A Clock-Gating Method for Low-Power LSI Design

Takeshi Kitahara Fumihiro Minami Toshiaki Ueda Kimiyoshi Usami
Seiichi Nishio Masami Murakata Takashi Mitsuhashi

Semiconductor DA & Test Engineering Center
TOSHIBA Corporation
580-1, Horikawa-cho, Saiwai-ku,
KAWASAKI, 210, JAPAN
Tel: +81-44-548-2346
Fax: +81-44-548-8306
kitahara@dad.eec.toshiba.co.jp

Abstract - This paper describes an automated layout design technique for the gated-clock design. Two issues must be considered for gated-clock circuits to work correctly. One is to minimize the skew for gated-clock nets. The other is to keep timing constraints for enable-logic parts. We propose the layout design technique to taking these things into consideration. We developed Gated-Clock Tree Synthesizer for the first issue, and Timing Constraints Generator and Clock Delay Estimator for the second. We applied it to a practical gated-clock circuit. By our technique, the clock-skew could be less than 0.2ns keeping timing constraints for enable-logic parts.

I. Introduction

Recently, the market for portable electric appliances has grown rapidly, generating great interest in low-power design. Gated-clock design[1] is one of the most important techniques to reduce power dissipation. From our experience, 30-50% of power is dissipated on clock-lines in a logic chip. By the gated-clock technique, power dissipated on clock-lines including synchronous storage elements such as flip-flops and latches, can be saved.

A number of approaches have been proposed for the gated-clock design. Benini et al.[2] introduced the concept of “Moore-state” on Mealy FSM(Finite State Machine). They showed that clock supply can be stopped when a present state lies in a Moore-state. He proposed the algorithm of generating a locally-Moore machine and gated-clock structure on FSM. Wu et al.[3] used quaternary representation for behaviors of signals. They proposed the method of finding a gated clock signal instead of a normal clock signal using Karnaugh maps, by checking quaternary value of each flip-flop in a circuit. Téllez et al.[4] showed the algorithm for gated-clock tree construction using CDFG(Control-Data Flow Graph), that is a result of scheduling and allocation. A binary clock tree is arranged according to its activity pattern obtained from CDFG. These are the techniques of finding enable signals, and generating gated-clock structures using FSMs[2], extended Boolean functions[3], and CDFGs[4].

The techniques of finding enable signals are important for the gated-clock design, however, there is another important issue from a practical point of view. That is a timing assurance method to ensure that a gated-clock circuit works correctly. There have been few proposals for such a timing assured implementation method, and it is the very reason why the gated-clock design is not so commonly used for practical circuits, especially ASICs. For the gated-clock design, timing related to enable signals and the clock-skew must be treated carefully. If timing constraints for enable signals are violated, irregular clock-rising at a clock-input of some registers may occur. This causes unexpected data-loading at the registers in the gated-clock circuit, and the circuit works incorrectly. We propose an automated design method to ensure such unexpected data-loading does not occur. Two issues must be considered for the timing assurance. One is to minimize the clock-skew even when there are both normal clock buffers and gated clock buffers in clock nets. The other is to keep timing constraints for enable-logic parts after placement and route.

In this paper, we propose an automated layout design method for the gated-clock design, taking the issues mentioned above into consideration. For this purpose, we have developed EDA tools named Clock Delay Estimator(CDE), Gated-Clock Tree Synthesizer(Gated-CTS), and Timing Constraints Generator(TCG). We have constructed a layout flow for the gated-clock design including these tools. The remainder of the paper is organized as follows. Section II introduces gated-clock design style, and defines a problem for timing issues. Section III presents the EDA tools developed for the gated-clock design. Section IV proposes the layout flow using the EDA tools mentioned above. Section V presents an experimental result using a practical circuit.

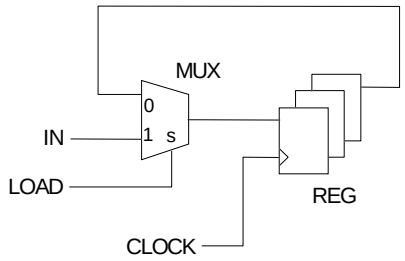
II. Preliminary

A. Gated-Clock Design

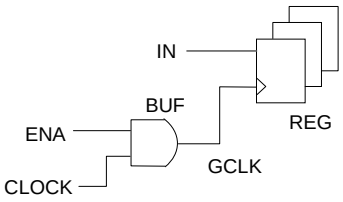
By the clock-gating technique, a non gated-clock circuit as illustrated in Fig.1(a) can be transformed into a gated-clock circuit shown in Fig.1(b), where *REG* is a register, and *MUX* is a multiplexer whose *s*-input is a control signal. In this case, the signal *ENA* must satisfy the following condition, where DC_{REG} is a don't-care condition of the register *REG*.

$$onset(LOAD - DC_{REG}) \subset onset(ENA) \subset onset(LOAD + DC_{REG})$$

Clock pulses are not supplied to the register, when the value of the signal *ENA* is equal to zero. Power dissipated in the register and some parts of clock-lines can be saved, when the clock pulses are not supplied. There are two reasons why it can be achieved. (1)By eliminating signal transitions at inputs of registers, dynamic power dissipated in the registers and clock nets can be saved. (2)If clock pulses are not supplied to registers when $DC_{REG} = 1$, the value of the registers don't change. So, signal transitions at transitive fanouts of the registers can be eliminated.



(a) Non Gated-Clock Circuit



(b) Gated-Clock Circuit

Fig.1 Example of Gated-Clock Circuit

Considering two circuits in Fig.1 from a functional(zero-delay) aspect, both of the registers have a same function. However, from a timing aspect timing constraints assigned to the signal *LOAD* and *ENA* are different. Actually, tighter timing constraints need to be assigned to the signal *ENA*. For example, signal transitions at the signal *ENA* must not occur when the signal *CLOCK* is equal to one in Fig.1(b). If such transitions occur, the transitions are propagated to the signal *GCLK*, and unexpected clock-rising or falling happens at the clock-input of the registers *REG*. Some measures should be taken to eliminate such transitions at the signal *ENA*.

Fig.2 shows one of the typical gated-clock design styles. We introduce terminology used in this paper. The "enable-logic" consists of combinational elements, and determines whether a clock pulse should be supplied to the registers or not. If the output of the enable-logic is equal to zero, a clock pulse is not propagated to the clock-inputs of the registers. An AND-gate or an OR-gate which satisfies following two conditions is called "gated buffer". (1)The gate is a (transitive) fanin of a clock-input of a register. (2)One input of the gate is an enable signal, and another is an original clock signal. On the contrary, a buffer without gating function is called "normal buffer". Glitches at the input of the gated buffer should be eliminated as mention above. The "de-glitch latch" is inserted between the enable-logic and the gated buffer to eliminate glitches which occur in the enable-logic. A clock signal which the gated-clock technique is applied, is called "gated clock signal"(say GCLK), and a one not "normal clock signal"(say NCLK).

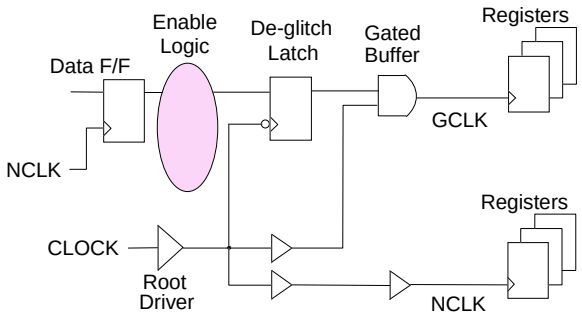


Fig.2 Example of Gated-Clock Design

B. Problem Definition

Here, we describe a sticky point for the timing assurance on the gated-clock design. In fact, timing issues for the enable signal *ENA* in Fig.1(b) are rather complex. Signal transitions at the enable signal *ENA* must not occur when the clock signal *CLOCK* is equal to one as mentioned above. So, the value of the enable signal *ENA* must be determined after the clock signal *CLOCK* falls, and before it rises. Clock-rising at the clock-input of the gated buffer *BUF* is a little earlier than that of the register *REG*. Let c be the delay from the gated buffer to the register, and T_s clock cycle. To be exact, when building timing constraints for the enable signal *ENA*, c must be taken into account. For example, let $\{x_i\}$ be a set of transitive fanins of the enable signal *ENA*, and assuming that each event of x_i occurs when the signal *CLOCK* rises. In this case, the maximum delay constraints for the

paths between each x_i and *ENA* must be less than T_s - c, if assuming that clock-skews of all registers are zero. It is difficult to determine the value c adequately. The reason for it is as follows. Gated-CTS inserts gated and normal buffers into both of the signals *CLOCK* and *GCLK*. The number of buffers on the signal *GCLK* cannot be found out till Gated-CTS is done. So, it is difficult to determine the adequate value c before the clock-route step. However, the timing constraints for the enable signal are needed to execute timing-driven placement, and the placement step is an earlier step than the clock-route. This dilemma is one problem for the gated-clock design. To overcome the problem, we introduced EDA tools named Clock Delay Estimator, and constructed the timing assurance method.

III. EDA Tools for Gated-Clock Design

As described before, there are two issues that should be considered for the gated-clock design. One is to minimize the clock-skew, and the other is to keep the timing constraints for the gated-clock parts. We developed Gated-Clock Tree Synthesizer for the first issue, and Timing Constraints Generator and Clock Delay Estimator for the second. In this section, we explain the features of these EDA tools.

A. Clock Delay Estimator(CDE)

First, Clock Delay Estimator determines the number of clock buffer stages between a root driver and each clock-input of flip-flops based on the number of flip-flops in a circuit. Next, CDE places only flip-flops randomly on a chip. After the placement of the flip-flops, CDE spans clock nets. The process of spanning them consists of the following three steps: clock planning, buffer cell placement, and global routing for clock trees. In the clock planning step, node-pairing is done to obtain the optimal node-pair combinations of the minimum delay and skew. The node-pairing is performed recursively in a bottom up manner, and a clock tree is constructed. In the buffer cell placement step, buffer cells are added to the clock tree, according to the number of clock buffer stages obtained in the beginning of CDE. The buffer cells are placed at ideal positions to minimize the skew. After the clock global routing step, CDE estimates delay of the clock nets. Then the following clock delay information can be obtained. Each piece of the clock delay information is shown in Fig.3.

(a)maximum and minimum delay from the root driver to the buffers in the first stage(*FIRSTSTAGE*).
(b)maximum and minimum delay from the root driver to the buffers in the last stage(*LASTSTAGE*).
(c)maximum and minimum delay from the root driver to the flip-flops(*FFIN*).

Strategy of generating balanced clock trees is similar to that of CTS[5]. The error between actual clock delay and delay obtained by CDE is within 15% from our experience.

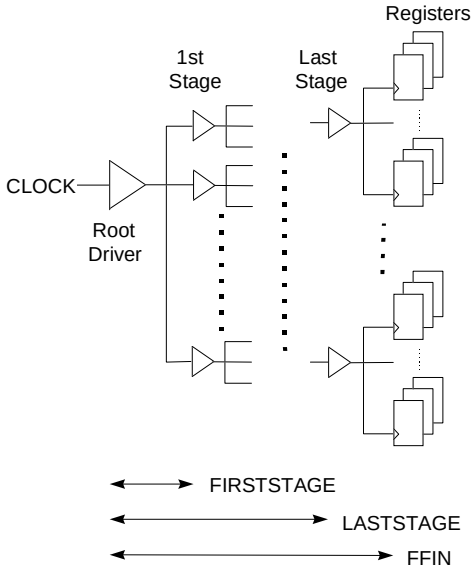


Fig.3 Clock Delay Information

B. Gated-Clock Tree Synthesizer(Gated-CTS)

Fig.4(a) shows an example of clock net structure before Gated-CTS. First, Gated-CTS divides flip-flops into groups. All flip-flops in one of the groups have a same enable signal on their clock-lines. Flip-flops driven by a normal clock signal constitute one group. In Fig.4(a), four groups are created. Next, it divides the groups into clusters. The clusters are generated so that the total capacitance in each cluster can be equalized. For the benefit of clustering, the delay in each cluster is balanced, and the skew minimized. In Fig.4, the flip-flop group whose enable signal is E1 is supposed to be divided into four clusters, and the group E2 two. The group E3 and the flip-flop group driven by the normal clock signal are supposed to be treated as clusters.

Tree construction and buffer insertion are performed, after clustering. Gated-CTS generates a clock tree by bottom up node-pairing stated in the explanation of CDE. Some gated buffers have been already added to the clock net as shown in Fig.4(a). Gated-CTS puts additional normal buffers and duplicates gated buffers in order to balance each clock path delay. Fig.4(b) shows the result of the tree construction and the buffer insertion for the example of Fig.4(a). The gated buffer controlled by the enable signal E1 is put at the first stage. On the contrary, the gated buffer controlled by the enable signal E2 is duplicated at the second stage. This gated buffer cannot be put at the first stage. In general, the buffers controlled by enable signals are placed as upstream as possible toward the root driver. There are two reasons for it.

One is to minimize the delay of enable signals, and the other is to reduce power dissipation of clock nets.

Finally, buffer cell placement and gated-clock tree routing is executed. Both gated and normal buffer cells are placed at ideal positions to minimize the skew. When some cells have already been placed at the positions, they are moved to actual positions as near ideal as possible. Then detailed routing is performed according to clock tree topology(Fig.4(c)). At this time, routing is performed in a bottom up manner and delay balanced points are modified one by one.

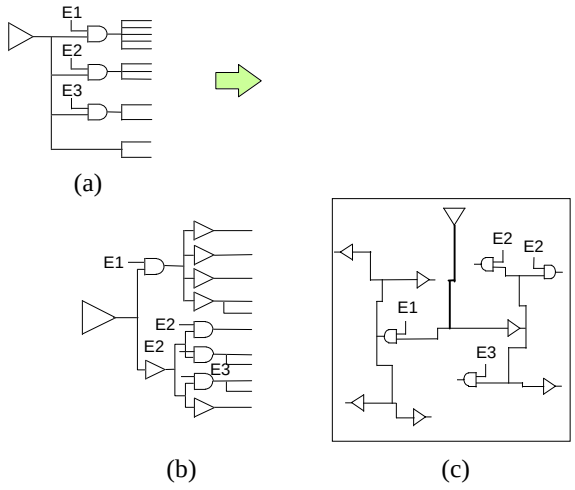


Fig.4 Gated-Clock Tree Synthesis

C. Timing Constraints Generator(TCG)

Timing Constraints Generator generates maximum and minimum delay constraints for the gated-clock parts. We explain delay constraints for the gated-clock design shown in Fig.2. First, maximum delay constraints are required for the enable-logic. Let *S* be the set of storage elements(say Data F/F) that are fanins of the enable-logic. If the de-glitch latch is assumed to be a flip-flop, the maximum delay value between each element in *S* and the de-glitch latch is calculated by the following expression.

$$HALF\_CYCLE - SKEW\_VALUE$$

HALF_CYCLE is half of the clock cycle T_s , and *SKEW_VALUE* can be calculated by the result of CDE using the following expression.

$$SKEW\_VALUE = MAX\_FFIN - MIN\_FFIN$$

Maximum delay constraints are also required for the AND gated buffer. In Fig.1(b), the value of the signal *ENA* must be determined before the signal *CLOCK* rises. Gated-CTS adds buffers into clock nets, and it cannot be decided in which stage gated buffers inserted before the Gated-CTS step. It is necessary to generate the delay constraints before the Gated-CTS step for timing-driven placement. So, we assume the worst case, and generate the delay constraints using the value of *FIRSTSTAGE* obtained by CDE. The maximum delay constraints between the de-glitch latch and the gated buffer are calculated by the following expression.

$$MAX\_DELAY = HALF\_CYCLE - SKEW\_VALUE - (MIN\_FFIN - MAX\_FIRSTSTAGE)$$

There is a case of using a transparent latch for the de-glitch instead of the flip-flop. Delay constraints for gated-clock parts can be also generated in the same way.

Here, we take up a subject of minimum delay constraints. In Fig.5, the hold-time assurance for the OR gated buffer must be considered, because the output of the data F/F is connected to the gated buffer directly. In this case, we also assume the worst case, and generate the minimum delay constraints using the value of *LASTSTAGE*. The minimum delay constraints between the data F/F and the gated buffer are calculated by the following expression.

$$MIN\_DELAY = SKEW\_VALUE - (MIN\_FFIN - MAX\_LASTSTAGE)$$

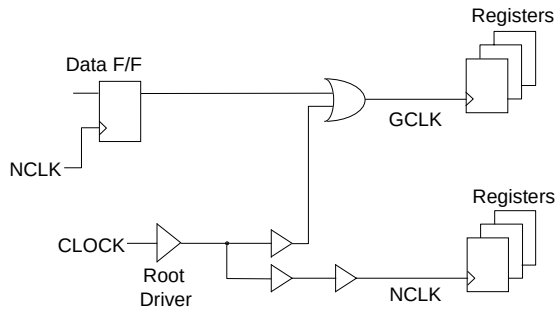


Fig.5 Example of Gated-Clock Design

IV. Layout Flow for Gated-Clock Design

We introduce a layout EDA flow using the tools described in the previous section with a placer and a router. Fig.6 shows our layout flow for the gated-clock design. First, CDE estimates clock delay information based on a netlist and die size. The gated-clock parts are supposed to be incorporated into the netlist. CDE creates a report related to the clock delay before placement. The factors, *FIRSTSTAGE*, *LASTSTAGE*, and *FFIN* shown in Fig.3, are calculated here. TCG generates timing constraints for the gated-clock parts using these factors in the CDE report. Timing-Driven Placer[6] and Post-Placement Netlist Optimizer(PNO)[7] assures timing constraints. Timing-Driven Placer does placement with optimizing timing of a gated-clock circuit

based on timing constraints generated by TCG and those of datapaths. When there exists timing violation after the placement, PNO inserts buffers and changes cell models called "repower" to assure the timing constraints. Next, Gated-CTS routes clock nets and minimizes the clock-skew. The PNO step after Gated-CTS is also needed. Gated-CTS duplicates gated buffers as shown in Fig.4. So, the number of fanouts of enable signals may increase. This may causes timing violation for the gated-clock parts. PNO can repair this timing violation. Finally, timing-driven routing[8] for ordinary nets is executed.

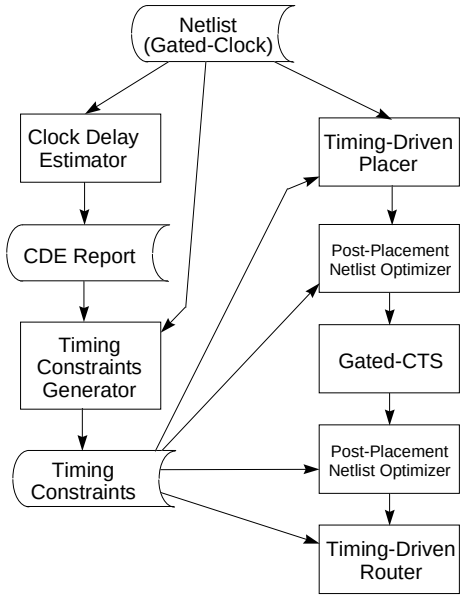


Fig.6 Layout Flow for Gated-Clock Design

V. Experimental Result

We tested our flow on the practical logic circuit with gated-clock structure. There are 54 thousand cells and 58 thousand nets in the circuit. The number of flip-flops is 55 hundreds. The layout result is shown in Fig.7. There are three clocks in this circuit, and each clock is gated. The nets constructed by solid lines in Fig.7 shows one clock net(CLK3). The clock delay and skew information is shown in Table I. In Table I, “#FFs” shows the number of flip-flops that are driven by each clock. “Gating Ratio” is a percentage of clock-gated flip-flops. For example, about 1050 flip-flops are driven by gated clock signals based on CLK1, and about 1450 are driven by the normal clock signal CLK1. “#Enables” is the number of enable signals in the netlist. Each enable signal has a different enable-logic. Gated-CTS could achieve a small clock-skew value for each clock. Actually, the skew value is less than 0.2ns and it is comparable to that of non gated-clock circuit. Power dissipation of the gated-clock circuit and the non gated-clock circuit is analyzed by Pro-Power[9,10]. ProPower is a power estimator employing probabilistic approach to calculate signal probabilities and switching activities. The result of the power analysis is shown in Table II. The modules from A to E have gated-clock structure. In Table II, “Power(Non-Gated)” shows the power dissipation of the circuit without gated-clock structure, and “Power(Gated)” shows the one with the gated-clock structure. Large power reduction could be achieved for these modules. “Total” means the whole chip including mega-cells and I/Os. So, sum of 5 modules is not equal to “Total”. About 30% power reduction could be achieved for the whole design by clock-gating.

Table I Clock Delay and Skew

Clock	#FFs	Gating Ratio	#Enables	Clock Delay	Clock Skew
CLK1	2504	42%	18	2.46ns	0.18ns
CLK2	528	97%	16	1.79ns	0.09ns
CLK3	2532	33%	2	2.27ns	0.13ns

Table II Power Dissipation

(mW)

Module	Power (Non-Gated)	Power (Gated)	Power Reduction
A	56.48	19.69	65.1%
B	237.42	144.88	39.0%
C	45.71	11.22	75.5%
D	52.20	16.51	68.4%
E	31.86	10.25	67.8%
Total	1193.54	828.01	30.6%

VI Conclusion

We have presented the design method including the timing-driven layout for the gated-clock design. We have developed three tools named CDE, Gated-CTS and TCG, to assure gated-clock circuits work correctly. Gated-CTS minimizes the clock-skew for gated-clock nets. CDE and TCG keep timing constraints for gated-clock parts. We have also proposed the layout flow including these tools. By the experiment, we could achieve 30% power reduction for the practical circuit. The clock-skew could be less than 0.2ns, and all of the timing constraints for the enable-logic parts are kept.

Acknowledgment

The authors would like to thank T.Matoba, Y.Maki, A.Fujimoto and H.Muraoka for providing the practical design data used for the experiment. We would also like to thank T.Aoki and T.Uchino for helping to execute Post-Placement Netlist Optimizer and ProPower, respectively.

Reference

[1]J.M.Rabaey and M.Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996.

[2]L.Benini, G. De Micheli, E.Macii, M.Poncino and R.Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of control-oriented synchronous networks", *Proceedings of European Design and Test Conference*, pp.514-520, 1997.

[3]Q.Wu, M.Pedram and X.Wu, "Clock-gating and its application to low power design of sequential circuits", *Proceedings of Custom Integrated Circuits Conference*, pp.479-482, 1997.

[4]G.E.Tellez, A.Farrahi and M.Sarrafzadeh, "Activity-driven clock design for low power circuits", *Proceedings of International Conference on Computer-Aided Design*, pp.62-65, 1995.

[5]M.Takano, F.Minami and N.Kojima, "Delay and skew minimized clock tree synthesis for embedded arrays", *IEICE Transaction on Information and Systems*, Vol.E79-D, No.10, October, pp.1405-1409, 1996.

[6]M.Murakata, M.Murofushi, M.Igarashi, T.Aoki, T.Ishioka, T.Mitsuhashi and N.Goto, "Concurrent logic and layout design system for high performance LSIs", *Proceedings of Custom Integrated Circuits Conference*, pp.465-468, 1995.

[7]T.Aoki, M.Murakata, T.Mitsuhashi and N.Goto, "Fanout-tree restruturing algorithm for post-placement timing optimization", *Proceedings of Asia and South Pacific Design Automation Conference*, pp.417-422, 1995.

[8]H.Tanaka, M.Igarashi, T.Akiyama and M.Murakata, "A timing-driven global routing method for deep submicron VLSI", *Proceedings of DA Symposium*, pp.263-268,1996 (in Japanese).

[9]T.Uchino, F.Minami, T.Mitsuhashi and N.Goto, "Switching activity analysis using Boolean approximation method", *Proceedings of International Conference on Computer-Aided Design*, pp.20-25, 1995.

[10]T.Uchino, F.Minami, M.Murakata and T.Mitsuhashi, "Switching activity analysis for sequential circuits using Boolean approximation method", *Proceedings of International Symposium on Low Power Electronics and Design*, pp.79-84, 1996.

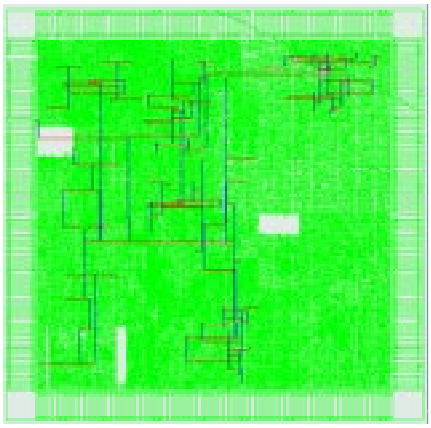


Fig.7 Layout Result