Design Methodology and Tools for NEC Electronics' Structured ASIC ISSP

Takumi Okamoto
NEC Corporation
1753 Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8666, Japan
okamoto@ct.jp.nec.com

Tsutomu Kimoto Naotaka Maeda
NEC Electronics Corporation
1753 Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8666, Japan
{tsutomu.kimoto, n.maeda}@necel.com

ABSTRACT
In this paper, we describe a design methodology and tools for NEC Electronics' structured ASIC, *Instant Silicon Solution Platform (ISSP)*, which is being developed to fill the gap between FPGAs and standard cell-based ASICs. The ISSP has a unique regular-fabric architecture designed to achieve both a short time to production and high-performance LSI design. We have developed a special design methodology and tools to fully exploit the capability of this new device. Experimental results for industrial data show that our approach has advantages for ISSP design.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aides

General Terms
Design, Performance, Algorithm

Keywords
Structured ASIC, Regular fabric, ISSP, Placement.

1. INTRODUCTION
While the design productivity gap resulting from the latest advances in deep-submicron process technology has been much discussed, another gap has received much less attention until recently – the gap between standard cell-based ASICs and FPGAs. This gap presents a problem, particularly for mid-volume designs, which often require ASIC performance and density that meet product specifications and cost goals but may also need FPGA time to production to achieve market goals. Bridging this gap requires not only a new ASIC architecture, but also optimized design methodology [1][2].

Design times for complex ASICs are increasing due to the need to insert and verify designs for test (DFT) circuitry, and address deep sub-micron design issues such as signal integrity. Even in the FPGA world, it is not unusual to spend weeks on converging timing, especially for highly utilized, large gate-count designs with clock speeds exceeding 100 MHz. The rising costs of development and non-recurring engineering (NRE) of the masks needed to design deep sub-micron ASICs make it difficult for designers of systems with mid-range production volumes to target multiple respins of their prototypes. For early-stage products, there are also risks in trying to predict the production volumes needed to amortize these development costs. A large percentage of today's designs are mid-volume, i.e., between several hundreds and 100,000 units are produced. In this range, designers must carefully balance performance, time to market, and total cost.

Recently, several structured ASICs or regular fabrics have been proposed by industries [3][4][5] and university researchers [6][7][8][9][10] to cope with this situation. Typically, these devices consist of an array of logic cells built using diffusion and a few metal layers. Each of the logic cells contains customizable combinatorial logic and may also contain register elements. The mask layers required to build the logic-cell array are common to all customer designs. The logic cells are then customized and interconnected using a few metal layers on top that are generated using custom masks for a given design. This approach minimizes the number of custom masks needed to build an ASIC, thereby reducing the upfront NRE costs, while preserving the advantages of low unit costs, low power consumption, and the high performance of a standard cell-based ASIC.

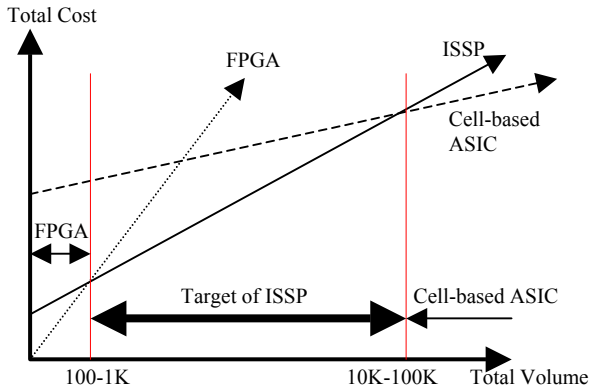
In terms of the design methodology and tools, many of the deep submicron issues can be directly resolved in the construction of the regular fabric with built-in clocks, scan chains, power distribution, and so on. In addition, testing is greatly simplified. In certain design phases, however, to extract the maximum performance from a device requires design methodology and tools specially developed for the specific device. Regular fabrics impose some constraints on design, which puts stress on different aspects of the design process. In some cases, for example, mapping and placement is becoming harder while routing may be getting easier.

In this paper, we describe a design methodology and tools for NEC Electronics' structured ASIC, *Instant Silicon Solution Platform (ISSP)*, which has been developed to fill the gap between FPGAs and standard cell-based ASICs [3]. ISSP has a unique regular-fabric architecture aimed at achieving both a short time to production and high-performance LSI design. We have developed a special design methodology and tools to fully exploit this new device capability. Experimental results for industrial data show that our approach has advantages for ISSP design.

ISPD'04, April 18-21, 2004, Phoenix, Arizona, USA.

2. INSTANT SILICON SOLUTION PLATFORM (ISSP)

The Instant Silicon Solution Platform (ISSP) is a new class of device based on a cost-effective, high-function, easy-to-design ASIC architecture ideal for designs requiring high-speed system clocks and mid-range production volumes [3]. Figure 1 shows the relationship between total volume and total cost (engineering cost + mask NRE + total volume × unit cost). The ISSP is targeted at the area between FPGAs and cell-based ASICs, i.e., mid-volume between several hundreds and 100K units.



Total cost = Engineering cost + Mask NRE + Total volume * Unit

Figure 1: Total volume vs. total cost.

Figure 2 shows the architecture of a device in the first-generation ISSP1-STD family (CMOS 0.15 µm technology)[1], which has the following functions embedded in a common mask (implemented using three metal-layers):

- Clock structure
- SRAM
- Analog phase-locked loops (APLLs)
- Delay-locked loops (DLLs).
- Complex multi-gate (CMG: multiplexer and flip-flop) array

User-logic is implemented by two individual metal layers on top that are generated using custom masks for a given design.

Users designing with ISSP need not worry about the impact that adding DFT might have on design time because ISSP is preconfigured with testing methodologies such as SCAN, BSCAN, and others. Built-in clock domains also conserve design time by maintaining predictable timing levels and lowering clock skew.

By providing a platform array-based architecture, an easy design flow, and NRE costs comparable to those of a gate array, the ISSP permits fast respins of prototypes and enables products to reach mid-range production volume levels at very reasonable costs. If the volumes increase, ISSP designers can migrate to cell-based products based on the same process technology, IP, and design flow.

The ISSP1-STD family comprises a number of versatile devices targeted for use in high-performance broadband communication and networking equipment, computer peripherals, instrumentation devices, and a wide range of other applications.

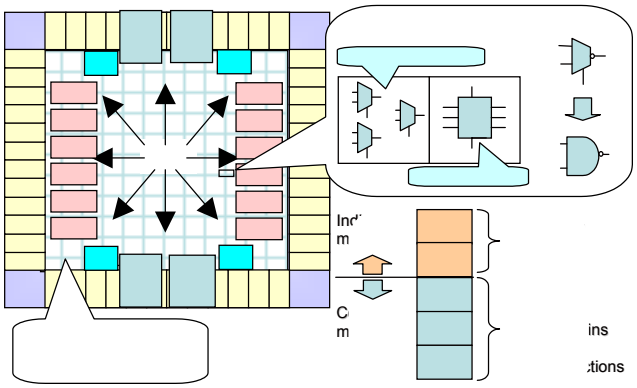


Figure 2: Architecture of the ISSP.

3. PHYSICAL DESIGN METHODOLOGY AND TOOLS FOR ISSP

In this section, we describe an algorithm used in our physical design tools which was developed to fully exploit ISSP device capability. Here, we focus on two key issues in the physical design: optimization of embedded clocks and interconnect delays.

3.1 Embedded Clock Structure

Figure 3 shows an example of the embedded clock structure used by the ISSP to design low-skew clocks in a short TAT[2]. The entire chip is divided into four regions (denoted by *first-level region*) and 16 regions (denoted by *second-level region*) in a hierarchical, two-level tree structure: each first-level region has four children (second-level regions) and each second-level region has one parent (first-level region). This structure imposes the following design constraints:

Embedded clock constraints:

- The chip has *two main and eight local embedded clock structures.* The clocks can also be implemented by using individual masks (user-customizable layer) without using an embedded clock structure, which is called a *non-embedded clock structure*. Each clock in a circuit implemented on the ISSP is assigned to one of three structures: main, local, or non-embedded. The clocks assigned to these structures are called the *main clock, local clock,* and *non-embedded clock*, respectively.
- Flip-flops (F/F or register) of the main clock can be placed in any region.

[1] The ISSP family includes several types of architecture. In this paper, the architecture in Figure 2 is used as an example.

[2] The ISSP family has several types of embedded clock structure. In this paper, the structure in Figure 3 is used as an example.

- Each first-level region can take four local clocks. F/Fs of more than four local clocks cannot be placed in one first-level region.
- Each second-level region can take two local clocks, which must be selected from the local clocks assigned to its parent region. F/Fs of more than two local clocks cannot be placed in one second-level region, and the local clock must be selected from the local clocks assigned to its parent first-level region. The number of clocks that one region can take is called the *clock capacity* of the region. Also note that one clock can be assigned to more than one first- or second-level region.
- F/Fs must be aligned on to the embedded F/F slots. Therefore, there is an upper bound on the number of F/Fs each first- or second- level region can take, which is called the *F/F capacity* of the region.

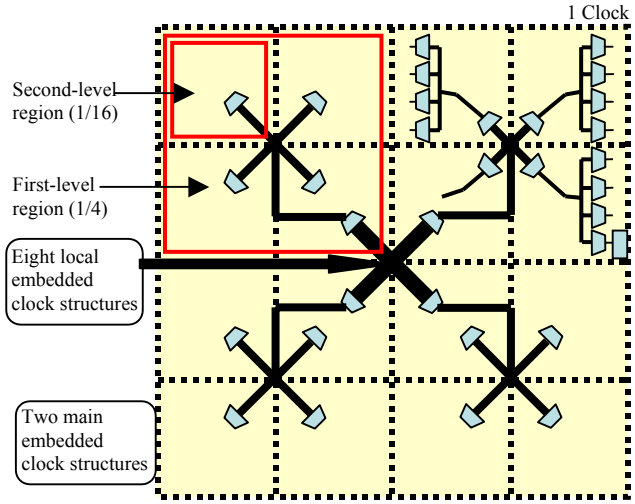


Figure 3: Embedded clock structure.

3.2 Embedded Clock-Aware Placement

Embedded clock aware-placement for the ISSP is designed to solve the placement problem under the embedded clock constraints described in 3.1. This includes:

***Main/local clock assignment*:** Determine which clock structure (main, local, or non-embedded) is assigned to each clock in the circuit[3].

***Regional clock assignment*:** Determine the local clocks that each of the first- and second-level regions take.

***Cell placement*:** Place cells including F/Fs under the constraints of regional clock assignment determined before or during the placement.

A simple approach to this problem is shown in Figure 4.

[3] In some cases, this assignment is specified by the circuit designer.

Method 1: "Clock assignment first" approach

Step 1. Perform main/local clock assignment and regional clock assignment based on the number of F/F, IO positions, and RAM positions.

Step 2. Carry out placement under the regional clock assignment generated in Step 1.

Figure 4: Simple embedded clock-aware placement.

However, this approach does not work well because it is difficult to predict good regional clock assignment without carrying out the actual placement. Regional clock assignment without considering placement sometimes results in infeasible constraints on placement. It is therefore necessary to develop a tool to generate a placement-friendly form of regional clock assignment. One possible way of doing this is to analyze and perform regional clock assignment based on actual placement results. Figure 5 shows an algorithm based on this idea.

Method 2: Placement-based embedded clock optimization

Step 1. Carry out global placement without considering embedded clock constraints to get an initial solution.

Step 2. Perform main/local clock assignment and regional clock assignment based on the results of the initial placement in Step 1.

Step 3. Carry out incremental placement based on the regional clock assignment generated in Step 2 by moving F/Fs that are in violation of the constraints into legal regions.

Step 4. Align the F/Fs on to embedded F/F slots.

Figure 5: Placement-based embedded clock optimization.

In Step 1, global placement is performed to get an initial solution; "global" means that the placements made are not necessarily legalized. This initial, unconstrained placement gives us information on how the clock domains are distributed on the chip, which is a good starting point for legal assignment of regional clocks.

Figure 6 shows an algorithm used in Step 2 to assign the main/local clocks and regional clocks. Hereafter, "*violation*" means a violation of embedded clock constraints.

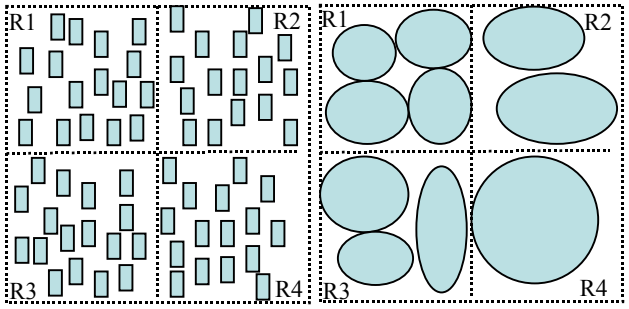
Procedure for Regional Clock Assignment

```
1.  For (each second-level region) {
2.    F/Fs of the same clock are formed into one cluster;
3.  }
4.  For (all combinations of main/clock assignment) {
5.    All clusters are set to be movable;
6.    While (violations or unplaced clusters exist) {
7.      Rip-up movable clusters until all violations are
          eliminated one by one in a small-cluster-first manner;
8.      For (each ripped-up cluster) {
9.        Move the cluster to a region that
            does not cause violations;
```

10. **If** (a legal region to move it to cannot be found) {
11. Move the cluster to the original region and set it to be fixed;
12. }
13. }
14. **If** (all clusters are fixed) break;
15. }
16. }
17. Select the best combination of main/local clock assignment;
18. Assign local clocks to regions that are not saturated for clock assignment;

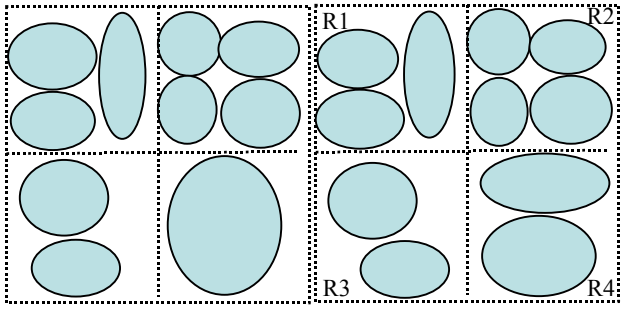
Figure 6: Regional clock assignment.

The algorithm is based on a simple heuristic approach: an iteration of ripping-up and replacing clusters causing violations.Figure 7 shows an example of *regional clock assignment*, where each rectangle represents an F/F numbered according to the clock that it belongs to and, for simplicity, the local clock capacity of each region is assumed to be 2. In Figure 7(a), the region R1 and R3 has violations that F/Fs of 4 and 3 local clocks are placed in one region. Figure 7(b) shows the result of lines 1-3: F/F clustering based on the initial placement in Figure 7(a). Figure 7 (c) shows the result of ripping-up and replacing violating clusters. In Figure 7 (d), a local clock 3 is added to the bottom right region (at line 18 in Figure 6).



(a) Initial placement of F/Fs. Number attached to F/F denotes the clock domain.

(b) F/F clustering based on region and clock. The number of each F/F cluster denotes the clock



(c) Moving F/F cluster under embedded clock constraints

(d) Local clock is added to non-saturated region.

Figure 7: Example of regional clock assignment. For simplicity, the local clock capacity of each region is assumed to be 2.

Since there are no that many combinations for main/local clock assignment, all the combinations are checked and the best one, for example, the combination that requires the least cluster movement to eliminate violations, is selected (line 17).

Next, based on the regional clock assignment in Step 2, F/Fs are moved to legal regions according to the embedded clock and the F/F capacity constraints in Step 3 (Note that F/Fs are not moved in Step 2). This is to minimize the total movement (distance) of F/Fs under the constraints of regional clock assignment and the F/F capacity of the region. Figure 8 shows an example of this incremental placement. Taking the current placement results (top left) and regional clock assignment (top right) as an input, violation-free global placement is generated (bottom).

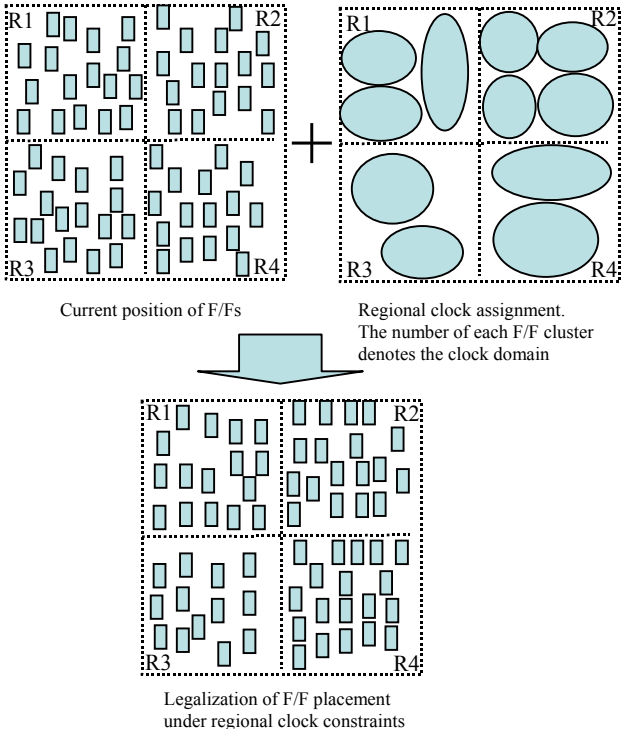


Figure 8: Example of incremental placement.

This problem can be solved by a minimum cost network-flow algorithm. Figure 9 shows the network used for the incremental placement problem, which has the following nodes and edges:

- Source node *S*
- Sink node *T*
- Set of region nodes: each node corresponds to one region.
- Set of F/F nodes: each node corresponds to one F/F.
- Set of edges from *S* to each region node: capacity is equal to the F/F capacity of the region; weight is equal to 0.
- Set of edges from each F/F node to *T*: capacity is equal to one; weight is equal to 0.
- Set of edges from region node to F/F node whose clock is assigned to the region: capacity is equal to one; weight is equal to the distance from the region to the F/F.

The minimum cost flow from *S* to *T* with the amount of flow equal to the total number of F/Fs results in the optimal incremental placement in terms of minimization of the F/Fs movement.

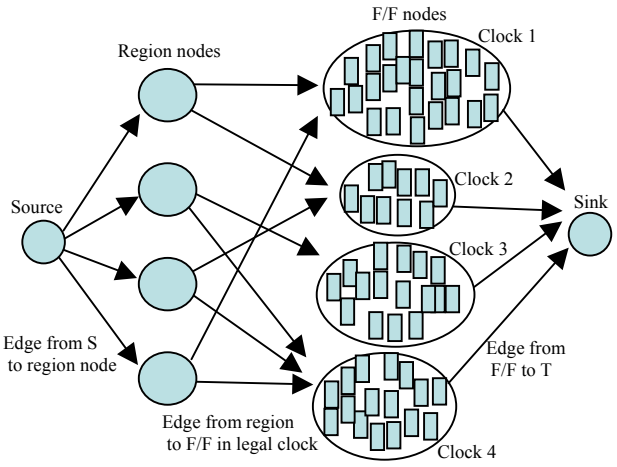
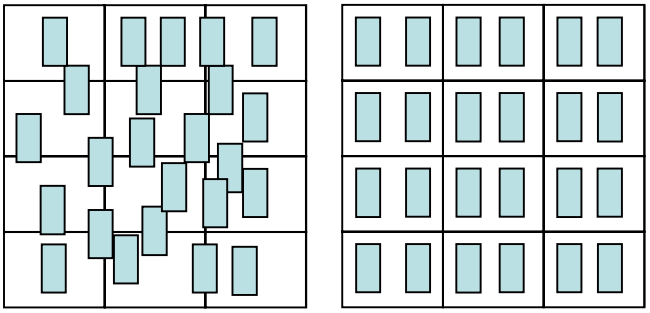


Figure 9: Network for incremental placement.

The final step in embedded clock optimization is detailed placement, where each cell is legalized on to the embedded slot. This is to minimize the total movement (distance) of F/Fs under the constraints of F/F slots. Figure 10 shows an example of detailed placement. Taking the current placement results (left) as an input, violation-free F/F slot assignment (right) is generated.



(a) Before detailed placement (b) After detailed placement

Figure 10: Example of detailed placement.

This kind of problem can often be solved by a linear-assignment algorithm. Figure 11 shows the linear assignment graph used for the detailed placement problem, which has the following nodes and edges:

- Set of F/F nodes: each node corresponds to one F/F.
- Set of slot nodes: each node corresponds to one slot.
- Set of edges from F/F node to slot node which can take the clock of F/F: weight is equal to the distance from the F/F to the slot.

Linear assignment on the graph results in the optimal detailed placement in terms of minimization of the F/F movement.

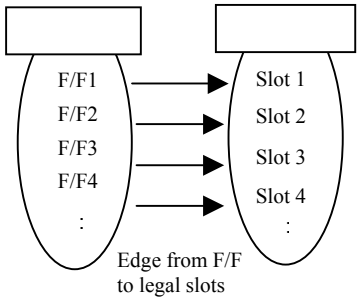


Figure 11: Linear assignment graph for detailed placement.

3.3 Design Methodology Enhancement Using Flooplanner

The problem with the approach used in the previous section is that the initial placement (Step 2 in Figure 5) may have too many regional clock assignment violations, which can seriously degrade the placement in Step 3 in Figure 5; numerous F/Fs have to be moved in accordance with the embedded clock constraints and placement becomes distorted. This has the potential to extend the design time.

To solve this problem, it is important to consider embedded clock constraints in the early design stages (before cell placement). For this purpose, we developed *floorplan-based embedded clock optimization* as shown in Figure 12.

Method 3: Floorplan-based embedded clock optimization

Step 1. Carry out embedded clock-aware floorplanning.

Step 2. Cary out placement-based embedded clock optimization.

Figure 12: Floorplan-based embedded clock optimization

Step 2 in Figure 12 is the same as the "placement-based embedded clock optimization" in Figure 5. In this approach, regional clock assignment is checked and optimized before proceeding to the level of detailed (cell) optimization in Step 2. This can resolve the problem of placement distortion that may occur in Step 2 inFigure 12.

More details of embedded clock-aware floorplanning (Step. 1 in Figure 12) are shown in Figure 13.

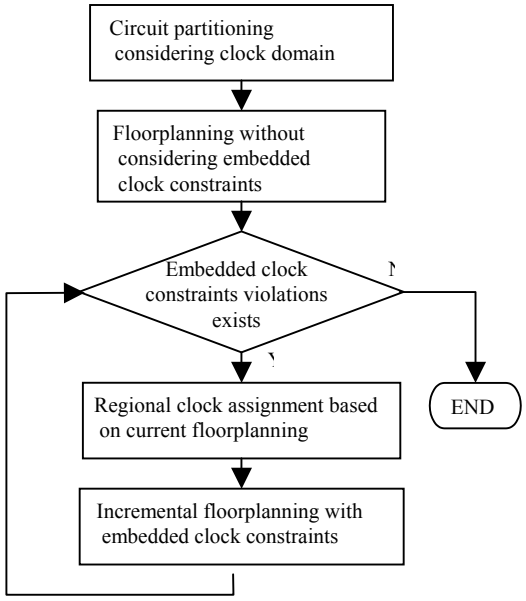


Figure 13: Embedded clock-aware floorplanning

This flow is based on the idea that "regional clock assignment" in Figure 6 can be naturally integrated into automatic circuit partitioning [11] and floorplannning[12]. First, circuit practitioner generates a set of clusters, where F/Fs of different clock are not mixed in one cluster. Next, initial floorplanning for the clusters is performed without considering embedded clock constraints, which is used for the following regional clock constraints generation. Then, incremental floorplanning is performed under the regional clock constraints. This constraints generation and floorplanning is incrementally iterated until all violations are eliminated with gradually increasing a penalty for the violations.

Design optimization in the early stages using circuit partitioning and floorplanning is also useful for optimizing interconnect delays To achieve higher performance, close to that of standard cell-based ASIC, we integrated the following performance optimization into the embedded clock-aware floorplanning in Figure 13, which includes RAM placement optimization.

- Circuit partitioning that considers logical hierarchy (absorbing timing-critical modules into one cluster) and physical hierarchy (optimizing number of interconnects between clusters).
- Floorplanning with a timing optimization capability which includes interconnect delay and timing analysis.

In order to fully exploit the capability of this new device, it is important to consider design constraints (for timing and embedded clock) from the early design stages by introducing this kind of approach.

4. EXPERIMENTAL RESULTS

The proposed design environment is used in ISSP design and more than 30 chips have already been taped out by the first-generation ISSP. In this section, our experimental results show how the proposed approach can optimize ISSP design.

4.1 Embedded Clock Optimization

To verify the effectiveness of our embedded clock-aware floorplanning, we compared the following methods:

Method 2: Placement-based embedded clock optimization.

Method 3: Floorplan-based embedded clock optimization.

The circuit used for the evaluation has 200K cell and seven embedded clocks. The total number of F/Fs is 56K and the number of local clock F/Fs is 15K.

Table 1 shows the number of F/Fs moved at the Step 3 in Figure 5.

Table 1: Embedded clock optimization

	# F/Fs moved in incremental placement
Method 2	1821
Method 3	0

This table shows that our approach reduces the number of F/Fs moved in incremental placement from 1821 down to 0. This means that the quality degradation of placement can be prevented by using our embedded clock-aware floorplanning.

4.2 Performance Optimization by Using Floorplanner

Next, we compared a conventional placement tool for standard cell-based ASICs (for global placement) and our specialized placement tool for ISSP to verify the effectiveness of using floorplanner. The circuit used for the evaluation has 380K cells and the target frequency is 200MHz.

Table 2 shows how the performance improved by our tools.

Table 2: Performance improvement with floorplanner

	Frequency (MHz)
Cell-based ASIC tool	153
Our tool	204

This table shows that a 30% improvement in performance can be obtained using our approach compared with using standard cell-based tools.

Figure 14 shows an example of floorplan used for the performance optimization.

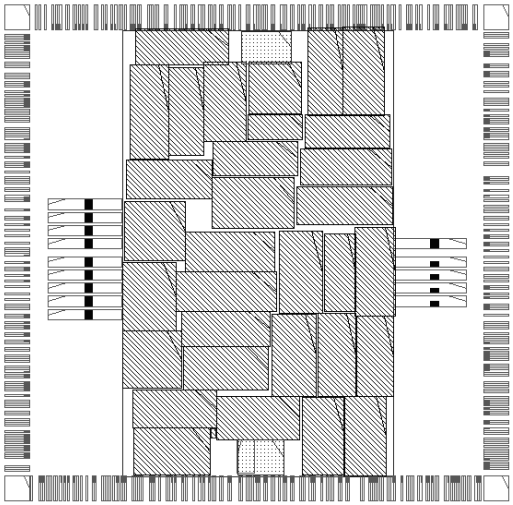


Figure 14: An example of floorplan

These experiments show that our newly developed tools can contribute to achieve both a short time to production and high-performance in the ISSP design.

5. CONCLUSION AND FUTURE WORK

In this paper, we have described a design methodology and tools for NEC Electronics' structured ASIC, *Instant Silicon Solution Platform (ISSP)*, which has been developed to fill the gap between FPGAs and standard cell-based ASICs. The ISSP has a unique regular-fabric architecture designed to achieve both a short time to production and high-performance LSI design. We have introduced a special design methodology and tools developed to fully exploit this new device capability with focusing on two key issues in the physical design: optimization of embedded clocks and interconnect delays.

Experimental results for industrial data show that performance improvements of up to 30% can be obtained using our approach compared with applying conventional standard cell-based tools, which confirm that the newly developed tools can contribute to both a short time to production and high-performance ISSP design.

Future work planned for the ISSP design methodology includes:

- Developing physical synthesis tools to fully exploit the device architecture capability.
- Enhancing the floorplanner for RAM and logic module placement.
- Developing a design methodology and tools which can efficiently handle circuit with more than 5 M gates in flat manner.

To achieve these goals, it will be very important to ensure that the development of device architecture and design methodology/tools is well integrated

6. ACKNOWLEDGMENTS

The authors would like to acknowledge their many colleagues in the ISSP project at NEC Electronics Corp. and the System CAD technology group at NEC Corp for their efforts in developing the ISSP devices and design methodologies and tools. The authors introduce the design methodologies and tools for the ISSP on behalf of the whole development team.

7. REFERENCES

[1] A. El-Gamal, I. Bolsens, A. Broom, C. Hamlin, P.Magarshack, Z. O.-Bach, L. Pileggi, "Fast, Cheap and Under Control: The Next Implementation Fabric," *in Proceedings of the 35th ACM/IEEE Design Automation Conference,* pp. 354-355, 2003

[2] A. Kahng, I. Bolsens, J. Cohn, B. Gupta, C. Hamlin, Z. Or-Bach, L. Pileggi, "What is the Next Implementation Fabric," *IEEE Design & Test of Computers, Vol. 20, no. 6,* pp. 86-95, Nov., 2003

[3] http://www.necelam.com/asics/index.php?Subject=ISSP

[4] http://www.lsilogic.com/products/rapidchip_platform_asic/index.html

[5] http://www.fma.fujitsu.com/accel/main01.asp

[6] L. Pileggi, H. Schmit, A.J. Strojwas, P. Gopalakrishnan, V.Kheterapal, A. Koorapaty, C. Patel, V. Rovner, K. Y. Tong, "Exploring Regular Fabrics to Optimize the Performance-Cost Trade-Off," *in Proceeding of the 34th ACM/IEEE Design Automation Conference,* pp. 782-787, 2003

[7] C. Patel, A. Cozzie, H. Schmit, L. Pilleggi, "An Architectural Exploration of Via Patterned Gate Arrays," *in Proceedings of 2003 International Symposium on Physical Design,* pp. 184-189, 2003

[8] J. Cong, Y. Fan, X. Yang, Z. Zhang, "Architecture and Synthesis for Multi-Cycle Communication," *in Proceedings of 2003 International Symposium on Physical Design,* pp. 190-196, 2003

[9] B. Hu, H. Jiang, Q. Liu, M. M.-Sadowska, "Synthesis and Placement Flow for Gain-Based Programmable Regular Fabrics," *in Proceedings of 2003 International Symposium on Physical Design,* pp. 197-203, 2003

[10] F. Mo, R.K. Brayton, "Fishbone: A Block-Level Placement and Routing Scheme," *in Proceedings of 2003 International Symposium on Physical Design,* pp. 204-209, 2003

[11] Y. Ono and T. Okamoto, "A Floorplan-Oriented Method of Circuit Partitioning," *in Proceedings of the Workshop on Synthesis and Systems Integration of Mixed Technologies*, pp. 301-307, 2001

[12] T. Okamoto and T. Yoshimura, "A New Approach to VLSI Floorplanning based on Quadratic Programming and Rectangle Packing," *in Proceedings of the Workshop on Synthesis and Systems Integration of Mixed Technologies*, pp. 257-263, 2001