SUMMARY The earlier the stage where we perform low power design, the higher the dynamic power reduction we achieve. In this paper, we focus on reducing switching activity in high-level synthesis, especially, in the problem of functional module binding, bus binding or register binding. We propose an effective low power bus binding algorithm based on the table decomposition method, to reduce switching activity. The proposed algorithm is based on the decomposition of the original problem into sub-problems by exploiting the optimal substructure. As a result, it finds an optimal or close-to-optimal binding solution with less computation time. Experimental results show the proposed method obtains a solution 2.3–22.2% closer to optimal solution than one with a conventional heuristic method, 8.0–479.2 times faster than the optimal one (at a threshold value of 1.0E+9).

Key words: low power, bus binding, switching activity, table decomposition, optimal substructure

1. Introduction

Mobile devices, such as hand-held phones, digital cameras, mp3 players, etc., are becoming increasingly more popular. The noticeable feature of a mobile device is that it is powered by a small-sized battery having limited electric capacity. Therefore, it is necessary to minimize the dissipated power of chips embedded in the mobile devices.

The total power dissipated in CMOS circuit consists of dynamic power, short-circuit power, leakage power, and static power. Among them, dynamic power contributes the dominant part of average power. Dynamic power is calculated as follows [1]:

$$P_{\text{dyn}} = P_{\text{trans}} \cdot C_L \cdot V_{dd}^2 \cdot f_{\text{clock}}$$

where $P_{\text{trans}}$ is the probability of an output transition, $C_L$ is the load capacitance, $V_{dd}$ is the supply voltage, and $f_{\text{clock}}$ is the system clock frequency. Due to the quadratic dependency of power on supply voltage, decreasing the supply voltage is highly effective in reducing dynamic power consumption, but at the expense of an increase in circuit performance. Techniques to reduce supply voltage by using multiple voltages have been proposed in [2]–[4].

On the other hand, reducing output transition also

Manuscript revised July 17, 2010.

††The author is with the System LSI Division, Samsung Electronics Co. Ltd., Yongin, Korea, and with Sungkyunkwan University, Suwon, Korea.

Copyright © 2011 The Institute of Electronics, Information and Communication Engineers
KIM and CHO: LOW POWER BUS BINDING EXPLOITING OPTIMAL SUBSTRUCTURE

Between data transfers an average of 7.50 bit lines out of 16 possible toggles be-

Table 1 Switching activity matrix of DFG in Fig. 1.

<table>
<thead>
<tr>
<th>x</th>
<th>dx</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>u1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.00</td>
<td>7.50</td>
<td>7.51</td>
<td>7.49</td>
<td>7.46</td>
<td>7.50</td>
<td>7.83</td>
<td>7.86</td>
<td>7.76</td>
<td>8.01</td>
</tr>
<tr>
<td>b</td>
<td>7.50</td>
<td>0.00</td>
<td>7.50</td>
<td>7.51</td>
<td>7.51</td>
<td>7.51</td>
<td>7.84</td>
<td>7.86</td>
<td>7.74</td>
<td>7.51</td>
</tr>
<tr>
<td>c</td>
<td>7.51</td>
<td>7.50</td>
<td>0.00</td>
<td>7.49</td>
<td>7.50</td>
<td>8.26</td>
<td>7.83</td>
<td>7.86</td>
<td>6.24</td>
<td>6.12</td>
</tr>
<tr>
<td>d</td>
<td>7.51</td>
<td>7.51</td>
<td>7.49</td>
<td>0.00</td>
<td>7.49</td>
<td>8.00</td>
<td>5.11</td>
<td>7.84</td>
<td>7.76</td>
<td>7.00</td>
</tr>
<tr>
<td>e</td>
<td>7.49</td>
<td>7.51</td>
<td>7.50</td>
<td>7.49</td>
<td>0.00</td>
<td>7.84</td>
<td>5.11</td>
<td>8.00</td>
<td>7.50</td>
<td>7.00</td>
</tr>
<tr>
<td>f</td>
<td>7.50</td>
<td>7.50</td>
<td>8.26</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>7.00</td>
<td>7.79</td>
</tr>
<tr>
<td>g</td>
<td>7.83</td>
<td>7.84</td>
<td>7.83</td>
<td>7.84</td>
<td>5.11</td>
<td>8.00</td>
<td>7.94</td>
<td>5.11</td>
<td>8.00</td>
<td>7.51</td>
</tr>
<tr>
<td>h</td>
<td>7.83</td>
<td>7.84</td>
<td>7.83</td>
<td>7.84</td>
<td>7.83</td>
<td>7.84</td>
<td>7.84</td>
<td>7.84</td>
<td>7.84</td>
<td>7.84</td>
</tr>
<tr>
<td>i</td>
<td>7.36</td>
<td>7.74</td>
<td>8.26</td>
<td>7.75</td>
<td>8.00</td>
<td>7.01</td>
<td>7.74</td>
<td>8.00</td>
<td>0.00</td>
<td>7.50</td>
</tr>
<tr>
<td>j</td>
<td>8.01</td>
<td>7.84</td>
<td>8.24</td>
<td>8.00</td>
<td>7.50</td>
<td>8.49</td>
<td>8.00</td>
<td>7.84</td>
<td>5.00</td>
<td>8.01</td>
</tr>
<tr>
<td>k</td>
<td>6.99</td>
<td>8.13</td>
<td>8.12</td>
<td>8.12</td>
<td>8.00</td>
<td>7.75</td>
<td>8.13</td>
<td>8.00</td>
<td>8.01</td>
<td>8.01</td>
</tr>
<tr>
<td>l</td>
<td>8.00</td>
<td>7.50</td>
<td>7.51</td>
<td>7.51</td>
<td>7.49</td>
<td>7.50</td>
<td>7.83</td>
<td>7.86</td>
<td>7.76</td>
<td>8.01</td>
</tr>
<tr>
<td>m</td>
<td>8.00</td>
<td>7.49</td>
<td>8.26</td>
<td>8.00</td>
<td>7.83</td>
<td>7.83</td>
<td>7.83</td>
<td>7.86</td>
<td>7.84</td>
<td>7.84</td>
</tr>
<tr>
<td>n</td>
<td>6.88</td>
<td>8.15</td>
<td>8.12</td>
<td>8.13</td>
<td>7.50</td>
<td>7.87</td>
<td>8.13</td>
<td>7.76</td>
<td>8.00</td>
<td>8.01</td>
</tr>
</tbody>
</table>

Fig. 1 Data flow graph of differential equation solver.

A schedule determines the set of data transfers to be executed in each clock cycle. However, it does not specify the bus on which a data transfer will take place. Bus binding assigns data transfers to buses for each control step (a control step will be described later).

Lyuh and Kim [7] used a probabilistic model to measure the switching activity on a bus since the exact patterns of input data streams are usually unknown in most cases. Let $SA(x, y)$ be the average number of bit transitions (i.e., Hamming distance) when data transfers $x$ and $y$ are successively implemented on the bus. The value of $SA(x, y)$ for every pair of data transfers in the DFG can be obtained by repeated simulation of the DFG. The signal values of all data transfers in a scheduling function $S$ is determined by the maximum number of variables located in one control step, e.g., ones in control step 1. In Fig. 2, a black circle denotes a variable that occupies a bus, and a white circle denotes a variable that does not exist in a bus. Note that two variables do not exist in buses at control steps 4 and 5.

In addition, a dotted line implies possible binding between two adjacent variables. The behavior of possible bindings between adjacent variables becomes complicated due to the empty nodes at control steps 4 and 5. The number of bit-lines of each bus is the same as the bit-width of each variable.

Let $SA(x, y)$ denote the expected number of bit lines on bus $k$ that toggle when data transfers $x$ and $y$ are successively implemented on the bus. Then, the problem we want to solve is to bind the data transfers to buses to minimize the quantity

$$TSA = \sum_{(k \notin \text{buses})} \sum_{(x \rightarrow y \text{ transitions of } k)} SA(x, y).$$

Based on the above discussion, our bus binding problem can be formulated as follows:

A scheduled $DFG = (O, V, C, S_f)$ consists of:

1. A finite set of operators, denoted $O = \{o_1, o_2, \ldots, o_p\}$.
2. A finite set of variables of operators, denoted $V = \{v_1, v_2, \ldots, v_q\}$.
3. A finite set of control steps, denoted $C = \{c_1, c_2, \ldots, c_r\}$.
4. A scheduling function $S_f : O \rightarrow C$, where $S(o_i) = c_j$ denotes that an operator corresponding to $o_i \in O$ is scheduled at control step $c_j$.

If an operator $o_i$ is scheduled at control step $c_j$ by scheduling function $S(o_i) = c_j$, the variables $u_k$ that are operands to the operator $o_i$ are located at control step $c_j$. Bus binding is performed for this scheduled DFG. Let

Figure 2 shows the bus binding model we adopt for the low power binding problem for DFG (Fig. 1). Its example for a scheduling and bus binding is shown in Fig. 3. We assume that the number of buses given is four. The number of buses corresponds to the maximum number of variables located in one control step, e.g., ones in control step 1. A scheduling function $S_f$ is implemented on a bus. The value of $S_f(x, y)$ is the average of the Hamming distances between primary input signals in the DFG. The value of $SA(x, y)$ is set to be the average of the Hamming distances between $x$ and $y$.

Figure 1 shows a scheduled DFG of a differential equation solver where variables $x'$ and $y'$ are cyclic variables, and are denoted as $x$ and $y$ in the next iteration instance of the loop, respectively. The bit width of each variable is 16. Table 1 shows the values for the data transfers in Fig. 1, which is obtained from simulating 100,000 random inputs to the DFG. For example, $SA(u, dx) = 7.50$ indicates that there is an average of 7.50 bit lines out of 16 possible toggles between data transfers $u$ and $dx$.
Bus binding problem is summarized from this discussion as follows:

**Low power bus binding problem**

- **Input:** A scheduled DFG \((O, V, C, S_f)\)
- **Output:** Bus binding solution with minimum TSA
- **Bus Binding:** \(M_f : V \times C \rightarrow B \times C\)

### 3. Previous Work

An optimal method finds an exact solution with small-sized instances. Chang and Pedram [5] proposed a technique to reduce power consumption during the binding of functional units. They formulated the problem as a max-cost multicommodity flow problem and solved it optimally. However, the low power binding problem is conjectured being NP-hard, and therefore, the computation time should increase exponentially with the problem size.

In contrast, a heuristic method finds a close-to-optimal solution with less computation time. Choi and Kim [8] proposed an efficient binding algorithm for power optimization in high-level synthesis. They exploited the property of efficient flow computations in a network, so it is applicable to practical designs while producing near-optimal results. Xing and Jong [9] proposed a look-ahead synthesis technique with backtracking to reduce switching activity in low power high-level synthesis, effectively reducing the probability for the solutions to fall into local minimum.

### 4. Our Algorithm

In this section, we present our low power binding algorithm based on the table decomposition method.

#### 4.1 Proposed Table Decomposition Method

Our table decomposition method is based on the divide-and-conquer approach. We define the optimal substructure of the binding problem to utilize the advantage of finding an optimal binding solution with less computation time. The optimal substructure of a problem is defined as a substructure whose optimal solution is contained in the globally optimal solution to the original problem [10].

The proposed algorithm is described in Fig. 4. Using the property of the optimal substructure, the proposed method can successfully decompose the original problem into sub-problems by exploiting the fixed boundary nodes between two adjacent sub-problems.

The concept of optimal substructure adopted in this paper is analogous to that of finding a shortest-path between
two vertices. Shortest path algorithms typically rely on the property that a shortest path between two vertices contains other shortest paths within it.

1) STEP-1 (Line 1): Divide binding table into consecutive and non-consecutive regions.

Figure 5 shows the definition of consecutive and non-consecutive regions in bus binding table. The consecutive region consists of two csteps, all of which have a maximum number of rows. On the other hand, the non-consecutive region has boundary csteps having a maximum number of rows, and internal csteps having less than the maximum number rows.

We decompose the entire binding table into these two kinds of regions, since these two regions have the property of an optimal substructure; we can find an optimal solution independently for these two regions, and combine these optimal solutions in the decomposed substructures simply into a global optimal solution (Refer to the Theorem in Sect. 4.3).

The concept of optimal substructure in our method is applied to only consecutive and non-consecutive regions, not to all possible sub-regions, e.g., the sub-region consisting of cstep 3–4 that does not match either consecutive region or non-consecutive region.

2) STEP-2 (Line 2–8): Find sub-solution for each consecutive and non-consecutive region.

The main idea is to attempt to find an optimal solution independently for each consecutive and non-consecutive region. It is beneficial to apply this approach owing to the optimal substructure of the bus binding problem (Refer to Sect. 4.3). According to the complexity of problem instances, either an optimal or a heuristic method is applied selectively. The problem complexity can be viewed as the following two cases.

1. Case-1: problem instance does not have the optimal substructure that cannot be divided into the subproblems. In this case, we need to resort to using heuristics rather than using the optimal method.

2. Case-2: the problem instance can be divided into the sub-problems, and the complexity of sub-problem may be greater than an appropriate size that can be adequately calculated by the optimal method in the specific machine. Therefore, we should decide an appropriate threshold that indicates the upper bound at which the optimal method can be successfully applied. Therefore, if the complexity of problem (or sub-problem) exceeds this threshold, it is desirable to apply a heuristic method rather than optimal method.

The sequential procedure deciding the appropriate threshold is described in Fig. 6. Initially, the threshold is set to the maximum complexity of sub-problems (how we calculate complexity is explained in Sect. 4.2). For the case where the threshold incurs memory overflow while we execute the optimal method on the sub-problem, then the threshold is decreased to the second greatest complexity. This procedure is repeated until the memory overflow problem does not occur.

Using this sequential threshold decision procedure, the proposed algorithm adaptively chooses either optimal or heuristic method according to the complexity of the original problem and the computing power of the machine. If the complexity is less than a predefined threshold, a conventional optimal method for minimum weight perfect matching in a bipartite graph is applied to the consecutive region,
and an optimal method for the min-cost multi-commodity network flow is applied to the non-consecutive region.

We start by introducing some basic graph terminology referred to [11] to understand minimum weight perfect matching problem:

A graph \( G = (V, E) \) consists of a set \( V \) of vertices and a set \( E \) of pairs of vertices called edges. For an edge \( e = (u, v) \), we say that the endpoints of \( e \) are \( u \) and \( v \); we also say that \( e \) is incident to \( u \) and \( v \). A graph \( G = (V, E) \) is bipartite if the vertex set \( V \) can be partitioned into two sets \( A \) and \( B \) (the bipartition) such that no edge in \( E \) has both endpoints in the same set of the bipartition.

A matching \( M \subseteq E \) is a collection of edges such that every vertex of \( V \) is incident to at most one edge of \( M \). If a vertex \( v \) has no edge of \( M \) incident to it, then \( v \) is said to be exposed (or unmatched). A matching is perfect if no vertex is exposed; that is, a matching is perfect if its cardinality is equal to \( |A| = |B| \).

The example is shown in Fig. 7. The edges \((1,6), (2,7), (5,8), \) and \((3,8)\) form a matching. Vertices \(4, 5, 9, \) and \(10\) are exposed.

The minimum weight perfect matching problem is summarized as follows:

Given a cost \( c_{ij} \) for all \((i, j) \in E\), find a perfect matching of minimum cost where the cost of a matching \( M \) is given by \( c(M) = \sum_{(i,j) \in M} c_{ij} \).

In this paper, \( SA(x, y) \) in our problem described in Sect. 2 corresponds to the cost \( c_{ij} \) described in the minimum weight perfect matching problem.

A consecutive region can be formed as a bipartite graph, and fortunately, there exists a polynomial-time algorithm for the minimum weight perfect matching problem in a bipartite graph. Munkres [12] proposed a refined algorithm of minimum weight perfect matching, and showed a polynomial running time.

In contrast, the min-cost multi-commodity network flow problem is formulated as follows [5]:

- **Instances:** A directed graph \( G = (V, E) \), edge capacity \( K(e) \in R^+ \) and edge cost \( C(e) \in R \) for \( e \in E \), demands \( D_i(v) \in R \) for all vertices \( v \in V \) and for \( m \) commodities \( i = 1, \ldots, m \).
- **Configurations:** All sequences of edge labelling \( f_i : E \rightarrow R^+, i = 1, \ldots, m \).
- **Solutions:** All sequences of edge labelling that satisfy the following constraints:

1. **Capacity Constraints:** For all \( e \in E \), \( \sum_{i=1}^{m} f_i(e) \leq K(e) \).
2. **Flow-Conversation Constraints:** Define the net flow of commodity \( i \) into vertex \( v \) to be \( f_i(v) = \sum_{e \in in-v} f_i(e) - \sum_{e \in out-v} f_i(e) \). Then, for all \( i = 1, \ldots, m \) and \( v \in V \), we have \( f_i(v) = D_i(v) \).

- Minimize: \( C(f) = \sum_{e \in E} C(e) \times \sum_{i=1}^{m} f_i(e) \).

Unfortunately, to the best of our knowledge, the min-cost multi-commodity network flow method has the property of NP-complete, and therefore the computation time increases exponentially with the size of inputs.

On the other hand, if the complexity of the calculation is equal to, or greater than a predefined threshold, a heuristic method is applied to consecutive and non-consecutive regions. In that case, any of the existing heuristic methods can be applied to consecutive and non-consecutive regions. For this purpose, we adopt the heuristic method named BIND\_LP [8] which 1) determines a feasible binding solution by partially utilizing the computation steps to find a maximum flow of minimum cost in a network, and then 2) refines it iteratively.

3) **STEP-3 (Line 9–10):** Combine all sub-solutions to obtain a preliminary solution, and match starting and ending nodes of each bus group to find a final solution.

All sub-solutions in consecutive and non-consecutive regions are simply combined to get a preliminary binding, since boundary nodes in each region are fixed. That is, the end node in the former region is connected to the same start node in the latter region. However, this preliminary binding does not guarantee that starting and ending nodes of each bus group are matched to meet the boundary condition for cyclic execution of DFG.

The boundary condition for cyclic execution is described in the following Definition.

**Definition:** Boundary Condition for Cyclic Execution of DFG

The starting and ending nodes of each bus binding group must be identical.

This condition is necessary to efficiently separate each bus group for circuit implementation in order that a separated bus group is routed independently in the layout step. We apply the heuristic method described in [7] to meet this boundary condition.

We explain the method using the illustrative example shown in Fig. 8. Suppose we have the three conflicting flow paths shown in Fig. 8(a). The dotted circle represents that the corresponding bus carries no data transfer at that control.
A conflict graph is constructed from the conflicting flow paths. Figure 8(b) shows the conflict graph of Fig. 8(a). The nodes of the graph are the collection of the first and last data transfers in the flow paths, and there is an edge between two nodes if the corresponding data transfers of the nodes are the first and last data transfers in a conflicting flow path, e.g. $a \rightarrow e$ denotes the flow path from boundary node $a$ (in cstep 4) to node $e$ (in cstep 1) in bus group 4 in (a); these two boundary nodes conflict, and hence, violate cyclic execution of boundary nodes.

Then, cost, $adj \ cost((\cdot))$, is assigned to each edge of the graph. The cost represents a (minimal) increase of the total flow cost required for either one of the two corresponding conflicting flow paths to be non-conflicting.

For example, the flow path $a \rightarrow e \rightarrow \cdots \rightarrow d$ in Fig. 8(a) becomes $a \rightarrow g \rightarrow \cdots \rightarrow a$ by switching its flow arc $a \rightarrow e$ with flow arc $e \rightarrow g$ in flow path $e \rightarrow g \rightarrow \cdots \rightarrow a$ as shown by the upper (dotted) circle in Fig. 8(c), since the increase of flow cost is the minimum among the permutations of flow arcs between other columns. This increases the total flow cost by 0.5. Similarly, $d \rightarrow \cdots \rightarrow h \rightarrow e$ can be converted to a non-conflicting flow path $d \rightarrow e \rightarrow \cdots \rightarrow d$ with an increase of flow cost by 0.8 as shown in the lower (dotted) circle in Fig. 8(c). Consequently, $adj \ cost(a, d) = \min\{0.5, 0.8\} = 0.5$.

For each of the connected components in a conflict graph with edge costs, we select the edge with the least cost, and resolve the corresponding flow path. The two nodes of the edge are then merged into one, and edge costs are updated accordingly. For example, Fig. 8(d) shows the reduced conflict graph with two connected components.

Consequently, $a \rightarrow e \rightarrow \cdots \rightarrow d$ is resolved in the first iteration, and $d \rightarrow \cdots \rightarrow h \rightarrow e$ (and thus $e \rightarrow e \rightarrow \cdots \rightarrow d$) will be resolved in the second iteration. This procedure is repeated to resolve the conflicts of all bus groups to meet the boundary condition.

4.2 Illustrative Example

We demonstrate the entire procedure to run our algorithm using the scheduled operator table shown in Fig. 5.

1) STEP-1: Divide binding table into consecutive and non-consecutive regions.

Figure 9 shows the decomposed consecutive and non-consecutive regions in the bus binding table. There are many ways to define the complexity of the problem instance. We choose the number of permutations of all nodes, except ones in the first cstep in the sub-region, since the optimal method conceptually calculates switching activities of all cases obtained by permuting all nodes, except ones in the first cstep in the sub-region. The number below each region in Fig. 9 denotes the complexity of problem instances.

For example, the complexity of the first region is $4! \times 24 = 24$ and the complexity of the third region is $4! \times 24 \times 4! \times 24 \times 4! \times 24 = 3456$. The optimal binding method is applied to each region, assuming that the threshold value is 5000. The effect of the threshold on the quality of a binding solution and computation time will be discussed in Sect. 5.

Note that all nodes in each region are not bound to any bus group at the current step.

2) STEP-2: Find sub-solution for each consecutive and non-consecutive region.

For each of the connected components in a conflict graph with edge costs, we select the edge with the least cost, and resolve the corresponding flow path. The two nodes of the edge are then merged into one, and edge costs are updated accordingly. For example, Fig. 8(d) shows the reduced conflict graph with two connected components.

Consequently, $a \rightarrow e \rightarrow \cdots \rightarrow d$ is resolved in the first iteration, and $d \rightarrow \cdots \rightarrow h \rightarrow e$ (and thus $e \rightarrow e \rightarrow \cdots \rightarrow d$) will be resolved in the second iteration. This procedure is repeated to resolve the conflicts of all bus groups to meet the boundary condition.

The value shown below each region is TSA, and the
STEP-3 has the property of an optimal substructure, assuming this property. In Sect. 4.1, the preliminary binding at each step has the property of optimality with respect to the entire binding table (Refer to the Theorem in Sect. 4.3).

3) STEP-3: Combine all sub-solutions to obtain a preliminary solution, and match starting and ending nodes of each bus group to find a final solution.

We resolve the conflict of boundary nodes to meet the boundary condition for cyclic execution of DFG. As mentioned earlier, we apply the heuristic method used in [7] to meet this boundary condition.

In Fig. 11, we show the result of applying the method to the binding table for our example in Fig. 10. The relations of boundary nodes among all bus groups are shown. Only ‘group 3’ that starts and ends with an identical node ‘x’ meets the boundary condition. The other groups violate the boundary condition since they rotate back-to-back, that is, group 1 starts with ‘3’ and ends with ‘dx’; group 4 starts with ‘dx’ and ends with ‘u’; and group 2 starts with ‘u’ and ends with ‘3’.

First, the conflicts of group 1 and 2 are resolved. As a result, group 1 meets the boundary condition since it starts and ends with an identical node ‘3,’ but the sum of TSAs increases by 2.73.

At the next step, the conflicts of group 2 and 4 are resolved, but the sum of TSAs also increases by 0.02. After the conflicts of all bus groups are resolved, the sum of TSAs increases from 101.40 to 104.15. This value is the minimum TSA found by our method.

Figure 12 shows the final binding solution that has no conflicts of boundary nodes in all the bus groups. The boundary node of each bus group in cstep 1 exactly matches the boundary node of each bus group in cstep 7.

4.3 Optimal Substructure of Table Decomposition Method

We found the optimal substructure of the bus binding problem, and developed a table decomposition method exploiting this property. In Sect. 4.1, the preliminary binding at STEP-3 has the property of an optimal substructure, assuming that the optimal method is applied to the consecutive and non-consecutive regions of this preliminary binding.

We employ the following Theorem.

<Theorem>Sub-bindings of minimum weight binding are also minimum weight bindings.

Given a weighted, directed graph $G = (V, E)$ with weight function $SA : E \rightarrow R$, let $p = (v_1, v_2, ..., v_k)$ be a minimum weight binding from boundary node $v_1$ to boundary node $v_k$ and, for any boundary node $i$ and $j$ such that $1 \leq i \leq j \leq k$, let $p_{ij} = (v_i, v_{i+1}, ..., v_j)$ be the sub-binding of $p$ from boundary node $v_i$ to boundary node $v_j$. Then, $p_{ij}$ is a minimum weight binding from $v_i$ to $v_j$. Also, $SA(p) = \sum SA(p_{ij})$, where $1 \leq i \leq k$ and $j = i + 1$.

(Proof) Without loss of generality, we prove this theorem using an illustrative example shown in Fig. 13. The graph is drawn from the bus binding table in Fig. 5.

We define a boundary node as the boundary grouping of starting or ending nodes in consecutive and non-consecutive regions. For example, a boundary node $v_1$ is the group of $(u, dx, 3, x)$ in Fig. 5, and a boundary node $v_4$ is the group of $(y, y_1, x, dx)$. Also, we define an internal node as the internal grouping of bus in non-consecutive region. For example, an internal node in cstep 4 corresponds to any grouping of $(r_5, r_6, two empty buses)$ in Fig. 5.

The multiple edges between boundary nodes in a consecutive region correspond to all possible mappings of starting and ending nodes. The mapping of the smallest $SA$ among those multiple edges is selected to an optimal solution in the substructure, e.g. the edge shown by a dashed line. Note that boundary nodes $v_1$ and $v_2$ are fixed since the boundary groupings are not changed although any edge between them is selected.

In contrast, there are no multiple edges for non-consecutive region. For non-consecutive regions, there are multiple paths between boundary nodes, but boundary nodes are also fixed as those in consecutive regions. The graph of consecutive regions differs from that of non-consecutive regions, due to the empty node that produces direct mapping between nodes in the former and latter cstep of current cstep.

If we decompose optimal binding $p$ into $v_1 \rightarrow (p_{ij}) \rightarrow v_j \rightarrow (p_{ij}) \rightarrow v_5$, then we have $SA(p) = SA(p_{ij}) + SA(p_{ij}) + SA(p_{ij})$, since the sub-bindings are independent of each other.
dependent owing to the fixed boundary nodes among them. Now, assume that there is a binding \( p'_{ij} \) from \( v_l \) to \( v_j \) with weight \( SA(p'_{ij}) < SA(p_{ij}) \), e.g., \( p_{23} \) and \( p'_{23} \) in Fig. 13. Then, \( v_l \rightarrow (p_{ij}) \rightarrow v_j \rightarrow (p'_{ij}) \rightarrow v_j \rightarrow (p_{j\beta}) \rightarrow v_\beta \) is a binding from \( v_l \) to \( v_\beta \) whose weight \( SA(p_{ij}) + SA(p'_{ij}) + SA(p_{j\beta}) \) is less than \( SA(p) \), contradicting the assumption that \( p \) is a minimum weight binding from \( v_l \) to \( v_\beta \).

From the above statement, we obtain a very useful equation to obtain a global optimal solution by adding each optimal solution in the substructure. As mentioned above, if we decompose optimal binding \( p \) into \( v_l \rightarrow (p_{ij}) \rightarrow v_j \rightarrow (p'_{ij}) \rightarrow v_j \rightarrow (p_{j\beta}) \rightarrow v_\beta \), then we have \( SA(p) = SA(p_{ij}) + SA(p'_{ij}) + SA(p_{j\beta})(1 \leq i \leq j \leq k) \). If we select all \( i \) such that \( 1 \leq i \leq k \) and \( j = i + 1 \), then this equation can be rewritten as \( SA(p) = \sum SA(p_{ij})(1 \leq i \leq k \text{ and } j = i + 1) \). □

5. Experimental Results

In our experiments, eight high-level datapath synthesis benchmark circuits in [9] were used in Tables 2–6 to show the effectiveness of our low-power binding method: 1) DIFF_EQ is a Differential Equation Solver, 2) EWF is an Elliptical Wave Filter, 3) IIR is a standard IIR filter, 4) FIR is a standard FIR filter, 5) TFIR is a transposed-FIR filter, 6) Lattice is a normalized Lattice filter, 7) FFT is an implementation of Fast Fourier Transformation, and finally 8) FDCT is an implementation of Fast Discrete Cosine Transformation.

Our proposed algorithm referred to as DECOMP was implemented in C++ and executed in a Sun Sparc64-V workstation.

5.1 Optimal Substructure of Benchmark Circuits

Table 2 shows the optimal substructures that exist in benchmark circuits. There are from zero to seven consecutive regions, and one to three non-consecutive regions. As mentioned in Sect. 4.2, we use the number of permutations of all nodes, except the nodes in the first cstep in the sub-region as the complexity of the problem instance.

Table 2 Optimal substructure of benchmark circuits.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of bus groups</th>
<th>Number of consecutive regions</th>
<th>Number of non-consecutive regions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The complexity of each region</td>
<td>The complexity of each region</td>
</tr>
<tr>
<td>DIFF_EQ</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>EWF</td>
<td>4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>IIR</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>FIR</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>TFIR</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Lattice</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>FFT</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>FDCT</td>
<td>13</td>
<td>24</td>
<td>1</td>
</tr>
</tbody>
</table>

The range of the complexity of non-consecutive regions is very wide, and therefore, the quality of the binding solution found by the proposed method is highly dependent upon the predefined threshold.

5.2 Total Switching Activity Comparison

Table 3 shows the results of TSA. BIND_OPT is the optimal binding method proposed in [5]. BIND_LP is the heuristic binding method in [8]. The number in parenthesis shows the error to the optimal binding solution obtained by BIND_OPT.

DECOMP shows that the proposed method achieves a solution that is 2.3–22.2% closer to optimal solution, compared to conventional heuristic method at the threshold value of 1.0E+9. Furthermore, the optimal solution can be obtained for FIR and TFIR. A maximum of 2.2% error to the optimal solution occurs in case of DIFF_EQ. The error to optimal binding solution occurs, since the optimal substructure is broken very little in the process of matching starting and ending nodes to meet the boundary condition of cyclic execution.

Another cause of error to an optimal solution can be due to applying a heuristic method instead of an optimal method, when the calculation complexity exceeds the predefined threshold. This occurs when the original problem is large, or the sub-problem decomposed into optimal substructure is also large. The examples show a 0.8% error to the optimal solution for FIR, and 1.4% for TFIR, at a threshold of 1.0E+3.

Table 3 Total switching activity (TSA) comparison.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>BIND_OPT</th>
<th>BIND_LP</th>
<th>DECOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Threshold (1.0E+3)</td>
<td>Threshold (1.0E+4)</td>
<td>Threshold (1.0E+6)</td>
</tr>
<tr>
<td>DIFF_EQ</td>
<td>101.90</td>
<td>126.77</td>
<td>111.98</td>
</tr>
<tr>
<td></td>
<td>(24.4%)</td>
<td>(9.9%)</td>
<td>(22.5%)</td>
</tr>
<tr>
<td>EWF</td>
<td>–</td>
<td>310.85</td>
<td>310.76</td>
</tr>
<tr>
<td>IIR</td>
<td>–</td>
<td>182.94</td>
<td>175.72</td>
</tr>
<tr>
<td>FIR</td>
<td>170.96</td>
<td>174.94</td>
<td>172.35</td>
</tr>
<tr>
<td>TFIR</td>
<td>112.85</td>
<td>118.03</td>
<td>114.43</td>
</tr>
<tr>
<td>Lattice</td>
<td>–</td>
<td>171.23</td>
<td>163.68</td>
</tr>
<tr>
<td>FFT</td>
<td>–</td>
<td>250.87</td>
<td>250.86</td>
</tr>
<tr>
<td>FDCT</td>
<td>–</td>
<td>222.88</td>
<td>221.13</td>
</tr>
</tbody>
</table>

Table 4 Computation time comparison.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>BIND_OPT</th>
<th>BIND_LP</th>
<th>DECOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Threshold (1.0E+3)</td>
<td>Threshold (1.0E+4)</td>
<td>Threshold (1.0E+6)</td>
</tr>
<tr>
<td>DIFF_EQ</td>
<td>589.41</td>
<td>0.01</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>(3.9E+5x)</td>
<td>(4.9E+6x)</td>
<td>(4.9E+7x)</td>
</tr>
<tr>
<td>EWF</td>
<td>–</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>IIR</td>
<td>–</td>
<td>3.28</td>
<td>605.51</td>
</tr>
<tr>
<td>FIR</td>
<td>5213.00</td>
<td>0.01</td>
<td>5.18</td>
</tr>
<tr>
<td>TFIR</td>
<td>31224.00</td>
<td>0.01</td>
<td>5.18</td>
</tr>
<tr>
<td>Lattice</td>
<td>–</td>
<td>1.00</td>
<td>12.07</td>
</tr>
<tr>
<td>FFT</td>
<td>–</td>
<td>0.01</td>
<td>53.36</td>
</tr>
<tr>
<td>FDCT</td>
<td>–</td>
<td>0.01</td>
<td>2.07</td>
</tr>
</tbody>
</table>

a. memory overflow problem
If we set the threshold to 1.0E+14, then the binding solution for all sub-regions is obtained by applying only the optimal method. In the case of EWF, the optimal method cannot obtain the binding solution since the complexity of non-consecutive regions is very high, and hence, the memory overflow problem occurs. Therefore, we see that it is undesirable to use too great a threshold compared to the computing power of the machine.

In other benchmark circuits, the binding results are the same at threshold values of 1.0E+9 and 1.0E+14. In contrast, the binding results at threshold of 1.0E+3 are 0.8–7.7% worse than those at other threshold values, since the heuristic method is applied to non-consecutive regions.

Note that BIND\(_{OPT}\) was unable to report the binding solutions for some designs, mainly due to the memory overflow problem. Even for these circuits, DECOMP found TSA of binding solutions that are smaller than those obtained by the heuristic method (BIND\(_{LP}\)).

### 5.3 Computation Time Comparison

Table 4 shows the results of computation time. The number in parentheses shows how fast DECOMP finds a binding solution compared to an optimal method (BIND\(_{OPT}\)). DECOMP finds a solution 8.0–479.2 times faster than does optimal method (BIND\(_{OPT}\)) at the threshold value of 1.0E+9. The computation time becomes faster as sub-problems are divided into smaller sizes.

The computation time by DECOMP decreases considerably as the threshold value becomes smaller, since the heuristic method is applied to the sub-region instead of the optimal method. However, for EWF, the computation time is the same at threshold values of 1.0E+3 and 1.0E+9, since the heuristic method is applied to sub-regions in both cases.

### 5.4 The Effect of Skipping the Resolution of Conflicts in TSA and Computation Time

In this section, we consider the effect of skipping the resolution of conflicts at STEP-3 in our proposed method. This experiment is conceptually identical to ignoring the boundary condition for cyclic execution of DFG.

Table 5 shows the comparison of TSA. For DIFF\(_{EQ}\), FIR, and TFIR, the TSAs are the same as those obtained by BIND\(_{OPT}\); the results coincide with the Theorem in Sect. 4.3.

The TSAs obtained by BIND\(_{OPT}\) are reduced by 0.50–6.03 compared to those in Table 3, since more design spaces are explored, i.e., more combinations including the permutations of nodes at the final cstep, due to no constraint for the boundary condition.

Table 6 shows the computation time comparison. Computation time by DECOMP is slightly reduced by 0.01–0.36 (sec) compared to those in Table 4, since the time to resolve the boundary node conflicts is not needed. However, the computation time by BIND\(_{OPT}\) increases about 24 times compared to those in Table 4, since the method explores more design spaces as mentioned earlier.

### 6. Conclusion

We proposed an effective low-power binding technique to reduce switching activity using a table decomposition method. This method is based on a divide-and-conquer approach. We define the optimal substructure of the bus binding problem to utilize the advantage of finding optimal solution with rapid computation.

Our method successfully decomposes the original problem into sub-problems, exploiting the fixed boundary nodes between two adjacent sub-problems. For practical applications, we attempt to combine the advantages of optimal and heuristic approaches to find a better (in terms of quality and computation time) solution compared to conventional optimal-only or heuristic-only methods.

Experimental results show that the proposed method attained a solution 2.3–22.2% closer to the optimal solution compared to one conventional heuristic method (at a threshold value of 1.0E+9). It finds the solution 8.0–479.2 times faster than the optimal method (at a threshold value of 1.0E+9).

While our method finds an optimal or close-to-optimal solution, we noticed that the solution depends on the size of the sub-problem. As the size of problem grows, it is harder to find an optimal solution. Hence, our future study might explore the extended decomposition method by di-
viding the original problem into sufficiently smaller-sized sub-problems.

Acknowledgments

The authors would like to acknowledge valuable comments and suggestions received from the editor and anonymous reviewers.

References


