# 論文 / 著書情報
# Article / Book Information

| | |
|---|---|
| 論題(和文) | |
| Title(English) | Delay Balancing by Min-Cut Algorithm for Reducing the Area of Pipelined Circuits |
| 著者(和文) | Rosdi Bakhtiar Affendi, 高橋 篤司 |
| Authors(English) | Bakhtiar Affendi Rosdi, Atsushi Takahashi |
| 出典(和文) | 第20回 回路とシステム軽井沢ワークショップ 論文集, Vol. , No. , pp. 643-648 |
| Citation(English) | Proc. the 20th Workshop on Circuits and Systems in Karuizawa, Vol. , No. , pp. 643-648 |
| 発行日 / Pub. date | 2007, 4 |
| URL | http://search.ieice.org/ |
| 権利情報 / Copyright | 本著作物の著作権は電子情報通信学会に帰属します。<br>Copyright (c) 2007 Institute of Electronics, Information and Communication Engineers. |

第20回　回路とシステム
軽井沢ワークショップ
The 20th Workshop on Circuits and Systems
in Karuizawa, April 23-24, 2007

# Delay Balancing by Min-Cut Algorithm for Reducing the Area of Pipelined Circuits

Bakhtiar Affendi Rosdi† Atsushi TAKAHASHI‡
†Department of Communications and Integrated Systems, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552 Japan
TEL:+81-3-5734-2665 FAX:+81-3-5734-2902
Email:†{fendi, atushi}@lab.ss.titech.ac.jp

## 1 Introduction

Circuit pipelining is one technique that has been used in order to shrink the clock period. Pipelining is a method in which a circuit is divided into a small number of stages and intermediate registers are inserted between stages to store the intermediate data. With this method, extra circuit area is required to situate the additional intermediate registers and the size of the clock tree is also increased.

Recently, to overcome this problem, several studies have been carried out on wave pipelining [1], which is a method of speeding up the circuit without the insertion of intermediate registers. However, wave pipelining requires tighter timing constraints. In wave pipelining, there may exist a number of 'waves' of data in a circuit at any given time. Therefore, to avoid data collisions, delay balancing is required, which increases the circuit area.

In [2], an algorithm that removes intermediate registers to reduce the circuit size while maintaining the circuit behavior was proposed. The obtained circuits contain multi-clock cycle paths as in wave pipeline, but timing constraints are relaxed by clock scheduling and satisfied without delay balancing.

In [3], the algorithm in [2] was enhanced by introducing a limited delay balancing. The algorithm in [3] inserts delay elements to which the removed intermediate registers were located if necessary. A smaller circuit is obtained since the algorithm can select either the removal of a register or the replacement of a register with delay elements.

In this paper, we enhance the algorithm proposed in [3] by introducing a more general delay balancing. In the proposed algorithm, the locations where delay elements are inserted are not restricted to the locations of the removed intermediate registers. The algorithm determines the locations where delay elements are inserted by using the concept of delay-demand and delay-slack proposed in [4]. In [4], delay elements are greedily inserted one by one to a circuit to minimize the feasible clock period according to delay-demand and delay-slack. While, in the proposed algorithm, in order to obtain a smaller circuit that works correctly within a given target clock period range, delay elements are inserted simultaneously to a circuit by using the well-known flow based min-cut algorithm.

Experiments with a multiplier verify that, given a particular target clock-period range, the proposed algorithm obtains a circuit with smaller area compared with the circuit obtained by the algorithm in [3]. Although the size of an obtained circuit is same as the circuit obtained by [3] in few cases in the experiments, we believe that the size reduction becomes larger if the proposed algorithm is applied to larger circuits.

## 2 Preliminaries

### 2.1 Timing Constraints

In this paper, we consider a circuit consisting of registers and gates, and wires connecting them. We refer to them as elements. A circuit is represented by the circuit graph $G = (V_g, E_g)$, where $V_g$ is the vertex set corresponding to elements in the circuit and $E_g$ is the directed edge set corresponding to signal propagations in the circuit. In this paper, we assume each element has minimum and maximum delay. Let $d_{min}(v)$ ($d_{max}(v)$) be the minimum (maximum) delay of $v \in V_g$. Let $V_r$ be the register set of $G$. Necessarily, the register set is a subset of $V_g$.

An example of the circuit graph is shown in Fig. 1. In Fig. 1, $\{u_1, u_2, v_1, v_2, v_3, w_1, w_2\}$ is the register set, and the figures in each vertex represent its minimum and maximum delay.

The clock timing $s(r)$ of register $r$ is the difference in clock signal arrival time between $r$ and an arbitrarily chosen (perhaps hypothetical) reference register. The set of clock timings is called a clock-schedule.

We make the basic assumption that a circuit works correctly if the following two types of constraint are satisfied for each register pair with signal propagation [5],[6]:

**Setup Const. :**
$s(r_i) - s(r_j) \leq \beta_{r_i,r_j} T - d_{\max}(r_i, r_j)$

**Hold Const. :**
$s(r_j) - s(r_i) \leq d_{\min}(r_i, r_j) - \alpha_{r_i,r_j} T$

where $T$ is the clock period, $d_{\max}(r_i, r_j)$ ($d_{\min}(r_i, r_j)$) is the maximum (minimum) propagation delay from register $r_i$ to register $r_j$ along a combinatorial circuit, and $\beta_{r_i,r_j}$ and $\alpha_{r_i,r_j}$ are given constants ($\beta_{r_i,r_j} > \alpha_{r_i,r_j} \geq 0$), respectively. Note that for a pair of registers with a single-clock cycle path, $\beta_{r_i,r_j}$ and $\alpha_{r_i,r_j}$ are given by 1 and 0, respectively. This formulation is sufficiently general to deal with multi-clock cycle paths and multi-clocks that have different periods.

If $\alpha_{r_i,r_j}$ is 0 for every pair, the feasible clock period has no upper bound, i.e. if the clock period $T$ is feasible then any $T^{'}$(where $T^{'} \geq T$) is feasible. However, the feasible clock period is bounded above if $\alpha_{r_i,r_j}$ is not 0 for some pair $(r_i, r_j)$.

From the above constraints, when the clock schedule and the signal propagation delay are known, the minimum and maximum feasible clock period, $T_{\min}$ and $T_{\max}$, can be determined from the setup and hold constraints, respectively.

If the clock timing is not fixed, then $T_{\min}$ and $T_{\max}$ depend on each other. In order for the circuit to tolerate clock jitter and delay variation, the circuit must works correctly throughout a certain clock-period range $\delta$. A circuit works correctly for a clock period between $T$ and $T+\delta$, if the following constraints are satisfied.

**Setup Const. :**
$s(r_i) - s(r_j) \leq \beta_{r_i,r_j} T - d_{\max}(r_i, r_j)$
**Hold Const. :**
$s(r_j) - s(r_i) \leq d_{\min}(r_i, r_j) - \alpha_{r_i,r_j}\delta - \alpha_{r_i,r_j} T$

In the following, our target is to get a circuit that works correctly for a clock period between $T$ and $T+\delta$, where $T$ and $\delta$ are given.

A pipeline consists of several pipeline stages. Let $p$ be the number of pipeline stages. Each pipeline stage is referred to by the stage number. The stage numbers of pipeline stages from the primary inputs to the primary outputs are assigned to 1, 2, ..., $p$, respectively. Registers between pipeline stages are also referred to by the stage number. The stage number of a register between stage-$i$ and stage-$(i+1)$ is $i$. The stage numbers of a register at the primary inputs and the primary outputs are 0 and $p$, respectively. Let $stage(r)$ be the stage number of register $r$.

In the proposed pipeline architecture, setup and hold constraints for register pair $r_i$ and $r_j$ with signal propagation from $r_i$ to $r_j$ are defined by using $\beta_{r_i,r_j} = stage(r_j) - stage(r_i)$ and $\alpha_{r_i,r_j} = stage(r_j) - stage(r_i) - 1$. Note that in the conventional pipeline architecture, $\beta_{r_i,r_j} = 1$ and $\alpha_{r_i,r_j} = 0$ since $stage(r_j) - stage(r_i) = 1$ if a signal propagates from $r_i$ to $r_j$.

An example of the circuit of the proposed pipeline architecture is shown in Fig. 2. In Fig. 2, "buff" is the delay elements, $v_1$ is the intermediate register and it is between stage-1 and stage-2, therefore $stage(v_1) = 1$.

### 2.2 Delay-demand and delay-slack

A circuit is not work correctly at target clock-period range, if the timing constraints are violated. The violated timing constraints can be eliminated by inserting some delay elements to the edges in a circuit graph. The concept of delay-demand and delay-slack proposed in [4] can be used to find the edges in a circuit graph where delay elements should be inserted.

For any edge $e$ in a circuit graph, let $break(e)$ be the time when the earliest signal arrives at $e$, $complete(e)$ be the time when the latest signal arrives at $e$, $hold(e)$ be the earliest possible time of signal arrival at $e$ that can keep Hold constraints, and $setup(e)$ be the latest possible time of signal arrival at $e$ that can keep Setup constraints. Moreover, let delay-slack $slack(e)$ be $setup(e) - complete(e)$, that means the amount of delay which can be inserted, and delay-demand $demand(e)$ be $hold(e) - break(e)$, that means the amount of delay which must be inserted.

The formal definition of *demand* and *slack* to make a circuit $G$ to works correctly for a clock period between $T$ and $T+\delta$ is as follows. Note that, since our target circuit contains multi-clock cycle paths that is different from the target circuit of the algorithm proposed in [4], the definition stated in this paper is little bit different from the definition stated in [4].

For a given clock period $T$, clock-period range $\delta$ and clock schedule $S$ of a circuit $G$, $break, complete, hold, setup, demand$, and $slack$ of $e = (u, v) \in E_g$ are defined as follows; If $u$ is a register ($u \in V_r$),

$break(e) = s(u) + d_{min}(u) + T * stage(u)$
$complete(e) = s(u) + d_{max}(u) + T * stage(u)$

otherwise,

$break(e) = \min_{e' \in Fi(e)}\{break(e') + d_{min}(u)\}$
$complete(e) = \max_{e' \in Fi(e)}\{complete(e') + d_{max}(u)\}$

where $Fi(e)$ is the edge set whose tail is $u$. If $v$ is a register ($v \in V_r$),

$hold(e) = s(v) + (T+\delta) * (stage(v) - 1)$
$setup(e) = s(v) + T * stage(v)$

otherwise,

$$hold(e) = \max_{e' \in Fo(e)}\{hold(e') - d_{min}(v)\}$$
$$setup(e) = \min_{e' \in Fo(e)}\{setup(e') - d_{max}(v)\}$$

where $Fo(e)$ is the edge set whose head is $v$. Moreover,

$$demand(e) = hold(e) - break(e)$$
$$slack(e) = setup(e) - complete(e)$$

Note that *demand* and *slack* depend on clock schedule $S$. Different clock schedule results different *demand* and *slack*.

*demand* and *slack* for the circuit and clock schedule shown in Fig. 3 are shown in the same figure.

## 3 Delay Balancing with Min-cut Algorithm

In our proposed algorithm, first, all intermediate registers are removed. When all intermediate registers are removed, the obtained circuit will not work correctly within a given target clock-period range since several timing constraints are violated. In order to make the circuit works correctly within a given target clock-period range, the violated timing constraints need to be eliminated. The delay-demand of an edge is positive when the timing constraint is violated. The violated timing constraints can be eliminated by forcing positive delay-demands to be smaller than or equal to 0.

In this paper, the delay-demand and delay-slack are computed according to the clock schedule for the pipelined circuit where some of the intermediate registers are removed and works correctly at target clock period range, which is obtained by [3]. The clock schedule is used for the delay-demand and delay-slack computation for the circuit without intermediate registers.

The delay-demand of an edge can be reduced by inserting delay elements to the edge. We only use one type of buffer as a delay element. The cost of an edge $e$ is defined as the number of buffers that need to be inserted on $e$ in order to make the delay-demand of $e$ becomes smaller than or equal to 0. However, the number of buffers that can be inserted on $e$ is limited by the delay-slack of $e$. Therefore, if the number of buffers that need to be inserted is larger than the number of buffers that can be inserted, the cost of the edge is defined as $\infty$. For an edge $e_r$ which corresponds to an edge where an intermediate register is located, if the total area of buffers that need to be inserted is larger than or equal to the area of an intermediate register, the cost of $e_r$ is defined corresponding to the area of the intermediate register. For example, if the area of an intermediate register is $m$ times larger than the area of a buffer, the cost of $e_r$ is $m$. The cost of an edge whose delay-demand is 0 or negative is 0.

In [4], delay elements are repeatedly inserted to each edge where *demand* is larger than 0, until *demand* for all edges is smaller than or equal to 0. This is time consuming, also there are some possibilities that the order of the delay element insertion will affect the total number of inserted delay elements. To avoid that kind of problem, here we use the minimum cut algorithm proposed in [7] to find the edges to which the delay elements are needed to be inserted.

A minimum cut is a cut whose total edge cost is minimum and divides the primary inputs and outputs into two different parts. In order to find a minimum cut that divides the primary inputs and outputs into two different parts, two hypothetical vertex *source* and *sink* are inserted to the circuit graph. *source* (*sink*) is connected to the primary inputs (outputs) by edges with infinite cost. The worst case of the found minimum cut corresponds to the solution obtained by the algorithm shown in [3].

The delay elements and intermediate registers are inserted on all edges in the found minimum cut simultaneously. By inserting the delay elements and intermediate register to the edges of a minimum cut, the delay-demand of all edges become smaller than or equal to 0, thus makes the circuit works correctly within the given target clock-period range. Therefore there are no order of delay elements insertion. Also, there are no repeated process of the finding of a minimum cut and the computation of delay-demand and delay-slack, thus reduced the computation time.

Our algorithm is heuristic. Different clock schedule may results different solution. There may exist a solution where the total of inserted buffers is smaller than the solution obtained by our proposed algorithm, since in our proposed algorithm the clock schedule is fixed.

The details of our proposed algorithm are as follows.

**Inputs** : Circuit graph $G$. The target clock-period range $\delta$.

**Outputs** : Circuit graph $G^{'}$ after delay insertion.

**Step 1** : Compute the clock schedule and the minimum clock period $T_{\min}$ of the circuit obtained by [3].

**Step 2** : Compute the *demand* and *slack* of the circuit obtained by removing all intermediate registers. *demand* and *slack* are computed by using the clock schedule and the minimum clock period obtained by Step 1.

**Step 3** : Based on the *demand* $(e)$, *slack* $(e)$, minimum delay and maximum delay of the delay elements, compute the cost of edge $e$.

**Step 4** : Find a minimum cut that divides the primary inputs and outputs of the circuit graph into two different parts by the Ford Fulkerson algorithm in [7].

**Step 5** : Output circuit graph $G'$ after the delay elements and intermediate registers are inserted to the location of the found minimum cut and terminate.

### 3.1 Example

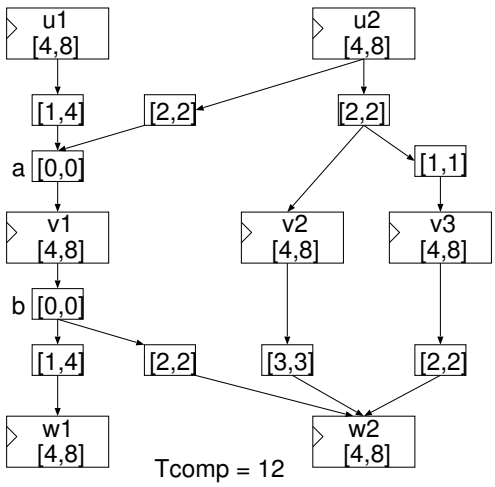


Fig. 1: Conventional pipelined circuit graph

To explain the behavior of the algorithm, we apply the algorithm to the pipelined circuit shown in Fig. 1. In this example, our target clock-period range $\delta$ is 3 and the timing of each register is scheduled. Parameters are set as follows: setup and hold time for registers are 0; the minimum and maximum delay of the intermediate registers are 4 and 8, respectively; the minimum and maximum delay of the buffers are 1 and 2, respectively; the size of an intermediate register is 4 times larger than the size of a buffer, that is, the upper bound of the cost for the edge where intermediate register is located is 4. For the original circuit with zero clock-skew, the minimum feasible clock period $T_{comp}$ is 12.

The circuit after the algorithm shown in [3] is applied is shown in Fig. 2. The minimum clock period $T_{\min}$ of the obtained circuit is 12 and the obtained clock schedule is as shown in Fig. 2. The circuit obtained by removing all intermediate registers is

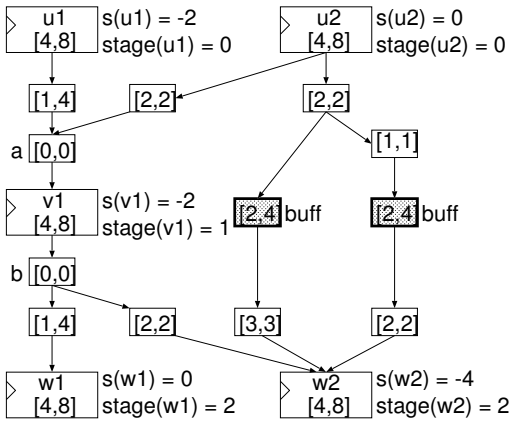


Fig. 2: Circuit graph of the proposed pipeline architecture

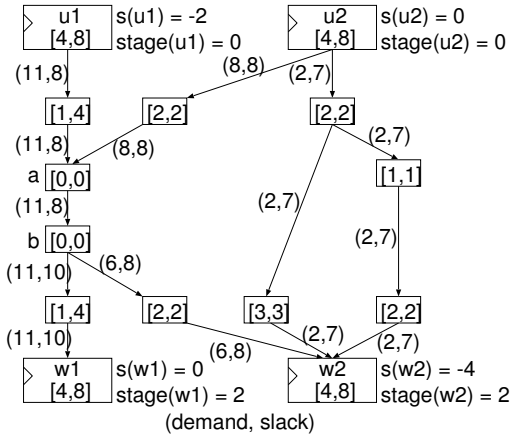


Fig. 3: Demand and slack of the circuit.

shown in Fig. 3. Delay-slack and delay-demand are determined as shown in Fig. 3. The cost of each edge is computed as shown in Fig. 4. In Fig. 4, the edge between gate $a$ and $b$ is the edge whose intermediate register $v1$ was located, and the number of buffers that need to be inserted is larger than the number of buffers that can be inserted. Therefore, the edge cost of the edge between gate $a$ and $b$ is 4 that is the total area of an intermediate register. The minimum cut of the obtained circuit graph is found as shown in Fig. 4. As shown in Fig. 5, by inserting 2 buffers between register $u_2$ and gate $c$ and an intermediate register between gate $a$ and $b$, the delay-demand for all edges become smaller than or equal to 0.

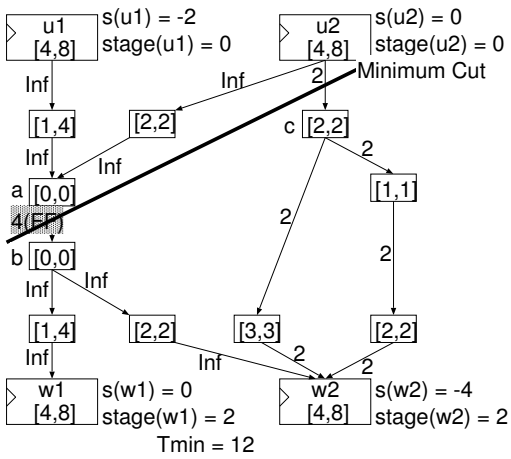


Fig. 4: The cost of each edge and the found minimum cut

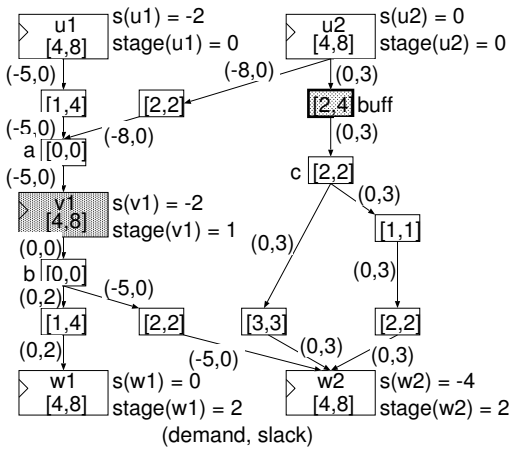


Fig. 5: The circuit after the insertion of delay elements and intermediate registers on the edges of the found minimum cut

Table 1: Statistics of multiplier

| | | Circuit delay [ps] | | | |
|---|---|---|---|---|---|
| circuit | Total #FF | 1st stage | | 2nd stage | |
| | | min | max | min | max |
| 6bit_mul | 42 | 949 | 3888 | 382 | 3581 |
| 16bit_mul | 120 | 807 | 5406 | 391 | 4423 |

## 4 Experiments

The proposed algorithm was implemented on a PC with Pentium 4 (CPU 3GHz, memory 513764kb). Since there are no benchmark examples of pipelined circuits, two simple examples, briefly described below, were constructed for our experiments.

- 6bit_mul: A 2-stage multiplier that multiplies two 6-bit numbers. The first stage uses a carry-save adder with Wallace tree structure [8] and the second stage uses a ripple-carry adder.
- 16bit_mul: A 2-stage multiplier that multiplies two 16-bit numbers. The first stage uses a carry-save adder with Wallace tree structure [8] and the second stage uses a carry-look-ahead adder.

The statistics of the circuit are shown in Table 1. The ROHM 0.35 $\mu$m process library was used for these experiments. In the library, the area of a register is four times larger than the area of a buffer. Therefore, in our experiments the upper bound of the cost for the edge where intermediate register is located is 4. The timing of each I/O pin was scheduled as well as the timing for each register.

The results of our experiments are shown in Table 2. "Delay Balancing with Min-Cut" is the result when our proposed algorithm is applied. "Delay Balancing with greedy" is the result when the delay elements are greedily inserted one by one to the edges whose delay-demand is positive until delay-demand for all edges is smaller than or equal to 0. "Replacement of FF with delay in [3]" is the result when the register replacement algorithm shown in [3] is applied. Ori. is the original circuit containing the intermediate registers and the clock timing of all registers are fixed at 0 (zero clock-skew). "$\delta$[ps]" and "$T_{\min}$[ps]" are the target clock-period range and the output minimum feasible clock period, respectively. "Buff. (#)" and "FF (#)" are the number of buffers and intermediate registers, respectively. "Buff. + FF (unit)" is the total area of the buffers and intermediate registers that had been inserted. Note that the area of a FF is 4 times larger than the area of a buffer. "Buff. + FF (%)" is the percentage of the total area of the buffers and intermediate registers compared with the total area of intermediate registers in the original circuit. "Time[$s$]" is the computation time of the respective algorithm.

The results show that the circuit area of the circuit obtained by our proposed algorithm is smaller compare with the circuit area of the circuit obtained by [3]. Although the size of an obtained circuit is same as the circuit obtained by [3] in few cases in the experiments, we believe that the size reduction becomes larger if the proposed algorithm is applied to larger circuits.

Table 2: Experimental Results

| | | | | Delay Balancing with Min-Cut | | | | | Delay Balancing with Greedy | | | | | Replacement of FF with Delay in [3] | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Circuit | $\delta$ | $T_{\min}$ | | Buff. | FF | Buff. + FF | | Time | Buff. | FF | Buff. + FF | | Time | Buff. | FF | Buff. + FF | | Time |
| | [ps] | [ps] | (%) | # | # | unit | % | [s] | # | # | unit | % | [s] | # | # | unit | (%) | [s] |
| 6bit_mul | Ori. | 3888 | (100) | 0 | 18 | 72 | (100) | - | 0 | 18 | 72 | (100) | - | 0 | 18 | 72 | (100) | - |
| | 0 | 3855 | (99) | 4 | 1 | 8 | (11) | 0.18 | 7 | 1 | 11 | (15) | 0.26 | 4 | 1 | 8 | (11) | 0.02 |
| | 200 | 3839 | (99) | 3 | 2 | 11 | (15) | 0.20 | 12 | 2 | 20 | (28) | 0.63 | 3 | 2 | 11 | (15) | 0.04 |
| | 400 | 3846 | (99) | 2 | 3 | 14 | (19) | 0.20 | 11 | 3 | 23 | (32) | 0.44 | 2 | 3 | 14 | (19) | 0.03 |
| | 600 | 3852 | (99) | 5 | 3 | 17 | (24) | 0.19 | 19 | 3 | 31 | (43) | 0.64 | 5 | 3 | 17 | (24) | 0.05 |
| | 800 | 3858 | (99) | 4 | 4 | 20 | (28) | 0.21 | 13 | 4 | 28 | (39) | 0.59 | 4 | 4 | 20 | (28) | 0.05 |
| | 1000 | 3864 | (99) | 8 | 4 | 24 | (33) | 0.22 | 19 | 5 | 39 | (54) | 0.66 | 5 | 5 | 25 | (35) | 0.05 |
| 16bit_mul | Ori. | 5406 | (100) | 0 | 56 | 224 | (100) | - | 0 | 56 | 224 | (100) | - | 0 | 56 | 224 | (100) | - |
| | 0 | 5390 | (99) | 19 | 2 | 27 | (12) | 70.40 | 14 | 8 | 46 | (21) | 926.67 | 4 | 8 | 36 | (16) | 2.97 |
| | 200 | 5393 | (99) | 11 | 7 | 39 | (17) | 62.67 | 8 | 11 | 52 | (23) | 572.88 | 1 | 11 | 45 | (20) | 3.49 |
| | 400 | 5395 | (99) | 9 | 17 | 77 | (34) | 62.46 | 9 | 19 | 85 | (38) | 620.07 | 7 | 19 | 83 | (37) | 6.31 |
| | 600 | 5390 | (99) | 37 | 25 | 137 | (61) | 67.43 | NG | NG | NG | (-) | 1797.32 | 20 | 30 | 140 | (63) | 9.57 |
| | 800 | 5378 | (99) | 17 | 42 | 185 | (83) | 71.31 | NG | NG | NG | (-) | 2146.73 | 17 | 42 | 185 | (83) | 10.94 |
| | 1000 | 5368 | (99) | 9 | 46 | 193 | (86) | 69.32 | NG | NG | NG | (-) | 1299.16 | 9 | 46 | 193 | (86) | 8.96 |

NG : Solution cannot be obtained.

## 5 Conclusion

It has been shown that the area of a pipelined circuit can be reduced by implementing a multi-clock cycle path technique together with clock scheduling and delay balancing with min-cut algorithm. The size of the clock tree also is reduced when the number of intermediate registers is reduced. There may exist a solution where the total number of inserted buffers is smaller than the solution obtained by our proposed algorithm. We believe that our proposed algorithm can be enhanced so that it can reduce the total number of inserted buffers. This is a topic for future investigation.


### Acknowledgements

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc., Cadence Design Systems, Inc., and Rohm Corporation.


## References


[1] W. J. Kim and Y. Kim, "Clocking for correct functionality on wave pipelined circuits," in *Proc. IEEE International ASIC/SOC Conference*, 2003, pp. 161–164.

[2] B. A. Rosdi and A. Takahashi, "Multi-clock cycle paths and clock scheduling for reducing the area of pipelined circuits," *IEICE Transactions on Fundamentals*, vol. E89-A, no. 12, pp. 3435–3442, 2006.

[3] B. Rosdi and A. Takahashi, "Replacement of register with delay element for reducing the area of pipelined circuits," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS)*, 2006, pp. 802–805.

[4] Y. Kohira and A. Takahashi, "Clock Period Minimization Method of Semi-Synchronous Circuits by Delay Insertion," *IEICE Transactions on Fundamentals*, vol. Vol.E88-A, no. 4, pp. 892–898, 2005.

[5] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–951, 1990.

[6] B. A. Rosdi and A. Takahashi, "Reduction on the usage of intermediate registers for pipelined circuits," in *Proc. the Workshop on Synthesis and System Integration of Mixed Technologies (SASIMI 2004)*, 2004, pp. 333–338.

[7] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[8] C. Wallace, "A suggestion for fast multiplier," *IEEE Trans. on Electronic Computers*, vol. 13, no. 2, pp. 14–17, 1964.