{l(v_i) \text{ sorted in ascending order } | f(v_i) = f, l(v_i) \notin L\}$

$Z(\Lambda, \tau) \leftarrow 0$

For each $t_1 \in E$, each $t_2 \in L$ and $t_2 - t_1 \geq d(\tau)$

$$\left\{ Z(\Lambda, \tau) \leftarrow \text{Max} \left(Z(\Lambda, \tau), \left\lceil \frac{\sum_{i=1}^n \eta_i(t_1, t_2)}{\lfloor (t_2 - t_1)/d(\tau) \rfloor} \right\rceil \right) \right\}$$

}

$a(1 - \text{nw}^*(V, I, M)) \leftarrow \sum_{\text{all } \tau} Z(\Lambda, \tau) * a(\tau)$

Theorem 2: Algorithm 1 computes an optimum area cost of $1 - \text{nw}(V, I, M)$ in $O(g * \text{Max}(n \log n, (nT^2/2)))$ where n is the number of operations, g is the number of functions and $T \geq \text{Max}_{v_i \in V} l(v_i)$.

Proof: As under the unicomponent selection, each type of operation can be performed by one and only one type of component. Hence, the $1 - \text{nw}(V, I, M)$ problem can be divided into several independent subproblems $1 - \text{nw}(V_k, I, M)$ where each V_k consists of only one type of operations that can be performed by only one type of component. By Theorem 1, P gives the minimum number of components required by the interval scheduling of V_k . Therefore, Algorithm 1 computes the minimal area cost required by the interval scheduling of V , i.e., the optimum lower bound of $1 - \text{nw}(V, I, M)$.

It can be seen that for each of g functions, the sorting of $\{e(v_i)\}$ and $\{l(v_i)\}$ takes $O(n \log n)$, and there are at most $T(T - 1)/2$ intervals of which each interval has a maximum of n nodes need to be checked, so the worst overall complexity is $O(g * \text{Max}(n \log n, (nT^2/2)))$. ■

IV. A BRUTE FORCE ALGORITHM FOR UPPER BOUND OF $1 - \text{nw}(V, I, M)$

As $1 - \text{nw}(V, I, M)$ is a special case of $1 - \text{nw}(V, I, M)$, the optimum solution of $1 - \text{nw}(V, I, M)$ is also a feasible solution of $1 - \text{nw}(V, I, M)$. Hence, we obtain the following theorem.

Theorem 3: The algorithm 1 produces an upper bound of area cost of $1 - \text{nw}(V, I, M)$. ■

To find the minimum upper bound, we can immediately obtain a brute force algorithm by enumerating all the possible combinations of component types for different types of operation functions described as below.

Algorithm 2: Computing the upper bound of $1 - \text{nw}(V, I, M)$

$\{Ms\} \leftarrow$ All the combinations of component types of all the functions

$\text{UB} \leftarrow \infty$

For each $M \in \{Ms\}$

{

Call Algorithm 1 to compute $a(1 - \text{nw}^*(V, I, M))$

$\text{UB} \leftarrow \text{Min}(\text{UB}, a(1 - \text{nw}^*(V, I, M)))$

}

Theorem 4: The complexity of Algorithm 2 is $O(g * \text{Max}(n \log n, (nT^2/2)) * \prod_{k=1}^g |\Psi(f_k)|)$. If there are m component implementations for each function, the complexity is $O(m^g * g * \text{Max}(n \log n, (nT^2/2)))$. ■

V. AN OPTIMUM SURROGATE LOWER BOUND OF $1 - \text{nw}(V, I, M)$

Although the computation of the upper bound of $1 - \text{nw}(V, I, M)$ looks straightforward, computing the lower bound of $1 - \text{nw}(V, I, M)$ is quite difficult. In this section, we first propose an efficient ILP model and then decompose it into several independent subproblems $1 - \text{nw}(V_k, I, M)$, and finally obtain a lower bound from the LP relaxation of the optimum surrogate relaxation of $1 - \text{nw}(V_k, I, M)$.

A. ILP Formulation of $1 - \text{nw}(V, I, M)$

Assume that there is a set of functions and each function f_k can be performed by the component type implementation set $\Psi(f_k) = \{\tau_j | f(\tau_j) = f_k, j = m_k, \dots, m_{k+1} - 1\}$. All the components are consecutively numbered as $m_1, m_1 + 1, \dots, m_2 - 1, m_2, \dots, m_k, \dots, m_{k+1} - 1, \dots, m_g, \dots, m_{g+1} - 1$ where $m_1 = 1, m_{g+1} = m + 1$. Note that the components $m_k, \dots, m_{k+1} - 1$ have the same type and m is the total number of components. As each component type is allowed to perform only one type of operation, $\Psi(f_k) \cap \Psi(f_l) = \emptyset$ for $\forall k \neq l$.

The problem of minimum functional area multicomponent selection for interval scheduling $1 - \text{nw}(V, I, M)$ can be formulated as the following ILP problem.

Problem 1: $\text{Min } a(\omega) = \sum_{k=1}^g \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j$
s.t.

$$\sum_{\tau_j \in \Psi(f(v_i))} \sum_{t=e(v_i)}^{l(v_i)-d(\tau_j)} y_{v_i, \tau_j, t} = 1 \quad \forall v_i \in V \quad (1-1)$$

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t'}^{t'+d(\tau_j)-1} y_{v_i, \tau_j, t} \leq x_j$$

$$j = 1, \dots, m; \quad t' = 0, \dots, T \quad (1-2)$$

where

$$x_j = Z(\Lambda, \tau_j) \geq 0, \quad \text{integer},$$

$$y_{v_i, \tau_j, t} = \begin{cases} 1, & v_i \text{ is performed by } \tau_j \text{ and scheduled at } t \\ 0, & \text{otherwise.} \end{cases}$$

Constraint (1-1) expresses that each operation node v_i must be performed by one component with type $\in \Psi(f(v_i))$, and scheduled at one time step in its active range $[e(v_i), l(v_i) - d(\tau_j)]$. Constraint (1-2) indicates that for each component type τ_j , and each time step t' , the number of operations that can be performed by the component type τ_j and scheduled at any time step in the time period $[t', t' + d(\tau_j) - 1]$ is no greater than the available number of components with type τ_j . The objective is to minimize the total area cost of components used.

Lemma 2: Constraint (1-2) is equivalent to the following constraint:

$$\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f(\tau_j), v_i \in V} d(\tau_j) * y_{v_i, \tau_j, t} \leq x_j * (t_2 - t_1)$$

$$\forall [t_1, t_2], \quad 0 \leq t_1 < t_2 \leq T; \quad j = 1, \dots, m \quad (2-2)$$

Proof: Without loss of generality, assume that there is a component type τ_j , which can perform any operation v_i with $f(v_i) = f(\tau_j)$. For $\forall [t_1, t_2], 0 \leq t_1 < t_2 \leq T$, there are at most $h = \lfloor (t_2 - t_1) / d(\tau_j) \rfloor$ intervals: $[t_1, t_1 + d(\tau_j) - 1], [t_1 + d(\tau_j), t_1 + 2d(\tau_j) - 1], \dots, [t_1 + (h-1)d(\tau_j), t_1 + h*d(\tau_j) - 1]$, where a component with type τ_j can perform operations. From Constraint (1-2)

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t_1}^{t_1+d(\tau_j)-1} y_{v_i, \tau_j, t} \leq x_j,$$

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t_1+d(\tau_j)}^{t_1+2d(\tau_j)-1} y_{v_i, \tau_j, t} \leq x_j, \dots,$$

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t_1+(h-1)d(\tau_j)}^{t_1+h*d(\tau_j)-1} y_{v_i, \tau_j, t} \leq x_j.$$

Multiply $d(\tau_j)$ to each constraint and sum them

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t_1}^{t_1+h*d(\tau_j)-1} d(\tau_j) * y_{v_i, \tau_j, t} \leq h * d(\tau_j) * x_j$$

$$= x_j * d(\tau_j) * \left\lfloor \frac{t_2 - t_1}{d(\tau_j)} \right\rfloor \leq x_j * (t_2 - t_1).$$

Note that in the interval $[t_1 + h * d(\tau_j), t_2]$, as

$$t_2 - (t_1 + h * d(\tau_j))$$

$$= t_2 - t_1 - \left\lfloor \frac{t_2 - t_1}{d(\tau_j)} \right\rfloor * d(\tau_j) < d(\tau_j)$$

a component with type τ_j cannot perform any operation.³ So,

$$\sum_{f(v_i)=f(\tau_j), v_i \in V} \sum_{t=t_1+h*d(\tau_j)}^{t_2} d(\tau_j) * y_{v_i, \tau_j, t} = 0.$$

Therefore

$$\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f(\tau_j), v_i \in V} d(\tau_j) * y_{v_i, \tau_j, t} \leq x_j * (t_2 - t_1).$$

On the other hand, Constraint (1-2) can be easily obtained from Constraint (2-2) by letting $t_2 = t_1 + d(\tau_j)$

$$\sum_{t=t_1}^{t_1+d(\tau_j)} \sum_{f(v_i)=f(\tau_j), v_i \in V} d(\tau_j) * y_{v_i, \tau_j, t}$$

$$\leq x_j * (t_1 + d(\tau_j) - t_1) = x_j * d(\tau_j)$$

So

$$\sum_{t=t_1}^{t_1+d(\tau_j)-1} \sum_{f(v_i)=f(\tau_j), v_i \in V} y_{v_i, \tau_j, t}$$

$$\leq \sum_{t=t_1}^{t_1+d(\tau_j)} \sum_{f(v_i)=f(\tau_j), v_i \in V} y_{v_i, \tau_j, t} \leq x_j.$$

Hence, Constraint (1-2) is equivalent to Constraint (2-2). ■

Therefore, Problem 1 can be transformed to the following equivalent problem:

Problem 2: $\text{Min } a(\omega) = \sum_{k=1}^g \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j$
s.t.

$$\sum_{\tau_j \in \Psi(f(v_i))} \sum_{t=e(v_i)}^{l(v_i)-d(\tau_j)} y_{v_i, \tau_j, t} = 1 \quad \forall v_i \in V \quad (2-1)$$

$$\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f(\tau_j), v_i \in V} d(\tau_j) * y_{v_i, \tau_j, t} \leq x_j * (t_2 - t_1)$$

$$\forall [t_1, t_2], \quad 0 \leq t_1 < t_2 \leq T; \quad j = 1, \dots, m \quad (2-2)$$

where

$$x_j = Z(\Lambda, \tau_j) \geq 0, \quad \text{integer}$$

$$y_{v_i, \tau_j, t} = \begin{cases} 1, & v_i \text{ is performed by } \tau_j \text{ and scheduled at } t \\ 0, & \text{otherwise.} \end{cases}$$

B. Problem Decomposition

Since $\Psi(f_k) \cap \Psi(f_l) = \emptyset$ for $\forall k \neq l$, i.e., the component type implementation sets for different functions are independent, Problem 2 can be decomposed into g independent sub-problems.

³ $x - \lfloor x/d \rfloor d < d$ can be seen as follow: let $x = kd + b$ where $b = x \% d < d$, $x - \lfloor x/d \rfloor d = kd + b - \lfloor (kd + b/d) \rfloor d = kd + b - kd = b < d$.

For $k = 1, \dots, g$

$$\{ \text{Problem S1: } \min_{a(\omega_k)} = \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j \\ \text{s.t.}$$

$$\sum_{\tau_j \in \Psi(f_k)} \sum_{t=e(v_i)}^{l(v_i)-d(\tau_j)} y_{v_i, \tau_j, t} = 1 \quad \forall v_i \in V, \quad f(v_i) = f_k \quad (\text{S1-1})$$

$$\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f_k, v_i \in V} d(\tau_j) * y_{v_i, \tau_j, t} \leq x_j * (t_2 - t_1) \\ \forall [t_1, t_2], \quad 0 \leq t_1 < t_2 \leq T; \quad j = m_k, \dots, m_{k+1} - 1 \quad (\text{S1-2})$$

where

$$x_j = Z(\Lambda, \tau_j) \geq 0, \quad \text{integer,} \\ y_{v_i, \tau_j, t} = \begin{cases} 1, & v_i \text{ is performed by } \tau_j \text{ and scheduled at } t \\ 0, & \text{otherwise.} \end{cases} \\ \} \\ a(\omega) = \sum_{k=1}^g a(\omega_k)$$

Therefore, instead of solving Problem 2, we solve each of the independent subproblem S1, respectively.

Define: $V_k = \{v_i \in V \mid f(v_i) = f_k\}$,

α_{k, t_1, t_2} be the number of operations whose type is f_k and interval is in $[t_1, t_2]$, i.e., the number of $v_i \in V_k$ with $t_1 \leq e(v_i) < l(v_i) \leq t_2$,

β_{j, t_1, t_2} be the number of operations performed by a component τ_j whose type is f_k and interval is in $[t_1, t_2]$,

The interval $[t_1, t_2]$ with

$$\text{Max}_{\forall [t_1, t_2], 0 \leq t_1 < t_2 \leq T, m_k \leq j \leq m_{k+1}-1} \frac{\beta_{j, t_1, t_2}}{t_2 - t_1}$$

be the critical interval, denoted by $[t_1^*, t_2^*]$.

Obviously, for $m_k \leq j \leq m_{k+1} - 1$,

$$\beta_{j, t_1, t_2} = \sum_{t=t_1}^{t_2} \sum_{f(v_i)=f_k, v_i \in V} y_{v_i, \tau_j, t}, \\ \alpha_{k, t_1, t_2} = \sum_{j=m_k}^{m_{k+1}-1} \beta_{j, t_1, t_2} \\ = \sum_{j=m_k}^{m_{k+1}-1} \sum_{t=t_1}^{t_2} \sum_{f(v_i)=f_k, v_i \in V} y_{v_i, \tau_j, t}.$$

As

$$\frac{x_j}{d(\tau_j)} \geq \frac{\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f_k, v_i \in V} y_{v_i, \tau_j, t}}{t_2 - t_1}$$

for $\forall [t_1, t_2], 0 \leq t_1 < t_2 \leq T, j = m_k, \dots, m_{k+1} - 1$ is equivalent to the equation shown at the bottom of the page. Hence, Problem S1 is equivalent to

$$\text{Problem S2: } \min_{a(\omega_k)} = \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j \\ \text{s.t.}$$

$$\sum_{\tau_j \in \Psi(f_k)} \sum_{t=e(v_i)}^{l(v_i)-d(\tau_j)} y_{v_i, \tau_j, t} = 1 \quad \forall v_i \in V_k \quad (\text{S2-1})$$

$$\sum_{t=t_1^*}^{t_2^*} \sum_{v_i \in V_k} d(\tau_j) * y_{v_i, \tau_j, t} \leq x_j * (t_2^* - t_1^*) \\ j = m_k, \dots, m_{k+1} - 1 \quad (\text{S2-2})$$

where

$$x_j = Z(\Lambda, \tau_j) \geq 0, \quad \text{integer,} \\ y_{v_i, \tau_j, t} = \begin{cases} 1, & v_i \text{ is performed by } \tau_j \text{ and scheduled at } t \\ 0, & \text{otherwise.} \end{cases}$$

We refer to Problem S2 as the $1 - n\omega(V_k, I, M)$ problem.

Theorem 5: The problem $1 - n\omega(V_k, I, M)$ is NP hard.

Proof: We show that the $1 - n\omega(V_k, I, M)$ problem is a generation of the bin packing problem [12] which is NP hard. The bin packing is stated as below:

Instance: Finite set U of items, a size $s(u) \in Z^+$ for $u \in U$, a positive integer bin capacity B , and positive integer K .

Question: Is there a partition of U into disjoint sets $U_1, U_2, \dots, U_K$ such that the sum of the sizes of the items in each U_i is B or less?

It can be seen that the bin packing problem is a particular case of $1 - n\omega(V_k, M)$ when all the active ranges of the operations are identical ($e(v_i) = e, l(v_i) = l$ for all v_i). That is, given $m_{k+1} - m_k$ bins: $\tau_{m_k}, \dots, \tau_j, \dots, \tau_{m_{k+1}-1}$, each with capacity $t_2^* - t_1^*$ and cost $a(\tau_j)$, and a set V of items $\{v_i\}$, each with size $d(\tau_j)$ if putting into bin τ_j , is there a packing of V with $m_{k+1} - m_k$ bins, such that the sum of the sizes of the items in each bin is $t_2^* - t_1^*$ or less?

Therefore, the problem $1 - n\omega(V_k, I, M)$ is NP hard. ■

As $1 - n\omega(V_k, I, M)$ is the subproblem of $1 - n\omega(V, I, M)$, we immediately obtain:

Theorem 6: The problem $1 - n\omega(V, I, M)$ is NP hard. ■

C. Surrogate Relaxation

As $1 - n\omega(V_k, I, M)$ is NP hard, to compute the lower bound efficiently, relaxation has to be made. There are several widely used relaxation techniques: linear programming (LP) relaxation, Lagrangian relaxation and surrogate relaxation [15], [18], [19]. It is known that for an integer programming

$$\frac{x_j}{d(\tau_j)} \geq \text{Max}_{\forall [t_1, t_2], 0 \leq t_1 < t_2 \leq T, m_k \leq j \leq m_{k+1}-1} \frac{\sum_{t=t_1}^{t_2} \sum_{f(v_i)=f_k, v_i \in V} y_{v_i, \tau_j, t}}{t_2 - t_1} \\ = \frac{\sum_{t=t_1^*}^{t_2^*} \sum_{f(v_i)=f_k, v_i \in V} y_{v_i, \tau_j, t}}{t_2^* - t_1^*}$$

minimization problem P , $\text{LPR}(P) \leq \text{LR}(P) \leq \text{SR}(P) \leq P^*$, where $\text{LPR}(P)$, $\text{LR}(P)$, and $\text{SR}(P)$ are, respectively, the solution values to the LP relaxation, Lagrangian relaxation and surrogate relaxation of problem P while P^* is the optimum solution value of P . That is, surrogate relaxation dominates the LP relaxation and Lagrangian relaxation and may provide the tightest lower bound for the original problem.

In general, for an integer programming problem $P = \text{Min}\{cx \mid Ax \geq b, x \geq 0, x \in Z^n\}$ where A is a $m \times n$ matrix, b, c are $m \times 1$ and $1 \times n$ vectors, for a given set of nonnegative multipliers $\lambda \in R^m$, called surrogate vector (or multipliers), the solution value to standard surrogate relaxation problem $S(P, \lambda) = \text{Min}\{cx \mid \lambda Ax \geq \lambda b, x \geq 0, x \in Z^n\}$ gives a lower bound to P . The optimum set of surrogate multipliers $\lambda^* \in R^m$ is the one that produces the maximum lower bound to P , i.e., the optimum solution value to the surrogate dual program $D(S(P, \lambda)) = \text{Max}_{\lambda \geq 0} \text{Min}\{cx \mid \lambda Ax \geq \lambda b, x \geq 0, x \in Z^n\}$.

However there is no known efficient method to derive λ^* . Most people consider it as an “art”. By studying the special structure of problem $1 - n\omega(V_k, I, M)$, we show that:

Lemma 3: The optimum surrogate multipliers for surrogate relaxation problem $S(1 - n\omega(V_k, I, M), \lambda)$ is $\lambda_j^* = \kappa$ (κ is any positive constant) for all $m_j \leq j \leq m_{j+1} - 1$.

Proof: The surrogate relaxation, $S(1 - n\omega(V_k, I, M), \lambda)$, of $1 - n\omega(V_k, I, M)$ is as follow:

$$S(1 - n\omega(V_k, I, M), \lambda) : \text{Min } a(\omega_k) = \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j$$

s.t.

$$\sum_{\tau_j \in \Psi(f_k)} \sum_{t=e(v_i)}^{l(v_i)-d(\tau_j)} y_{v_i, \tau_j, t} = 1 \quad \forall v_i \in V_k \quad (\text{S3-1})$$

$$\frac{\sum_{j=m_k}^{m_{k+1}-1} \lambda_j \sum_{t=t_1^*}^{t_2^*} \sum_{v_i \in V_k} y_{v_i, \tau_j, t}}{t_2^* - t_1^*} \leq \sum_{j=m_k}^{m_{k+1}-1} \frac{\lambda_j * x_j}{d(\tau_j)} \quad (\text{S3-2})$$

where

$$x_j = Z(\Lambda, \tau_j) \geq 0, \quad \text{integer},$$

$$y_{v_i, \tau_j, t} = \begin{cases} 1, & v_i \text{ is performed by } \tau_j \text{ and scheduled at } t \\ 0, & \text{otherwise} \end{cases}$$

Let $\hat{j} = \arg \text{Min}\{\lambda_j : m_k \leq j \leq m_{k+1} - 1\}$. Suppose that (x_j^*) and $(y_{v_i, \tau_j, t}^*)$ is an optimal solution to $S(1 - n\omega(V_k, I, M), \lambda)$. We can obtain a feasible solution of the same objective value by setting, for each v_i and t , $y_{v_i, \hat{\tau}_j, t}^* = 1$ and $y_{v_i, \tau_j, t}^* = 0$ for $j \neq \hat{j}$, as it does not change (x_j^*) .

Hence, $S(1 - n\omega(V_k, I, M), \lambda)$ is equivalent to

$$\text{Min } a(\omega_k) = \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j$$

s.t.

$$\begin{aligned} \sum_{j=m_k}^{m_{k+1}-1} \frac{\lambda_j * x_j}{d(\tau_j)} &\geq \frac{\lambda_j \sum_{t=t_1^*}^{t_2^*} \sum_{v_i \in V_k} y_{v_i, \tau_j, t}}{t_2^* - t_1^*} \\ &= \frac{\lambda_j \sum_{j=m_k}^{m_{k+1}-1} \sum_{t=t_1^*}^{t_2^*} \sum_{v_i \in V_k} y_{v_i, \tau_j, t}}{t_2^* - t_1^*} \\ &= \frac{\lambda_j \alpha_{k, t_1^*, t_2^*}}{t_2^* - t_1^*} \end{aligned}$$

where x_j integer.

Obviously, it is an integer knapsack problem in minimization form [16]. It can be seen that for a given knapsack problem with given knapsack capacity $(\lambda_j \alpha_{k, t_1^*, t_2^*} / t_2^* - t_1^*)$, the smaller λ_j , the smaller of each item size $\lambda_j / d(\tau_j)$ (for given $d(\tau_j)$), and the larger the solution value. Hence, the maximum solution value of $D(S(1 - n\omega(V_k, I, M), \lambda)) = \text{Max}_{\lambda \geq 0} D(S(1 - n\omega(V_k, I, M), \lambda))$ is given by letting $\lambda_j = \text{Min}_{1 \leq j \leq k} \lambda_j = \lambda_j$ for all $m_j \leq j \leq m_{j+1} - 1$, i.e., all the surrogate multipliers are equal. Therefore, the optimum surrogate lower bound for the problem $1 - n\omega(V_k, I, M)$ is $\lambda_j^* = \kappa$ (κ is any positive constant) for all $m_j \leq j \leq m_{j+1} - 1$. ■

Therefore, by choosing surrogate multipliers $\lambda_j^* = \kappa$ for $m_j \leq j \leq m_{j+1} - 1$, the surrogate dual problem $D(S(1 - n\omega(V_k, I, M), \lambda))$ becomes $D(S(1 - n\omega(V_k, I, M), \lambda^*)) : \{\text{Min } a(\omega_k) = \sum_{j=m_k}^{m_{k+1}-1} a(\tau_j) * x_j \mid \sum_{j=m_k}^{m_{k+1}-1} (x_j / d(\tau_j)) \geq (\alpha_{k, t_1^*, t_2^*} / t_2^* - t_1^*), \text{ where } x_j \text{ integer}\}$ which is an integer knapsack problem in minimization form that is also NP hard [12], [16]. So,

Theorem 7: The optimum surrogate relaxation of $1 - n\omega(V, I, M)$ is NP hard. ■

D. LP Relaxation

However, different from the strongly NP hard problems such as bin packing problem, the integer knapsack problem belongs to a class of weakly NP hard problem [12]. A tight lower bound can be obtained by its LP relaxation.

Due to G. B. Dantzig [17], the optimal solution to the LP relaxation of $D(S(1 - n\omega(V_k, I, M), \lambda^*))$ is given by

$$x_s = \frac{d(\tau_s) \alpha_{k, t_1^*, t_2^*}}{t_2^* - t_1^*},$$

$$x_j = 0, \quad m_k \leq j \leq m_{k+1} - 1, \quad j \neq s$$

the optimal solution value is $(a(\tau_s) d(\tau_s) \alpha_{k, t_1^*, t_2^*} / t_2^* - t_1^*)$ where $s = \arg \min\{a(\tau_j) d(\tau_j) : m_k \leq j \leq m_{k+1} - 1\}$.

Notice that $d(\tau_s)$ must be no less than $t_2^* - t_1^*$, otherwise τ_s cannot perform any operation. We call τ_s the most cost-effective component. Therefore, we immediately obtain:

Theorem 8: A lower bound to the optimum surrogate dual problem $D(S(1 - n\omega(V_k, I, M), \lambda^*))$ and, thus, problem $1 - n\omega(V_k, I, M)$, is $L = \lceil (a(\tau_s) d(\tau_s) \alpha_{k, t_1^*, t_2^*} / t_2^* - t_1^*) \rceil$, where $s = \arg \min\{a(\tau_j) d(\tau_j) \mid d(\tau_j) \geq t_2^* - t_1^*, m_k \leq j \leq m_{k+1} - 1\}$. ■

E. Algorithm of Computing the Optimum Surrogate Lower Bound for $1 - n\omega(V, I, M)$

Algorithm 3: Computing the lower bound for $1 - n\omega(V, I, M)$

For function $f_k \in f(V)$

{/* Step 1: Sort intervals */}

$E \leftarrow \{e(v_i) \text{ sorted in ascending order } | f(v_i) = f_k, e(v_i) \notin E\}$

$L \leftarrow \{l(v_i) \text{ sorted in ascending order } | f(v_i) = f_k, l(v_i) \notin L\}$

{/* Step 2: Compute the critical time slot $[t_1^*, t_2^*]$ */}

$t_1^* \leftarrow 0, t_2^* \leftarrow 0, \alpha \leftarrow 0$

For each $t_1 \in E$, each $t_2 \in L$ and $t_2 > t_1$

{ $\alpha_{k,t_1,t_2} \leftarrow 0$

For each $v_i \in V$

if $f(v_i) = f_k$ and $t_1 \leq e(v_i) < l(v_i) \leq t_2$ then $\alpha_{k,t_1,t_2} \leftarrow$

$\alpha_{k,t_1,t_2} + 1$

If $(\alpha_{k,t_1,t_2}/t_2 - t_1) > \alpha$ then

{ $\alpha \leftarrow (\alpha_{k,t_1,t_2}/t_2 - t_1), \alpha_{k,t_1^*,t_2^*} \leftarrow \alpha_{k,t_1,t_2}, t_1^* \leftarrow t_1, t_2^* \leftarrow t_2$ }

}

{/* Step 3: Find the most cost-effective component */}

MinValue $\leftarrow \infty$

For each $\tau_j \in \Psi(f_k)$ and $d(\tau_j) \geq t_2^* - t_1^*$

{ If $(a(\tau_j) * d(\tau_j)) < \text{MinValue}$ Then

{ MinValue $\leftarrow a(\tau_j) * d(\tau_j), s \leftarrow j$ }

{/* Step 4: Compute the lower bound */}

$$L(a(\omega_k)) \leftarrow \left\lceil \frac{a(\tau_s)d(\tau_s)\alpha_{k,t_1^*,t_2^*}}{t_2^* - t_1^*} \right\rceil$$

}

$L(a(\omega)) = \sum_{\text{all } k} L(a(\omega_k))$

Theorem 9: The complexity of Algorithm 3 is $O(g(n \log n + (nT^2/2) + m))$ where m is the maximum number of implementations of a function. ■

VI. AN EFFICIENT ALGORITHM FOR THE UPPER BOUND OF $1 - n\omega(V, I, M)$

As a by-product of Section V, applying Theorem 1 on the most cost-effective component, we have,

Theorem 10: An upper bound for $1 - n\omega(V, I, M)$ is

$$U = a(\tau_s) \left\lceil \frac{\alpha_{k,t_1^*,t_2^*}}{\left\lfloor \frac{t_2^* - t_1^*}{d(\tau_s)} \right\rfloor} \right\rceil$$

where $s = \arg \min \{a(\tau_j)d(\tau_j) \mid d(\tau_j) \geq t_2^* - t_1^*, m_k \leq j \leq m_{k+1} - 1\}$ ■.

Hence, we obtain a more efficient algorithm for upper bound of $1 - n\omega(V, I, M)$ than the brute force algorithm given in Section IV.

Algorithm 4: Computing upper bound for $1 - n\omega(V, I, M)$

For function $f_k \in f(V)$

{

Step 1-3 is the same as Algorithm 3

$$U(a(\omega_k)) \leftarrow a(\tau_s) \left\lceil \frac{\alpha_{k,t_1^*,t_2^*}}{\left\lfloor \frac{t_2^* - t_1^*}{d(\tau_s)} \right\rfloor} \right\rceil$$

}

$U(a(\omega)) = \sum_{\text{all } k} U(a(\omega_k))$

Theorem 11: The complexity of Algorithm 4 is $O(g(n \log n + (nT^2/2) + m))$ where m is the maximum number of implementations of a function. ■

VII. ERROR ESTIMATION OF THE BOUNDS

For an instance A of a minimization problem, let $UB(A), LB(A)$ and $OPT(A)$ be the upper bound, lower bound and optimum solution value for the instance A , respectively.

Define: the *absolute error of an upper bound* or a *lower bound* be $\rho(UB(A)) = UB(A) - OPT(A)$ or $\rho(LB(A)) = OPT(A) - LB(A)$, the *performance ratio of an upper bound* or a *lower bound* be $\gamma(UB(A)) = (UB(A)/OPT(A))$ or $\gamma(LB(A)) = (OPT(A)/LB(A))$, the *asymptotic performance ratio of an upper bound* or a *lower bound* be $R^\infty(UB) = \inf\{r \mid \exists N_0, \gamma(UB(A)) \leq r \text{ for all instances } A \text{ with } OPT(A) \geq N_0\}$ or $R^\infty(LB) = \inf\{r \mid \exists N_0, \gamma(LB(A)) \leq r \text{ for all instances } A \text{ with } OPT(A) \geq N_0\}$.

In general, for a NP hard optimization problem, it is very difficult to find the optimum solution. Hence, instead of computing $OPT(A)$, we use $LB(A)$ and $UB(A)$ to estimate it and obtain the upper bound of the absolute error and the performance ratio.

As $LB(A) \leq OPT(A) \leq UB(A)$, we have, $\rho(UB(A)) \leq UB(A) - LB(A)$, $\rho(LB(A)) \leq UB(A) - LB(A)$, $\gamma(UB(A)) \leq (UB(A)/LB(A))$, $\gamma(LB(A)) \leq (UB(A)/LB(A))$.

We call $UB(\rho(LB, UB, A)) = UB(A) - LB(A)$ the *upper bound of absolute error* of both $UB(A)$ and $LB(A)$, and $UB(\gamma(LB, UB, A)) = (UB(A)/LB(A))$ the *upper bound of performance ratio* of both $UB(A)$ and $LB(A)$.

Theorem 12: Shown at the bottom of the next page, the proof can be found in the Appendix. ■

It can be seen that for $1 - n\omega(V_k, I, M)$, when the length of critical interval $t_2^* - t_1^*$ is divisible by the delay of the most cost-effective component $d(\tau_s)$, and the number of operations in the critical interval α_{k,t_1^*,t_2^*} is divisible by the floor of the ratio $\lfloor (t_2^* - t_1^*)/d(\tau_s) \rfloor$ of the length of critical interval and the delay of the most cost-effective component, both the upper bound and lower bound reach the optimum solution value. Even when these conditions are not satisfied, the asymptotic performance ratio for both the upper bound and lower bound is less than two (the number of operations in the critical interval is sufficient large, $\alpha_{k,t_1^*,t_2^*} \gg \lfloor (t_2^* - t_1^*)/d(\tau_s) \rfloor$), and closes to the optimum asymptotic performance ratio 1 when the length of critical interval is divisible by the delay of the most cost-effective component.

The upper bound of the overall absolute error and the overall performance ratio for both the upper bound $U = U(1 - n\omega(V, I, M))$ and the lower bound $L = L(1 - n\omega(V, I, M))$ are given as

$$\begin{aligned} UB(\rho(L, U, A)) &= \sum_{k=1}^g U_k - \sum_{k=1}^g L_k \\ &< g + \sum_{k=1}^g a(\tau_{s(k)}) \\ &\quad + \sum_{k=1}^g \frac{a(\tau_{s(k)}) \alpha_{k,t_1^*(k),t_2^*(k)}}{\left\lfloor \frac{t_2^*(k) - t_1^*(k)}{d(\tau_{s(k)})} \right\rfloor^2} \end{aligned}$$

$$UB(\gamma(L, U, A)) = \frac{\sum_{k=1}^g U_k}{\sum_{k=1}^g L_k}.$$

VIII. EXPERIMENTAL RESULTS

We use two well-known benchmarks [20]: the differential equation (DE) and the elliptic wave filter (EWF) to test our bounds and algorithms. Fig. 2 is the generic component library which is used for the experiments. For each benchmark we use different time constraints to generate different test cases. For each test case with a given time constraint, we select the fastest component in the component library to compute the ASAP time and ALAP time for each operation in the DFG and take them as the intervals.

Tables I and II show the lower bounds, upper bounds, the upper bound of the absolute errors, and the upper bound of the performance ratios of $1 - n\omega(V, I, M)$ for the DE and the EWF, respectively. It can be seen that the upper bound of the overall performance ratio is ranged 1.00–1.97 for the DE, and 1.03–1.69 for the EWF, which show that both our upper bound and lower bound are very tight. The CPU time for producing each of the bounds is all less than 0.001 s. All experiments are performed on a 350-MHz Pentium II processor PC with 64-M memory and LINUX operating system.

To compare with the traditional $1 - 1\omega(V, I, M)$, we select for each type of operation, a slow component and a fast component to build two types of 1-1 mapping component library, namely, the slow component library and the fast component

```

/*      A generic component library      */
Adder1_40:      area = 40, delay = 1;
Adder2_15:      area = 15, delay = 2;
Adder3_12:      area = 12, delay = 3;
Adder4_10:      area = 10, delay = 4;
Subtractor1_50:  area = 50, delay = 1;
Subtractor2_15: area = 15, delay = 2;
Subtractor3_13: area = 13, delay = 3;
Subtractor4_10: area = 10, delay = 4;
Comparator1_45: area = 45, delay = 1;
Comparator2_15: area = 15, delay = 2;
Comparator3_12: area = 12, delay = 3;
Multiplier1_250: area = 250, delay = 1;
Multiplier2_100: area = 100, delay = 2;
Multiplier3_70: area = 70, delay = 3;
Multiplier4_52: area = 52, delay = 4;
Divisor1_450:   area = 450, delay = 1;
Divisor2_220:   area = 220, delay = 2;
Divisor3_160:   area = 160, delay = 3;
Divisor4_120:   area = 120, delay = 4;

```

Fig. 2. A generic component library.

library, as shown in Figs. 3 and 4, respectively. We call Algorithm 1 to obtain the optimum area cost of $1 - 1\omega(V, I, M)$ for each of the component library on all the test cases. Tables III and IV show the results for the DE and the EWF, respectively. For comparison, we also include the lower bound and upper bound of $1 - n\omega(V, I, M)$ in the tables. The symbol “NA” in the tables refers to as “no solution.” It can be seen that the optimum area costs for both the slow and the fast $1 - 1\omega(V, I, M)$ are all much larger than the upper bounds of $1 - n\omega(V, I, M)$ if there exist solutions, which means that the traditional unicomponent selection will result in surplus area cost comparing

Let $L_k = L(1 - n\omega(V_k, I, M))$, $U_k = U(1 - n\omega(V_k, I, M))$

$$\begin{aligned}
 & UB(\rho(L_k, U_k, A)) \\
 & \begin{cases} < a(\tau_s) + 1 + \frac{a(\tau_s)\alpha_{k,t_1^*,t_2^*}}{\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]^2} & (t_2^* - t_1^*) \% d(\tau_s) \neq 0, \quad \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) \neq 0 \\ & \leq a(\tau_s) & (t_2^* - t_1^*) \% d(\tau_s) = 0, \quad \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) \neq 0 \\ & = 0 & (t_2^* - t_1^*) \% d(\tau_s) = 0, \quad \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) = 0 \end{cases} \\
 & UB(\gamma(L_k, U_k, A)) \\
 & \begin{cases} < 1 + \frac{1}{\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]} + \frac{1}{\frac{\alpha_{k,t_1^*,t_2^*}}{\frac{t_2^* - t_1^*}{d(\tau_s)}}} & (t_2^* - t_1^*) \% d(\tau_s) \neq 0, \text{ and } \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) \neq 0 \\ & \leq 1 + \frac{1}{\frac{\alpha_{k,t_1^*,t_2^*}}{\frac{t_2^* - t_1^*}{d(\tau_s)}}} & (t_2^* - t_1^*) \% d(\tau_s) = 0, \text{ and } \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) \neq 0 \\ & = 1 & (t_2^* - t_1^*) \% d(\tau_s) = 0, \text{ and } \alpha_{k,t_1^*,t_2^*} \% \left(\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]\right) = 0 \end{cases} \\
 & UB(R^\infty(L_k)) = UB(R^\infty(U_k)) \\
 & \begin{cases} < 1 + \frac{1}{\left[\frac{t_2^* - t_1^*}{d(\tau_s)}\right]} \leq 2 & (t_2^* - t_1^*) \% d(\tau_s) \neq 0 \\ & = 1 & (t_2^* - t_1^*) \% d(\tau_s) = 0 \end{cases}
 \end{aligned}$$

TABLE I
LOWER/UPPER BOUNDS OF
 $1 - n\omega(V, I, M)$ FOR DE

T	LB	UB	UB(ρ)	UB(γ)
4	375	580	5	1.00
5	340	360	20	1.06
6	271	345	74	1.27
7	226	245	19	1.08
8	195	245	50	1.26
9	170	245	75	1.44
10	152	245	93	1.61
11	136	245	109	1.80
12	124	245	121	1.97
13	114	145	31	1.27

TABLE II
LOWER/UPPER BOUNDS OF $1 - n\omega(V, I, M)$ FOR EWF

T	LB	UB	UB(ρ)	UB(γ)
14	578	605	27	1.05
15	265	275	10	1.04
16	218	275	57	1.26
17	187	260	73	1.39
18	173	260	87	1.50
19	160	245	85	1.53
20	149	245	96	1.64
21	140	145	5	1.03
22	133	145	12	1.09
23	124	145	21	1.17
24	119	145	26	1.22
25	112	145	33	1.29
26	107	130	23	1.21
27	102	130	28	1.27
28	98	130	32	1.32
29	94	130	36	1.38
30	90	130	40	1.44
31	88	130	42	1.48
32	85	130	45	1.52
33	82	130	48	1.59
34	79	130	51	1.65
35	77	130	53	1.69

```

/*    A slow component library    */
Adder3_12:      area = 12, delay = 3;
Subtractor3_13: area = 13, delay = 3;
Comparator3_12: area = 12, delay = 3;
Multiplier3_70: area = 70, delay = 3;

```

Fig. 3. A slow component library.

```

/*    A fast component library    */
Adder1_40:      area = 40, delay = 1;
Subtractor1_50: area = 50, delay = 1;
Comparator1_45: area = 45, delay = 1;
Multiplier1_250: area = 250, delay = 1;

```

Fig. 4. A fast component library.

to the multicomponent selection. The slow $1 - n\omega(V, I, M)$ often fails to give solutions for the cases which have critical time constraints, while the fast $1 - n\omega(V, I, M)$ often produces more costly results although it can implement all the cases. It should be noted that the upper bound of $1 - n\omega(V, I, M)$

TABLE III
OPTIMUM $1 - n\omega(V, I, M)$ VERSUS LOWER/UPPER BOUNDS OF
 $1 - n\omega(V, I, M)$ FOR DE

T	Slow	Fast	LB	UB
4	NA	635	375	580
5	NA	635	340	360
6	NA	635	271	345
7	NA	385	226	245
8	NA	385	195	245
9	NA	385	170	245
10	NA	385	152	245
11	NA	385	136	245
12	177	385	124	245
13	177	385	114	145

TABLE IV
OPTIMUM $1 - n\omega(V, I, M)$ VERSUS LOWER/UPPER BOUNDS OF
 $1 - n\omega(V, I, M)$ FOR EWF

T	Slow	Fast	LB	UB
14	NA	620	578	605
15	NA	370	265	275
16	NA	330	218	275
17	NA	330	187	260
18	NA	330	173	260
19	NA	330	160	245
20	NA	330	149	245
21	NA	330	140	145
22	NA	330	133	145
23	NA	330	124	145
24	NA	330	119	145
25	NA	330	112	145
26	NA	290	107	130
27	NA	290	102	130
28	NA	290	98	130
29	NA	290	94	130
30	NA	290	90	130
31	NA	290	88	130
32	NA	290	85	130
33	NA	290	82	130
34	NA	290	79	130
35	NA	290	77	130

is actually the optimum area cost of $1 - n\omega(V, I, M)$ for the most cost-effective components as stated in Section VI.

To see how large the design space of the component selection would be when the multicomponent selection is considered, and how many inferior component selections could be pruned when the lower bound and upper bound produced in the paper are used, we also conduct another experiment. First we enumerate all the number of possible component selections, which we call the total number of component selections, that the area cost is in the range between 0 (we use 0 as the trivial lower bound) and the area cost of the fastest, i.e., the most expensive implementation. Then we calculate how many of these component selections, which we call the number of effective component selections, whose area cost is within the lower bound and upper bound. We define a pruning ratio, the ratio of the number of effective component selections over the total number of component selections, to characterize the pruning capability of the lower bound and upper bound.

Take the EWF as an example. When $T = 35$, using the component library of Fig. 2, for the area cost between 0 and 290 (290 is the fastest implementation as shown in Table IV), where 0–40 is the area cost for the additions and 0–250 for the multiplications, there are 20 component selections for the additions, 53 for the multiplications, and the total number of component selections is 1060. However, with the lower bound 77 (as shown in Table II) where 23 is the area cost for the additions, 54 for the multiplications, and the upper bound 130 (as shown in Table II) where 30 for the additions, 100 for the multiplications, there are 5 component selections for the additions and 6 for the multiplications as shown below:

$\{Adder2_15, Adder4_10\}, \{Adder2_15, Adder3_12\},$
 $\{Adder2_15, Adder2_15\}, \{Adder3_12, Adder3_12\},$
 $\{Adder4_10, Adder4_10, Adder4_10\},$
 $\{Multiplier2_100\}, \{Multiplier3_70\}, \{Multiplier4_52\},$
 $\{Multiplier5_45\}, \{Multiplier5_45, Multiplier5_45\},$
 $\{Multiplier4_52, Multiplier5_45\}.$

The number of effective component selections is 30, and the pruning ratio for this case is $30/1060 = 0.028$.

That is to say, of all the 1060 component selections, only 30 possible component selections can lead to the optimal solution while all the others are either infeasible (under the lower bound) or suboptimal (above the upper bound). With the help of lower bound and upper bound, we only need to search for 2.8% of the entire design space of multicomponent selections that can find the optimal solution.

Tables V and VI summarize the results for DE and EWF, respectively. In the tables, column II is the total number of the component selections, column III is the number of effective component selections, and column IV gives the pruning ratio. From these figures, we can see that although the whole design space of multicomponent selection is very huge, the number of component selections falls within the lower bound and upper bound is relatively much smaller and often accounts for a very small percentage of the total number of component selections. It shows that by using the lower bound and upper bound, most of the inferior component selection can be pruned and the space of multicomponent selection for searching for the optimal solution is greatly narrowed down.

To test the performance of our lower bound and upper bound on large cases, we also generate some large random DFG's with the number of operations ranged from 500 to 6000 and perform the experiments. Figs. 5–7 show the performance ratios for three selected test cases while Figs. 8–10 depict the pruning ratios, respectively. Fig. 11 draws the profiles of the run time of the algorithm. It can be seen that the performance ratios for the three test cases are all within the range of 1.0 to 1.8 while the pruning ratios are all below 0.0025, which shows that the lower bound and upper bound have very good performance guarantee and strong pruning capability. From the figures, it also can be seen that when the size of the DFG goes larger, both the performance ratio and pruning ratio are getting better while the CPU time keeps increasing almost linearly. The runtime of the algorithm is very fast, even for big DFG with more than 5000 nodes, the CPU time is only about 3.5 s. It shows that our algorithm has very good scalability and runs fast on large cases.

TABLE V
NUMBER OF COMPONENT SELECTIONS AND PRUNING RATIO FOR DE

T	# of total	# of effective	pruning ratio
4	2896520	3	0.000001
5	2896520	42	0.000015
6	2896520	387	0.000134
7	338140	36	0.000106
8	338140	324	0.000958
9	338140	720	0.002129
10	338140	960	0.002839
11	338140	1008	0.002981
12	338140	1056	0.003123
13	338140	48	0.000142

TABLE VI
NUMBER OF COMPONENT SELECTIONS AND PRUNING RATIO FOR EWF

T	# of total	# of effective	pruning ratio
14	162532	140	0.000861
15	18974	30	0.001581
16	5777	564	0.097629
17	5777	320	0.055392
18	5777	460	0.079626
19	5777	21	0.003635
20	5777	88	0.015233
21	5777	6	0.001039
22	5777	30	0.005193
23	5777	48	0.008309
24	5777	52	0.009001
25	5777	56	0.009694
26	1060	4	0.003774
27	1060	12	0.011321
28	1060	12	0.011321
29	1060	15	0.014151
30	1060	20	0.018868
31	1060	20	0.018868
32	1060	20	0.018868
33	1060	25	0.023585
34	1060	30	0.028302
35	1060	30	0.028302

IX. FURTHER DISCUSSIONS

A. Relations Between $1 - \eta\omega(V, I, M)$ and Multicomponent Selection for Precedence-Constrained Scheduling

In most of the applications of high-level synthesis, the design constraint is given as the precedence constraints between the operations rather than the interval constraint of each operation, it is, thus, desired to see whether the bounds of $1 - \eta\omega(V, I, M)$ are still applicable to the multicomponent selection for precedence constrained scheduling. Similarly to the definition of $1 - \eta\omega(V, I, M)$, we give a formal description to this problem.

Let DFG be a directed acycle graph $G = (V, E)$ where V is the set of operations, and E is the set of precedence relations between operations.

Given a DFG, a component subset $\Lambda = \Gamma \times Z$, and a time constraint T , the *integrated component allocation and precedence constrained scheduling*, denoted by ICAPCS(DFG, T , Λ), is to find a component allocation $\mu : V \rightarrow \Gamma \times Z$ and a precedence constrained scheduling $\Theta : V \rightarrow S$, satisfying:

- 1) *time constraint*: $|S| \leq T$, i.e., for $\forall v_i \in V$, $0 \leq \Theta(v_i) < \Theta(v_i) + d(\mu(v_i)) \leq T$;

Fig. 5. Performance ratio of test case 1.

Fig. 6. Performance ratio of test case 2.

Fig. 7. Performance ratio of test case 3.

- 2) *precedence constraint*: for $\forall v_i, v_j \in V, (v_i, v_j) \in E$, $\Theta(v_j) \geq \Theta(v_i) + d(\mu(v_i))$;
- 3) *resource sharing constraint*: for $\forall v_i, v_j \in V$, if $f(v_i) = f(v_j)$, $\mu(v_i) = \mu(v_j)$, then $S(v_i) \cap S(v_j) = \phi$ where $S(v_i) = \{s \in S \mid \Theta(v_i) \leq s \leq \Theta(v_i) + d(\mu(v_i)) - 1\}$;
- 4) *component constraint*: $\forall \tau \in \Gamma, Z(\Lambda, \tau) \geq \text{Max}_{0 \leq s \leq T} |\{v_i \in V \mid \Theta(v_i) \leq s \leq \Theta(v_i) + d(\mu(v_i)) - 1, \tau(\mu(v_i)) = \tau\}|$.

Given a DFG, a component library M , and a time constraint T , the *component selection for precedence constrained scheduling*, denoted by $\omega(\text{DFG}, T, M)$, is to find a component subset $\Lambda = \Gamma \times Z$, such that $\exists$ an integrated component allocation and precedence constrained scheduling (ICAPCS DFG, T, Λ).

Fig. 8. Pruning ratio of test case 1.

Fig. 9. Pruning ratio of test case 2.

Fig. 10. Pruning ratio of test case 3.

The terms $a(\omega(\text{DFG}, T, M))$, $\omega^*(\text{DFG}, T, M)$, $1 - 1\omega(\text{DFG}, T, M)$, and $1 - n\omega(\text{DFG}, T, M)$ are similarly defined to denote the area cost of component selection, the optimal component selection, the unicomponent selection and the multi component selection for the precedence constrained scheduling, respectively. We simply use $\omega(V, I, M)$ or $\omega(\text{DFG}, T, M)$ to denote the interval scheduling or the precedence constrained scheduling when we disregard the type of the component selection, and use $1 - n\omega$ or $1 - 1\omega$ to denote the multi component selection or the unicomponent selection when we do not concern about the type of the scheduling.

It can be seen that $\omega(V, I, M)$ is a partial relaxation of $\omega(\text{DFG}, T, M)$, any solution to $\omega(\text{DFG}, T, M)$ still satisfy $\omega(V, I, M)$, hence, $\text{LB}(\omega(V, I, M)) \leq \omega^*(V, I, M) \leq \omega^*(\text{DFG}, T, M)$.

As a result, $\text{LB}(1 - 1\omega(V, I, M)) \leq 1 - 1\omega^*(V, I, M) \leq 1 - 1\omega^*(\text{DFG}, T, M)$ and $\text{LB}(1 - n\omega(V, I, M)) \leq 1 - n\omega^*(V, I, M) \leq 1 - n\omega^*(\text{DFG}, T, M)$. That is to say, the lower bound of $1 - n\omega(V, I, M)$ is still a valid lower bound of $1 - n\omega(\text{DFG}, T, M)$.

However, the relation of the upper bounds is reversed, i.e., the upper bound of $1 - n\omega(\text{DFG}, T, M)$ is an upper bound of $1 -$

Fig. 11. CPU run time profile.

$n\omega(V, I, M)$, it is because $\omega^*(V, I, M) \leq \text{UB}(\omega(V, I, M)) \leq \omega^*(\text{DFG}, T, M) \leq \text{UB}(\omega(\text{DFG}, T, M))$. That is the upper bound of $1 - n\omega(V, I, M)$ is not valid for $1 - n\omega(\text{DFG}, T, M)$.

On the other hand, as $1 - 1\omega$ is a special case of $1 - n\omega$, the optimum solution of $1 - 1\omega$ is also a feasible solution of $1 - n\omega$, i.e., $1 - n\omega^* \leq 1 - 1\omega^*$. As a result, $\text{LB}(1 - n\omega(\text{DFG}, T, M)) \leq \text{LB}(1 - 1\omega(\text{DFG}, T, M))$.

It should be noted that a lower bound or the optimum solution value of the traditional unicomponent selection, even for precedence constrained scheduling, does not give a lower bound of multicomponent selection for precedence constrained scheduling. The reason is that although $\omega^*(V, I, M) \leq \omega^*(\text{DFG}, T, M)$, as $1 - n\omega^* \leq 1 - 1\omega^*$, there is no dominating relation between $1 - 1\omega^*(V, I, M)$ and $1 - n\omega^*(\text{DFG}, T, M)$, while for both with precedence constrained scheduling, the relation is reversed, $\text{LB}(1 - n\omega(\text{DFG}, T, M)) \leq \text{LB}(1 - 1\omega(\text{DFG}, T, M))$.

In summary, for the multicomponent selection, the lower bound of the interval scheduling is still a lower bound of the precedence constrained scheduling while the upper bound is on the contrary. But for the traditional unicomponent selection, the lower bound, no matter with the interval scheduling or the precedence constrained scheduling, does not give a lower bound for the multicomponent selection.

B. The Two Assumptions

There are two assumptions that are taken throughout the paper as stated in the first section. Here, we discuss whether the bounds are still valid when the assumptions are removed.

1) *The Area Costs of the Storage and Interconnection:* The assumption that the area costs of the storage and interconnection are ignorable comparing with that of the functional components was taken by most of the previous estimation methods [4]–[11]. With this assumption, the component binding which is also a computationally intensive problem can be relaxed during scheduling and component selection. This simplifies the computation of the lower bound but does not affect the validity of the lower bound, as eliminating the assumption can lead only to a tighter lower bound.

2) *Complex Multifunctional Component:* There exist cases in high-level synthesis that it produces much cheaper designs by use of complex multifunctional component such as ALU that can perform multiple different types of functions. The

gives rise to a new class of heterogeneous multifunctional component selection problem which is more generalized and difficult than the multicomponent selection. Although the multifunctional component can be treated as a single functional component which is the way that most of the traditional methods did, however, as the complex multifunctional component introduces larger design space that has to be fully explored, trivially treating it as a single functional component will not give the valid lower/upper bound for this problem. This is a new class of problem that needs to be further investigated and beyond the scope of this paper.

X. CONCLUSION

This research work intends to draw attentions on component selection in the process of estimating the lower bound and upper bound of scheduling. We show that as the traditional unicomponent selection restricted the freedom of the component selection so that the design space of component selection was not fully explored, it resulted in the local lower/upper bounds and, hence, the suboptimal designs produced. Different from the previous unicomponent selection methods, we compute, in this paper, the lower/upper bounds which consider the scheduling and the multicomponent selection simultaneously. We present a very interesting and important result that both the lower bound and upper bound of $1 - n\omega(V, I, M)$ are obtained on the most cost-effective components which have the minimum area-delay products. This property greatly narrows down the large design space of the multicomponent selection so that both the lower bound and upper bound of multicomponent selection can be calculated efficiently. Although the bounds look very natural, the derivation and proof are not trivial. By introducing an efficient ILP model and a surrogate relaxation technique, we obtain an optimum surrogate lower bound of $1 - n\omega(V_k, I, M)$ which has the worst asymptotic performance ratio less than 2. We also derive an upper bound of $1 - n\omega(V_k, I, M)$ with same asymptotic performance ratio which turns out to be the optimum solution value of the traditional unicomponent selection. Intensive experimental results also validate our theoretical results.

Although the paper uses the interval scheduling, the traditional partial relaxation to the precedence constrained scheduling, to derive the lower bound and upper bound of multicomponent selection, it has been shown that the lower bound is still applicable to the precedence constrained scheduling. As

the future work, we will further investigate the problem of multi-component selection integrated with the precedence constrained scheduling. Meanwhile, we will extend the lower bound and upper bound on the more generalized heterogeneous multifunctional component selection problem, as well as consider the area costs of storage and interconnections, various realistic behavioral structures such as pipelining, loop and condition in the estimation.

APPENDIX

Assume a, b, c, d be positive integers, we have the following lemmas.

Lemma 4:

$$\left\lfloor \frac{a}{b} \right\rfloor \leq \frac{a}{b} \leq \left\lceil \frac{a}{b} \right\rceil \leq \left\lfloor \frac{a}{b} \right\rfloor + 1, \quad 0 \leq a \% b \leq b - 1.$$

Proof: Obvious. ■

Lemma 5:

$$a - a \% b = b \left\lfloor \frac{a}{b} \right\rfloor \leq a.$$

Proof: As

$$a = b \left\lfloor \frac{a}{b} \right\rfloor + a \% b. \quad \blacksquare$$

Lemma 6:

$$a - a \% b \leq b \left\lfloor \frac{a}{b} \right\rfloor \leq b \left(1 + \left\lfloor \frac{a}{b} \right\rfloor \right) = b + a - a \% b.$$

Proof:

$$\begin{aligned} a - a \% b &= b \left\lfloor \frac{a}{b} \right\rfloor \leq b \left\lceil \frac{a}{b} \right\rceil \\ &\leq b \left(1 + \left\lfloor \frac{a}{b} \right\rfloor \right) = b + a - a \% b. \end{aligned} \quad \blacksquare$$

Lemma 7:

$$\frac{a - a \% b}{b} \leq \left\lceil \frac{a}{b} \right\rceil \leq 1 + \left\lfloor \frac{a}{b} \right\rfloor.$$

Proof: From Lemma 6. ■

Lemma 8:

$$c \left\lceil \frac{a}{b} \right\rceil - \left\lceil \frac{ca}{b} \right\rceil \leq c \left(1 - \frac{a \% b}{b} \right).$$

Proof: As

$$\begin{aligned} c \left\lceil \frac{a}{b} \right\rceil &\leq c \left(1 + \left\lfloor \frac{a}{b} \right\rfloor \right) = c + c \left\lfloor \frac{a}{b} \right\rfloor, \quad \text{and} \\ \left\lceil \frac{ca}{b} \right\rceil &= \left\lceil \frac{c(b \left\lfloor \frac{a}{b} \right\rfloor + a \% b)}{b} \right\rceil = c \left\lfloor \frac{a}{b} \right\rfloor + \left\lceil \frac{c(a \% b)}{b} \right\rceil \\ &\geq c \left\lfloor \frac{a}{b} \right\rfloor + \frac{c(a \% b)}{b}. \end{aligned} \quad \blacksquare$$

Lemma 9:

$$\frac{1}{\left\lfloor \frac{a}{b} \right\rfloor} - \frac{1}{\frac{a}{b}} = \frac{a \% b}{a \left\lfloor \frac{a}{b} \right\rfloor}.$$

Proof:

$$\frac{1}{\left\lfloor \frac{a}{b} \right\rfloor} - \frac{1}{\frac{a}{b}} = \frac{\frac{a}{b} - \left\lfloor \frac{a}{b} \right\rfloor}{\frac{a}{b} \left\lfloor \frac{a}{b} \right\rfloor} = \frac{a - b \left\lfloor \frac{a}{b} \right\rfloor}{a \left\lfloor \frac{a}{b} \right\rfloor} = \frac{a \% b}{a \left\lfloor \frac{a}{b} \right\rfloor}. \quad \blacksquare$$

Lemma 10:

$$\left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{c}{\frac{a}{b}} \right\rceil \leq 1 + \frac{c(a \% b)}{a \left\lfloor \frac{a}{b} \right\rfloor}.$$

Proof:

$$\left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{c}{\frac{a}{b}} \right\rceil \leq 1 + \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} - \frac{c}{\frac{a}{b}} = 1 + \frac{c(a \% b)}{a \left\lfloor \frac{a}{b} \right\rfloor}. \quad \blacksquare$$

Lemma 11:

$$d \frac{\left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil}{\left\lceil \frac{dc}{\frac{a}{b}} \right\rceil} \begin{cases} < 1 + \frac{1}{\left\lfloor \frac{a}{b} \right\rfloor} + \frac{1}{\frac{c}{a}} & a \% b \neq 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0 \\ \leq 1 + \frac{1}{\frac{c}{a}} & a \% b = 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0 \\ = 1 & a \% b = 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) = 0. \end{cases}$$

Proof: When $a \% b \neq 0, c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0$

$$\begin{aligned} \frac{d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil}{\left\lceil \frac{dc}{\frac{a}{b}} \right\rceil} &\leq \frac{d \left(\left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil + 1 \right)}{\frac{dc}{\frac{a}{b}}} = \frac{\frac{a}{b}}{\left\lfloor \frac{a}{b} \right\rfloor} + \frac{a}{c} \\ &\leq \frac{\left\lfloor \frac{a}{b} \right\rfloor + 1}{\left\lfloor \frac{a}{b} \right\rfloor} + \frac{1}{\frac{c}{a}} = 1 + \frac{1}{\left\lfloor \frac{a}{b} \right\rfloor} + \frac{1}{\frac{c}{a}}. \end{aligned}$$

When $a \% b = 0, c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0$

$$\frac{d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil}{\left\lceil \frac{dc}{\frac{a}{b}} \right\rceil} \leq \frac{d \left(\frac{c}{\frac{a}{b} + 1} \right)}{\frac{dc}{\frac{a}{b}}} = 1 + \frac{a}{b} = 1 + \frac{1}{\frac{c}{a}}.$$

When $a \% b = 0, c \% (\left\lfloor \frac{a}{b} \right\rfloor) = 0$

$$\frac{d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil}{\left\lceil \frac{dc}{\frac{a}{b}} \right\rceil} = \frac{d \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor}}{\frac{dc}{\frac{a}{b}}} = 1. \quad \blacksquare$$

Lemma 12:

$$d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{dc}{\frac{a}{b}} \right\rceil \begin{cases} < d + 1 + \frac{dc}{\left\lfloor \frac{a}{b} \right\rfloor^2} & a \% b \neq 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0 \\ \leq d & a \% b = 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0 \\ = 0 & a \% b = 0, \quad c \% (\left\lfloor \frac{a}{b} \right\rfloor) = 0. \end{cases}$$

Proof: When $a \% b \neq 0, c \% (\left\lfloor \frac{a}{b} \right\rfloor) \neq 0$

$$\begin{aligned} d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{dc}{\frac{a}{b}} \right\rceil &= d \left\lceil \frac{c}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{dc}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil + \left\lceil \frac{dc}{\left\lfloor \frac{a}{b} \right\rfloor} \right\rceil - \left\lceil \frac{dc}{\frac{a}{b}} \right\rceil \\ &\leq d \left(1 - \frac{c \% (\left\lfloor \frac{a}{b} \right\rfloor)}{\left\lfloor \frac{a}{b} \right\rfloor} \right) + 1 + \frac{a \% b}{a \left\lfloor \frac{a}{b} \right\rfloor} dc. \end{aligned}$$

As

$$\begin{aligned} \frac{c \% (\left\lfloor \frac{a}{b} \right\rfloor)}{\left\lfloor \frac{a}{b} \right\rfloor} &\geq 0, \quad \text{and} \quad a \% b < b \\ d \left(1 - \frac{c \% (\left\lfloor \frac{a}{b} \right\rfloor)}{\left\lfloor \frac{a}{b} \right\rfloor} \right) + 1 + \frac{a \% b}{a \left\lfloor \frac{a}{b} \right\rfloor} dc &< d + 1 + \frac{b}{a \left\lfloor \frac{a}{b} \right\rfloor} dc = d + 1 + \frac{dc}{\frac{a}{b} \left\lfloor \frac{a}{b} \right\rfloor} \\ &\leq 1 + d + \frac{cd}{\left\lfloor \frac{a}{b} \right\rfloor^2}. \end{aligned}$$

When $a\%b = 0, c\%(\lfloor a/b \rfloor) \neq 0$

$$\begin{aligned} d \left\lceil \frac{c}{\lfloor \frac{a}{b} \rfloor} \right\rceil - \left\lceil \frac{dc}{b} \right\rceil &= d \left\lceil \frac{c}{\lfloor \frac{a}{b} \rfloor} \right\rceil - \left\lceil \frac{dc}{\lfloor \frac{a}{b} \rfloor} \right\rceil \\ &\leq d \left(1 - \frac{c\%(\lfloor \frac{a}{b} \rfloor)}{\lfloor \frac{a}{b} \rfloor} \right) \leq d. \end{aligned}$$

When $a\%b = 0, c\%(\lfloor a/b \rfloor) = 0$

$$\begin{aligned} d \left\lceil \frac{c}{\lfloor \frac{a}{b} \rfloor} \right\rceil - \left\lceil \frac{dc}{b} \right\rceil &= d \left\lceil \frac{c}{\lfloor \frac{a}{b} \rfloor} \right\rceil - \left\lceil \frac{dc}{\lfloor \frac{a}{b} \rfloor} \right\rceil \\ &= d \frac{c}{\lfloor \frac{a}{b} \rfloor} - \frac{dc}{b} = 0. \quad \blacksquare \end{aligned}$$

Proof of Theorem 12: $UB(\rho(L_k, U_k, A))$, and $UB(\gamma(L_k, U_k, A))$ can be immediately obtained from Lemma 11 and 12.

$UB(R^\infty(L_k)), UB(R^\infty(U_k))$ can be seen as follows:
When

$$\frac{\alpha_{k,t_1^*,t_2^*}}{\frac{t_2^* - t_1^*}{d(\tau_s)}}$$

is sufficiently large

$$\frac{1}{\frac{\alpha_{k,t_1^*,t_2^*}}{\frac{t_2^* - t_1^*}{d(\tau_s)}}} \rightarrow 0$$

both U_k and L_k is sufficiently large and, hence, OPT_k is also sufficiently large, i.e., $\geq N_0$, as $L_k \leq OPT_k \leq U_k$. Therefore, by Lemma 11, $UB(R^\infty(L_k)) = UB(R^\infty(U_k)) < UB(\gamma(L_k, U_k, A))$

$$\begin{cases} < 1 + \frac{1}{\left\lceil \frac{t_2^* - t_1^*}{d(\tau_s)} \right\rceil} \leq 2 & (t_2^* - t_1^*) \% d(\tau_s) \neq 0 \\ = 1 & (t_2^* - t_1^*) \% d(\tau_s) = 0. \quad \blacksquare \end{cases}$$

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable detailed comments and helpful suggestions.

REFERENCES

- [1] R. Jain, A. C. Parker, and N. Park, "Predicting area-time tradeoffs for pipelined designs," in *Proc. DAC*, 1987, pp. 35–41.
- [2] —, "Predicting system-level area and delay for pipelined and non-pipelined designs," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 955–965, Aug. 1992.
- [3] A. Sharma and R. Jain, "Estimating architectural resources and performance for high-level synthesis applications," in *Proc. DAC*, 1993, pp. 355–360.
- [4] J. M. Rabaey and M. Potkonjak, "Estimating implementation bounds for real time DSP application specific circuits," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 669–683, 1994.
- [5] Y. Hu, A. Ghouse, and B. S. Carlson, "Improved lower bounds for the scheduling optimization problem," in *Proc. ISCAS*, 1994, pp. 295–298.
- [6] —, "Lower bounds on the iteration time and the number of resources for functional pipelined data flow graphs," in *Proc. ICCD*, 1993, pp. 21–24.
- [7] S. Y. Ohm, F. J. Kurdahi, and N. Dutt, "Comprehensive lower bound estimation from behavioral descriptions," in *Proc. ICCAD*, 1994, pp. 182–187.

- [8] —, "A unified lower bound estimation technique for high-level synthesis," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 458–471, 1997.
- [9] M. Rim and R. Jain, "Lower-bound performance estimation for the high-level synthesis scheduling problem," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 451–458, 1994.
- [10] M. Langevin and E. Cerny, "A recursive technique for computing lower-bound performance of schedules," *ACM Trans. Design Automation Electron. Syst.*, vol. 1, no. 4, pp. 443–456, 1996.
- [11] S. Chaudhuri and R. A. Walker, "Computing lower bounds on functional units before scheduling," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 273–279, June 1996.
- [12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
- [13] J. D. Ullman, "NP-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, pp. 384–393, 1975.
- [14] C. L. Liu, *An Introduction to Discrete Mathematics*. New York: McGraw-Hill, 1985.
- [15] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: Wiley, 1989.
- [16] S. Martello and P. Toth, *Knapsack Problems Algorithm and Compute Implementations*. New York: Wiley, 1990.
- [17] G. B. Dantzig, "Discrete variable extremum problems," *Operation Res.*, vol. 5, pp. 266–277, 1957.
- [18] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management Sci.*, vol. 27, pp. 1–18, 1981.
- [19] F. Glover, "Surrogate constraint duality in mathematical programming," *Operations Res.*, vol. 23, pp. 434–451, 1975.
- [20] Collaborative Benchmarking Lab., Dept. Comput. Sci., North Carolina State Univ., Raleigh, NC. HLSynth'92 Benchmarks. [Online]. Available: HTTP: http://www.cbl.ncsu.edu/CBL_Docs/hls92.html
- [21] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 375–395, 1984.

Zhao-Xuan Shen (M'97) received the B.Sc. degree in computer science from Zhejiang University, Hang Zhou, China, and the M.Eng. degree in electrical and electronic engineering from Nanyang Technological University, Singapore.

From August 1987 to January 1994, he was a Research Associate at Key Laboratory of CAD & Graphics, Chinese Academy of Sciences, Beijing, China. During 1996–1999, he worked as a Senior Engineer at Institute of High Performance Computing, Singapore, developing parallel optimization algorithms for very large scale integration placement, floorplanning, and high-level synthesis. He is now with Arcadia Design systems Inc., San Jose, CA, developing next generation high-performance data-path placement tools. His research interests include high-level synthesis and layout synthesis. He has published numerous papers in these areas.

Mr. Shen received the First Class Science & Technology Progressing Award from Chinese Academy of Sciences in 1991 for his outstanding work in logic partitioning and circuit simulation.

Ching Chuen Jong received the B.Sc.(Eng.) degree in electronics with computer science and the Ph.D. degree in electronic engineering from Queen Mary College, University of London, London, U.K., in 1983 and 1988, respectively.

From July 1987 to October 1990, he worked in the area of high-level synthesis of digital systems first in the University of Southampton, Southampton, U.K., and then in Racal Research Limited, U.K. In 1991, he joined the Nanyang Technological University, Singapore, as a Faculty Member. He is now an Associate Professor in the Division of Circuits and Systems, School of Electrical and Electronic Engineering. His technical interests include high-level synthesis, ASIC design and fast-prototyping of digital designs.

Dr. Jong is a Chartered Engineer, a member of the Institute of Electrical Engineering (IEE) and a member of BCS.