PAPER

A Fast Gate-Level Register Relocation Method for Circuit Size Reduction in General-Synchronous Framework*

Yukihide KOHIRA[†a)] ***and*** **Atsushi TAKAHASHI**[†], ***Members***

SUMMARY Under the assumption that the clock can be inputted to each register at an arbitrary timing, the minimum feasible clock period might be reduced by register relocation while maintaining the circuit behavior and topology. However, if the minimum feasible clock period is reduced, then the number of registers tends to be increased. In this paper, we propose a gate-level register relocation method that reduces the number of registers while keeping the target clock period. In experiments, the proposed method reduces the number of registers in the practical time in most circuits.

key words: *register relocation, retiming, clock scheduling, general-synchronous framework*

1. Introduction

The semiconductor manufacturing process technology has improved the scale, speed and power consumption of LSI circuits. However, increasing the ratio of the routing delay in the propagation delay bounds the amount of improvements in complete-synchronous framework (c-frame) in which the simultaneous clock distribution to every register is assumed. The increases of the size and power consumption of a clock distribution circuit have become serious issues in c-frame. While, general-synchronous framework (g-frame) [3]–[5], in which the clock is assumed to be distributed periodically to each individual register though not necessarily to all registers simultaneously, is expected to give an essential solution. By using g-frame, the quality of circuit such as the clock frequency, clock distribution circuit size, power consumption, and etc. are expected to be improved. The efforts toward improvements of qualities in g-frame are summarized in [6].

The framework of synchronization by a global clock without restriction of simultaneity was discussed in the context of clock scheduling, useful-skew, semi-synchronous, and etc. In this paper, we call the framework g-frame to emphasize the framework includes c-frame. In the beginning of studies of g-frame, clock scheduling algorithms [3]–[5], [7] and clock distribution circuit synthesis algorithms [8], [9] for given logic circuits were proposed. However, given logic circuits are synthesized for c-frame. Since the clock period might not be reduced in g-frame even if the maximum delay is reduced, the effort in c-frame might degrade the circuit performance in g-frame. Therefore, the optimization of circuit synthesis that takes g-frame into account must be investigated.

As logic circuit modification methods that improve the performance in g-frame, delay insertion methods [10]–[13], a gate sizing method [14], a multi-clock cycle path method [15], and register relocation methods [16], [17] are proposed.

In c-frame, the circuit modification in which registers are relocated while maintaining the circuit behavior and topology is called retiming [18]. However, in g-frame, retiming may be confused with the change of the clock input timing of a register. Therefore, we call it register relocation in g-frame.

In [16], a mixed integer linear programming (MILP) formulation and a heuristic algorithm of the register relocation in g-frame are proposed. The objective of these algorithms is the clock period minimization or the tolerance maximization against clock signal delay variations. However, since the computation time of these algorithms is too long, these algorithms cannot be applied to circuits with thousands of gates. In [17], the register relocation method for the clock period minimization in g-frame is proposed. We call this method P-CR (Period reduction by Cone Relocation). P-CR is guaranteed to achieve the lower bound of the minimum clock period by register relocation in g-frame if the maximum delay of each element is equal to its minimum delay. Moreover, it is empirically shown that P-CR can be applied to large circuits. However, the number of registers in a circuit obtained by P-CR tends to be increased, since P-CR applies register relocations to reduce the minimum clock period in g-frame even if the numbers of registers are increased.

Since LSIs are often designed under the target clock period as the specifications, the problem that minimizes the number of registers in a circuit that is obtained by register relocation and that works at the target clock period in g-frame is very important in LSI design in practical sense. Therefore, we focus on this problem. If the maximum delay of each element is equal to its minimum delay, the minimum clock period by register relocation in g-frame is obtained in a polynomial time [17]. The minimum number of registers by register relocation in g-frame is obtained in a polynomial time [18]. However, the computational complexity of the problem that minimizes the number of registers in a circuit that is obtained by register relocation and that works at the

Manuscript received February 22, 2008.
Manuscript revised May 15, 2008.

[†] The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.

[*] The preliminary vesion was presented at [1], [2].

a) E-mail: kohira@lab.ss.titech.ac.jp

DOI: 10.1093/ietfec/e91–a.10.3030

target clock period in g-frame is not known, although we conjecture that this problem is NP-hard. Therefore, we investigate heuristic algorithms for this problem in this paper.

In this paper, we propose a gate-level register relocation method that reduces the number of registers while keeping the target clock period in g-frame. We call this method S-CR (Size reduction by Cone Relocation). S-CR is a greedy local circuit modification method in g-frame. In S-CR, if a circuit that works at the target clock period is inputted, a register relocation that reduces the number of registers most while keeping the target clock period is applied iteratively. S-CR reduces the number of registers even if the maximum delay of each element is not equal to its minimum delay. In order to obtain a circuit that works at the target clock period, we use P-CR as pre-processing of S-CR, because P-CR is fast and obtains the circuits which achieve the target clock periods in most cases even if the maximum delay of each element is not equal to from its minimum delay. We call it P&S-CR. If the target clock period is different, the required number of registers in a circuit is different. For the same target clock period, the number of registers in a circuit obtained by P&S-CR is not larger than that by P-CR. In experiments, it is smaller in most cases. Although an optimal circuit in terms of the number of registers cannot be obtained by P&S-CR in general, optimal circuits are obtained by P&S-CR for small circuits. The computation time of P&S-CR is less than 10 minutes for most circuits with thousands of gates.

2. Preliminaries

In this paper, we consider a circuit consisting of registers and gates, and wires connecting them. We call them elements. A circuit is represented by the graph $G = (V, E)$, where vertex set V corresponds to elements in the circuit and directed edge set E corresponds to signal propagations in the circuit. An example of a circuit graph is shown in Fig. 1. In the circuit graph shown in Fig. 1, $\{a, b, c, I/O\}$ is the register set, and the figure in each vertex represents its delay. For simplicity, the maximum delay of each element is equal to its minimum delay and each register does not have delay in this example.

2.1 General-Synchronous Framework

In c-frame, the clock arrival timing of a register is the same as those of the other registers. C-frame, in which the clock arrival timings of registers are assumed to be equal, is a kind of g-frame. In g-frame [3]–[5], [7], the clock arrival timing of a register may be different from other registers. The clock timing $S(r)$ of a register r is defined as the difference in the clock arrival time between r and an arbitrary chosen reference register.

A circuit works correctly with a clock period T if the following two types of constraints are satisfied for every register pair with signal propagations (Fig. 2) [3].

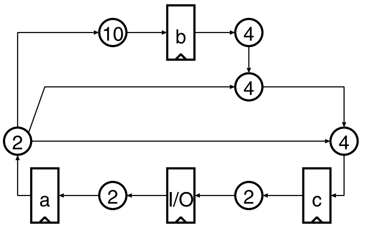


Fig. 1 Circuit graph G.

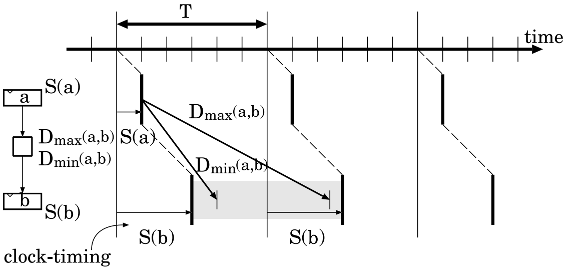


Fig. 2 Timing chart.

Setup (No-Zero-Clocking) Constraints

$$S(a) - S(b) \leq T - D_{\max}(a, b)$$

Hold (No-Double-Clocking) Constraints

$$S(b) - S(a) \leq D_{\min}(a, b),$$

where $D_{\max}(a, b)(D_{\min}(a, b))$ is the maximum (minimum) delay from a register a to a register b.

Since c-frame has the premise that a clock ticks all the register simultaneously, the clock period must be larger than the maximum delay between registers. On the other hand, in g-frame, the circuit can work correctly with the clock period which is smaller than the maximum delay between registers, if all the register pair with the signal path satisfies two types of constraints.

Let $T_S(G)$ be the minimum clock period of a circuit G in g-frame under the assumption that the clock can be inputted to each register at an arbitrary designated timing. Hereafter, we simply call $T_S(G)$ the minimum clock period of G in g-frame. $T_S(G)$ is determined by the *constraint graph* $H(G) = (R, A)$ for G, where vertex set R corresponds to registers in G and directed edge set A corresponds to two types of constraints. An edge in A from a register a to a register b with weight $D_{\min}(a, b)$, called the D-edge, corresponds to the Hold constraint, and an edge from a register b to a register a with weight $T - D_{\max}(a, b)$, called the Z-edge, corresponds to the Setup constraint. Let $H(G, t)$ be the constraint graph in which the clock period T is set to t. Let the weight of a directed cycle in $H(G, t)$ be the sum of edge weights on the directed cycle. It is known that the minimum clock period in g-frame is determined as in the following theorem.

Theorem 1 ([5]): $T_S(G)$ is the minimum t such that there is no cycle with negative weight in the constraint graph $H(G, t)$.

A cycle whose weight is zero in $H(G, T_S(G))$ and is

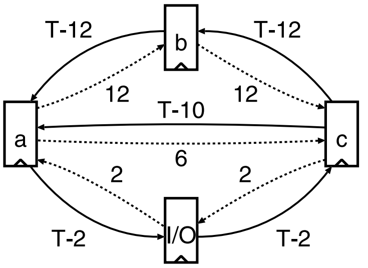


Fig. 3 Constraint graph $H(G)$.

negative in $H(G, t)$ where $t < T_S(G)$ is said to *be critical* and determines the minimum clock period $T_S(G)$ in g-frame.

For example, the constraint graph of circuit graph G shown in Fig. 1 is shown in Fig. 3. In $H(G, 9)$, since the weight of cycle (a, c, b, a) is zero and it is negative with $T <$ 9, cycle (a, c, b, a) is critical and the minimum clock period $T_S(G)$ is 9.

2.2 Register Relocation

Register relocation method is a circuit modification method maintaining the circuit behavior and topology. In [17], *cone relocation* which is a kind of register relocation method is proposed. We also apply cone relocation in the proposed method.

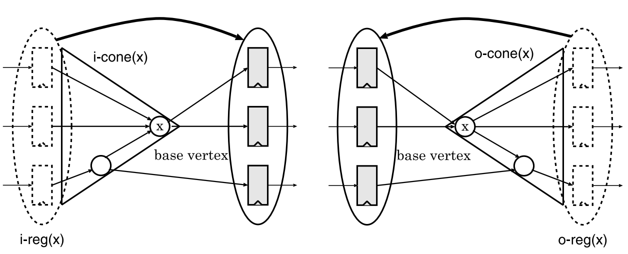
Let i-cone(x), the input cone of a vertex x in G, be the set of vertices of G from which a signal propagates to x without go through registers in G. Let i-reg(x) be the set of input registers of i-cone(x). An edge (u, v) in G is called an output of the i-cone(x) if u is in i-cone(x) and v is not in i-cone(x).

The forward cone relocation of a vertex x (f-reloc(x)) is a modification of G in which all i-reg(x) are removed and which a register is inserted to each output of i-cone(x) (Fig. 4(a)). Similarly, the backward cone relocation of x (b-reloc(x)) is defined (Fig. 4(b)). In f-reloc(x) or b-reloc(x), x is called the base vertex. Since i-reg(x) and outputs of i-cone(x) (o-reg(x) and inputs of o-cone(x)) in forward (backward) cone relocation can be determined by depth first search in G, the time complexity of a cone relocation is $O(|E|)$ if a base vertex x is given.

In a cone relocation, we can consider that a register is relocated along a path in the circuit with duplication when the path branches and with merging when the path converges. A cone relocation is an enhancement of the well-known register relocation of a vertex [18] which we call *vertex relocation*. A cone relocation can be defined as the set of vertex relocations.

The minimum clock period of a circuit in g-frame is changed by register relocation. Let the *lowest clock period* $T_L(G)$ of a circuit G be a lower bound of the minimum clock periods of all circuits which are obtained by register relocations. It is known that the lowest clock period $T_L(G)$ is determined as in the following theorem.

Theorem 2 ([10], [17]): The lowest clock period $T_L(G)$ is defined as



(a) Forward (f-reloc(x)). (b) Backward (b-reloc(x)).

Fig. 4 Cone relocation.

$$T_L(G) = \max_{C \in \text{cycles in } G} \frac{D_{\max}(C)}{N(C)},$$

where $D_{\max}(C)$ and $N(C)$ are the maximum delay and the number of registers of a directed cycle C in G, respectively.

3. Proposed Method

In this paper, we propose a greedy register relocation method in g-frame. The proposed method iteratively applies cone relocations that reduce the number of registers most and that keep the target clock period.

3.1 Size Gain and Period Slack

The following two types of values are defined to select a cone relocation to be applied in our register relocation method. Let the *size gain* of a cone relocation be the difference in the number of registers before and after the cone relocation. Let the *period slack* of a cone relocation be the difference between the target clock period and the minimum clock period in g-frame after the cone relocation.

The size gain of a cone relocation is obtained by depth first search in G in $O(|E|)$ time. The period slack of a cone relocation is obtained by computing the minimum clock period in the circuit obtained by the cone relocation. Before computing the minimum clock period, the constraint graph is needed to be updated. Updating the constraint graph takes $O(|E| \cdot |R|)$ time and computing the minimum clock period in g-frame takes $O(k \cdot |R| + k^2 \cdot |A|)$ time [7], where k is the maximum number of edges in a shortest trail in the constraint graph which can be regarded as a small constant in most cases. Since $|R|$ and $|A|$ are smaller than $|E|$ in general, the time complexity of the period slack computation depends on updating the constraint graph and it is larger than that of the size gain computation.

3.2 Size Reduction by Cone Relocation

In our proposed method S-CR (Size reduction by Cone Relocation), it is assumed that a circuit which works correctly at the target clock period is inputted. S-CR reduces the number of registers while keeping the target clock period by iteratively applying a cone relocation that reduces the number

```
Procedure  S_CR($G, T_{tar}(G)$)
Input :  circuit $G$ which work correctly at the target clock
         period $T_{tar}(G)$, target clock period $T_{tar}(G)$
Output :  circuit $G$ obtained by cone relocations
Step 1 :  Compute the size gains of all cone relocations of
          $G$ and sort cone relocations with positive size gains
          in non-increasing order.
Step 2 :  Compute the period slack of a cone relocation
          according to the size gain order. If a cone reloca-
          tion with non-negative period slack is found, then
          obtain $G$ by the cone relocation and go to Step 1.
          Otherwise, output circuit $G$ and terminate.
```

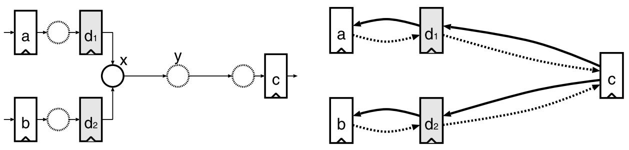
Fig. 5 S-CR.

of registers and that keeps the target clock period. By applying a cone relocation with positive size gain and non-negative period slack, the number of registers is reduced while the target clock period is kept.

In S-CR, among the cone relocations with positive size gain and non-negative period slack, a cone relocation with maximum size gain is applied iteratively until there exists no such cone relocation. Since the computation time of period slack is larger than that of size gain, the number of period slack computations is better to be small. S-CR decides a cone relocation that is applied efficiently as shown in Fig. 5.

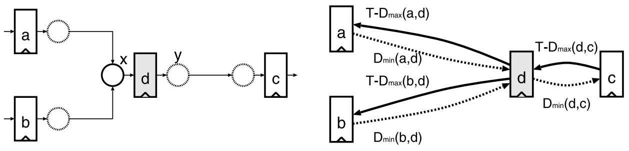
In S-CR, first, the size gains of all cone relocations are computed and cone relocations with positive size gains are sorted in non-increasing order. Then, S-CR computes the period slack of a cone relocation with positive size gain according to the size gain order until a cone relocation with non-negative period slack is found. If a cone relocation with non-negative period slack is found, then it is applied to the current circuit and S-CR repeats from the size gain computation. Otherwise, S-CR terminates and outputs the current circuit. Note that S-CR decides a cone relocation that is applied without computing period slacks of cone relocations with non-positive size gain. Moreover, once S-CR finds a cone relocation with non-negative period slack, S-CR does not compute period slacks for cone relocations with small positive size gain.

S-CR reduces the number of registers while keeping the target clock period even if the maximum delay of each element is not equal to its minimum delay since the definitions of the size gain and the period slack do not depend on the existence of delay range of each element. If the maximum delay of each element is equal to its minimum delay to save the computation time, S-CR excludes several cone relocations before computing their size gains without losing chances to reduce the number of registers. The forward and backward cone relocations of a vertex with single incoming and single outgoing edges that corresponds to a buffer, an inverter, and etc. are excluded in S-CR since these size gains are zero or there exist other cone relocations with the same size gain and period slack if the maximum delay of each element is equal to its minimum delay. For example, the circuit and constraint graph obtained from the circuit shown in Fig. 6(a) by f-reloc(x) and f-reloc(y) are shown in Fig. 7 and Fig. 8, respectively. It is not necessary to apply f-reloc(y), since the size gain and period slack of f-reloc(y) are same as those of f-reloc(x).



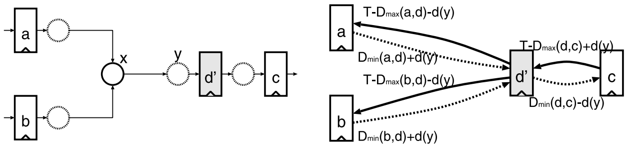
(a) circuit. (b) constraint graph.

Fig. 6 A part of original circuit.



(a) circuit. (b) constraint graph.

Fig. 7 After f-reloc(x).



(a) circuit. (b) constraint graph.

Fig. 8 After f-reloc(y).

3.3 Target Clock Period Realization

The obtained circuit depends on the target clock period. No circuit that works correctly at a clock period which is less than the lowest clock period T_L of the given circuit is obtained by register relocation. The target clock period in circuit improvement by register relocation should be larger than or equals to T_L of the given circuit. In S-CR, it is assumed that a circuit that works at the target clock period is inputted. In order to apply S-CR to reduce the number of registers, a circuit that works at the target clock period should be obtained. There are several approaches to reduce the minimum clock period of a circuit.

For example, S-CR itself is possible to use to reduce the minimum clock period. If the given circuit contains a cone relocation with positive size gain with non-negative period slack, a desired circuit is obtained. However, the given circuit contains no such relocation if the constraint graph at the target clock period contains more than one negative weight cycle that is not broken by a cone relocation simultaneously. Therefore, the assumption that the given circuit contains such relocations is not appropriate, and S-CR is not good to use to reduce the minimum clock period. In order to reduce the minimum clock period, a cone relocation that breaks a negative weight cycle should be applied. However, the period slack is not appropriate index to find a cone relocation that breaks a negative weight cycle. A cone relocation that does not break any negative weight cycle might

```
Procedure P_CR($G, T_{tar}(G)$)
Input : circuit $G$, target clock period $T_{tar}(G)$
Output : circuit $G$ obtained by cone relocations
Step 1 : Determine $T_S(G)$ from the constraint graph. If
    $T_S(G) \leq T_{tar}(G)$, then output $G$ and terminate.
Step 2 : Obtain a cone relocation with the maximum
    size gain among cone relocations which break critical cycles.
Step 3 : Obtain $G$ by the cone relocation, and go to Step 1.
```

Fig. 9 P-CR [17].

```
Procedure P&S_CR($G, T_{tar}(G)$) or P&S_CR($G$)
Input : circuit $G$, (target clock period $T_{tar}(G)$)
Output : circuit obtained by cone relocations
Step 1 : If $T_{tar}(G)$ is not given, determine $T_L(G)$ from
    the constraint graph consisting of only Z-edges ([11]),
    and set $T_{tar}(G)$ to $T_L(G)$.
Step 2 : $G :=$ P_CR($G, T_{tar}(G)$).
Step 3 : $G :=$ S_CR($G, T_{tar}(G)$), output circuit $G$ and
    terminate.
```

Fig. 10 P&S-CR.

have the maximum period slack as well as a cone relocation that breaks a negative weight cycle.

Therefore, we adopt P-CR (see Fig. 9) to reduce the minimum clock period if a circuit that does not work at the target clock period is given. P-CR finds a cone relocation that breaks a negative weight cycle efficiently without computing period slacks, and applies the found cone relocation iteratively. Moreover, if the maximum delay of each element in circuit is not equal to its minimum delay, the found cone relocation does not generate new negative weight cycles, and a circuit that works at the target clock period is obtained by P-CR if it is larger than T_L of the given circuit [17]. In case that the maximum delay of each element is not equal to its minimum delay, there is no guarantee that a circuit that works at the target clock period is obtained, even if the target clock period is larger than T_L. There might be no circuit that works at T_L. The found cone relocation might generate new negative weight cycles, and the original P-CR might not terminate. So, an enhanced P-CR that stops cone relocation if the repetition is detected is used. In experiments, we found that, in most cases, a circuit that works at the target clock period is obtained by P-CR even if the maximum delay of each element in circuit is not equal to its minimum delay. We call the register relocation method in which P-CR is used as pre-processing of S-CR P&S-CR (see Fig. 10).

4. Experiments

We implement P&S-CR in C++, which compiled by gcc3.5.5, and executed on a PC with 3.06 GHz/512 K Intel Pentium-4 CPU and 512 MB RAM. We also implement S-VR-C(MILP) and S-VR-G(MILP) which is modified from the MILP formulations for clock period minimization in c-frame [18] and in g-frame [16] to MILP formulations for register minimization in c-frame and in g-frame, respectively. S-VR-C(MILP) and S-VR-G(MILP) obtain the optimal solutions in c-frame and in g-frame, respectively. MILP is solved by CPLEX 9.0.0 [19]. We perform these methods on the ISCAS89 benchmark suite. In the following three sub-sections, three kinds of experimental results are described.

Table 1 Benchmark circuits.

model	#gate	#reg	T_C	T_S	T_L
s298	119	14	18	12.0	10.00
s344	160	15	37	34.0	19.00
s349	161	15	37	34.0	19.00
s382	158	21	18	12.0	11.25
s400	164	21	18	12.0	11.25
s444	181	21	20	13.0	11.67
s499	152	22	23	19.0	11.50
s526	193	21	18	12.0	11.00
s526n	194	21	18	12.0	11.00
s635	286	32	162	158.0	88.50
s991	519	19	117	110.0	109.00
s1269	569	37	70	61.0	39.34
s1423	657	74	164	156.0	146.00
s1512	780	57	54	43.0	40.50
s3271	1572	116	58	34.0	27.72
s3330	1789	133	66	40.0	32.00
s3384	1685	183	168	154.0	75.50
s4863	2342	104	144	129.0	75.00
s6669	3080	239	231	197.0	56.50
s9234	5597	228	107	72.0	63.00
s9234.1	5597	211	107	72.0	63.00
prolog	1601	136	68	40.0	31.00
s13207	7951	669	106	76.0	75.00
s15850	9772	597	141	104.0	78.00
s15850.1	9772	534	141	124.0	103.00
s38417	22179	1636	85	61.0	60.00

T_C: minimum clock period in c-frame
T_S: minimum clock period in g-frame
T_L: lowest clock period

4.1 Comparison of Number of Registers

First, the numbers of registers obtained by four register relocation methods are evaluated. The maximum delay of each element is set to be equal to its minimum delay: NOT gate delay is set to 1, NAND and NOR gate delay are set to 2, AND and OR gate delay are set to 3, and routing and register delays are set to 0. The target clock period is set to the lowest clock period T_L of each circuit.

In 26 circuits among 48 ISCAS89 benchmark circuits, the minimum clock periods in g-frame are decreased by the register relocation. These 26 circuits are shown in Table 1. The results of these 26 circuits are shown in Table 2. Among these 26 circuits shown in Table 2, the solutions of 21 circuits cannot be obtained by S-VR-G(MILP) because of the lack of memory. Also, the solutions of 4 circuits cannot be obtained by S-VR-C(MILP) because of the lack of memory or not obtained within a day.

The number of registers by P-CR is increased from original since the minimum clock period is reduced. On the

Table 2 Experimental results.

	g-frame														c-frame		
	original			P&S-CR									S-VR-G(MILP)		S-VR-C(MILP)		
				P-CR[17]				S-CR									
						#CR				#CR							
model	T_S	#reg	T_L	#reg	time[s]	size	appl.	#reg	time[s]	size	period	appl.	#reg	time[s]	T_R	#reg	time[s]
s298	12	14	10.00	17	0.01	4	2	17	0.01	226	6	0	17	7823.23	10	47	0.07
s344	34	15	**19.00**	26	0.05	14	11	**22**	0.10	696	42	2	22	1400.93	20	27	0.12
s349	34	15	**19.00**	26	0.05	14	11	**22**	0.10	716	42	2	22	1965.21	20	27	0.13
s382	12	21	**11.25**	25	0.03	10	4	**23**	0.17	786	40	2	23	72180.79	12	29	0.14
s400	12	21	**11.25**	27	0.03	10	4	**23**	0.17	1004	42	3	23	37356.44	12	29	0.17
s444	13	21	**11.67**	35	0.08	17	8	**28**	0.28	1352	58	4	N/A	N/A	13	29	0.33
s499	19	22	**11.50**	109	0.70	936	90	**76**	3.80	4328	750	14	N/A	N/A	12	89	0.10
s526	12	21	11.00	22	0.01	2	1	22	0.01	396	3	0	N/A	N/A	11	63	0.26
s526n	12	21	11.00	22	0.01	2	1	22	0.01	394	3	0	N/A	N/A	11	63	0.25
s635	158	32	**88.50**	76	1.95	101	53	**61**	2.03	4655	180	14	N/A	N/A	89	63	1.57
s991	110	19	109.00	20	0.01	2	1	20	0.77	810	3	0	N/A	N/A	109	26	7.63
s1269	61	37	**39.34**	90	2.76	103	50	**80**	2.06	4183	109	2	N/A	N/A	40	123	5.88
s1423	156	74	146.00	81	0.80	8	7	**80**	1.03	2036	27	1	N/A	N/A	146	87	193.59
s1512	43	57	**40.50**	61	0.11	2	2	61	0.29	1078	6	0	N/A	N/A	41	72	25.72
s3271	34	116	**27.72**	199	6.07	159	54	**174**	27.49	23196	369	11	N/A	N/A	28	185	8.80
s3330	40	133	32.00	147	0.93	20	8	**74**	3.29	11280	48	7	N/A	N/A	32	123	7.34
s3384	154	183	**75.50**	292	13.27	214	76	**222**	29.37	39974	279	25	N/A	N/A	76	183	111.10
s4863	129	104	75.00	219	20.50	281	103	**144**	182.40	80483	1406	35	N/A	N/A	75	159	253.55
s6669	197	239	**56.50**	975	588.67	4344	798	**450**	11160.09	535228	23181	154	N/A	N/A	58	448	133.49
s9234	72	228	63.00	240	2.30	12	4	**231**	83.15	54778	221	7	N/A	N/A	63	263	445.65
s9234.1	72	211	63.00	223	2.23	12	4	223	135.20	100265	321	0	N/A	N/A	63	255	444.04
prolog	40	136	31.00	154	0.89	20	8	**78**	16.76	74599	123	30	N/A	N/A	31	144	6.04
s13207	76	669	75.00	670	1.40	2	1	**611**	656.00	1101415	497	35	N/A	N/A	N/A	N/A	N/A
s15850	104	597	78.00	643	39.66	34	14	**603**	357.20	188929	195	22	N/A	N/A	N/A	N/A	N/A
s15850.1	124	534	103.00	544	15.80	13	7	**541**	78.76	29656	40	1	N/A	N/A	N/A	N/A	N/A
s38417	61	1636	60.00	1638	10.22	2	1	**1602**	245.40	185943	24	12	N/A	N/A	N/A	N/A	N/A

T_S: minimum clock period of the original circuit in g-frame
T_L: lowest clock period (= target clock period)
T_R: minimum clock period achieved by S-VR-C(MILP)
size: number of size gain computations
period: number of period slack computations
appl.: number of applied cone relocations
In column T_L, the cases are highlighted if $T_L < T_R$.
In column #reg of S-CR, the cases are highlighted if "#reg" of S-CR is less than "#reg" of P-CR.

other hand, in 4 circuits the number of registers by P&S-CR is reduced from original even if the minimum clock period is reduced. The number of registers by P&S-CR is less than that by P-CR in 20 circuits among 26 circuits while keeping the lowest clock period T_L. Although the optimality of P&S-CR in terms of the number of registers is not guaranteed, an optimal circuit is obtained by P&S-CR when S-VR-G(MILP) outputs an optimal circuit. The computation time of P&S-CR is the sum of those of P-CR and S-CR. The computation time of P&S-CR is practical in most circuits. Although S-CR saves the redundant computations of period slack, the computation time of S-CR is longer than that of P-CR in most circuits.

In the other 22 circuits among 48 circuits, the minimum clock periods in g-frame are not decreased by the register relocation. This means that these 22 circuits are optimal in terms of the clock period. The minimum number of registers without clock period constraint is obtained by solving the minimum-cost flow problem [18]. In 18 circuits among these 22 circuits, the minimum numbers of registers without clock period constraint are equal to the numbers of registers in the original circuits. This means that these 18 circuits are also optimal in terms of the number of registers. In the other 4 circuits, the minimum numbers of registers without clock period constraint are less than the originals. Although these 4 circuits are not optimal in terms of the number of registers, the minimum number of registers with clock period constraint cannot be obtained by S-VR-C(MILP) and S-VR-G(MILP) since they are relatively large. Since the results of S-CR at these 4 circuits are similar to the results shown in Table 2, the detailed results of these 4 circuits are omitted.

4.2 Trade-off between Clock Period and Number of Registers

Next, in order to observe the relations between the target clock period and the number of registers, the target clock period is changed from T_L to larger values. In experiments, P-CR and P&S-CR are applied to s344, s3330, s3384, and s9234.

The results of s344, s3330, s3384, and s9234 are shown in Fig. 11, 12, 13, and 14, respectively. #reg-optimal is a circuit which has the minimum number of registers without the clock period constraint and which is obtained by solving the minimum-cost flow problem corresponding to the circuit [18]. In s344 and s3330, P&S-CR achieves the lower bound of the number of registers if the target clock periods are set large enough. Moreover, the number of registers by P&S-CR becomes smaller than that by P-CR at most target clock

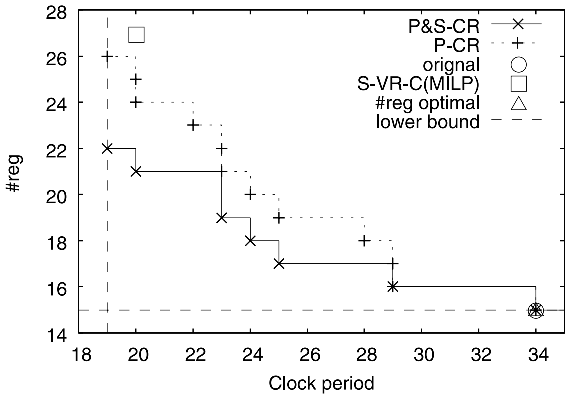


Fig. 11 Result of s344.

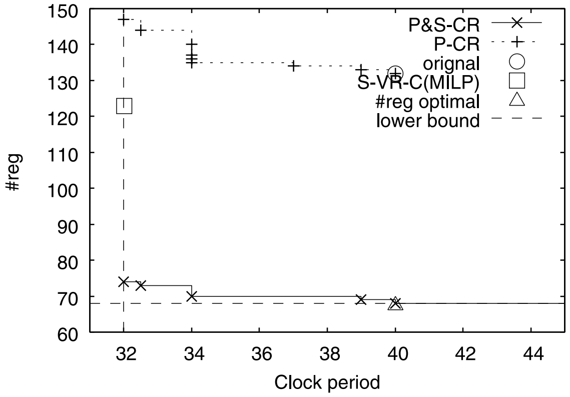


Fig. 12 Result of s3330.

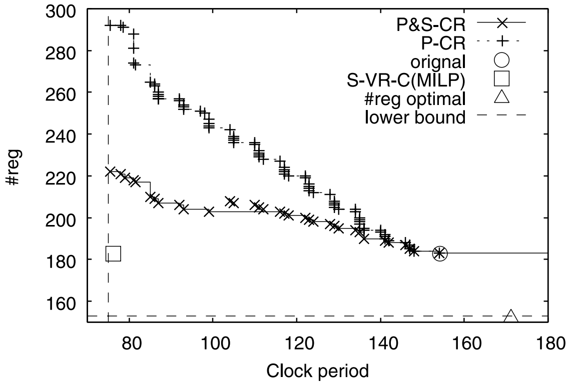


Fig. 13 Result of s3384.

periods. On the other hand, P&S-CR cannot achieve the lower bound of the number of registers in s3384 and s9234, and the number of registers cannot be reduced enough at some target clock periods in s3384. These facts imply that P&S-CR is heuristic.

4.3 Dependency on Delay Model

Last, in order to evaluate S-CR in more practical delay model, the maximum delay of each element is set to twice the minimum delay of each element and the target clock period is set to the lowest clock period T_L of each circuit. In 30 circuits among 48 circuits, the lowest clock period of the given circuit is smaller than the minimum clock period of the given circuit. In 29 circuits among these 30 circuits, P-CR obtains the circuit which works correctly at lowest clock period T_L of the given circuit. In 22 circuits among these 29 circuits the number of registers by P&S-CR is less than that by P-CR. The detail results are omitted since results are similar to Table 2.

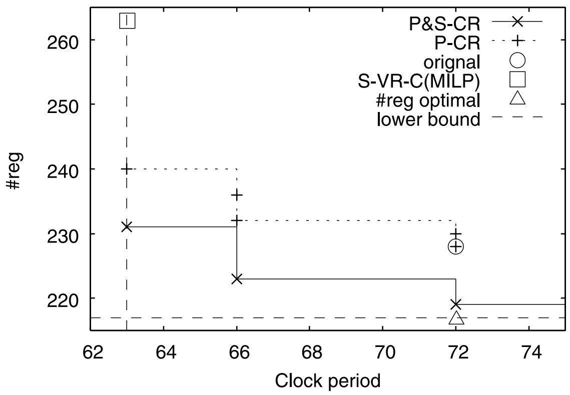


Fig. 14 Result of s9234.

5. Conclusions

In this paper, we propose a gate-level register relocation method that reduces the number of registers while keeping the target clock period in general-synchronous framework. The proposed method is a fast greedy local circuit modification method in general-synchronous framework in order to reduce the number of registers while keeping the given target clock period. Experiments show that the number of registers is reduced by the proposed method, and the computation time is practical in most circuits.

Since the proposed method is heuristic, there are some cases that the number of registers cannot be reduced enough or the computation time is large. Improvements of the proposed method in register size and in computation time are in our future works. Although the clock distribution circuit synthesis that realizes the clock schedule is not in the scope of this paper, the enhancement of the proposed method to take the clock distribution circuit synthesis into account is necessary to improve the final quality of the circuit.

Acknowledgments

This research was partially supported by Japan Society for the Promotion of Science (JSPS), Research Fellow-ship for Young Scientist 19-6015, 2007.

References

[1] Y. Kohira and A. Takahashi, "A fast register relocation method for circuit size reduction in generalized-synchronous framework," IEICE Technical Report, VLD2006-70, 2006.

[2] Y. Kohira and A. Takahashi, “A fast register relocation method for circuit size reduction in generalized-synchronous framework,” ISCAS, pp.1795–1798, 2007.

[3] J. Fishburn, “Clock skew optimization,” IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.

[4] R.B. Deoker and S.S. Sapatneker, “A graph-theoretic approach to clock skew optimization,” ISCAS, pp.407–410, 1994.

[5] A. Takahashi and Y. Kajitani, “Performance and reliability driven clock scheduling of sequential logic circuits,” ASP-DAC’97, pp.37–43, 1997.

[6] A. Takahashi, “General synchronous circuits using global clock — design methodologies, tools, and prospects,” IPSJ SIG Technical Report, 2006-SLDM-126, vol.2006, no.111, pp.159–164.

[7] A. Takahashi, “Practical fast clock-schedule design algorithms,” IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.1005–1011, April 2006.

[8] K. Inoue, W. Takahashi, A. Takahashi, and Y. Kajitani, “Schedule-clock-tree routing for semi-synchronous circuits,” IEICE Trans. Fundamentals, vol.E82-A, no.11, pp.2431–2439, Nov. 1999.

[9] S. Ishijima, T. Utsumi, T. Oto, and A. Takahashi, “A semi-synchronous circuit design method by clock tree modification,” IEICE Trans. Fundamentals, vol.E85-A, no.12, pp.2596–2602, Dec. 2002.

[10] T. Yoda and A. Takahashi, “Clock period minimization of semi-synchronous circuits by gate-level delay insertion,” IEICE Trans. Fundamentals, vol.E82-A, no.11, pp.2383–2389, Nov. 1999.

[11] Y. Kohira and A. Takahashi, “Clock period minimization method of semi-synchronous circuits by delay insertion,” IEICE Trans. Fundamentals, vol.E88-A, no.4, pp.892–898, April 2005.

[12] B. Taskin and I.S. Kourtev, “Delay insertion method in clock scheduling,” IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst., vol.25, no.4, pp.651–663, 2006.

[13] C. Lin and H. Zhou, “Clock skew scheduling with delay padding for prescribed skew domains,” ASP-DAC, pp.541–546, 2007.

[14] T. Yasui, K. Kurokawa, M. Toyonaga, and A. Takahashi, “A circuit optimization method by the register path modification in consideration of the range of feasible clock timing,” DA Symposium, pp.259–264, 2002.

[15] B.A. Rosdi and A. Takahashi, “Multi-clock cycle paths and clock scheduling for reducing the area of pipelined circuits,” IEICE Trans. Fundamentals, vol.E89-A, no.12, pp.3435–3442, Dec. 2006.

[16] X. Liu and M.C. Papaefthymiou, “Retiming and clock scheduling for digital circuit optimization,” IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst., vol.21, no.2, pp.184–203, 2002.

[17] Y. Kohira and A. Takahashi, “Gate-level register relocation in generalized synchronous framework for clock period minimization,” IEICE Trans. Fundamentals, vol.E90-A, no.4, pp.800–807, April 2007.

[18] C.E. Leiserson and J.B. Saxe, “Retiming synchronous circuitry,” Algorithmica, vol.6, no.1, pp.5–35, 1991.

[19] ILOG, CPLEX, http://www.ilog.com/

Yukihide Kohira received his B.E., M.E., and D.E. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2003, 2005, and 2007, respectively. He is currently a researcher of Department of Communications and Integrated Systems in Tokyo Institute of Technology. His research interests are in VLSI design automation and combinational algorithms. He is a member of IEEE and IPSJ.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997 and has been an associate professor since 1997. He visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2001 to 2002. He is currently with Department of Communications and Integrated Systems, Graduate School of Science and Engineering, Tokyo Institute of Technology. His research interests are in VLSI layout design and combinational algorithms. He is a member of IEEE and IPSJ.