International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:7, 2011

Dynamic Bus Binding for Low Power Using Multiple Binding Tables

Jihyung Kim, Taejin Kim, Sungho Park, and Jun-Dong Cho

***Abstract*—**A conventional binding method for low power in a high-level synthesis mainly focuses on finding an optimal binding for an assumed input data, and obtains only one binding table. In this paper, we show that a binding method which uses multiple binding tables gets better solution compared with the conventional methods which use a single binding table, and propose a dynamic bus binding scheme for low power using multiple binding tables. The proposed method finds multiple binding tables for the proper partitions of an input data, and switches binding tables dynamically to produce the minimum total switching activity. Experimental result shows that the proposed method obtains a binding solution having 12.6-28.9% smaller total switching activity compared with the conventional methods.

***Keywords*—**low power, bus binding, switching activity, multiple binding tables

I. Introduction

Low power binding technique has been one of classical issues in a high-level synthesis. Many techniques for low power binding to minimize total switching activity (TSA) have been proposed [1]-[5]. The conventional methods assume that a random or user-specific data is given as an input, and find an optimal or close-to-optimal binding solution. By using a single binding table, a fixed bus binding is implemented in the circuit, and is not changed during operation.We show that a binding method which uses multiple binding tables gets better solution rather than ones which use only one binding table, and propose a dynamic bus binding scheme using multiple binding tables. The proposed method finds multiple binding tables for the proper partitions of an input data, and finds the best binding solution periodically for the specific input data. In this paper, the problem of minimizing switching activity in on-chip bus is considered, and however, the proposed method is easily extended to the other resources, e.g. functional units, registers, .etc, with slight modifications.

The main contributions of this paper are as follows:

1) It is shown that that a conventional binding solution found by a single binding table can be improved by using multiple binding tables because using multiple binding tables explores wider design space.
2) Dynamic bus binding is performed periodically by using a real input data, and therefore the proposed method is robust to the variation of the statistical property of an input data.

The remainder of this paper is organized as follows. Section II introduces low power bus binding for switching activity minimization, and the proposed dynamic bus binding scheme is described in Section III. Section IV discusses experimental result, and conclusion is drawn in Section V.

Jihyung Kim is with the System LSI Division, Samsung Electronics Co. Ltd., Yongin, Korea, and with Sungkyunkwan University, Suwon. Korea (e-mail: kim_ji_hyung@samsung.com, johnny71@skku.edu).
Taejin Kim is with the System LSI Division, Samsung Electronics Co. Ltd., Yongin, Korea (e-mail: taejinkim@samsung.com).
Sungho Park is with the System LSI Division, Samsung Electronics Co. Ltd., Yongin, Korea (e-mail: sh603.park@samsung.com).
Jun-Dong Cho is a corresponding author, and with Sungkyunkwan University, Suwon, Korea (e-mail: jdcho@skku.edu).

II. Bus Binding for Switching Activity Minimization

A. Problem Definition of Low Power Bus Binding

Dynamic power is calculated as follows:

$$P_{dyn} = P_{trans} \cdot C_L \cdot V_{dd}^2 \cdot f_{clock} \qquad (1)$$

where P_{trans} is the probability of an output transition, and C_L is the load capacitance, and V_{dd} is the supply voltage, and f_{clock} is the frequency of system clock [6].

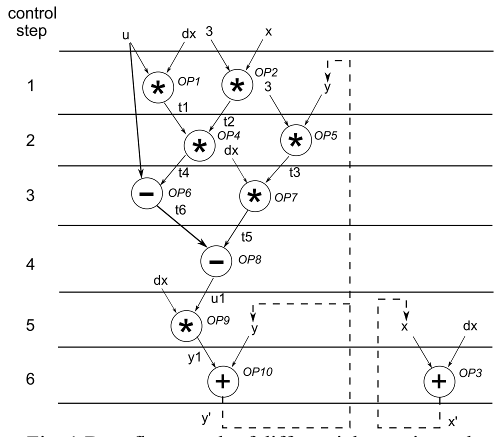


Fig. 1 Data flow graph of differential equation solver

Fig. 1 shows a scheduled DFG of 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.

We assume that a scheduled data flow graph (DFG) is given as an input, and the technique of minimizing switching activity is applied to bus binding. Bit width of each variable is 16.

Conventionally, many researches pay attention to the problem of minimizing total switching activity (TSA) that is the summation of SA in each bus group, that is,

$$\mathrm{TSA} = \Sigma_{(\forall k \text{ of buses})} \mathrm{SA}^k \qquad (2)$$

where, $\mathrm{SA}^k(x, y)$ denote the expected number of bit lines on bus k that toggle when data transfers x and y are successively implemented on the bus, and SA^k is the sum of all $\mathrm{SA}^k(\cdot)$ for every pair of consecutive data transfers on bus k [3].

TABLE I shows the Switching Activity Matrix, which is obtained from simulating 100,000 random inputs to the DFG shown in Fig. 1. For example, SA(u, $t2$) = 7.37 indicates that there is an average of 7.37 bit lines out of 16 possible toggles between data transfers u and $t2$.

TABLE I
SWITCHING ACTIVITIES MATRIX OF DFG IN FIG. 1

	u	*dx*	*3*	*x*	*y*	*t1*	*t2*	*t3*	*t4*	*t5*	*t6*	*u1*	*y1*	*x'*	*y'*
u	0.00	7.50	7.51	7.51	7.49	7.50	7.37	7.83	7.76	8.01	6.99	0.00	8.00	8.00	7.88
dx	7.50	0.00	7.50	7.51	7.51	7.50	7.84	7.84	7.74	7.51	8.13	7.50	7.49	8.00	8.13
3	7.51	7.50	0.00	7.49	7.50	8.26	7.83	7.83	8.26	8.24	8.12	7.51	8.26	8.00	8.12
x	7.51	7.51	7.49	0.00	7.49	8.00	5.11	7.84	7.75	8.00	8.12	7.51	8.01	8.00	8.13
y	7.49	7.51	7.50	7.49	0.00	8.00	7.84	5.11	8.00	7.50	8.00	7.49	7.82	8.00	7.50
t1	7.50	7.50	8.26	8.00	8.00	0.00	8.01	8.00	7.01	7.59	7.75	7.50	7.01	8.00	7.87
t2	7.83	7.84	7.83	5.11	7.84	8.01	0.00	7.94	7.74	8.00	8.13	7.83	8.00	8.00	8.13
t3	7.83	7.84	7.83	7.84	5.11	8.00	7.94	0.00	8.00	7.51	7.00	7.83	7.90	7.99	7.99
t4	7.76	7.74	8.26	7.75	8.00	7.01	7.74	8.00	0.00	7.50	8.01	7.76	7.34	8.12	8.00
t5	8.01	7.51	8.24	8.00	7.50	7.49	8.00	7.51	7.50	0.00	8.01	8.01	7.34	8.12	8.00
t6	6.99	8.13	8.12	8.12	8.00	7.75	8.13	8.00	8.01	8.01	0.00	6.99	7.99	7.84	7.75
u1	0.00	7.50	7.51	7.51	7.49	7.50	7.83	7.83	7.76	8.01	6.99	0.00	8.00	8.00	7.89
y1	8.00	7.49	8.26	8.01	7.82	7.01	8.00	7.90	7.34	7.34	7.99	8.00	0.00	7.99	8.01
x'	8.00	8.00	8.00	8.00	8.00	8.00	8.00	7.99	8.12	8.00	7.84	8.00	7.99	0.00	7.88
y'	7.88	8.13	8.12	8.13	7.50	7.87	8.13	7.70	8.00	8.01	7.75	7.88	8.01	7.88	0.00

Bus binding can be mathematically formulated as follows [7]:

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

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

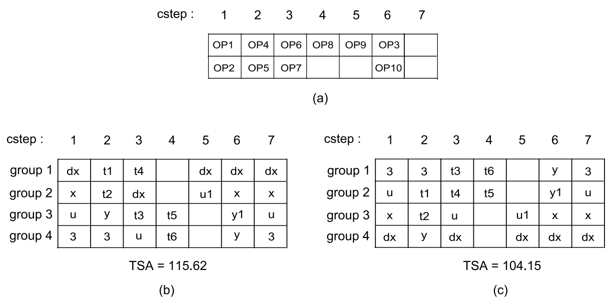
Note that if an operator o_i is scheduled at control step c_j by scheduling function $S(o_i) = c_j$, the variables v_k that are operands to the operator o_i are also located at control step c_j.

Bus binding is performed for this scheduled DFG. Let B is a finite set of buses, denoted $B = \{ b_1, b_2, \dots, b_s \}$ and, $N_v(c_j)$ be the number of variables located at control step c_i. We assume that the number of buses is limited to the maximum number of variables located in one control step, i.e. s = max { $N_v(c_j)$ } where $j = 1, 2, \dots, r$.

Bus Binding is described in the following Definition.

<Definition> Bus Binding
Bus binding is a mapping M_f: $V \times C \to B \times C$, where $M_f(v_i, c_k) = (b_j, c_k)$ denotes that variable corresponding to $v_i \in V$ scheduled at control step c_k, is bound to bus $b_j \in B$ at control step c_k, where V is a set of variable of operators and B is a set of buses.

Fig. 2-(a) is a scheduled operator table extracted from a scheduled DFG in Fig. 1, and the results of bus binding and corresponding TSA are shown in (b) and (c).



cstep :	1	2	3	4	5	6	7
	OP1	OP4	OP6	OP8	OP9	OP3	
	OP2	OP5	OP7			OP10	

(a)

cstep :	1	2	3	4	5	6	7
group 1	dx	t1	t4		dx	dx	dx
group 2	x	t2	dx		u1	x	x
group 3	u	y	t3	t5		y1	u
group 4	3	3	u	t6		y	3

TSA = 115.62

(b)

cstep :	1	2	3	4	5	6	7
group 1	3	3	t3	t6		y	3
group 2	u	t1	t4	t5		y1	u
group 3	x	t2	u		u1	x	x
group 4	dx	y	dx		dx	dx	dx

TSA = 104.15

(c)

Fig. 2 (a) Scheduled operator table (b), (c): Typical examples of bus bindings

For that DFG, four bus groups are allocated to implement all variables of ten operators. In (b) and (c) of Fig. 2, 'group' denotes the specific group of bus binding that starts at cstep 1, and ends at cstep 7. For example, 'group 1' in Fig. 2-(b) shows bus binding that consists of $dx \to t1 \to t4 \to dx$, where empty node in cstep 4 denotes that there is no variable in cstep 4. Note that cstep 7 is inserted for cyclic execution with cstep 1.

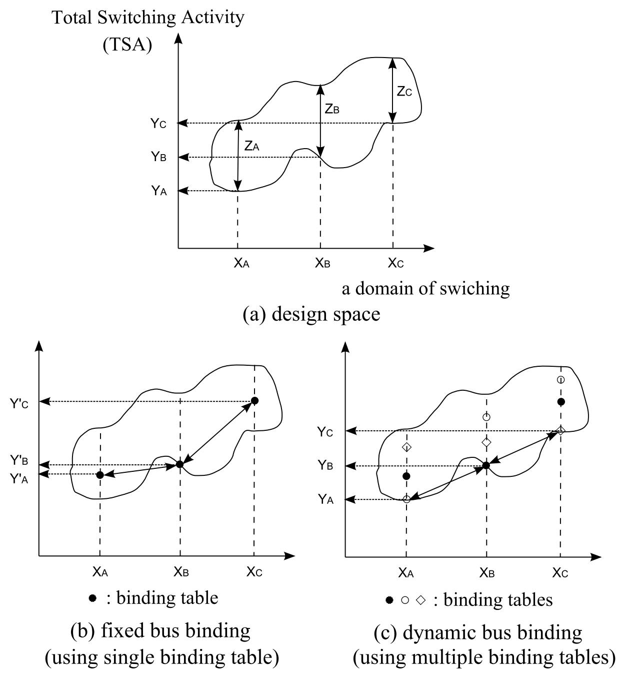
Low power bus binding problem is summarized as follows:

< *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: $V \times C \to B \times C$ ■

The motivation for this paper is described as follows.

We show that binding method using multiple binding tables gets smaller TSA rather than using single binding table, and employ Theorem stating that the binding method using ideally infinite binding tables gets optimal binding solution.

The relation between design space and binding table is shown in Fig. 3. In Fig. 3-(a), X-axis denotes a domain of switching activity matrix, and Y-axis denotes TSA. X_A, X_B, and X_C are three different switching activity matrices calculated by assuming three different probability density functions for input data to DFG. Y_A, Y_B, and Y_C are the minimum TSA found by optimal binding method. Z_A, Z_B, and Z_C denotes that there are many binding solution for each switching activity having TSA ranging from X_A to $X_A + Z_A$, from X_B to $X_B + Z_B$, and from X_C to $X_C + Z_C$, respectively.



(a) design space

(b) fixed bus binding (using single binding table)

(c) dynamic bus binding (using multiple binding tables)

Fig. 3 Design space and a binding table

In Fig. 3-(b), fixed bus binding by conventional binding method is shown. The method uses single switching activity matrix, e.g., X_B, and this matrix is calculated by assuming a random or user-specific input data. For X_B, the optimal bind solution (black dot) having the minimum TSA, e.g., Y'_B is found. However, this binding table do not always find optimal binding solution for different switching activity matrix such as X_A or X_C. For example, Y'_A is not the minimum TSA for X_A. To overcome the fundamental limitation of the existing method, dynamic binding method using multiple binding tables is proposed as shown in Fig. 3-(c). For each of three different switching activity matrices X_A, X_B, and X_C, the optimal binding table is found respectively (block dot, white dot, and white diamond). We showed only three binding tables in Fig. 3-(c), and however, there might be infinitely many binding tables for each of infinitely many different switching activity matrices.

B. Related Work

Conventional binding methods assume that *a priori* switching activity matrix is given. They calculate a fixed single switching activity matrix in advance by inputting a random or application specific pattern to DFG and by calculating Hamming (average) distances between two consecutive variables. How to get a switching activity matrix in conventional binding methods is referred to as follows:

- H(x1, x2) is the Hamming distance of the binary representations of x1 and x2 [1].
- The value of SW(x, y) is set to be the average of the Hamming distances between x and y. [2]-[3].
- We use the Switching Activity Matrix, which is obtained from simulating 100,000 random inputs to the DFG, to obtain the SA of several different schedules and bindings of the DFG, which illustrates different schedules and bindings could result in very different SA [4].
- Given a DFG G(V, E), execution profiling is done where execution traces on all edges E are collected by simulating the behavioral description with a set of input sequences provided by the user. It iterates over all data transfers in partition Ei which can be bounded to a single bus [5].

However, the statistical property of data pattern input to DFG in real applications can be different from that of the assumed specific data pattern, and as the difference becomes greater, the error to optimal binding solution becomes greater. Therefore, it is impossible to get minimum total switching activity using only one binding table.

III. Dynamic Bus Binding Using Multiple Binding Tables

A. Proposed Bus Binding Method

```
Dynamic_bus_binding ( variables, buses, binding tables )
  1  Store input data in advance periodically.
  2  Calculate switching activity matrix.
  3  Find binding table to produce minimum total switching activity.
  4  Switching the binding table with the previous binding table.
  5  Repeat Step 1 ~ 4.
```

Fig. 4 Dynamic bus binding procedure

Fig. 4 shows the bus binding method using multiple binding tables. Conventional method corresponding to Fig. 4-(b) uses a single binding table, and bus binding is fixed. If the switching pattern of input data such as { V_1, V_2, … , V_n } is exactly the same as the assumed one, e.g. X_B in Fig. 3-(b), the switching activity in bus such as { B_1, B_2, … , B_m } shows the minimum value. However, if the switching pattern of input data is X_A, then the switching activity might not show the minimum value. On the other hand, the proposed bus binding scheme switches dynamically binding tables that are optimal for periodically calculated switching activity patterns of input data.

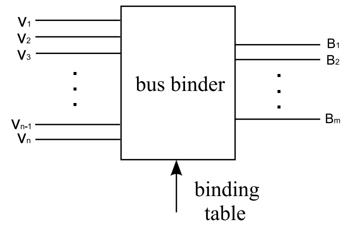
The functional diagram of the proposed binding scheme is shown in Fig. 5. There are four components such as memory, switching activity calculator, binding table finder, and bus binder. Memory is used to store input data for some pre-defined period. For the stored data, switching activity matrix (SAM) is calculated in switching activity calculator. After SAM is calculated, the optimal binding table is obtained for the SAM. When new binding table is obtained, the previous binding table in bus binder is replaced with the new binding table. At the same time, 'done' signal is sent to memory and new input data is stored in memory for the next period.

We employ the following Theorem.

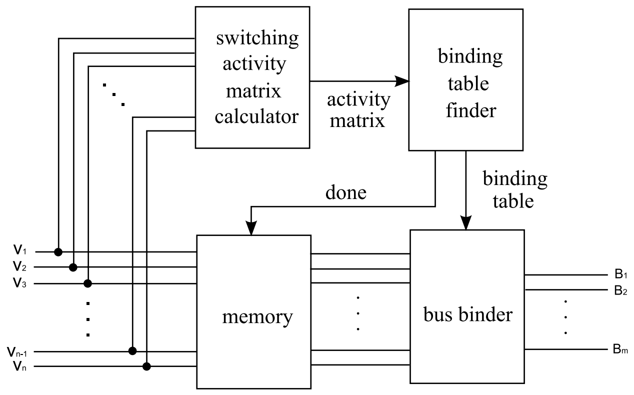
<Theorem> The proposed method which uses multiple binding tables finds optimal solution for total switching activity (TSA) if the number of multiple binding tables is infinite.

(Proof) We prove this theorem by contradiction. The proposed method switches dynamically bus binding tables, each of which produces the minimum (optimal) TSA for the corresponding switching activity matrices which are calculated periodically.

Suppose that total switching activity is not the minimum for bus binding by the proposed method. Then there should be any (at least one) binding table that is not optimal for the specific switching activity matrix. This contradicts that our proposed method always found optimal binding table to all switching activity matrix input to bus binder. □



(a) conventional fixed bus binding scheme



(b) proposed adaptive bus binding scheme

Fig. 5 Functional diagram for dynamic bus binding scheme

B. Limitation for the implementation of Original Work

Theoretically, the proposed method finds optimal binding solution as proved by Theorem. However, there is some limitation for the implementation of bus binder. Suppose that optimal bus binding method such as [1] is adopted in bus binder. To the best of our knowledge, optimal bus binding method is NP-hard, then it cannot be applied in real-time because it takes considerably long time to find optimal bus binding.

There are two choices for implementation of real hardware.

1) First, heuristic bus binding method rather than optimal method is applied in bus binder. However, it still takes relatively long time for the operation frequency of the real circuit.

2) Second, the optimal bus binding tables are found in advance, and saved in memory. Then, for switching activity calculated periodically, the binding table to produce minimum TSA is selected among them. There should be limitation for the number of binding tables due to the limited space for memory.

In this paper, we choose the latter approach, and describe the modified bus binding method in the next section.

C. Modified Dynamic Bus Binding Method

We modify the original work considering the limitation for implementation. First, we generate multiple binding tables for the proper partitions of input data. Fig. 6 describes the procedure of generating multiple binding tables.

```
Generation_of_multiple_binding_tables ( Scheduled DFG )
 1  Get input variables Xi (i = 1, 2, ... Ni)  /* Ni : number of input variables */
 2  /* generate coarse-grain binding table */
 3  Set index p = 0.
 4  for 'j' from 1 to Nj   /* Nj : number of iteration */
 5      Generate random data from 0 to W for all Xis.
 6      Calculate switching activity, and average it.
 7  end for loop
 8  Get switching activitity matrix SAM0.
 9  Apply binding method to get binding table BT0.
10  Save BT0 in memory.
11  /* generate fine-grain binding tables */.
12  Increase p by 1.
13  for 'each input variable Xi' from 1 to Ni
14      Partition the probability density function into 'n' parts with equal size.
15      for 'each q' from 1 to n-1
16          for 'j' from 1 to Nj   /* Nj : number of iteration */
17              Generate random data from qxW/n to (q+1)xW/n
18              Generate random data from 0 to W for Xk. (k != i).
19              Calculate switching activity, and average it.
20          end for loop
21          Get switching activitity matrix SAMp.
22          Apply binding method to get binding table BTp.
23          Save BTp in memory.
24      end for loop
25      Increase p by 1.
26  end for loop
```

Fig. 6 Generation of multiple binding tables

To guarantee the performance of the conventional method at the minimum, we add optimal binding table found by conventional binding method. We denote the optimal binding table obtained by conventional method as *coarse-grain* binding table, because it is found for the switching activity matrix representing the overall statistical property of input data.

In contrast, we denote the other subsidiary binding tables as *fine-grain* binding tables, because they are found for the switching activity matrices representing the local statistical property of input data.

1) STEP-1 (Line 1-10): Generate coarse-grain binding table.

Coarse-grain binding table is obtained for the overall probability density function of the input data. This binding table is obtained by conventional binding method. The generated binding table, i.e. BT_0 is stored in memory to be used for bus binder module.

2) STEP-2 (Line 11-26): Generate fine-grain binding table.

Fine-grain binding tables are obtained for the local probability density function of the input data. These binding tables are used to compensate for the error of coarse-grain binding table when the statistical property of input data is far from that of the random pattern.

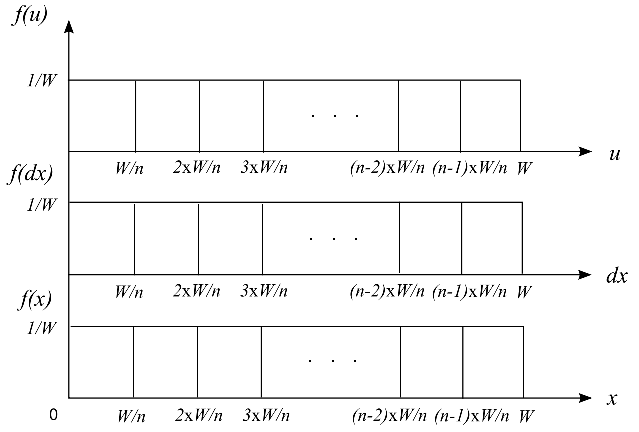


Fig. 7 Partition of probability function of input variables for DFG in Fig.1

Fig. 7 shows the proper partition for input data. For each input variable, the probability density is partitioned into '*n*' parts having equal probability density. For each iteration (Line 13-26), random data is generated for each part, and switching activity matrix is calculated, and finally, optimal binding table corresponding to each switching activity matrix is obtained.

With the limitation for the number of binding tables, the binding solution found by this approach incurs the error to the optimal one obtained by our original work. We compare the result of this modified approach with that of the convention method in Section IV.

IV. Experimental Result

To show the effectiveness of the proposed dynamic bit ordering method, eight high-level datapath synthesis benchmark circuits are used in Tables II ~ III: 1)DIFF_EQ is a Differential Equator, 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.

The proposed algorithm was implemented in C++ and executed in a Sun Sparc64-V workstation.

A. Comparison of Total Switching Activity

Table II shows the comparison of total switching activity (TSA). BIND_OPT is optimal binding method proposed in [1], and BIND_LP is heuristic binding method in [2]. Dynamic bus binding is performed to each conventional method, and the proposed method is denoted as BIND_OPT + MBT and BIND_LP + MBT, respectively, where MBT denotes Multiple Binding Tables. The number in parenthesis shows a reduction factor to conventional methods.

The proposed method denoted as BIND_OPT + MBT gets the solution having 17.4-25.9% (average 22.5%) smaller TSA compared with BIND_OPT by using seven binding tables. And the proposed method denoted as BIND_LP + MBT obtains the solution having 16.1-26.3% (average 20.6%) smaller TSA compared with BIND_OPT by using seven binding tables.

TABLE II
Comparison of total switching activity (TSA) (number of binding tables = 7)

	BIND_ OPT	BIND_OPT + MBT	BIND_LP	BIND_LP + MBT
DIFF_EQ	101.90	77.21 (24.2%)	124.17	98.59 (20.6%)
EWF	–[a]	–[a]	307.41	257.92 (16.1%)
IIR	–[a]	–[a]	177.28	130.66 (26.3%)
FIR	168.54	139.25 (17.4%)	170.18	138.02 (18.9%)
TFIR	110.82	82.10 (25.9%)	112.25	85.31 (24.0%)
Lattice	–[a]	–[a]	163.38	130.05 (20.4%)
FFT	–[a]	–[a]	249.28	202.79 (18.7%)
FDCT	–[a]	–[a]	220.18	176.03 (20.1%)
Average (%)	–[a]	22.5		20.6

a. memory overflow problem

The proposed dynamic bus binding method gets better solution regardless of the type of conventional methods, i.e. optimal or heuristic method because our method finds better binding solution by switching proper binding tables that produce the minimum TSA for specific input data sets, and it does not matter how to find each binding table. Also, our method shows the similar performance for various benchmark circuits.

B. Influence of Iterations on Total Switching Activity

Table III shows the influence of the number of binding tables on total switching activity (TSA).

As the number of binding tables increases, the reduction ratio of switching activity also increases from 12.6% to 28.9% (on average) because wider design space can be explored. Therefore, it is desirable to increase the number of binding tables as at the maximum as possible.

But, as the more binding tables are used, the overhead of the dynamic power dissipated in the binding table finder increases because the more calculation should be performed to find minimum switching activity between binding tables. Therefore, that is, there is a trade-off between the number of binding tables and the overhead of additional power dissipation.

TABLE III
COMPARISON OF TSA ACCORDING TO NUMBER OF BINDING TABLES

	BIND_ LP	**BIND_LP + MBT**				
		Number of Binding Tables				
		3	**5**	**7**	**9**	**11**
DIFF_EQ	124.17	109.27 (12.0%)	105.42 (15.1%)	98.59 (20.6%)	89.65 (27.8%)	80.24 (35.4%)
EWF	307.41	280.67 (8.7%)	272.37 (11.4%)	257.92 (16.1%)	242.24 (21.2%)	228.71 (25.6%)
IIR	177.28	146.26 (17.5%)	139.70 (21.2%)	130.66 (26.3%)	124.63 (29.7%)	119.13 (32.8%)
FIR	170.18	152.82 (10.2%)	145.50 (14.5%)	138.02 (18.9%)	131.72 (22.6%)	121.85 (28.4%)
TFIR	112.25	97.83 (12.9%)	91.82 (18.2%)	85.31 (24.0%)	80.60 (28.2%)	77.23 (31.2%)
Lattice	163.38	136.91 (16.2%)	132.99 (18.6%)	130.05 (20.4%)	124.99 (23.5%)	118.61 (27.4%)
FFT	249.28	225.10 (9.7%)	216.38 (13.2%)	202.79 (18.7%)	194.94 (21.8)	188.46 (24.4%)
FDCT	220.18	189.57 (13.9%)	181.87 (17.4%)	176.03 (20.1%)	170.86 (22.4%)	162.49 (26.2%)
Average (%)		12.6	16.2	20.6	24.7	28.9

V. CONCLUSION

An effective dynamic bus binding technique for low power is proposed to obtain a binding solution having smaller total switching activity by using multiple binding tables rather than by using a single binding tables. Experimental result shows that the proposed method obtains a binding solution having 12.6-28.9% smaller TSA compared with conventional methods for various number of binding tables from three to eleven.

The proposed method obtains a better binding solution because wider design space is explored. Our future work is to explore wider design space by combining binding technique with data encoding technique such as data reordering technique.

REFERENCES

[1] J. Chang and M. Pedram, “Module assignment for low power,” ” in *Proc. Eur. Design Automation Conf.*, pp.376-381, 1996.

[2] Y. Choi and T. Kim, “An efficient low-power binding algorithm in high-level synthesis,” IEEE Int. Symp. On Circuits and Systems, vol. 4, pp. 321-324, 2002.

[3] C. Lyuh and T. Kim, “High-level synthesis for low power based on network flow method,” ” IEEE Trans. VLSI, vol. 1, no. 3, pp. 309–320, 2003

[4] X. Xing and C. C. Jong, “A look-ahead synthesis technique with backtracking for switching activity reduction in low power high-level synthesis,” Microelectronics Journal, vol. 38, no. 4-5, pp. 595-605, 2007.

[5] H. Sankaran and S. Katkoori, “Bus Binding, Re-ordering, and Encoding for Crosstalk-producing Switching Activity Minimization during High Level Synthesis,” in *Proc. 4th IEEE Intl. Symp. On Electronics Design, Test & Applications*, pp. 454-457, 2008.

[6] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, Low power methodology manual : for system-on-chip design, Springer, pp. 4-7, 2007.

[7] J. Kim and J. Cho, “Low power bus binding exploiting optimal substructure,” IEICE Trans. on Fundamentals of Electronics, Communications, and Computer Sciences, vol. E94-A, no. 1, pp. 332-341, 2011.