World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:5, No:12, 2011

Extended Low Power Bus Binding Combined with Data Sequence Reordering

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

***Abstract*—**In this paper, we address the problem of reducing the switching activity (SA) in on-chip buses through the use of a bus binding technique in high-level synthesis. While many binding techniques to reduce the SA exist, we present yet another technique for further reducing the switching activity. Our proposed method combines bus binding and data sequence reordering to explore a wider solution space. The problem is formulated as a multiple traveling salesman problem and solved using simulated annealing technique. The experimental results revealed that a binding solution obtained with the proposed method reduces 5.6-27.2% (18.0% on average) and 2.6-12.7% (6.8% on average) of the switching activity when compared with conventional binding-only and hybrid binding-encoding methods, respectively.

***Keywords*—**low power, bus binding, switching activity, multiple traveling salesman problem, data sequence reordering

I. Introduction

DYNAMIC power is proportional to $P_{trans} \cdot C_L \cdot V_{dd}^2 \cdot f_{clock}$, where P_{trans} is the probability of an output transition (also referred to as the switching activity), C_L is the load capacitance, V_{dd} is the supply voltage, and f_{clock} is the frequency of the system clock. In low-power high-level synthesis, it is usually effective to reduce the switching activity (referred to as SA) in functional units, buses, and registers without degrading the circuit performance. Some conventional low power methods to reduce the SA include scheduling, binding, data sequence reordering, and data encoding techniques [1]-[9].

An existing data sequence reordering technique is outlined in Refs. [6, 7], where the problem is formulated using the well-known Travelling Salesman Problem (TSP) and solved through the application of a greedy method. One data encoding technique is bus invert coding [9], where data is sent inverted or non-inverted depending on which mode leads to the least number of transitions. Conventional bus binding and data encoding techniques can be effectively combined.

The first hybrid technique to combine bus binding and data encoding with bus invert coding was proposed by Sankaran et al. [10], who proposed a technique for combining bus binding, bus-invert encoding, and bit-line re-ordering primarily for minimizing the crosstalk-induced switching activity between bit-lines in a bus during high-level synthesis. Bit-line re-ordering switches bit-lines in order to reduce crosstalk [3]. However, they did not consider the switching of the data sequence between control steps, but instead considered only bus line reordering within a control step to reduce the crosstalk between bit lines. If the exchange of data between control steps is considered, the quality of the binding solution can be improved.

Motivated by this fact, we compared previous methods with the technique proposed in this paper under the same experimental conditions (the results are given in Section IV). To the best of our knowledge, this is the first attempt to combine bus binding with data sequence reordering in order to reduce the switching activity in on-chip buses.

The contributions of this work can be summarized as follows:

1) The spectrum of the solution space was increased by combining data reordering with bus binding. We also formulated the problem using the multiple TSP (mTSP) [11] and solved the mTSP using simulated annealing .

2) When data reordering was applied to bus binding, conventional bus binding did not resolve the problem of data dependency in a data flow graph. However, we successfully implemented a data reordering technique by taking advantage of line buffer logic without sacrificing throughput.

We demonstrated that the overhead on the line buffer logic is small and thus, the proposed technique can be used in practical applications.

II. Problem Definition

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,

$$\mathrm{TSA} = \Sigma_{(\forall k \text{ of buses})} \mathrm{SA}^k \quad (1)$$

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].

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).

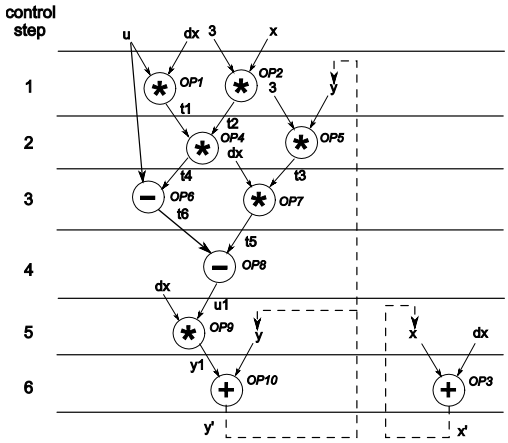

Fig. 1 Data flow graph of differential equation solver

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

The problem of bus binding can be formulated as in Ref. [12].

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

1) A finite set of operators, denoted as $O = \{o_1, o_2, \ldots, o_p\}$.

2) A finite set of variables of operators, denoted as $V = \{v_1, v_2, \ldots, v_q\}$.

3) A finite set of control steps, denoted as $C = \{c_1, c_2, \ldots, c_r\}$

4) A scheduling function $S_f : O \rightarrow C$, where $S(o_i) = c_j$ denotes that the operator corresponding to $o_i \in O$ is scheduled at control step c_j.

Let B represent a finite set of buses, denoted as $B = \{b_1, b_2, \ldots, b_s\}$, 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 $j = 1, 2, \ldots, r$.

The problem of "bus binding" can now be stated as follows.

Problem of 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 a variable corresponding to $v_i \in V$ and scheduled at control step c_k is bound to bus $b_j \in B$ at control step c_k, V is a set of variable of operators, and B is a set of buses. ■

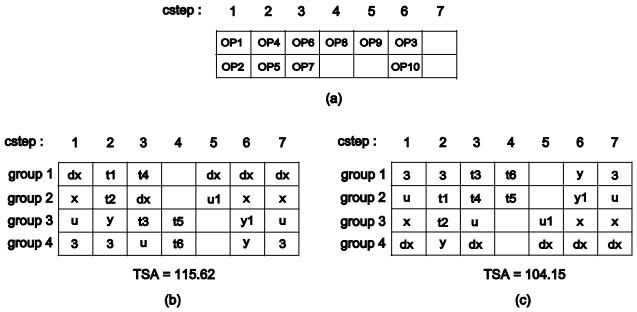

Fig. 2 (a) Scheduled operator table; (b) and (c) show examples of bus bindings with different TSAs

A scheduled operator table extracted from the scheduled DFG in Fig. 1 is shown in Fig. 2(a); two different examples of bus bindings are shown in Fig. 2(b) and 2(c). In this DFG, four buses are allocated to implement the input variables of ten operators. Here, "group" denotes the specific group of bus binding that starts at cstep 1 and ends at cstep 7.

The problem of low-power bus binding is now formally defined as follows:

Problem of Low-power Bus Binding:
Input: A scheduled DFG = (O, V, C, S_f)
Output: Bus binding with minimum TSA
Bus Binding: $M_f : V \times C \rightarrow B \times C$ ■

III. THE PROPOSED HYBRID BUS BINDING METHOD

In this section, the proposed hybrid bus binding algorithm and its logical implementation are described. The performance of the algorithm is then evaluated.

A. Bus Binding Combined with Data Sequence Reordering

The combination of bus binding and data sequence reordering serves to rearrange variables in a bus binding table in two dimensional directions, as illustrated in Fig. 3. That is, while the aim of bus binding is to locate variables to buses in the same control step vertically, data sequence reordering serves to arrange variables in a different control step horizontally.

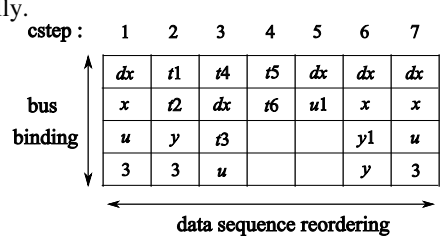

Fig. 3 Bus binding combined with data sequence reordering

We formulate the problem of combining bus binding with data sequence reordering (we refer to the problem as the "e*xtended low power bus binding problem*") based on a mTSP with a fixed destination [11]. The mTSP is a generalization of the well-known TSP, where more than one salesman is allowed to be used in the solution. The fixed destination comes from the fact that if multiple depots exist with a number of salesmen located at each depot, the salesmen can return to their original depot after completing their tour.

In our mTSP formulation, there exists a special node denoted as the *super node* that corresponds to *depot* in the original mTSP, as shown in Fig. 4. Each super node contains a *node* that corresponds to a variable in the same control step. The node in the super node must not be traversed more than once by the same traveling salesman and every salesman should visit each super node at least once. That is, each salesman at a start node is required to visit k nodes (excluding the virtual nodes shown in Fig. 4), $k \leq m$, where m is the number of control steps, and return to the starting node.

The cost (i.e., switching activity) assigned to each edge is obtained from Table 1. The mTSP will determine the traveling route so as to minimize the total switching activities acquired by all salesmen.

The problem of extended low-power bus binding is now formally stated as follows:

Problem of extended low-power bus binding

- Instances:
  1) A directed graph $G = (V, E)$, where edge cost $C(e) \in R$ for $e \in E$.
  2) Set of super nodes $S = \{ s_1, s_2, \ldots, s_r \}$, where r is the number of control steps, each of which contain m nodes, where m is the number of buses. All nodes in a super node exist in the same control step.
  3) Set of traveling salesmen, $T = \{ t_1, t_2, \ldots, t_m \}$.

- Configurations:
  1) f_i: $E \times N \rightarrow Z$, $i = 1, \ldots, m$, and $Z = \{0, 1\}$
     That is, $f_i(e, n) = 1$ if traveling salesman 'i' visits a node v (in super node s) as an 'n'-th destination (from the start node) connected to edge 'e' among all possible edges. If no salesmen goes along edge 'e', then $f_i(e, n) = 0$.
  2) g_i: $E \times N \rightarrow V$, $i = 1, \ldots, m$, and $N = \{ 0, 1, \ldots, r \}$
     That is, $g_i(e, n) = v$, where traveling salesman i visits a node v, connected to edge e, as an n-th destination. If no salesmen goes along edge e, then $g_i(e, n) =$ NULL.
  3) h_i: $E \times N \rightarrow S$, $i = 1, \ldots, m$, and $N = \{0, 1, \ldots, r\}$
     That is, $h_i(e, n) = s$, where traveling salesman i enters a super node s, connected to edge e, as an n-th destination. If no salesmen goes along edge e, then $h_i(e, n) =$ NULL.

- Constraints:
  1) Each node is to be visited by one salesman only.
     For $n \in N$ and i,
     $\sum_{e \in E} f_i(e, n) = 1$.
     For $e_1, e_2 \in E$, $n_1, n_2 \in N$ and all i, j,
     $g_i(e_1, n_1) \neq g_j(e_2, n_2)$ if $(i \neq j \vee e_1 \neq e_2 \vee n_1 \neq n_2)$
  2) Each salesman should visit a super node once.
     For $e_1, e_2 \in E$, $n_1, n_2 \in N$ and all i, j,
     $h_i(e_1, n_1) \neq h_j(e_2, n_2)$ if $n_1 \neq n_2$
  3) All salesmen should visit the nodes in the same super node as the next visiting nodes.
     For $e_1, e_2 \in E$, $n_1, n_2 \in N$ and all i, j,
     $h_i(e_1, n_1) = h_j(e_2, n_2)$ if $(i \neq j \wedge n_1 = n_2)$
  4) The starting node of each bus must match the ending node of each bus.
     For $e \in E$ and all i,
     $g_i(e, r\text{-}1) = s_r = s_1$, where r is the last control step.

- Objective:
  Minimize: For $e \in E$ such that $f_i(e, n) = 1$ for all i, n,
  $C(f) = \sum_{e \in E} C(e)$ ■

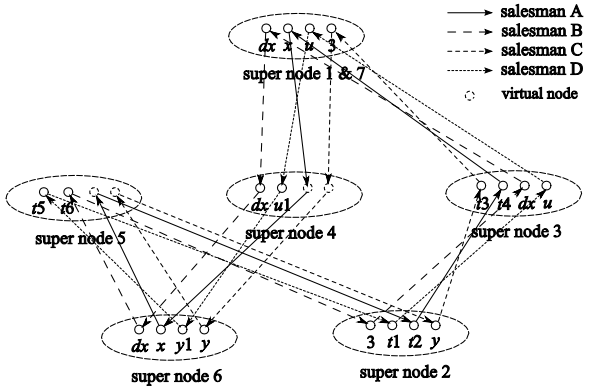

Fig. 4 A solution to extended low-power bus binding that solves an mTSP for the DFG in Fig. 1 (TSA = 95.12)

A solution to the problem of extended low-power bus binding with respect to the DFG in Fig. 1 is shown in Fig. 4.

We solved the formulated mTSP using simulated annealing. In our extended low-power binding problem, the number of nodes in each super node may vary due to empty nodes, which are denoted as *virtual nodes* in Fig. 4.

B. Functional Implementation

The functional diagram of the DFG in Fig. 1 is shown in Fig. 5. The diagram is used to implement a binding solution by solving the mTSP. A data reordering process is implemented by line buffer logic. Line buffer logic receives input data and the results generated by functional units, stores the data temporarily in the line buffer, and finally reorders the data sequence to resolve data dependency between data of the previous and next control step. The stored data are loaded into on-chip buses and multiplexed with other on-chip bus data. The multiplexed data is then sent to functional units.

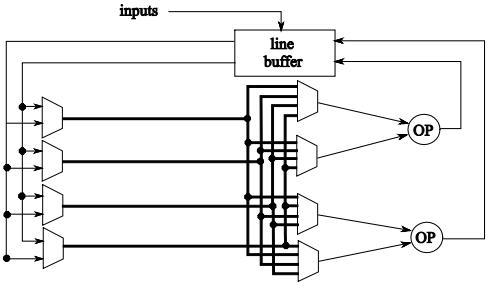

Fig. 5 Functional implementation of the DFG shown in Fig. 1

The manner in which the issue of data dependency is resolved in the proposed method is shown in Fig. 6; data flows of the conventional method and the proposed method are compared.

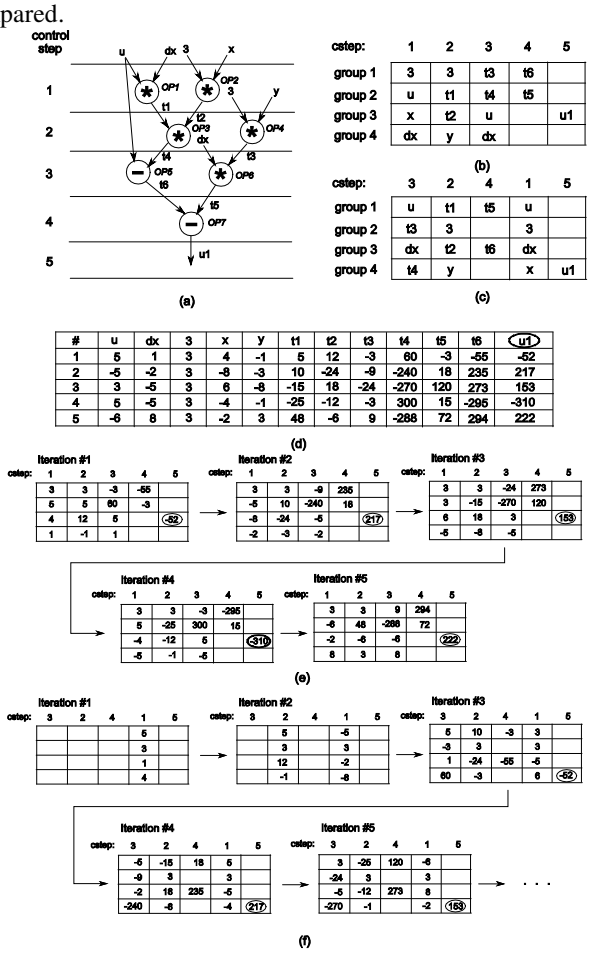

Fig. 6 Functional Data flows of conventional and proposed methods. (a) DFG example, (b) bus binding by conventional method, (c) bus binding by proposed method, (d) input and output data for DFG (e) data flow by conventional method, and (f) data flow by proposed method

The basic idea behind the proposed method is that the data processed in the previous iteration are stored in line buffer and used in the next iteration for the case where there is a reverse order in the control steps.

A reduced form of the DFG in Fig. 1, redrawn for convenience, is shown in Fig. 6(a). A binding table found by the conventional binding method is displayed in Fig. 6(b), while the binding table obtained with the proposed method is shown in Fig. 6(c). In Fig. 6(c), the control steps are reordered as $3 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 5$.

Assume that the stream of input data is applied as shown in Fig. 6(d) for five iterations. The results of the conventional binding table for consecutive iterations are shown in Fig. 6(e); the output (depicted with circles) is generated per iteration.

The results of the proposed binding method are shown in Fig. 6(f). In the first iteration, the variables in control steps 3, 2, and 4 cannot be processed due to data dependency. For example, to process the operations in control step 3, we need outputs from the previous control steps 1 and 2. Therefore, only variables in control step 1 are processed in the first iteration and are stored in line buffer.

In the second iteration, the variables in control step 2 can be processed since the result of control step 1 in the first iteration is now available in the line buffer. At the same time, new values corresponding to the #2 input data set (shown in Fig. 6(d)) are assigned to the input variables for control step 1. Consecutive iterations are performed in the above manner and the first output corresponding to the #1 input data set is obtained after three iterations.

C. Analysis of Latency and Throughput

Due to the data reordering process, overheads such as buffer and latency are required. In the example shown in Fig. 6, three iterations are needed to generate the first output and thus, the latency is three (iterations).

A comparison of the latency and throughput between the conventional binding method and the proposed method is shown in Fig. 7; the total number of control steps of a DFG is assumed to be three.

The results obtained with the conventional binding method are shown in Fig. 7(a); the latency is one (iteration) and the throughput is one (output/iteration). The results of the proposed method when there is no feedback variable in a DFG are given in Fig. 7(b). For example, the DFG in Fig. 1 has feedback variables *x*' and *y*'. The latency and throughput for the DFG with the given feedback variables are shown in Fig. 7(c).

In Fig. 7(b), the latency is three (iterations) and the throughput is one (output/iteration). As such, it acts like a pipeline process. The throughput becomes one after three control steps because output variables, such as *x*' and *y*' in the DFG of Fig. 1, are obtained at every iteration of the DFG after the initial latency of three iterations. In Fig. 7(c), it is assumed that the A, B, and C data sets are independent from each other, i.e., there are no interactive paths between them. For example, in a multi-core CPU system, a group of threads processed in one CPU is data set A, while another group of threads processed in another CPU is data set B. In this case, the latency is three (iterations) and the throughput is one (output/iteration), as in the case without feedback variables in Fig. 7(b).

In Fig. 7(b) and 7(c), the worst latency of the proposed method is the same as the number of control steps. However, if the sequence of the reordered index of control steps is in a specific order, the average latency can be reduced. For example, in the second iteration in Fig. 6(f), the remaining steps 3, 4, and 5 constitute a monotonic increasing sequence and thus, the variables in these steps can be processed together in the third iteration.

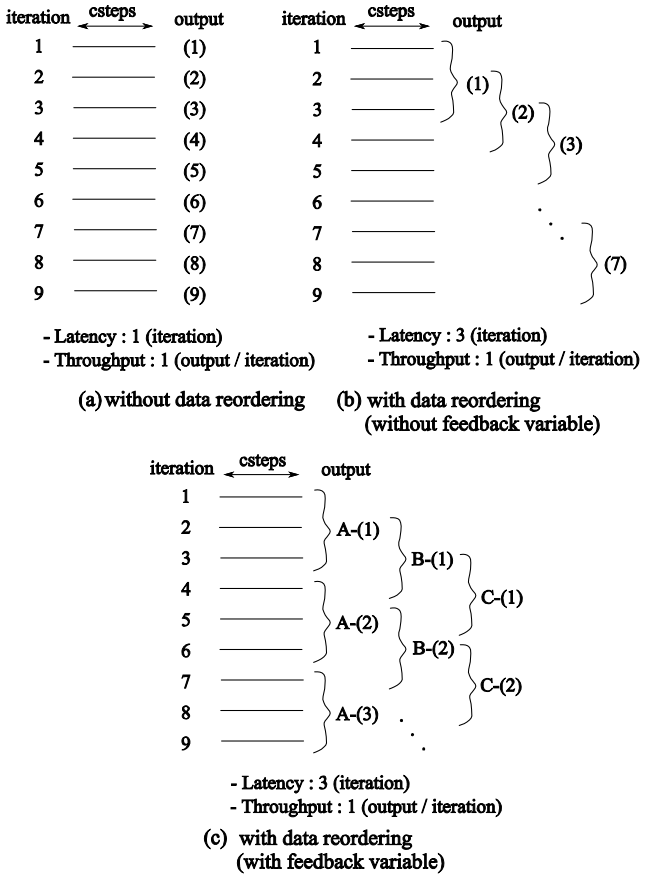

Fig. 7 Comparison of latency and throughput

IV. Experimental Result

To demonstrate the effectiveness of our solution to the extended low-power binding problem, eight high-level datapath synthesis benchmark circuits were used in our experiments, as shown in Tables 2~4. These circuits are as follows: 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. Our proposed algorithm was implemented in C++ and executed in a Sun Sparc64-V workstation.

A. Comparison of Total Switching Activity

A comparison of the TSA results is shown in Table 2; BIND_LP is a heuristic binding method proposed in Ref. [4], BIND_BI is a hybrid binding-encoding method in Ref. [10], and BIND_DSR is the proposed method (DSR denotes data sequence reordering).

In Table II, the number in parentheses denotes a reduction factor to a binding solution obtained by BIND_LP. Note that the bit ordering algorithm that was incorporated into the original BIND_BI method is not used in this experiment so that the binding results could be compared under the same conditions.

The proposed method yielded a binding solution with an SA that was 5.6-27.2% (18.0% on average) and 2.6-12.7% (6.8% on average) lower than those of the BIND_LP and BIND_BI methods, respectively.

TABLE II
COMPARISON OF TOTAL SWITCHING ACTIVITY (TSA)

(Initial temperature = $1.0*10^{3}$℃, Final temperature = $1.0*10^{-2}$ ℃, step: T = 0.95T)

DFG	(# of functional units, # of variables)	BIND_LP	BIND_BI	BIND_DSR
DIFF_EQ	(10, 15)	126.77	98.43 (22.4%)	95.12 (25.0%)
EWF	(20, 38)	310.85	251.28 (19.2%)	227.29 (26.9%)
IIR	(15, 29)	182.94	162.13 (11.4%)	144.57 (21.0%)
FIR	(15, 22)	174.94	169.57 (3.1%)	165.20 (5.6%)
TFIR	(13, 26)	118.03	112.58 (4.6%)	107.79 (8.7%)
Lattice	(20, 20)	171.23	142.84 (16.6%)	124.69 (27.2%)
FFT	(24, 32)	250.87	224.12 (10.7%)	211.73 (15.6%)
FDCT	(24, 32)	222.88	201.29 (9.7%)	190.56 (14.5%)

B. Buffer Logic Overhead and Latency/Throughput

An RTL code was written with Verilog hardware description language (HDL) after obtaining a binding solution by solving the mTSP. The RTL code was simulated with a control step of 10 ns. A comparison of the buffer logic overhead and latency/throughput between the binding-only method and the proposed method is shown in Table III.

TABLE III
COMPARISON OF THE BUFFER LOGIC OVERHEAD AND LATENCY/THROUGHPUT

(control step = 10 ns)

*Assumption: Independent data sets are applied to a DFG.

DFG	# of control steps	Buffer logic (# of F/Fs)	Latency (us)	Throughput (output / iteration)
		Binding-only/ Proposed Method	Binding-only/ Proposed Method	Binding-only/ Proposed Method
DIFF_EQ	7	0 / 7	0.07 / 0.21	1 / 1
EWF	16	0 / 16	0.16 / 1.28	1 / 1
IIR	9	0 / 9	0.09 / 0.63	1 / 1
FIR	9	0 / 9	0.09 / 0.45	1 / 1
TFIR	10	0 / 10	0.10 / 0.80	1 / 1
Lattice	12	0 / 12	0.12 / 0.36	1 / 1
FFT	13	0 / 13	0.13 / 1.04	1 / 1
FDCT	13	0 / 13	0.13 / 0.78	1 / 1

It is assumed that there are independent data sets and the number of the independent data sets is same as the number of control steps. The number of flip-flops regarded as the buffer logic overhead is also the same as the number of control steps. All benchmark circuits are successfully implemented with the latency and throughput, as described earlier. The latency is equal to or smaller than (number of control steps)^2 × (control step), while the throughput is equivalent to (number of control steps) × (control step).

While the response time to obtain the first outcome is delayed due to the increased latency and the area overhead increases due to the added buffer logic, the benefit of the reduced switching activity in low-power on-chip buses is more critical to power consumption in an overall chip when compared with the side effects.

The throughput is more important than the latency in many applications, such as a video streaming (throughput determines the quality of the video), microprocessors that execute millions or billions of instructions per second, and non-recursive digital filtering.

V. Conclusion

We propose an extended low-power bus binding solution combined with the data reordering technique to reduce the total switching activity in on-chip buses. It was shown that the bus binding solution obtained by the proposed method has a lower switching activity since the technique explores a wider spectrum of solution space.

If a greater increase in the latency to obtain the first output is permissible, a specific constraint for the mTSP formulation, such as "the same super node as the next destination, i.e. constraint 3," can be released. In this case, a better binding solution can be expected due to the exploration of a wider solution space.

References

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

[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] C. Lyuh and T. Kim, "Coupling-Aware High-Level Interconnect Synthesis," IEEE Trans. Computer-aided design of integrated circuits and systems, vol. 23, no. 1, pp. 157–164, 2004.

[4] 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.

[5] 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.

[6] M. Yoon, "Sequence-switch coding for low-power data transmission," IEEE Trans. on VLSI Syst., vol. 12, no. 12, pp. 1381-1385, 2004.

[7] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 12, pp. 1325-1333, 1998.

[8] S. K. Wong and C. Y. Tsui, "Re-configurable bus encoding scheme for reducing power consumption of the cross coupling capacitance for deep sub-micron instruction bus," *in Proc. DATE*, vol. 1, pp. 130–135, 2004.

[9] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," IEEE Trans. VLSI Syst., vol. 3, pp. 49-58, 1995.

[10] 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.

[11] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," Omega, vol. 34, no. 3, pp. 209-219, 2006.

[12] 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, 2011.