論文 / 著書情報
Article / Book Information

Title	Low Area Pipelined Circuits by the Replacement of Registers with Delay Elements
Authors	Bakhtiar Affendi Rosdi, Atsushi Takahashi
Citation	IEICE Trans. Fundamentals, Vol. E90-A, No. 12, pp. 2736-2742
Pub. date	2007, 12
URL	http://search.ieice.org/
Copyright	(c) 2007 Institute of Electronics, Information and Communication Engineers

IEICE TRANS. FUNDAMENTALS, VOL.E90–A, NO.12 DECEMBER 2007

PAPER Special Section on VLSI Design and CAD Algorithms

Low Area Pipelined Circuits by the Replacement of Registers with Delay Elements*

Bakhtiar Affendi ROSDI[†a)] ***and*** **Atsushi TAKAHASHI**[†], ***Members***

SUMMARY A new algorithm is proposed to reduce the area of a pipelined circuit using a combination of multi-clock cycle paths, clock scheduling and delay balancing. The algorithm analyzes the circuit and replaces intermediate registers with delay elements under the condition that the circuit works correctly at given target clock-period range with the smaller area. Experiments with pipelined multipliers verify that the proposed algorithm can reduce the area of a pipelined circuit without degrading performance.

key words: *pipelined circuits, multi-clock cycle paths, clock scheduling, delay balancing*

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 [2], 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 [3], they proposed an algorithm to reduce the number of intermediate registers of a pipelined circuit by using a combination of multi-clock cycle paths and clock scheduling. A multi-clock cycle path is a path from register to register where data transmission takes more than one clock period. Note that in wave pipelining, all paths are multi-clock cycle paths. Introducing a multi-clock cycle path into a pipelined circuit allows some intermediate registers to be removed. Their algorithm removes an intermediate register without delay balancing. Clock scheduling is a technique in which the clock skew of a register is intentionally introduced to improve circuit performance by relaxing the timing constraints. Using clock scheduling, more intermediate registers can be removed, without the need for delay balancing. Note that the reduction on the number of intermediate registers will reduce the complexity of clock tree synthesis.

The algorithm proposed in [3] removes some intermediate registers, while the others remain so that the circuit works correctly at given target clock-period range. However, there are some possibilities that a remaining intermediate register can be replaced with some delay elements whose total area is smaller than the register. In this paper, we propose a new algorithm that replaces intermediate registers with delay elements whose total area is smaller than the registers, under the condition that the pipelined circuit works correctly at given target clock-period range. By replacing an intermediate register with delay elements, the number of intermediate registers in a pipelined circuit is reduced.

An intermediate register is a register that stores one of the data between stages of the data flow. Here we consider a pipelined circuit such as adder or multiplier, whose data flow is in one way without any feedback. However, we believe that our algorithm can be enhanced so that it can be used to a pipelined circuit with feedback.

In our proposed algorithm, all intermediate registers of the pipelined circuit are initially removed same as in [3]. Then using the algorithm in [4], the obtained circuit is checked whether it works correctly at given target clock-period range or not. If the circuit does not work correctly at given target clock-period range, some timing constraints are violated. The violated timing constraints can be eliminated by intermediate register insertion or delay element insertion. The algorithm eliminates the violated timing constraints iteratively by inserting registers or buffers with small area cost. As the result, our proposed algorithm replaces an intermediate register with delay elements whose total area is smaller than the register.

Our algorithm is an enhancement of the algorithm proposed in [3], since an intermediate register is removed if no buffer is inserted to the location where the removed intermediate register is located. Delay balancing is implemented as the replacement of intermediate register with some delay elements.

Experiments with multipliers verify that, given a particular target clock-period range, the proposed algorithm can obtain a circuit with smaller area compared with the circuit obtained by the algorithm in [3]. The number of intermediate registers in the circuit obtained by proposed algorithm is smaller compared with the number of intermediate registers

Manuscript received March 6, 2007.
Manuscript revised June 14, 2007.
Final manuscript received July 31, 2007.

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

*The preliminary version was presented at [1].

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

DOI: 10.1093/ietfec/e90–a.12.2736

in the circuit obtained by the algorithm in [3].

2. Preliminaries

We consider a circuit with a single clock consisting of registers linked by combinatorial circuits. The clock timing $s(v)$ of register v is the difference in clock signal arrival time between v 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. : $s(u) - s(v) \leq \beta_{u,v}T - d_{\max}(u,v)$
Hold Const. : $s(v) - s(u) \leq d_{\min}(u,v) - \alpha_{u,v}T$

where T is the clock period, $d_{\max}(u,v)$ ($d_{\min}(u,v)$) is the maximum (minimum) propagation delay from register u to register v along a combinatorial circuit, and $\beta_{u,v}$ and $\alpha_{u,v}$ are given integer constants ($\beta_{u,v} > \alpha_{u,v} \geq 0$). Note that for a pair of registers with a single-clock cycle path, $\beta_{u,v}$ and $\alpha_{u,v}$ 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 $\alpha_{u,v}$ 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 $\alpha_{u,v}$ is not 0 for some pair (u,v).

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. $T_{\min}$ has to be minimized under the constraint that the circuit works correctly throughout a certain clock-period range, in order for the circuit to tolerate clock jitter and delay variation. The above constraints become:

Setup Const. : $s(u) - s(v) \leq \beta_{u,v}T_{\min} - d_{\max}(u,v)$
Hold Const. : $s(v) - s(u) \leq d_{\min}(u,v) - \alpha_{u,v}\delta - \alpha_{u,v}T_{\min}$

where δ is the clock-period range, i.e. $\delta = T_{\max} - T_{\min}$. Therefore if δ is given, then, by using the above constraints, clock timings can be determined so that the circuit works correctly for a clock period between $T_{\min}$ and $T_{\min}+\delta$. In the following, our target is to minimize $T_{\min}$ under the constraint that the circuit is feasible throughout the given clock-period range δ.

These constraints are represented by the constraint graph $G(V,E)$ of the circuit, which is defined as follows: a vertex $v \in V$ corresponds to a register; a directed edge $(u,v) \in E$ corresponds to either type of constraint; an edge (u,v) corresponding to the setup (hold) constraint is called a Z-edge (D-edge), and the weight $w(u,v)$ of (u,v) is $\beta_{v,u}T - d_{\max}(v,u)(d_{\min}(u,v) - \alpha_{u,v}\delta - \alpha_{u,v}T)$. A Z-edge (D-edge) corresponding to a single-clock cycle path, i.e. $\beta_{v,u} = 1$ and $\alpha_{u,v} = 0$, is called single Z-edge (single D-edge). While, a Z-edge (D-edge) corresponding to a multi-clock cycle path is called multi Z-edge (multi D-edge). The constraint graph G corresponding to clock period t and clock-period range δ is denoted by $G_\delta(t)$. We may denote $G_\delta(t)$ as $G(t)$ if no confusions occur.

Let the weight of a directed cycle in a constraint graph be the sum of weights of edges on the directed cycle. We refer to a directed cycle whose weight is negative as negative cycle. It is known that a circuit can works correctly if and only if there is no negative cycle in the constraint graph of the circuit [7].

3. Delay and Register Insertion Effects on the Constraint Graph

There is a negative cycle in the constraint graph of a circuit if the circuit does not work correctly for any clock schedule. To eliminate the negative cycle, a certain amount of weight needs to be added to edges on the negative cycle by an operation. However, it is important to make sure that the operation will not create other negative cycles. This section discusses about the changes of the weight and topology of the constraint graph when delay element and intermediate register are removed and inserted.

In our proposed algorithm, first, all intermediate registers are removed. When an intermediate register is removed, the weight of D-edge and Z-edge of the constraint graph are changed, as well as its topology. As for example, when an intermediate register v is removed, single-clock cycle paths (u,v) and (v,w) whose total minimum (maximum) delay is σ (σ') become two-clock cycle path (u,w) whose minimum (maximum) delay is $\sigma - \omega$ ($\sigma' - \omega'$), where ω ($\omega > 0$) and ω' are the minimum and maximum delay of the intermediate register v, respectively. In the constraint graph of the circuit, the single D-edges (u,v) and (v,w) whose total weight is σ are removed and a multi D-edge (u,w) whose weight is $\sigma - \omega - T$ is inserted. Similarly, the single Z-edges (w,v) and (v,u) whose total weight is $2T - \sigma'$ are removed and a multi Z-edge (w,u) whose weight is $2T - (\sigma' - \omega')$ is inserted. The weight of the multi D-edge is smaller than the corresponding original single D-edges. While the weight of the multi Z-edge is larger than the corresponding original single Z-edges.

When the weight and topology of the constraint graph are changed, there are some possibilities that the constraint graph contains a negative cycle thus makes the circuit infeasible. In order to make the circuit feasible, a negative cycle should be eliminated by increasing the total weight of the cycle.

Total weight of a cycle can be increased by increasing the weight of D-edge or Z-edge. The weight of D-edge can be increased by increasing the minimum delay between registers or by recovering back an intermediate register to the original location. While, the weight of Z-edge can be increased by reducing the maximum delay between registers.

In this paper, we choose to increase the weight of D-

edge to increase the total weight of a cycle because it is more easy to do it.

As for example, let consider a multi D-edge (u, w) whose weight is $\sigma - \omega - T$ corresponding to a multi-clock cycle path (u, v, w). When a delay element whose minimum delay is λ is inserted to the multi-clock cycle path (u, v, w), the weight of the multi D-edge (u, w) increases by λ from $\sigma - \omega - T$ to $\sigma - \omega - T + \lambda$. While, when an intermediate register v whose minimum delay is ω is recovered back to the multi-clock cycle path (u, v, w), the total weight of the single D-edges (u, v) and (v, w) is $\omega+T$ larger than the weight of the corresponding multi D-edge (u, w).

A negative cycle becomes non-negative if a certain amount of weight is added. For a cycle C in constraint graph, let weight demand $w^d(C)$ be the weight needed to be added in order to make the total weight $w(C)$ of C becomes 0, that is $w^d(C) = 0$ if $w(C) \geq 0$, and $w^d(C) = -w(C)$ if $w(C) < 0$. As for example, if $w(C) = -4$, then $w^d(C) = 4$.

The minimum delay of a multi-clock cycle path can be increased by inserting some delay elements to the path. However, inserting some delay elements may increases the maximum delay of the path thus reduces the weight of corresponding multi Z-edge. Especially, if a negative cycle consists of one multi D-edge (u, w) and one multi Z-edge (w, u) corresponding to the multi-clock cycle path (u, v, w), the cycle remains negative even if $w^d(C)$ is added to point v. The weight of the cycle remains negative since the reduction of the weight of multi Z-edge (w, u) is larger than or equal to the increase of the weight of multi D-edge (u, w).

Note that, the adding of the weight to multi D-edge may reduce the weight of multi Z-edge thus reduces the weight of other cycles and makes the other cycles become negative, that we do not consider here. In our experiments, it is sufficient to make sure a negative cycle does not consist of one multi D-edge (u, w) and one multi Z-edge (w, u) corresponding to the multi-clock cycle path (u, v, w).

4. Reduction on the Area of Pipelined Circuits

In this paper we consider a problem on how to reduce the area of a pipelined circuit, subject to the minimum feasible clock period is lower than or equal to the original circuit's clock period T_{comp} and the obtained pipelined circuit works correctly at given target clock-period range.

4.1 Reduction on the Number of Intermediate Registers

In [3], they have proposed an algorithm that reduces the area of a pipelined circuit by reducing the number of intermediate registers.

In their proposed algorithm, all intermediate registers of the pipelined circuit are initially removed. Then the obtained circuit is checked whether it works correctly at T_{comp} by the algorithm shown in [4]. When an intermediate register is removed, the obtained circuit may not work correctly at T_{comp}. If the circuit does not work correctly at T_{comp}, a negative cycle is found in the corresponding constraint graph and it contains a multi D-edge. In [3], the found negative cycle is eliminated by repeatedly recovering back an intermediate register to the corresponding multi D-edge until the circuit works correctly at T_{comp}.

4.2 Replacement of Registers with Delay Elements

We propose a new algorithm that reduces the area of a pipelined circuit by the replacement of an intermediate register with delay elements whose area is smaller than the register.

In the proposed algorithm, all intermediate registers of the pipelined circuit are initially removed. Then the obtained circuit is checked whether it works correctly at T_{comp} by the algorithm shown in [4]. When an intermediate register is removed, the obtained circuit may not work correctly at T_{comp}. If the circuit does not work correctly at T_{comp}, a negative cycle is found in the constraint graph and it contains a multi D-edge. In order to eliminate the found negative cycle, it is necessary to increase the weight of the found negative cycle. The weight of the found negative cycle can be increased by increasing the weight of multi D-edge. The weight of multi D-edge can be increased either by recovering back the intermediate register or inserting some delay elements such as buffers. In [3], the found negative cycle is eliminated by recovering back an intermediate register only. Our algorithm repeatedly recovers back an intermediate register or inserts buffer until the circuit works correctly at given target clock-period range. For each iteration, the algorithm chooses the operation that has small area cost.

In our algorithm, the replacement of intermediate register with buffers is implemented by inserting the buffers to the location where the removed intermediate register is located. We choose only one type of buffer. The limit of the number of buffers that can be inserted is set at a certain number, so that the area of the total inserted buffers is smaller than the area of a register. Let l be the limit of the number of buffers that can be inserted to each location. If the area of a register is m times larger than the area of a buffer, then $l = m - 1$. When a negative cycle C is found, if C consists of one multi D-edge and one multi Z-edge corresponding to the same multi-clock cycle path, an intermediate register is inserted. Otherwise, based on the minimum delay of the buffer and $w^d(C)$, the number of buffers needed to be inserted is computed. Let n be the number of buffers needed to be inserted. If $n > l$, an intermediate register is inserted. While, if $n \leq l$, n buffers are inserted.

If there is more than one multi-D-edge in the negative cycle, our algorithm chooses a multi-D-edge corresponding to a multi-clock cycle path in which the number of allowed clock cycles to complete the data transmission is large since its weight tends to be small and the probability that it is contained in many negative cycles is high. If the chosen multi-D-edge corresponds to a path with three clock cycles or more, then more than one register was removed from the corresponding path. In such cases, the corresponding path has more than one candidate location to recover back a reg-

Inputs : Constraint graph G^{in} of a pipelined circuit with intermediate registers, the minimum clock period T_{comp} of the original circuit at zero-clock skew framework and clock-period range δ.

Outputs : Constraint graph G^{out} of the obtained circuit, clock timing $s(u)$ and minimum clock period $T_{\min}(G^{out})$.

Step 0 : Remove all of the intermediate registers. Let G_δ be the constraint graph of the obtained circuit.

Step 1 : Check whether there are any negative cycles in the constraint graph $G_\delta(T_{comp})$ or not by the algorithm shown in [4]. If a negative cycle C is found, then based on the area cost, insert an intermediate register or buffers to the multi-clock cycle path which corresponds to a multi D-edge contained in C, and update G_δ. Repeat this step until there are no negative cycles in the constraint graph $G_\delta(T_{comp})$

Step 2 : Let G^{out} be the constraint graph of the obtained circuit and output G^{out}, $T_{\min}(G^{out})$ and the clock timing for all registers.

Fig. 1 An algorithm to reduce the area of pipelined circuits by the replacement of registers with delay elements.

ister or to insert buffers. If a register is recovered back to the middle location among candidates, then timing constraints will be relaxed much since the number of multi-D-edges corresponding to a multi-clock cycle path in which the number of allowed clock cycles to complete the data transmission is large and the number of multi-D-edges are expected to be reduced much. Therefore, our algorithm recovers back registers or inserts buffers to the middle location among candidates. That is, if the path has n locations then $\lceil \frac{n}{2} \rceil$-th location is selected.

Our algorithm is heuristic. A different circuit might be obtained depending on the found negative cycle. Furthermore, if there are more than one multi D-edge with the same clock cycle path in the negative cycle, our algorithm chooses one of them randomly. Thus, a different circuit might be obtained depending on the chosen multi D-edge.

The details of our proposed algorithm is shown in Fig. 1. In Fig. 1, G^{in} is the constraint graph corresponding to the pipelined circuit with all intermediate registers, while G^{out} is the constraint graph corresponding to the circuit obtained by the algorithm. Note that, in Step 1 of our proposed algorithm, if there is no more multi D-edge, the algorithm is stopped and outputs the input constraint graph.

4.3 Example

To explain the behavior of the algorithm, we apply the algorithm to the pipelined circuit shown in Fig. 2. In this example, our target clock-period range δ 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 limit number of buffers that can be inserted is 3. For the original circuit with zero clock-skew, the minimum feasible clock period T_{comp} is 12. The circuit after removing the intermediate registers $v1$, $v2$, $v3$ is shown in Fig. 3. G_3^0 is the constraint graph of the obtained circuit.

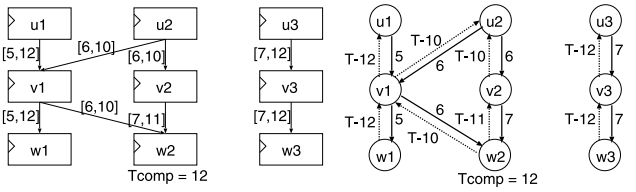

Fig. 2 Pipelined circuit with intermediate registers and the corresponding constraint graph G^{in}. $T_{comp} = 12$.

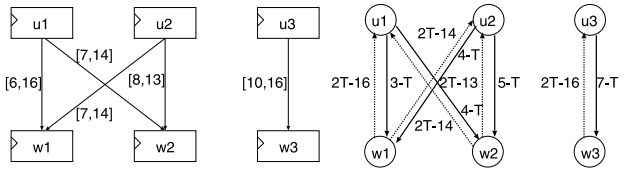

Fig. 3 Pipelined circuit after removing all intermediate registers and the corresponding constraint graph G_3^0. Found negative cycle $C^0 = (u1, w1, u1)$ in $G_3^0(12)$.

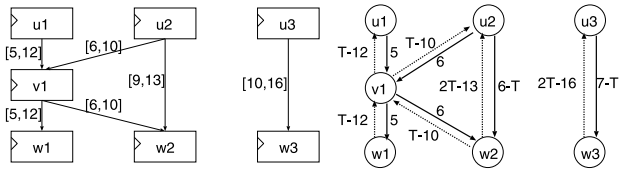

Fig. 4 Pipelined circuit after inserting the intermediate register $v1$ and the corresponding constraint graph G_3^1. Found negative cycle $C^1 = (u2, w2, v1, u2)$ in $G_3^1(12)$.

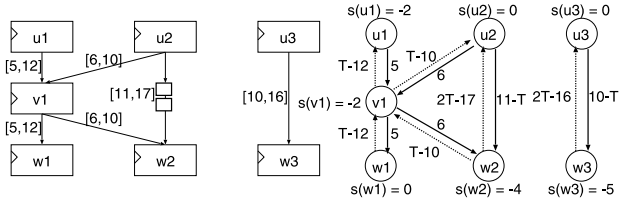

Fig. 5 Pipelined circuit after inserting the intermediate register $v1$ and 2 buffers at $v2$, and the corresponding constraint graph G_0^2. $T_{\min}(G_0^2) = 12$.

Negative cycle $C^0 = (u1, w1, u1)$ where $w(C^0) = -1$ is found in the constraint graph $G_3^0(12)$. Since, the multi D-edge $(u1, w1)$ and the multi Z-edge $(w1, u1)$ correspond to the multi-clock cycle path $(u1, v1, w1)$, an intermediate register is inserted to point $v1$. The circuit after inserting the intermediate register to point $v1$ is shown in Fig. 4. G_3^1 is the constraint graph of the obtained circuit. Negative cycle $C^1 = (u2, w2, v1, u2)$ where $w(C^1) = -2$ is found in the constraint graph $G_3^1(12)$. Since $w^d(C^1) = 2$, the number of buffers that has to be inserted is 2, which is smaller than the limit number of the buffers that can be inserted. Therefore, 2 buffers are inserted to point $v2$ between multi-clock cycle path $(u2, v2, w2)$ which corresponds to multi D-edge $(u2, w2)$ in C^1. The circuit after inserting the buffers is shown in Fig. 5. G_3^2 is the constraint graph of the obtained circuit. There are no negative cycles in the constraint graph

Table 1 Statistics of multiplier.

		Circuit delay [ps]					
circuit	Total	1st stage		2nd stage		3rd stage	
	#FF	min	max	min	max	min	max
2-stages 6bit_mul	42	884	3691	368	3488	-	-
2-stages 16bit_mul	120	757	5075	373	4050	-	-
2-stages 32bit_mul	245	1543	8533	408	5118	-	-
3-stages 32bit_mul	433	1543	6806	408	2881	408	5118

$G_3^2(12)$, so the algorithm stops and outputs the circuit and clock timings as shown in Fig. 5. Note that in Fig. 5, the clock timings are computed based on the constraint graph G_0^2. The output circuit works correctly between 12 to 15, while the original circuit works correctly between 12 to ∞.

5. Experiments

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

- 2-stages 6bit_mul: A 2-stages 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.
- 2-stages 16bit_mul: A 2-stages 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.
- 2-stages 32bit_mul: A 2-stages multiplier that multiplies two 32-bit numbers. The first stage uses a carry-save adder with Wallace tree structure [8] and the second stage uses a brent-kung adder [9].
- 3-stages 32bit_mul: A 3-stages multiplier that multiplies two 32-bit numbers. The first and second stages use a carry-save adder with Wallace tree structure [8] and the third stage uses a brent-kung adder [9].

The statistics of the circuits are shown in Table 1. The ROHM 0.35 μm 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 limit number of the buffer that can be inserted at a location where the removed intermediate register was located is 3. The timing of each I/O pin was scheduled as well as the timing for each register.

Table 2 shows the results when the algorithm shown in section 4.2 was applied. Ori. is the original circuit containing the intermediate registers and the clock timing of all registers are fixed at 0 (zero clock-skew). "δ[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.

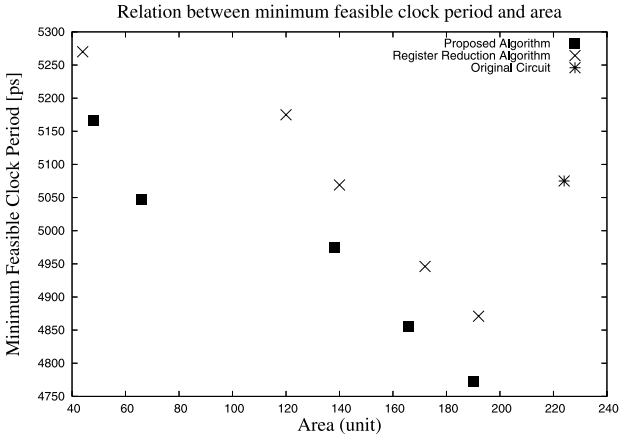

Fig. 6 Relation between $T_{\min}$ [ps] and Area (unit) of the 2-stages 16 bit multiplier. $\delta = 400$ [ps].

From the results shown in Table 2, 19 out of 24 types of experimental conditions, the area of the circuits obtained by our proposed algorithm is reduced compared with the area of the circuits obtained by register reduction algorithm in [3]. It shows the effectiveness of delay balancing on reducing the area of pipelined circuits.

The relation between the minimum feasible clock period and the total area of the buffers and intermediate registers of the 2-stages 16 bit multiplier is shown in Fig. 6. In the graph, the label "Proposed Algorithm" indicates results using the proposed algorithm for insertion of the intermediate registers and buffers, while "Register Reduction Algorithm" label indicates results when the algorithm in [3] is applied. The experiments were conducted on a few target of clock periods with clock period range 400 [ps].

The graph shows that there is trade off between the minimum feasible clock period and the size of the circuit obtained by both algorithms. However, with almost the same minimum feasible clock period, our proposed algorithm obtains a circuit with smaller area.

6. Conclusion

It has been shown that the area of the circuit obtained by our proposed algorithm is smaller than the area of the circuit obtained by the algorithm shown in [3] in most cases. The size of the clock tree will be reduced when the number of intermediate registers is reduced.

For the future work, a clock tree synthesis that can realize our target clock schedule based on the clustering based algorithm proposed in [10] is urgent.

The proposed algorithm inserts delay elements only at

Table 2 Experimental results.

		Proposed Algorithm									Register Reduction Algorithm in [3]								
				1st Stage		2nd Stage		Total					1st Stage		2nd Stage		Total		
Circuit	δ	T_{min}		Buff.	FF	Buff.	FF	Buff. + FF		Time	T_{min}		Buff.	FF	Buff.	FF	Buff. + FF		Time
	[ps]	[ps]	(%)	#	#	#	#	unit	%	[s]	[ps]	(%)	#	#	#	#	unit	(%)	[s]
2-stages	Ori.	3691	(100)	0	18	-	-	72	(100)	-	3691	(100)	0	18	-	-	72	(100)	-
6bit_mul	0	3634	(98)	1	2	-	-	9	(13)	0.01	3557	(96)	0	3			12	(17)	0.01
	200	3615	(98)	1	3	-	-	13	(18)	0.02	3426	(93)	0	4			16	(22)	0.03
	400	3673	(100)	2	3	-	-	14	(19)	0.02	3626	(98)	0	4			16	(22)	0.02
	600	3684	(100)	2	4	-	-	18	(25)	0.03	3450	(93)	0	6			24	(33)	0.03
	800	3650	(99)	5	4	-	-	21	(29)	0.03	3650	(99)	0	6			24	(33)	0.03
	1000	3677	(100)	3	6	-	-	27	(38)	0.04	2812	(76)	0	13			52	(72)	0.05
2-stages	Ori.	5075	(100)	0	56	-	-	224	(100)	-	5075	(100)	0	56	-	-	224	(100)	-
16bit_mul	0	5062	(100)	11	5	-	-	31	(14)	0.89	4870	(96)	0	11	-	-	44	(20)	0.69
	200	5070	(100)	0	11	-	-	44	(20)	1.68	5070	(100)	0	11	-	-	44	(20)	0.69
	400	5047	(99)	10	14	-	-	66	(29)	2.40	5069	(100)	0	35	-	-	140	(63)	1.93
	600	5074	(100)	16	29	-	-	132	(59)	3.48	5071	(100)	0	48	-	-	192	(86)	2.46
	800	3336	(66)	0	49	-	-	196	(88)	5.36	3336	(66)	0	49	-	-	196	(88)	2.58
	1000	3536	(70)	0	49	-	-	196	(88)	4.59	3536	(70)	0	49	-	-	196	(88)	2.54
2-stages	Ori.	8533	(100)	0	117	-	-	468	(100)	-	8533	(100)	0	117	-	-	468	(100)	-
32bit_mul	0	8484	(99)	3	1	-	-	7	(02)	2.36	8441	(99)	0	3	-	-	12	(03)	2.43
	400	8488	(99)	1	5	-	-	21	(05)	8.77	8488	(99)	0	6	-	-	24	(05)	3.94
	800	8506	(100)	12	7	-	-	40	(09)	13.36	8277	(97)	0	16	-	-	64	(14)	8.94
	1200	8524	(100)	10	15	-	-	70	(15)	24.56	8524	(100)	0	23	-	-	92	(20)	12.83
	1600	8517	(100)	10	25	-	-	110	(24)	39.55	8514	(100)	0	32	-	-	128	(27)	17.74
	2000	8506	(100)	6	38	-	-	158	(34)	57.58	8495	(100)	0	44	-	-	176	(38)	24.66
3-stages	Ori.	6806	(100)	0	188	0	117	1220	(100)	-	6806	(100)	0	188	0	117	1220	(100)	-
32bit_mul	0	6783	(99)	10	35	0	0	150	(12)	335.23	6783	(99)	0	44	0	0	176	(14)	136.53
	200	6801	(100)	11	50	0	0	211	(17)	470.96	6801	(100)	0	65	0	0	260	(21)	204.52
	400	6805	(100)	16	74	0	0	312	(26)	644.55	6795	(100)	0	88	0	0	352	(29)	277.38
	600	6801	(100)	10	173	0	0	702	(58)	1583.14	5314	(78)	0	182	0	0	728	(60)	571.70
	800	6722	(99)	6	180	0	0	726	(59)	1196.32	5414	(80)	0	182	0	0	728	(60)	586.94
	1000	5514	(81)	0	182	0	0	728	(60)	1130.00	5514	(80)	0	182	0	0	728	(60)	591.55

the location where the removed intermediate register is located. We believe that if delay elements can be inserted at any location, the circuit area can be further reduced. This is also 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] B.A. Rosdi and A. Takahashi, "Replacement of register with delay element for reducing the area of pipelined circuits," Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS), pp.802–805, 2006.
[2] W.J. Kim and Y. Kim, "Clocking for correct functionality on wave pipelined circuits," Proc. IEEE International ASIC/SOC Conference, pp.161–164, 2003.
[3] 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.
[4] A. Takahashi, "Practical fast clock-schedule design algorithms," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.1005–1011, April 2006.

[5] J.P. Fishburn, "Clock skew optimization," IEEE Trans. Comput., 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," Proc. Workshop on Synthesis and System Integration of Mixed Technologies (SASIMI 2004), pp.333–338, 2004.
[7] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp.37–42, 1997.
[8] C. Wallace, "A suggestion for fast multiplier," IEEE Trans. Electronic Computers, vol.13, no.2, pp.14–17, 1964.
[9] R. Brent and H. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol.C-31, no.3, pp.260–264, 1982.
[10] M. Saitoh, M. Azuma, and A. Takahashi, "A clustering based fast clock schedule algorithm for light clock-trees," IEICE Trans. Fundamentals, vol.E85-A, no.12, pp.2756–2763, Dec. 2002.

Bakhtiar Affendi Rosdi received his B.E., and M.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1999 and 2004, respectively. He is currently a D.E. student of Department of Communications and Integrated Systems in Tokyo Institute of Technology. His research interests are in VLSI design automation and combinational algorithms.

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.