Low Power Bus Binding Based on Dynamic Bit Reordering

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

***Abstract*—**In this paper, the problem of reducing switching activity in on-chip buses at the stage of high-level synthesis is considered, and a high-level low power bus binding based on dynamic bit reordering is proposed. Whereas conventional methods use a fixed bit ordering between variables within a bus, the proposed method switches a bit ordering dynamically to obtain a switching activity reduction. As a result, the proposed method finds a binding solution with a smaller value of total switching activity (TSA). Experimental result shows that the proposed method obtains a binding solution having 12.0-34.9% smaller TSA compared with the conventional methods.

***Keywords*—**bit reordering, bus binding, low power, switching activity matrix

I. Introduction

SINCE the functions of mobile devices such as a mp3 player, a hand-help phone, a tablet PC, .etc become complex and performance improves, gate counts in a system-on-chip (SOC) embedded in the devices increase considerably and operating frequency becomes very fast. Combined with a deep sub-micron process, this trend incurs the sharp increase of dynamic power dissipation, and hence hot temperature due to dynamic power dissipation brings the malfunction of operations of the SOC and the increase of package cost, e.g. adding a cooling device.

Moreover, as the deep sub-micron process becomes popular, power consumption in on-chip buses becomes a critical problem. For example, dynamic power consumption in on-chip buses consists of 20-36% of total dynamic power dissipation [1].

Many techniques have been developed to reduce dynamic power in on-chip buses, and among them, it is known as effective to minimize switching activity (SA) in bus wires because reducing switching activity, a.k.a. output transition does not influence the performance of the circuit and can be applied at any level of design time, i.e. architectural level, RTL level, and gate level.

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).

The techniques of reducing SA are proposed in [2]-[6] and they are divided into two categories: bus binding and bus encoding. Whereas bus binding techniques such as [2]-[3] are mainly applied at the stage of high-level synthesis, bus encoding methods such as [4]-[6] are generally applied at the step of physical implementation. It is known that as low power binding technique is applied at the higher design step, more dynamic power reduction can be achieved [7].

In this paper, the reduction of dynamic power in on-chip buses at the architectural level a.k.a. behavioral level or high-level, is considered, and low power bus binding technique is proposed to minimize switching activity in on-chip buses by switching a bit ordering dynamically between variables within a bus.

The main contributions of this paper are as follows:

1) It is noticed that that the distribution of switching activity between the bits of two adjacent variables within a bus is not uniform, and therefore the reduction of switching activity can be achieved by switching a bit ordering dynamically rather than using a fixed ordering.
2) The problem of obtaining an optimal bit ordering to produce minimum switching activity between the bits of two adjacent variables is defined as the problem of minimum weight bipartite matching, and a refined method is adopted to solve it in a polynomial running time.
3) Dynamic bit reordering 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 method is described in Section III. Section IV discusses experimental result, and conclusion is drawn in Section V.

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, C_L is the load capacitance, V_{dd} is the supply voltage, and f_{clock} is the frequency of system clock [8].

Fig. 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.

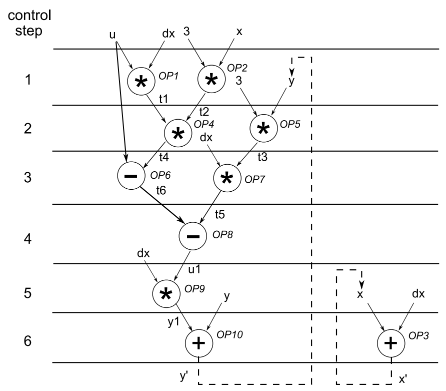


Fig. 1 Data flow graph of differential equation solver

It is assumed 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* [2].

TABLE I shows the values for the data transfers in Fig. 1. For example, SA(*u*, *t*2) = 7.37 indicates that there is an average of 7.37 bit lines out of 16 possible toggles between data transfers *u* and *t*2. Note that this matrix is generated by assuming the fixed bit ordering between the bits of two variables shown in Fig.3-(a).

Bus binding can be mathematically formulated as follows [9]:

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 $S_f : O \rightarrow C$, 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 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, $N(v_i, c_j)$ be the number of variables v_i located at control step c_i. It is assumed that the number of buses is limited to the maximum number of variables located in one control step, i.e. $s = \max \{ N(v_i, c_j) \}$ where $j = 1, 2, \dots, r$.

TABLE I
LONG-TERM SWITCHING ACTIVITIES MATRIX OF DFG IN FIG. 1
(BETWEEN VARIABLES, ITERATION = 100,000)

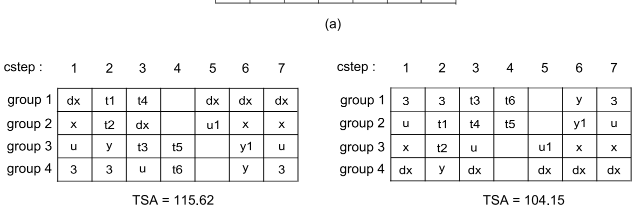
	u	*dx*	3	*x*	*y*	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6	*u*1	*y*1	*x'*	*y'*
u	0.00	7.50	7.51	7.51	7.49	7.50	7.37	7.83	7.76	8.01	6.99	0.00	8.00	8.00	7.88
dx	7.50	0.00	7.50	7.51	7.51	7.50	7.84	7.84	7.74	7.51	8.13	7.50	7.49	8.00	8.13
3	7.51	7.50	0.00	7.49	7.50	8.26	7.83	7.83	8.26	8.24	8.12	7.51	8.26	8.00	8.12
x	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
*t*1	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
*t*2	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
*t*3	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
*t*4	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
*t*5	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
*t*6	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
*u*1	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
*y*1	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 is described in the following Definition.

<Definition> Bus Binding

Bus binding is a mapping M_f*:* $V \times C \rightarrow B \times C$*, 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 $dx \rightarrow t1 \rightarrow t4 \rightarrow dx$, where an empty node in cstep 4 denote 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: $V \times C \rightarrow B \times C$ ■

The motivation for this work is described as follows.

TABLE II shows the example of switching activity between bits of u and $t2$ generated by inputting a random data, and simulating the DFG for 30 iterations. Note that TABLE I is the switching activity matrix (SAM) used by conventional binding method (denoted as long-term SAM), and the proposed method uses additional switching activity matrix between bits of variables as shown in TABLE II (denoted as short-term SAM).

The number of iterations from 10 to 50 rather than 100,000 is used to generate a short-term SAM because the proposed method uses a real input data and it finds switching activity periodically for small iterations to compensate the variations of the distribution of switching activity. As explained in Section IV, if the number of iterations increases, the distribution of switching activity of input data becomes more uniform, and therefore there is no possibility to reduce total switching activity although dynamic bit reordering is performed.

TABLE II shows that the switching activity between the bits of variables is not uniform, and there is an optimal bit ordering between them. Moreover, the switching activity changes because a real input data pattern varies, and therefore an optimal bit ordering should be changed periodically in accordance with the changed switching activity.

TABLE II
SHORT-TERM SWITCHING ACTIVITIES MATRIX FOR U AND $T2$ IN FIG. 1
(BETWEEN BITS, ITERATION = 30)

$t2$ / u	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}
l_0	0.43	0.50	0.57	0.40	0.60	0.57	0.63	0.57	0.37	0.53	0.43	0.57	0.67	0.43	0.23	0.60
l_1	0.50	0.57	0.43	0.60	0.47	0.63	0.50	0.57	0.37	0.53	0.63	0.43	0.60	0.43	0.43	0.53
l_2	0.57	0.43	0.50	0.33	0.60	0.43	0.50	0.57	0.70	0.47	0.37	0.37	0.40	0.57	0.37	0.60
l_3	0.33	0.40	0.53	0.37	0.37	0.40	0.53	0.60	0.40	0.57	0.40	0.67	0.70	0.47	0.33	0.50
l_4	0.60	0.40	0.53	0.57	0.50	0.47	0.40	0.47	0.67	0.50	0.47	0.60	0.50	0.60	0.60	0.23
l_5	0.33	0.60	0.40	0.70	0.43	0.40	0.33	0.47	0.67	0.37	0.47	0.47	0.50	0.53	0.47	0.50
l_6	0.47	0.33	0.60	0.57	0.43	0.67	0.40	0.60	0.47	0.70	0.47	0.60	0.43	0.67	0.67	0.23
l_7	0.33	0.53	0.60	0.50	0.63	0.53	0.33	0.53	0.53	0.50	0.60	0.60	0.57	0.47	0.47	0.57
l_8	0.60	0.27	0.67	0.43	0.50	0.47	0.53	0.53	0.53	0.57	0.67	0.67	0.50	0.47	0.40	0.43
l_9	0.57	0.57	0.57	0.60	0.40	0.37	0.43	0.57	0.37	0.47	0.63	0.63	0.47	0.50	0.57	0.40
l_{10}	0.57	0.43	0.57	0.40	0.67	0.50	0.43	0.57	0.57	0.40	0.37	0.50	0.60	0.50	0.50	0.40
l_{11}	0.43	0.63	0.50	0.47	0.53	0.43	0.57	0.50	0.23	0.47	0.50	0.57	0.47	0.30	0.57	0.60
l_{12}	0.47	0.53	0.40	0.57	0.50	0.53	0.47	0.40	0.67	0.57	0.47	0.53	0.37	0.60	0.53	0.23
l_{13}	0.53	0.40	0.53	0.50	0.43	0.60	0.40	0.60	0.53	0.57	0.53	0.53	0.43	0.40	0.53	0.63
l_{14}	0.60	0.47	0.60	0.37	0.50	0.60	0.73	0.53	0.53	0.63	0.60	0.60	0.57	0.53	0.53	0.50
l_{15}	0.53	0.40	0.60	0.50	0.37	0.60	0.47	0.47	0.53	0.50	0.47	0.60	0.37	0.53	0.53	0.43

Fig. 3 compares a fixed bit ordering and a bit reordering between two variables, e.g. u and $t2$. Conventional method uses a fixed bit ordering as shown in Fig. 3-(a), and generally a fixed bit ordering denotes that each bit position from zero to fifteen coincides with each other. In contrast, a proposed method uses a bit reordering which scrambles a bit position in order to find an optimal bit ordering having a minimum switching activity as shown in Fig. 3-(b). In fact, the bit ordering in Fig. 3-(b) is the optimal one having minimum switching activity between u and $t2$. In this way, optimal bit orderings for all pairs of variables in a DFG are found.

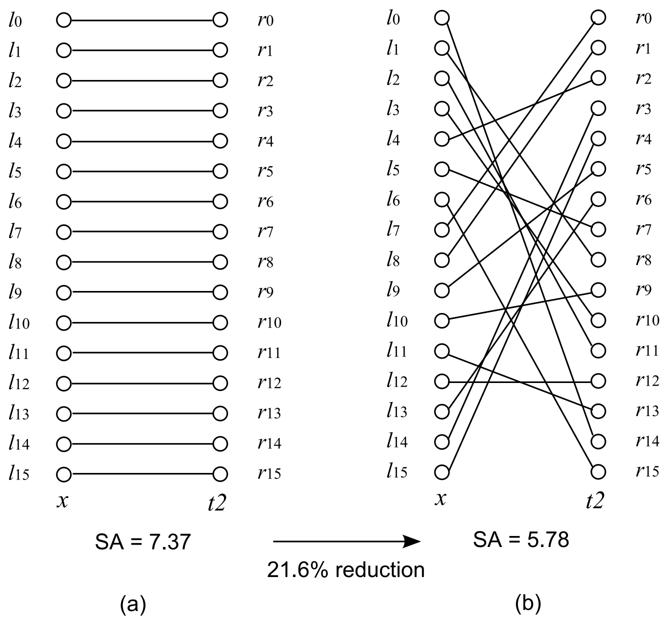


Fig. 3 (a) fixed bit ordering (b) bit reordering (optimal matching)

B. Minimum Weight Bipartite Matching Problem

A bit reordering problem can be formulated as a minimum weight bipartite matching (MWBM) problem. To understand a minimum weight perfect matching problem, some basic graph terminology referred to [10] is explained:

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 $e = (u, v)$, it is said that the endpoints of e are u and v; it is also said 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.

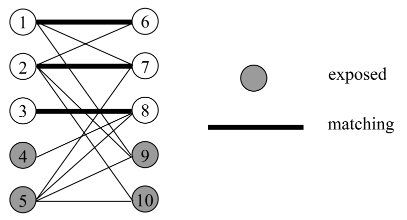


Fig. 4 Example for bipartite matching problem [10]

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; in other words, a matching is perfect if its cardinality is equal to $|A| = |B|$.

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

A 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, variables related to MWBM problem is defined as follows:

1) Left nodes in bipartite graph: $L = \{ l_1, l_2, \dots, l_n \}$, where n is the number of bit width of left variable. Left node denotes the bit position of left variable.
2) Right nodes in bipartite graph: $R = \{ r_1, r_2, \dots, r_n \}$, where n is the number of bit width of right variable. Right node denotes the bit position of right variable.
3) Edge incident to left node and right node: $E = \{ (i, j) \mid i = 1, 2, \dots, n, j = 1, 2, \dots, n\}$, where (i, j) is the edge incident to i-th left node and j-th right node.
4) Cost for each edge: c_{ij} for all $(i, j) \in E$ is the average switching activity calculated for a short-term input data pattern.

Jonker and Volgenant [11] proposed a refined algorithm of a minimum weight perfect matching, and showed a polynomial running time. It is notably faster than the Hungarian algorithm (a.k.a. Munkres' algorithm [12]) and several other linear assignment algorithms. The proposed method adopts the refined algorithm by Jonker and Volgenant to solve a MPMW problem in a polynomial running time.

C. Related Work

The conventional binding methods aim at obtaining binding solution to minimize switching activity in a specific component, i.e. functional unit, bus, register, .etc. Generally speaking, the methods use switching activity matrix (SAM) generated by inputting a random pattern to a DFG, and simulating the DFG for a sufficiently long iterations. When generating SAM, they use a fixed ordering, that is, each bit position from zero to fifteen coincides with each other.

The existing methods can be divided into the approach of finding optimal solution and that of obtaining close-to-optimal solution. In this paper, the former is denoted as optimal method, and the latter is denoted as heuristic method. The optimal method such as [13] finds optimal solution. But, because it is NP-hard, the calculation time increases exponentially with the size of problem. On the other hand, the heuristic methods such as [3], [14] find close-to-optimal solution with fast calculation time.

Chang, et al. [13] proposed a technique for reducing power consumption during the bindings of hardware components (registers, buses, and functional units). The problem is formulated as a min-cost multi-commodity flow problem and solved optimally. Because the multi-commodity flow problem is NP-hard, they restricted the domain of pipelined designs with a short latency.

In contrast, heuristic method finds close-to-optimal solution with faster computing time. Choi and Kim [3] proposed an efficient binding algorithm for power optimization in high-level synthesis. They exploited the property of efficient flow computations in a network so that it is applicable to practical designs while producing near-optimal results. Xing and Jong [14] proposed a look-ahead synthesis technique with backtracking for the reduction of switching activity in low power high-level synthesis, effectively reducing the probability for the solutions to fall into local minimum.

III. Binding Method Using Dynamic Bit Reordering

The proposed algorithm is described in Fig. 5. The proposed method consists of three parts: Part-1 is performed before implementing circuits, Part-2 is for the implementation of bus binding, and finally Part-3 is performed after implementing circuits.

```
Proposed_Method ( Scheduled DFG )
 1  Simulate DFG with random input patterns for 100,000 times.
 2  Obtain binding solution using conventional binding method.
 3  Implement bus binding by using binding solution obtained at STEP-2.
 4  Get variables Vi (i = 1, 2, ... , Nv) in DFG. /* Nv : number of variables */
 5  Save switching activity with real input patterns for a number of iterations.
 6  For 'Vi', i from 1 to Nv
 7      For 'Vj', j from 1 to Nv (i != j)
 8          Find optimal bit ordering by solving MWBM problem.
 9          Save the optimal bit ordering.
10      end for loop
11  end for loop
12  Switch bit ordering by using saved bit ordering information.
13  Repeat STEP 4 ~ 12 periodically.
```

Fig. 5 Proposed bus binding algorithm

1) Part -1 (Line 1-2): Obtain a bus binding solution.

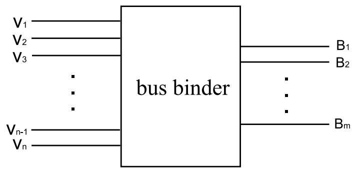
A bus binding solution is obtained using conventional bus binding method that is described in section II-C. Note that the proposed method adopts a fixed bus binding as conventional method does.

2) Part-2 (Line 3): Implement a bus binding.

The obtained bus binding solution in Part-1 is implemented in a real hardware circuit. Fig. 6 shows the functional block diagram to implement the proposed method. The diagram consists of memory, optimal bit ordering finder, and bus binder. Memory is necessary to save input data for a period during which a new optimal bit ordering is found. When new optimal bit ordering is found, then memory receives 'done' signal from optimal bit ordering finder and the saved input data is sent to bus binder. A bus binder with the new optimal bit ordering produces the reduced switching activity.

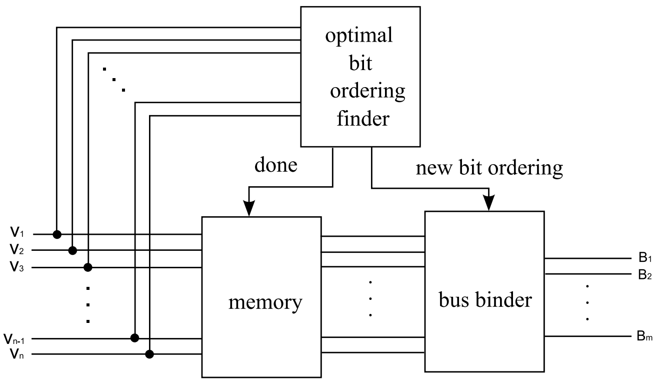
3) Part-3 (Line 4-13): Switch bit ordering periodically.

In Line 4-5, all variables in DFG are recognized and switching activity with real input data patterns are saved for short-term iterations. The number of iterations is the main factor that influences the performance of the proposed method, and is determined by experiment. The influence of the number of iterations on the performance of the proposed method will be described in Section IV.



(fixed bus binding & fixed bit ordering)

(a) conventional fixed bit ordering scheme



(fixed bus binding & dynamic bit ordering)

(b) proposed dynamic bit reordering scheme

Fig. 6 Dynamic bit reordering scheme

For the saved switching activity, the optimal bit ordering is found by solving minimum weight bipartite matching (MWBM) problem for all pair of variables. Note that it is not necessary to perform a bit reordering between the same variables because the fixed bit ordering with zero to fifteen coincided always produce minimum switching activity, i.e. zero.

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 III ~ IV: 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 method adopts the refined method by Jonker and Volgenant [11] to solve the problem of MWBM, and uses from 10 to 50 iterations to generate a short-term SAM. The proposed algorithm was implemented in C++ and executed in a Sun Sparc64-V workstation.

A. Comparison of Total Switching Activity

Table III shows the comparison of total switching activity (TSA). BIND_OPT is optimal binding method proposed in [13], and BIND_LP is heuristic binding method in [3]. Dynamic bit reordering is performed to each conventional method, and the proposed method is denoted as BIND_OPT + BRO and BIND_LP + BRO, respectively, where BRO stands for Bit Re-Ordering. The number in parenthesis shows a reduction factor to conventional methods.

The proposed method denoted as BIND_OPT + BRO gets the solution having 22.4-23.6% (average 22.9%) smaller TSA compared with BIND_OPT at the iteration number of 30. And the proposed method denoted as BIND_LP + BRO obtains the solution having 18.7-24.0% (average 20.3%) smaller TSA compared with BIND_OPT at the iteration number of 30.

The proposed dynamic bit ordering method gets better solution regardless of the type of conventional methods, i.e. optimal or heuristic method because the proposed method finds optimal bit ordering by solving MWBM problem and the bit ordering solution to this problem is independent of the bus binding solution. Also, the proposed method shows the similar performance for various benchmark circuits.

TABLE III
Comparison of total switching activity (TSA) (iteration = 30)

	BIND_ OPT	**BIND_OPT + BRO**	**BIND_LP**	**BIND_LP + BRO**
DIFF_EQ	101.90	88.96 (22.7%)	124.17	97.57 (21.4%)
EWF	–[a]	–[a]	307.41	249.62 (18.8%)
IIR	–[a]	–[a]	177.28	141.98 (19.9%)
FIR	168.54	147.64 (22.4%)	170.18	138.02 (18.9%)
TFIR	110.82	95.75 (23.6%)	112.25	85.32 (24.0%)
Lattice	–[a]	–[a]	163.38	130.05 (20.4%)
FFT	–[a]	–[a]	249.28	201.79 (18.7%)
FDCT	–[a]	–[a]	220.18	176.03

			(20.1%)
Average (%)	–[a]	22.9	20.3

a. memory overflow problem

B. Influence of Iterations on Total Switching Activity

Table IV shows the influence of iterations on total switching activity (TSA). As the number of iterations increases, the reduction of switching activity decreases from 34.9% to 12.0% (on average) because the distribution of switching activity of input data pattern becomes more uniform. Therefore, it is desirable to reduce the number of iterations as minimum as possible.

But, as the dynamic bit ordering is performed more often, the overhead of the dynamic power dissipated in the optimal bit ordering finder increases. That is, there is a trade-off between the number of iterations and the overhead of additional power dissipation.

TABLE IV
COMPARISON OF TSA ACCORDING TO ITERATION

	BIND_LP	**BIND_LP + BRO**				
		Iteration				
		10	**20**	**30**	**40**	**50**
DIFF_EQ	124.17	80.24 (35.4%)	86.53 (30.3%)	97.57 (21.4%)	106.24 (14.4%)	109.31 (12.0%)
EWF	307.41	204.55 (33.5%)	218.60 (28.9%)	249.62 (18.8%)	260.84 (15.2%)	271.50 (11.7%)
IIR	177.28	119.82 (32.4%)	134.22 (24.3%)	141.98 (19.9%)	145.64 (17.9%)	156.95 (11.5%)
FIR	170.18	110.48 (35.1%)	124.55 (26.8%)	138.02 (18.9%)	142.13 (16.5%)	148.11 (13.0%)
TFIR	112.25	69.51 (38.1%)	83.88 (25.3%)	85.32 (24.0%)	94.30 (16.0%)	97.83 (12.9%)
Lattice	163.38	98.60 (39.7%)	121.77 (25.5%)	130.05 (20.4%)	134.95 (17.4%)	143.81 (12.0%)
FFT	249.28	173.10 (30.6%)	179.66 (27.9%)	202.79 (18.7%)	213.21 (14.5%)	221.21 (11.3%)
FDCT	220.18	143.78 (34.7%)	152.12 (30.9%)	176.03 (20.1%)	185.22 (15.9%)	194.26 (11.8%)
Average (%)		34.9	27.5	20.3	16.0	12.0

V. CONCLUSION

An effective low bus binding technique is proposed to obtain binding solution having smaller total switching activity by switching an optimal bit ordering between the bits of variables within a bus.

Experimental result shows that the proposed method obtains a binding solution having 12.0-34.9% smaller TSA compared with conventional methods.

A fixed bus binding is used in the proposed method as conventional methods do. But, whereas conventional methods use a fixed ordering between bits of variables, dynamic bit reordering is adopted in the proposed method. Extending the proposed method to adopt a dynamic bus binding rather than a fixed bus binding is future work.

REFERENCES

[1] V. Soteriou and L. Peh, "Design space exploration of power-aware on/off interconnection networks," in *Proc. Int. Conf. Comput. Des.*, Oct. 2004, pp. 510–517.
[2] 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
[3] 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.
[4] W. C. Cheng and M. Pedram, "Power-optimal encoding for DRAM address bus," in *Proc. Int. Symp. Low-Power Electron. Design*, pp. 250-252, 2000.
[5] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power efficient bus interfaces, " IEEE Trans. Computer-Aided Design, vol. 19, pp. 969-980, Sept. 2000.
[6] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf, "An adaptive dictionary encoding scheme for SOC data buses, " in Proc. Design Automation Test Eur. Conf. Exihib., pp.1059-1064, 2002.
[7] A. P. Chandrakasan and R. W. Brodersen, Low power digital CMOS design, Kluwer Academic Publishers, pp. 235-245, 1995.
[8] 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.
[9] J. Kim and J. Cho, "Low power bus binding exploiting optimal substructure," IEICE Trans. on fundamentals of electronics communications and computer sciences, will be published in Jan. 2011.
[10] M. X. Goemans, Lecture notes on bipartite matching, unpublished, 2007.
[11] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, vol. 38, issue 4, pp. 325-340, 1987.
[12] J. Munkres, "Algorithms for the assignment and transportation problems," Journal of the Society for Industrial and Applied Mathematics, vol. 5, no. 1, pp. 32-38, 1957.
[13] J. Chang and M. Pedram, "Module assignment for low power," " in *Proc. Eur. Design Automation Conf.*, pp.376-381, 1996.
[14] 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.