IPSJ Transactions on System LSI Design Methodology Vol. 3 81–90 (Feb. 2010)

Regular Paper

Globally Optimal Time-multiplexing of Inter-FPGA Connections for Multi-FPGA Prototyping Systems

Masato Inagi,[†1] Yasuhiro Takashima[†2]
and Yuichi Nakamura[†3]

Multi-FPGA prototyping systems are widely used to verify logic circuit designs. To implement a large circuit using such a system, the circuit is partitioned into multiple FPGAs. Subsequently, sub-circuits assigned to FPGAs are connected using interconnection resources among the FPGAs. Because of limited resources, time-multiplexed I/Os are used to accommodate all signals in exchange for system speed. In this study, we propose an optimization method of inter-FPGA connections for multi-FPGA systems with time-multiplexed I/Os to shorten the verification time by accelerating the systems. Our method decides whether each inter-FPGA signal is transferred by a normal I/O or a time-multiplexed I/O, which is slower than a normal I/O but can transfer multiple signals. Our method optimizes inter-FPGA connections not only between a single FPGA pair, but among all the FPGAs. Experiments showed that for four-way partitioned circuits, our method obtains an average system clock period 16.0% shorter than that of a conventional method.

1. Introduction

With increasing size of VLSI circuits, the design and fabrication costs of VLSIs are also rapidly increasing. In this situation, if a bug is found after fabrication begins, the cost of revising the VLSI design and refabricating the VLSIs is large. Thus, VLSI designs are verified at each stage of the design flow to find and remove bugs at as early a stage as possible. In one of the verification processes, logic verification, a circuit is verified by determining whether its required logic functions are implemented appropriately. Logic verification is one of the most important functions of the VLSI circuit verifications, since it is performed at an early stage of the VLSI design flow. Several methods are used to perform logic verification of a VLSI circuit design. The simplest is to simulate the functions of the circuit design by software on a processor. This is a reasonable way because it requires only a general-purpose processor and a software circuit simulator. In contrast, it is very slow compared with the target VLSI circuit, since such a software simulator emulates each logic gate one by one and the number of logic gates is very large (e.g., 100 million). Since test bench data sets are often very large (e.g., a dozen-hour video data stream processed by the target VLSI circuit), the slow speed of software simulators is critical for logic verification of a large circuit. Another way is to use a hardware simulation accelerator. That greatly accelerates software circuit simulation by up to several MHz. In contrast, such an accelerator costs more than one million dollars in some cases. Thus, it is economically difficult to perform logic verification of a circuit design in parallel by using several hardware simulation accelerators. The third way is to use an FPGA prototyping system. Since an FPGA is a VLSI chip whose circuit is electrically reconfigurable, it is often used to prototype a circuit. An FPGA prototyping system is also used to verify the circuit. Due to the limited capacity of an FPGA, a multi-FPGA prototyping system [4),8)] composed of multiple FPGAs is often used to verify a large circuit design. Such a system is much cheaper than hardware simulation accelerators because it consists of general-purpose FPGAs. Besides, such a system is faster (e.g., several dozen MHz) than accelerators. Thus, multi-FPGA prototyping systems are widely used.

To perform logic verification using a multi-FPGA prototyping system, a large circuit must be partitioned into sub-circuits, each of which fits a single FPGA. This process, called circuit partitioning, has been studied for a few decades [e.g., Ref. 12), 6), 11) and 1)]. Since the number of I/O pins of an FPGA is very limited, the number of I/O signals of a sub-circuit has typically been minimized to fit a single FPGA. However, even the minimized number of I/O signals of a sub-circuit often exceeds the number of I/O pins of an FPGA. To overcome the problem, Babb, et al. [2)] introduced time-multiplexing of I/O pins. A time-multiplexed I/O pin transfers multiple I/O signals by time-division. This technique dramatically increases the acceptable number of I/O signals of a sub-circuit. In contrast, this technique makes a prototyping system much slower since inter-FPGA signal delay is considerably increased by time-multiplexing of I/O

†1 Hiroshima City University
†2 The University of Kitakyushu
†3 System IP Core Laboratories, NEC Corporation

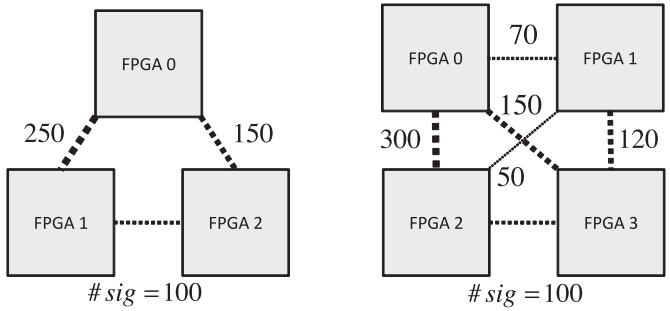

Fig. 1 Variation in the number of signals #sig among FPGA pairs.

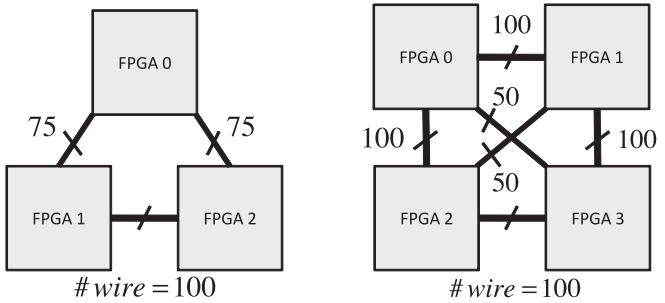

Fig. 2 Variation in the number of wires #wire among FPGA pairs.

pins. Each signal must wait for its turn to transfer. To ease this problem, we proposed an optimization method [10] of inter-FPGA connections for 2-FPGA systems, in which both normal I/O pins and time-multiplexed I/O pins are used for fast signal transfer and wide signal transfer, respectively. Hereinafter, we call this method OTM-2. In this study, we extend this method to handle multi-FPGA prototyping systems with many FPGAs. OTM-2 is also applicable to multi-FPGA prototyping systems. However, it does not distinguish the inter-FPGA signals of a pair of FPGAs, from those of another. Thus, it ignores the variation in the number of I/O signals from one pair of FPGAs to another (**Fig. 1**). Our new method considers this variation and obtains globally optimal solutions. It can also handle the variation in the number of wires between a pair of FPGAs (**Fig. 2**). This makes the method more practical, because in industrial multi-FPGA prototyping systems, different pairs of FPGAs often have different numbers of wires.

The rest of this paper is organized as follows. In Section 2, some definitions are given. In Section 3, we describe our proposed method. Subsequently, we present our experiments in Section 4 and give conclusions in Section 5.

2. Preliminaries

2.1 Definitions

A circuit is modeled as a hypergraph $G(V, E)$, called a circuit graph, which represents a circuit net-list. V is the set of vertices corresponding to cells (e.g., gates, flip-flops), including primary I/Os. Hereinafter, a vertex $v \in V$ and the cell corresponding to v are used interchangeably. E is the set of hyper-edges corresponding to nets that electrically connect cells. Each element $e \in E$ is the set of vertices to which the net corresponding to e connects. Hereinafter, a hyper-edge $e \in E$ and the net corresponding to e are used interchangeably.

Decomposition of a circuit into sub-circuits is called circuit partitioning. k-way partitioning of a circuit graph $G(V, E)$ gives vertex sets $V_1, V_2, \ldots, V_k$ such that $V_i \neq \emptyset$, $\bigcup_{i=1}^{k} V_i = V$, and $V_i \cap V_j = \emptyset$ $(i \neq j)$. V_i $(i = 1, 2, \ldots, k)$ are called blocks. A set $\Phi = \{V_1, V_2, \ldots, V_k\}$ of vertex sets is called a k-way partition. $e \in E$ is called a cutnet iff $\exists i, j, \{i \neq j, e \cap V_i \neq \emptyset \wedge e \cap V_j \neq \emptyset\}$. Since in this study it is assumed that one FPGA contains only one block and each block is assigned to an FPGA in advance, a block and its corresponding FPGA are used interchangeably. Let S be the set of all the pairs of FPGAs (V_i, V_j) $(1 \leq i < j \leq k)$.

Given a circuit graph G and its partition Φ, we represent the connections between inter-FPGA signals by a directed graph $G_{\mathrm{i/o}}(V_{\mathrm{i/o}}, E_{\mathrm{i/o}})$, called an inter-FPGA signal graph, where $V_{\mathrm{i/o}} = \{v_1, v_2, \ldots, v_m\}$ is the set of vertices corresponding to the inter-FPGA signals, and $E_{\mathrm{i/o}} = \{e_1, e_2, \ldots, e_l\}$ is the set of directed edges corresponding to the direct signal paths between inter-FPGA signals (**Fig. 3**). Let $V_{\mathrm{FI}}(v)$ $(v \in V_{\mathrm{i/o}})$ be the set of inter-FPGA signals (vertices in $V_{\mathrm{i/o}}$) from which there is at least one signal path to v through no other inter-FPGA signals. A vertex $v \in V_{\mathrm{i/o}}$ which has no incoming edges is called a source of $G_{\mathrm{i/o}}$. A vertex $v \in V_{\mathrm{i/o}}$ which has no outgoing edges is called a sink of $G_{\mathrm{i/o}}$. Each FPGA pair $a \in S$ has its inter-FPGA signals $V_{\mathrm{i/o},a}$.

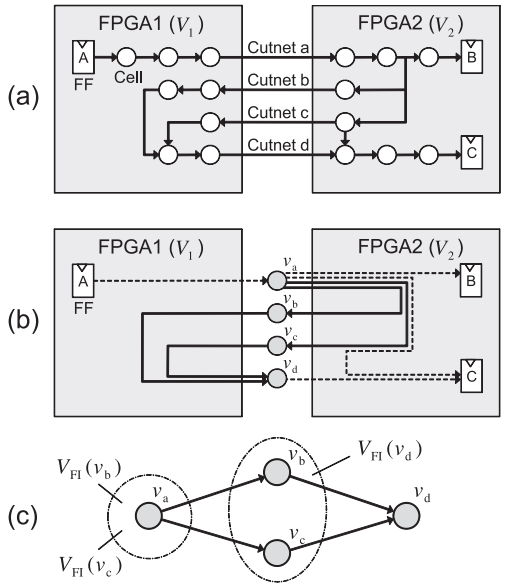

Fig. 3 (a) Circuit graph, (b), (c) Inter-FPGA signal graphs.

2.2 Time-multiplexed I/O

A time-multiplexed I/O (TM I/O) [e.g., used in Ref. 2)] is an I/O that serially transfers multiple signals from one device to another. It eases the problem of the limited number of I/O pins on a device (e.g., an LSI, an FPGA). Several implementations of TM I/Os exist. We adopt a simple implementation (**Fig. 4**) to maintain the reliability of our prototyping system. In our implementation, a TM I/O consists of a parallel/serial converter, an I/O pin of the sender device, an I/O pin of the receiver device, a wire between them, and a serial/parallel converter. The converters are triggered by an I/O clock, which is set to be faster than the system clock in order to transfer multiple signals within a system clock period. A signal is transferred in an I/O clock period, which is called a *slot*. A set of signals are transferred by using a set of slots, called a *stage*. #slot refers to the number of slots in a stage, which is the size of the stage. In a system clock period, a TM I/O repeats stages until the signals of the logic circuit converge. #stage refers to the number of stages repeated in a system clock period. In our

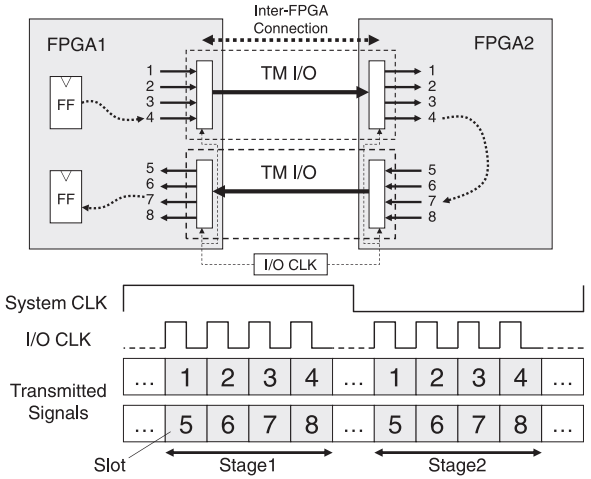

Fig. 4 Time-multiplexed I/O.

system, each TM I/O works with the same #slot and #stage. Empirically, #slot is set to be greater than or equal to 4. Each signal transferred by a TM I/O waits for the completion of the transfer for at most #slot I/O clock periods at the TM I/O. Thus, TM I/Os introduce a large delay to the circuit. Due to this large delay, the delays caused by other parts can be ignored for practical purposes. In our prototyping system, both normal I/O and TM I/O are used to accommodate inter-FPGA signals, alleviating system slowdown. We say a selection of signals to be time-multiplexed, a TM I/O signal selection, or simply a signal selection.

2.3 I/O Constraint

Let W_a and X_a be the number of wires for inter-FPGA I/Os and the number of time-multiplexed signals between FPGAs of an FPGA pair $a \in S$, respectively. Let T_a be the number of TM I/Os between an FPGA pair a. The number of signals transferred by a normal I/O can be represented by $|V_{\mathrm{i/o},a}| - X_a$. Then,

$$T_a + |V_{\mathrm{i/o},a}| - X_a \leq W_a$$

must be satisfied to transfer the signals using W_a wires. This is the I/O constraint of multi-FPGA prototyping systems with TM I/Os.

2.4 Delay in Multi-FPGA Systems with TM I/Os

Let M_i be the number of signals transferred by a TM I/O i. Then, #slot $\geq$

$\max_i M_i$. Let N_p be the number of TM I/Os on a FF-to-FF signal path p. N_p is called the *TM-depth* of path p. N_p is determined by the signal selection. Then, #stage $\geq \max_p N_p$. $\max_p N_p$ is called the *TM-depth* of the system, which is also the lower limit of #stage. Let $C_{\rm sys}$ and $C_{\rm i/o}$ be the system clock period and I/O clock period, respectively. A slot corresponds to an I/O clock period, a stage consists of #slot slots, and a system clock period contains #stage stages. Thus,

$C_{\rm sys} \geq$ #slot $\cdot$ #stage $\cdot C_{\rm i/o}$.

Since the minimum value of $C_{\rm i/o}$ is a constant given by a prototyping system, we assume

$C_{\rm sys} \propto$ #slot $\cdot$ #stage.

Since our TM I/Os transfer signals equally, we set $M_i = \max_{a \in S} \lceil X_a / T_a \rceil$ regardless of the value of i to minimize #slot $\geq \max_i M_i$ under the signal selection. Note that the minimum values of both #slot and #stage are determined by the signal selection. For more details, refer to Ref. 10).

2.5 Optimization of Inter-FPGA Connections between FPGAs

We proposed an optimization method of TM I/O signal selection for minimizing $C_{\rm sys}$ for 2-FPGA systems, OTM-2 [10]. OTM-2 is based on integer linear programming (ILP) [5]. We briefly explain the method here for the completeness of this paper. Note that OTM-2 is for 2-FPGA systems and thus does not distinguish FPGA pairs signals belong to. Since only one FPGA pair exists in a 2-FPGA system, we denote X_a, W_a, and $V_{{\rm i/o},a}$ by X, W and $V_{\rm i/o}$, respectively.

An inter-FPGA signal graph $G_{\rm i/o}(V_{\rm i/o}, E_{i/o})$, which represents signal propagation between inter-FPGA I/Os, is given. Let x_v represent whether a signal $v \in V_{\rm i/o}$ is time-multiplexed. $x_v = 1$ indicates that the signal v is time-multiplexed, and $x_v = 0$ indicates that v is not time-multiplexed. That is, the set of x_v ($v \in V_{\rm i/o}$) represents a TM I/O signal selection. Then, the number of time-multiplexed signals $X = \sum_{v \in V_{\rm i/o}} x_v$. Then, the lower limit of #slot in 2-FPGA systems,

$$\#\text{slot} \geq \left\lceil \frac{X}{W - |V_{\rm i/o}| + X} \right\rceil,$$

is derived from the I/O constraint. Since $X > W - |V_{\rm i/o}| + X$ when TM I/Os are necessary, $\left\lceil \frac{X}{W-|V_{\rm i/o}|+X} \right\rceil$ is a monotonically decreasing function of X. Thus, #slot can be minimized when X is maximized. Let y_v be the number of TM I/Os on a path from an FF to a signal $v \in V_{\rm i/o}$. Then, y_v can be calculated recursively on the inter-FPGA signal graph by $y_v = x_v + \max_{v \in V_{\rm FI}(v)}$. Note that $\max_p N_p = \max_{v \in V_{\rm i/o}} y_v$. Thus,

$$\#\text{stage} \geq \max_{v \in V_{\rm i/o}} y_v.$$

A maximum function can be substituted by parallel inequalities in linear programming.

The final goal is to minimize #slot $\times$ #stage. However, it cannot be directly minimized by using ILP. Thus, we decompose the problem into sub-problems by #stage to minimize #slot under the given #stage $= N$ as described below.

N-depth Signal Selection problem (N-SS problem)

Constants:

N: the number of stages (#stage)

Variables:

$x_{v_1}, x_{v_2}, \ldots, x_{v_m}$: 0-1 integers

$y_{v_1}, y_{v_2}, \ldots, y_{v_m}$: integers

Subject to:

$$0 \leq y_{v_i} \leq N \ (1 \leq i \leq m)$$

$$\forall v \in V_{\rm i/o} \begin{cases} y_v \geq x_v & \text{if } V_{\rm FI}(v) = \emptyset \\ y_v \geq x_v + y_{v'}, \forall v' \in V_{\rm FI}(v) & \text{if } V_{\rm FI}(v) \neq \emptyset \end{cases}$$

Maximize:

$$\sum_{v \in V_{\rm i/o}} x_v$$

The maximum number of gates on an FF-to-FF path is empirically less than 50, and the number of inter-block signals on an FF-to-FF path is empirically less than 20. Since #stage is less than or equal to the number of inter-block signals on an FF-to-FF path, #stage is also less than 20. Thus, we can obtain

the minimum value of C_{sys} by solving a small number of N-SS problems (fewer than 20). OTM-2 obtains the minimum value of C_{sys} in this manner.

In Ref. 10), we also formulated an extended problem, the N-depth Directed Signal Selection problem (N-D-SS problem), to consider signal direction. A signal between FPGA p and FPGA q has the direction p to q, or q to p. Only signals that have the same direction can be transmitted by a single TM I/O. This is considered in the N-D-SS problem. However, we omit the consideration in the rest of this paper because the effect changes the number of necessary TM I/Os by at most only one, though it is important in some cases.

The N-SS problem can be efficiently solved by LP solvers (e.g., CPLEX [9] and GLPK [7], which are also ILP solvers) by integer relaxation. In addition, ILP solvers can empirically solve the N-D-SS problem efficiently, though it is not guaranteed theoretically. Thus, the minimum value of C_{sys} for a 2-FPGA system can be obtained efficiently by OTM-2. For more details, refer to Ref. 10).

Now, let us consider how to apply OTM-2 to multi-FPGA systems with more than two FPGAs. An inter-FPGA signal graph $G_{\mathrm{i/o}}(V_{\mathrm{i/o}}, E_{\mathrm{i/o}})$ and the set of signals $V_{\mathrm{i/o},a} \subseteq V_{\mathrm{i/o}}$ for each FPGA pair $a \in S$ are given. In addition, the number of inter-FPGA wires W_a for each FPGA pair $a \in S$ is given. Since neither W nor W_a appears in the N-SS problem, it can be solved by using the N-SS problem ignoring FPGA pairs. Once a TM signal selection is obtained, its #slot can be evaluated by the following expression, which is from the I/O constraint,

$$\#\mathrm{slot} \geq \max_{a \in S} \left\lceil \frac{X_a}{W_a - |V_{\mathrm{i/o},a}| + X_a} \right\rceil,$$

where X_a is the number of time-multiplexed signals in $V_{\mathrm{i/o},a}$. Note that OTM-2 does not obtain optimal TM signal selections for multi-FPGA systems with more than two FPGAs since it ignores FPGA pairs.

3. ILP-based Optimization of Inter-block Connections for k-way Partitioned Circuit

In this section, we propose a method to consider, in an ILP formulation, the variation of the number of I/O signals from one FPGA pair to another.

An inter-FPGA signal graph $G_{\mathrm{i/o}}(V_{\mathrm{i/o}}, E_{\mathrm{i/o}})$ and the set of signals $V_{\mathrm{i/o},a} \subseteq V_{\mathrm{i/o}}$ for each FPGA pair $a \in S$ are given. In addition, the number of inter-FPGA wires W_a for each FPGA pair $a \in S$ is given.

#slot is limited only by the I/O constraint

$$\left\lceil \frac{X_a}{\#\mathrm{slot}} \right\rceil + |V_{\mathrm{i/o},a}| - X_a \leq W_a.$$

By transposing the second and third terms of the left side to the right side,

$$\left\lceil \frac{X_a}{\#\mathrm{slot}} \right\rceil \leq W_a - |V_{\mathrm{i/o},a}| + X_a.$$

Since $W_a - |V_{\mathrm{i/o},a}| + X_a$ is an integer,

$$\frac{X_a}{\#\mathrm{slot}} \leq W_a - |V_{\mathrm{i/o},a}| + X_a.$$

By solving the inequality about #slot,

$$X_a \leq (W_a - |V_{\mathrm{i/o},a}| + X_a) \cdot \#\mathrm{slot}$$

$$\frac{X_a}{W_a - |V_{\mathrm{i/o},a}| + X_a} \leq \#\mathrm{slot}.$$

This inequality cannot be handled by ILP because the left side of it is non linear. In OTM-2, the number of time-multiplexed signals $X = \sum_{a \in S} X_a$ is maximized instead of directly minimizing #slot, since #slot is monotonically decreasing function of X. In our new method, OTM-k, we transform the inequalities for all $a \in S$ into linear inequalities and a linear objective function. Let $X'_a = W_a - |V_{\mathrm{i/o},a}| + X_a$. Then, we get:

$$\frac{X'_a - W_a + |V_{\mathrm{i/o},a}|}{X'_a} \leq \#\mathrm{slot}$$

$$1 + (|V_{\mathrm{i/o},a}| - W_a)/X'_a \leq \#\mathrm{slot}.$$

Considering all $a \in S$,

$$1 + \max_{a \in S} (|V_{\mathrm{i/o},a}| - W_a)/X'_a \leq \#\mathrm{slot}.$$

Thus, #slot is minimized if

$$\max_{a \in S} (|V_{\mathrm{i/o},a}| - W_a)/X'_a$$

is minimized. That is, #slot is minimized if

$$\min_{a \in S} X'_a/(|V_{\mathrm{i/o},a}| - W_a)$$

is maximized. Thus, the I/O constraints and the minimization of #slot are

represented by

$$\forall a \in S, \left\{ X'' \leq \frac{1}{|V_{\mathrm{i/o},a}| - W_a} X'_a \right\},$$

Maximize:X'',

by introducing a variable X'' to obtain the minimum value. Summing up, we formulate the problem to minimize #slot under the given #stage considering the variation in the number of I/O signals as described below.

---

k-way N-depth Signal Selection problem (k-way N-SS problem)

Constants:

N**:** the number of stages (#stage)

W_a**:** the numbers of wires of an FPGA pair $a \in S$

$V_{\mathrm{i/o},a}$**:** the set of inter-FPGA signals of an FPGA pair $a \in S$

Variables:

$x_{a,v}$**:** 0-1 integers for all $v \in V_{\mathrm{i/o},a}$ for all $a \in S$

$y_{a,v}$**:** integers for all $v \in V_{\mathrm{i/o},a}$ for all $a \in S$

X_a, X'_a**:** real numbers for all $a \in S$

X''**:** a real number

Subject to:

$$X_a = \sum_{v \in V_{\mathrm{i/o},a}} x_{a,v}$$

$$\forall a \in S, \{X'_a = W_a - |V_{\mathrm{i/o},a}| + X_a\}$$

$$\forall a \in S, \forall v \in V_{\mathrm{i/o},a}, \{0 \leq y_{a,v} \leq N\}$$

$$\forall a \in S, \forall v \in V_{\mathrm{i/o},a} \begin{cases} y_{a,v} \geq x_{a,v} & \text{if } V_{\mathrm{FI}}(v) = \emptyset \\ y_{a,v} \geq x_{a,v} + y_{v'}, \forall v' \in V_{\mathrm{FI}}(v) & \text{if } V_{\mathrm{FI}}(v) \neq \emptyset \end{cases}$$

$$\forall a \in S, \left\{ X'' \leq \frac{1}{|V_{\mathrm{i/o},a}| - W_a} X'_a \right\} \text{ if } |V_{\mathrm{i/o},a}| > W_a$$

Maximize:

X''

As mentioned in the previous section, the mixed ILP problem is solved empirically fewer than 20 times to minimize C_{sys}.

4. Experimental Results

We performed experiments to evaluate our proposed method OTM-k compared to the ILP-based conventional method OTM-2 [10]. ILP problem generation programs of both methods were implemented in C++, and the problems were solved by a well-known LP/ILP solver, GNU Linear Programming Kit (GLPK [7]). The experiments were conducted on a PC equipped with an AMD Opteron 1.8 GHz CPU and 3 GB of memory. **Table 1** shows the benchmark circuits from industrial circuits used in the experiments. They are peripheral circuits for media processors, which need to be verified with long test benches such as video streams. In the experiments, all partitions were generated by using the well-known circuit partitioner hMetis [11].

Table 2 and **Table 3** show the numbers of time-multiplexed I/O signals of FPGA pairs of benchmark circuit A with 3- and 4-way partitioning, respectively. The number of wires between FPGAs of an FPGA pair for A was set to 100. **Table 4** and **Table 5** show the numbers of time-multiplexed I/O signals of FPGA pairs of benchmark circuit G with 3- and 4-way partitioning, respectively. The number of wires between FPGAs of an FPGA pair for G was set to 400. In these tables, a represents an FPGA pair in S, and #sig represents the number of I/O signals between a pair of FPGAs. For each #stage, the numbers of time-multiplexed I/O signals by OTM-k and OTM-2 are shown in the -k and -2 columns, respectively. The parenthetic number in each cell is the #slot value derived from the number of time-multiplexed I/O signals. These tables demonstrate that OTM-k selects signals on congested inter-FPGA connections prioritized to

Table 1 Benchmark circuits.

Circuit	#gates	#nets	#FFs
A	251,636	253,215	24,119
B	58,011	52,217	8,181
C	202,693	204,582	18,708
D	33,036	33,460	3,794
E	177,818	181,962	14,672
F	164,204	167,232	17,487
G	1,858,702	1,870,027	180,789

Table 2 Number of TM signals of A (3-way, W = 100).

a	#sig	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5	
		-k	-2	-k	-2	-k	-2	-k	-2	-k	-2
0-1	464	378 (27)	353 (N/A)	448 (6)	445 (6)	463 (5)	463 (5)	464 (5)	464 (5)	464 (5)	464 (5)
1-2	368	279 (26)	343 (5)	330 (6)	351 (5)	341 (5)	368 (4)	342 (5)	368 (4)	342 (5)	368 (4)
2-0	436	350 (25)	376 (10)	414 (6)	432 (5)	428 (5)	436 (5)	429 (5)	436 (5)	429 (5)	436 (5)
#slot		27	N/A	6	6	5	5	5	5	5	5
C_{sys}		27	N/A	12	12	15	15	20	20	25	25

Table 3 Number of TM signals of A (4-way, W = 100).

a	#sig	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5	
		-k	-2	-k	-2	-k	-2	-k	-2	-k	-2
0-1	366	285 (15)	333 (5)	327 (6)	344 (5)	334 (5)	360 (4)	337 (5)	366 (4)	339 (5)	366 (4)
0-2	191	98 (14)	185 (2)	112 (6)	191 (2)	115 (5)	191 (2)	116 (5)	191 (2)	116 (5)	191 (2)
0-3	137	40 (14)	126 (2)	46 (6)	136 (2)	47 (5)	136 (2)	47 (5)	137 (2)	48 (5)	137 (2)
1-2	168	73 (15)	155 (2)	84 (6)	161 (2)	86 (5)	165 (2)	87 (5)	167 (2)	87 (5)	168 (2)
1-3	78	0 (0)	62 (1)	0 (0)	73 (1)	0 (0)	78 (1)	0 (0)	78 (1)	0 (0)	78 (1)
2-3	469	394 (16)	365 (N/A)	453 (6)	437 (7)	463 (5)	462 (5)	467 (5)	464 (5)	469 (5)	469 (5)
#slot		16	N/A	6	7	5	5	5	5	5	5
C_{sys}		16	N/A	12	14	15	15	20	20	25	25

Table 4 Number of TM signals of G (3-way, W = 400).

a	#sig	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5	
		-k	-2	-k	-2	-k	-2	-k	-2	-k	-2
0-1	1,600	1,277 (17)	1,485 (6)	1,402 (7)	1,570 (5)	1,408 (7)	1,598 (5)	1,408 (7)	1,598 (5)	1,408 (7)	1,599 (5)
1-2	945	580 (17)	903 (3)	637 (7)	930 (3)	640 (7)	944 (3)	640 (7)	945 (3)	640 (7)	945 (3)
2-0	2,712	2,459 (17)	2,435 (20)	2,701 (7)	2,696 (8)	2,712 (7)	2,710 (7)	2,712 (7)	2,712 (7)	2,712 (7)	2,712 (7)
#slot		17	20	7	8	7	7	7	7	7	7
C_{sys}		17	20	14	16	21	21	28	28	35	35

reduce the maximum value of the lower limits of #slot derived from the I/O constraints on FPGA pairs. All the ILP problems in our proposed method were solved within 10 seconds.

Table 6 and **Table 7** show the minimum system clock periods for each #stage for each circuit, which are intermediate solutions of OTM-k and OTM-2. The unit of the system clock periods is one I/O clock period. The minimum system clock period for each circuit with OTM-k is underlined in Table 6 and Table 7. W represents the number of wires between FPGAs of an FPGA pair. #ineq. represents the number of constraints in the ILP formulation of the k-way N-SS problem. Note that #ineq. does not depend on #stage. "N/A" means that the intermediate solution was infeasible. *Imp.* represents the improvement ratio compared to the result using OTM-2. Time represents the total time in seconds to solve k-way N-SS problems for $N = 1, 2, \ldots, 5$. The *Ave.* row shows the average improvement ratio and the average total time. Note that we assume the system clock period C_{sys} is proportional to #stage$\cdot$#slot, and the lower limit of #slot is derived from the I/O constraints when a signal selection is given. Compared with OTM-2, the average minimum system clock periods of 3- and 4-way partitioned circuits were improved by 3.1% and 16.0% by OTM-k.

Let us consider the effect of the variation in the number of signals between an FPGA pair. Suppose a partition Φ of a circuit is given. Then, the variation can

Table 5 Number of TM signals of G (4-way, W = 400).

a	#sig	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5	
		-k	-2	-k	-2	-k	-2	-k	-2	-k	-2
0-1	2,543	2,426 (9)	2,372 (10)	2,527 (7)	2,515 (7)	2,533 (7)	2,533 (7)	2,534 (7)	2,534 (7)	2,534 (7)	2,534 (7)
0-2	886	553 (9)	841 (3)	576 (7)	884 (3)	577 (7)	883 (3)	578 (7)	885 (3)	578 (7)	886 (3)
0-3	1,118	817 (9)	1,014 (4)	851 (7)	1,091 (3)	853 (7)	1,109 (3)	853 (7)	1,115 (3)	853 (7)	1,118 (3)
1-2	720	364 (9)	717 (2)	379 (7)	720 (2)	380 (7)	720 (2)	380 (7)	720 (2)	380 (7)	720 (2)
1-3	469	79 (8)	411 (2)	82 (7)	449 (2)	82 (7)	465 (2)	82 (7)	468 (2)	82 (7)	468 (2)
2-3	439	45 (8)	389 (2)	47 (6)	428 (2)	47 (6)	436 (2)	47 (6)	439 (2)	47 (6)	439 (2)
#slot		9	10	7	7	7	7	7	7	7	7
C_{sys}		9	10	14	14	21	21	28	28	35	35

Table 6 Minimum clock period (3-way).

Circuit	W	#ineq.	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5		Imp.	Time
			-k	-2	-k	-2	-k	-2	-k	-2	-k	-2		
A	100	7,310	27 (27)	N/A (N/A)	12 (6)	12 (6)	15 (5)	15 (5)	20 (5)	20 (5)	25 (5)	25 (5)	0.0%	7 s
B	50	3,416	N/A (N/A)	N/A (N/A)	20 (10)	22 (11)	24 (8)	24 (8)	32 (8)	32 (8)	40 (8)	40 (8)	9.1%	2 s
C	100	6,713	N/A (N/A)	N/A (N/A)	12 (6)	12 (6)	15 (5)	18 (6)	20 (5)	24 (6)	25 (5)	25 (5)	0.0%	9 s
D	100	2,323	35 (35)	35 (35)	18 (9)	18 (9)	27 (9)	27 (9)	36 (9)	36 (9)	45 (9)	45 (9)	0.0%	2 s
E	400	7,135	4 (4)	4 (4)	8 (4)	8 (4)	12 (4)	12 (4)	16 (4)	16 (4)	20 (4)	20 (4)	0.0%	4 s
F	100	7,328	11 (11)	11 (11)	10 (5)	10 (5)	15 (5)	15 (5)	20 (5)	20 (5)	25 (5)	25 (5)	0.0%	2 s
G	400	23,605	17 (17)	20 (20)	14 (7)	16 (8)	21 (7)	21 (7)	28 (7)	28 (7)	35 (5)	35 (5)	12.5%	49 s
Ave.													3.1%	11 s

Table 7 Minimum clock period (4-way).

Circuit	W	#ineq.	#stage = 1		#stage = 2		#stage = 3		#stage = 4		#stage = 5		Imp.	Time
			-k	-2	-k	-2	-k	-2	-k	-2	-k	-2		
A	100	9,859	16 (16)	N/A (N/A)	12 (6)	14 (7)	15 (5)	15 (5)	20 (5)	20 (5)	25 (5)	25 (5)	14.3%	12 s
B	50	3,735	12 (12)	15 (15)	14 (7)	18 (9)	18 (6)	21 (7)	24 (6)	24 (6)	30 (6)	30 (6)	20.0%	1 s
C	100	4,554	3 (3)	4 (4)	6 (3)	6 (3)	9 (3)	9 (3)	12 (3)	12 (3)	15 (3)	15 (3)	25.0%	3 s
D	100	4,476	21 (21)	59 (59)	26 (13)	30 (15)	36 (12)	42 (14)	48 (12)	48 (12)	60 (12)	60 (12)	30.0%	2 s
E	400	11,396	3 (3)	3 (3)	6 (3)	6 (3)	9 (3)	9 (3)	12 (3)	12 (3)	15 (3)	15 (3)	0.0%	11 s
F	100	11,313	7 (7)	8 (8)	8 (4)	10 (5)	12 (4)	12 (4)	16 (4)	16 (4)	20 (5)	20 (5)	12.5%	11 s
G	400	22,281	9 (9)	10 (10)	14 (7)	14 (7)	21 (7)	21 (7)	28 (7)	28 (7)	35 (7)	35 (7)	10.0%	47 s
Ave.													16.0%	12 s

be briefly represented by the ratio of #sig-max to #sig-min, where #sig-max is the maximum number of signals between an FPGA pair among the FPGA pairs, and #sig-min is the minimum number of signals between an FPGA pair among the FPGA pairs. Since OTM-k preferentially selects signals from those between the FPGA pair with #sig-max to minimize #slot under the given #stage, it is expected to work effectively in cases with a large variation in the number of signals. This is confirmed by the experimental results: the improvement ratios by OTM-k of the 3- and 4-way partitions of A are 0% and 14.3%, and that of the 3-way partition of G is 12.5%, while the variations in the 3- and 4-way partitions of A are 1.26 (= 464/368) and 6.01 (= 469/78), and that of the 3-way partition of G is 2.87 (= 2,712/945).

5. Concluding Remarks

In this study, we proposed an extended method for optimizing time-multiplexing in inter-FPGA connections of FPGA prototyping systems to consider the variation in the number of signals between a pair of FPGAs from one pair to another. Our method is based on integer linear programming, for which efficient algorithms have been proposed. We transformed the non-linear constraints of inter-FPGA communication into linear constraint inequalities and a linear objective function to solve the optimization problem by integer linear programming. Experiments showed that our method obtains the minimum system clock periods 16.0% shorter than those of a method ignoring the variations, for 4-way partitioned circuits on average. Our future work includes the development of a partitioning algorithm considering optimal time-multiplexing and a time-multiplexing method considering data transfer scheduling.

References

1) Azegami, K.R., Inagi, M., Takahashi, A. and Kajitani, Y.: An improvement of network-flow Based multi-way circuit partitioning algorithm, *IEICE Trans. Fundamentals*, Vol.E85-A, No.3, pp.655–663 (2002).
2) Babb, J., Tessier, R. and Agarwal, A.: Virtual Wires: Overcoming pin limitations in FPGA-based logic emulators, *IEEE Workshop FCCM Proceedings*, pp.142–151 (1993).
3) Babb, J., Tessier, R., Dahl, M., Hanono, S., Hoki, D. and Agarwal, A.: Logic emulation with virtual wires, *IEEE Trans. Computer Aided Design*, Vol.16, No.6, pp.609–629 (1997).
4) Brown, R., Hayes, J. and Mudge, T.: Rapid prototyping and evaluation of high-performance computers, *Proc. Conf. Experimental Research in Computer Systems, NFS Experimental Systems*, pp.159–168 (1996).
5) Cook, W.J., Cunningham, W.H., Pullayblank, W.R. and Schrijver, A.: *Combinatorial Optimization*, John Willy & Sons Inc. (1998).
6) Fiduccia, C.M. and Mattheyses, R.M.: A linear time heuristic for improving network partitions, *IEEE DAC82 Proceedings*, pp.175–181 (1982).
7) Free Software Foundation, Inc.: GNU Linear Programming Kit, http://www.gnu.org.
8) Gateley, J., Blatt, M., Chen, D., Cooke, S., Desai, P., Doreswamy, M., Elgood, M., Feierbach, G., Goldsbury, T. and Greenley, D.: UltarSparc-I Emulation, *ACM/IEEE DAC95 Proceedings*, pp.13–18 (1995).
9) ILOG, Inc.: IBM ILOG CPLEX, http://www.ilog.com.
10) Inagi, M., Takashima, Y., Nakamura, Y. and Takahashi, A.: Optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA prototyping systems, *IEICE Trans. Fundamentals*, Vol.E91-A, No.12, pp.3539–3547 (2008).
11) Karypis, G., Aggarwal, R., Kumar, V. and Shekhar, S.: Multilevel hypergraph partitioning: Applications in VLSI domain, *ACM/IEEE DAC97 Proceedings*, pp.526–529 (1997).
12) Kernighan, B.W. and Lin, S.: An efficient heuristic procedure for partitioning graphs, *The Bell Systems Tech. J.*, Vol.49, pp.291–307 (1970).

(Received June 1, 2009)
(Revised September 5, 2009)
(Accepted October 31, 2009)
(Released February 15, 2010)

(Recommended by Associate Editor: *Shinji Kimura*)

Masato Inagi received his B.E. degree in computer science and M.E. and Ph.D. degrees in communications and integrated systems from Tokyo Institute of Technology in 2000, 2002 and 2008, respectively. He was a researcher at Kitakyushu Foundation for Advancement of Industry, Science and Technology from 2005 to 2007. He was also a researcher at the University of Kitakyushu from 2005 to 2008. Since 2008, he has been a research associate at the Graduate School of Information Sciences, Hiroshima City University. His research interests include combinatorial algorithms for circuit layout design.

Yasuhiro Takashima received his B.E., M.E., and Ph.D. degrees in electrical and electronic engineering from Tokyo Institute of Technology in 1993, 1995, and 1998, respectively. He had been a research associate in School of Information Science at Japan Advanced Institute of Science and Technology from 1998 to 2003. He had been with Kitakyushu Foundation for Advancement of Industry, Science and Technology as an invited researcher from 2003 to 2005. Since 2005, he has been an associate professor at the University of Kitakyushu. His research interest is in combinatorial algorithms applied to VLSI layout design.

Yuichi Nakamura received his B.E. in information engineering and his M.E. in electrical engineering from Tokyo Institute of Technology in 1986 and 1988. He received his Ph.D. degree from Graduate School of Information, Production, and Systems, Waseda University. He is currently a senior principal researcher at System IP Core Laboratories, NEC Corp. His research interests include the design and verification of high-speed and complex VLSIs.