Design Flow and Methodology for 50M gate ASIC

Alok Mehrotra, Lukas van Ginneken, Yatin Trivedi
Magma Design Automation Inc.
Cupertino, CA 95014

Abstract: This paper presents a methodology for full chip RTL timing closure for very large ASIC's. The methodology is based on the concept of a "Silicon Virtual Prototype". The methodology is based on the scalable technique of clustering and cluster placement and leverages the tight integration between the algorithms by means of a common, unified data model.

1. Introduction

The complexity of today's largest IC designs is over ten million gates. Looking ahead it is prudent to prepare a methodology for the 50M gate ASIC. Three forces are at work which make designing chips at the edge of the capability of the fabrication technology increasingly difficult. First, the size of today's 10M gate designs taxes even the largest and fastest computers. Second, the deep sub micron (DSM) effects are breaking existing design flows. The third force is the shrinking market window. The capacity and complexity problems impact the productivity causing a product to miss its market window.

This leads to a requirement for a **fast, high capacity and scalable** technology that provides early estimates of post-layout performance and identifies many issues that would typically have been found only after detailed place & route in a conventional flow. This saves numerous time-consuming iterations and enhances the productivity to enable fast time-to-market. An additional advantage of such a technology is that engineers can explore design architectures and implementation alternatives and have a high level of confidence that performance goals can be achieved.

The approach being discussed in this paper is that of a silicon virtual prototype (SVP). A *silicon virtual prototype* is a fast physical implementation of the design. This implementation has been coarsely tuned to reduce some of the most time consuming analyses and optimizations. However, the silicon virtual prototype still has sufficient accuracy to identify long-wire type timing implementation issues that are prevalent in large designs. Virtual prototyping leverages its high capacity to have a global view of the chip design to identify problems and also automate the creation of a floor plan. While most of the steps can be done automatically, the opportunity for manual intervention exists in most intermediate stages. Thus, the process of quickly building a virtual physical prototype of the final design from RTL or netlist in order to estimate chip area, performance and power early in the design cycle and determine changes needed to the RTL or constraints, if any, in order to successfully achieve timing closure in implementation, is called silicon virtual prototyping.

The organization of the paper is as follows. In section 2 we review previous work in this area. Section 3 outlines the two goals of the methodology. Section 4 describes an overview of the methodology as well as the use model. Section 5 describes extensively and in detail all of the different technologies which are required in this methodology. Section 6 introduces the products that embody this methodology.

2. Previous work

This paper touches on many issues in many different areas of EDA, but we will only review previous work in some of the core areas of prototyping, estimation and clustering.

Early work in estimation [3][4][6][8] primarily focused on prediction of area, wire length and congestion. For instance [3] and [4] describe methods for forecasting the size and wire length of a design or partition. [6] and [8] use floorplanning methods to forecast congestion and routability. [5] and [11] describe approaches where floor planning information is used to support behavioral synthesis and can hence be seen as a form of prototyping.

Beyond area and routability, performance and timing are an important concern [1][10][13].The authors of [17] describe an algorithm which simultaneously addresses logic synthesis and placement. [16] describes a more comprehensive hierarchical floorplanning methodology addressing the performance and timing budgeting issues.

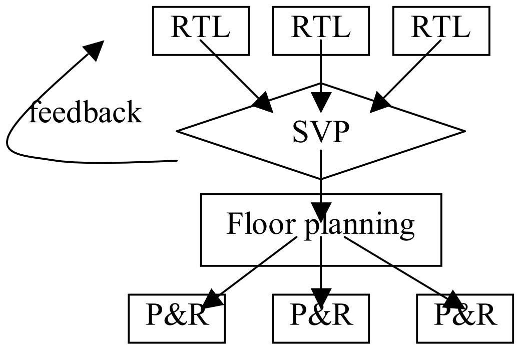
Prototyping technology raises the question of accuracy of fast estimation [18]. If the virtual prototype shows the design to be easily achievable then this approach provides a fast implementation flow.

The methods that we use to develop a prototype are based on clustering. Clustering has been described in a number of placement and partitioning contexts. [2] and [7] describe the application of clustering in placement using simulated annealing algorithms, while [12][15] describe the application of clustering in force directed or quadratic placement optimization algorithms. [14] describes the application of multilevel clustering to graph partitioning.

3. Two Goals

Building a silicon virtual prototype (Figure 1) has two distinct goals. The first goal is to determine the feasibility of the design. The second goal is to develop a floor plan and constraints that can be used for the actual implementation of the design.

Figure 1: Building a Silicon Virtual Prototype



These joint goals are addressed by quickly creating a fast, flat implementation of the design. This flat implementation leverages the unique scalability of the placement problem. By creating clusters, the number of objects to place can be reduced dramatically. Making the clusters larger creates nearly unlimited speedup of the placement algorithm. This creates a continuous trade-off between run time and placement quality.

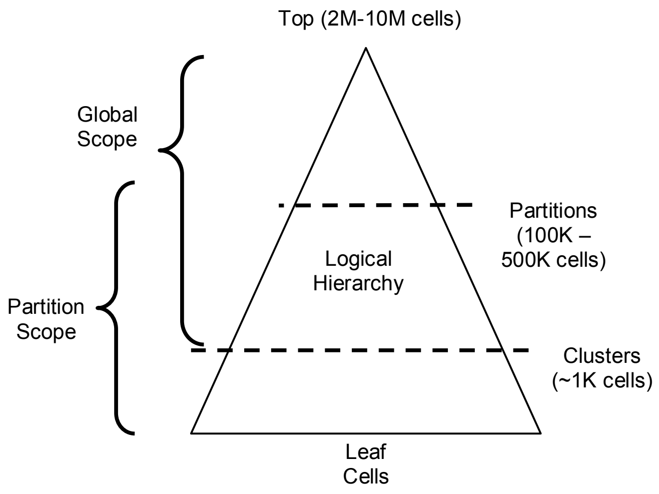
Feasibility concerns, especially early in the design cycle, are dominated by timing and performance issues. To do a feasibility check, it is essential that timing analysis is performed with reasonably accurate delays. Especially with today's multi-million gate chips, delay is very much affected by wire length. Long wires determine the performance of the design, and therefore cannot be ignored. Luckily, the cluster placement is sufficient to determine which wires are the long wires and to create a reasonably accurate wire length estimate for these wires. The short wires have much less of an impact on performance, and hence it is not necessary to complete a detailed placement to get accurate wire lengths for the short wires.

The second goal of virtual prototyping is to create a floor plan for the further hierarchical implementation of the design. The cluster placement can here be used as a guide to determine the appropriate partitions and to shape and position them. To drive implementation process for each of the partitions, it is necessary to set up the constraints in both the physical and the logical domains. In the physical domain, besides the shape of the partition, the most important constraints are the pin positions. In the logical domain, the relevant constraints are the timing constraints for the pins of the partition.

4. Proposed Prototyping Methodology

The prototype is a flat chip level design, in which global timing has been done. In order to speed up the placement, not actually each of the cells in the entire design is placed. Rather, the placement and optimization is done with clusters, which each consist of a number of cells (Figure 2). The number of clusters is several orders of magnitude smaller than the number of cells. The distances inside each cluster are small enough and the cluster sizes are more or less uniform, that the wiring does not play a role in the delay calculation.

Figure 2 : Tuning cluster size at different levels



Once timing closure has been achieved at the prototype level, the prototype can either be refined for final implementation or the design can be partitioned into *partitions*. Each of these partitions is then implemented flat. By making the clusters significantly smaller than the partitions, there is overlap of the global scope and the partition scope. Overlapping the global scope and the partition scope reduces the effects of the boundaries of the partitions on the quality of the placement.

The use model is one where a system integrator does daily integration. The timing closure loop consists of iterative prototyping runs while the RTL designer addresses timing issues from an architecture point of view. The methodology lets the system architect / system integrator learn about timing problems, by running the cluster placement and doing global timing. As timing problems are discovered, they can be solved at the "architectural" level. This can mean changing timing, pipelining, handshaking, FIFO queues, prefetching, speculative execution, parallel execution or by putting things physically close together. This is how the "architect" designs his chip. The end product of his/her design is the "golden RTL source".

To build a prototype, the system reads a Verilog or VHDL description, chip or partition level SDC timing constraints and any constraints for pad or macro placement. The system includes fast logic synthesis, static timing analysis, and placement. A daily prototype is built, starting with RTL and ending with global timing. Timing problems are identified as a result. The RTL designers work on the timing problems and try to resolve them. Now, in addition to all of the things a designer can do to speed up the design, the RTL can also be changed to impose physical proximity. By adding cluster placement constraint, the RTL designer has an additional tool to resolve timing closure problems at the architectural level. A design team will build several prototypes before committing to one particular partitioning.

The first step in the process consists of reading the Verilog source and clustering. In the case where the Verilog source is at the RTL level, this includes fast RTL synthesis. Once the net list has been read, the clusters in the hierarchy are identified. Ideally, most clusters already exist as modules in the logic hierarchy. Clusters should not be too large or too small. A global placement of the clusters is performed, which is fast, because there are only a few thousand clusters. Global static timing analysis can be performed on the placed clusters. This shows critical paths due to long wires. The cells inside the cluster are positioned in the center of the cluster. The global timing can also be used to derive timing budgets.

Once the prototype is complete, the design can be partitioned, a floor plan can be developed. Usually, partitions are chosen along the lines of the logical hierarchy, due to constraints imposed by the functional verification and test methodologies. Floorplanning involves giving each of the blocks a shape and a position. Both abutment style as well as channel style floor plans are supported. Once the floor plan has been determined, the timing constraints and the pin positions can be derived. This information is required for the separate implementation of each of the partitions.

5. Technologies Required for Prototyping

a. Fast, high quality RTL synthesis technology

One of the two goals of SVP is to support RTL design decisions and to arrive at full chip RTL timing closure. To do this, it is very important to perform accurate timing analysis and delay calculation. SVP provides much more accurate timing analysis as a global placement allows the identification of the long wires. The other requirement for SVP is to have very high capacity. It depends on synthesis technology which can synthesize several million gates at once— a 10x increase in capacity over conventional synthesis solutions. This means that, unlike conventional flows in which the RTL has to be arbitrarily partitioned into numerous submodules, our methodology allows design engineers to create their RTL based on functional intent rather than synthesis capacity limitations. Synthesizing the entire chip – rather than numerous submodules – produces superior optimization results because the algorithms are not limited by arbitrary boundaries.

The large ASIC designs are typically composed of multiple functional partitions that are in different stages of definition and implementation during the design process. If a SVP tool has to provide the early insight into the design issues it has to read in the design net list that would be a mixture of gates, RTL and black boxes (e.g. Hard IP). The SVP tool requires a synthesis engine, which supports black box models and can start prototyping from incomplete RTL.

As conventional synthesis tools are too time-consuming for SVP, some prototyping tools use "fast and dirty" synthesis like direct RTL mapping. This compromises the correlation between the results obtained from such a technology and the quality synthesis engine that will be used for the implementation. Poor correlation minimizes the value of the early feedback and not only makes the flow unpredictable but also does not prevent iterations between logic and layout.

b. Clustering Technology

The proposed virtual prototyping methodology is based on the unique scalability of placement by means of clustering, which is an abstraction technique for placement. It reduces the number of objects that need to be considered in an intuitive way: To partition a design of 10M cells into 10 paritions, it should not be necessary to consider the individual standard cells. The clustering factor gives the designer an easy way to trade off accuracy versus run time, which is important when cluster placement is used to evaluate floor-planning decisions.

Clustering is primarily driven by the connectivity of the net list. Placement clusters should be small enough to be uninteresting to the designer. A clustering algorithm should generate them automatically without user interaction.

The objective is to select clusters that have few connections to other clusters. The actual number of cells vary from cluster to cluster to keep the areas of the clusters as uniform as possible. Having uniform cluster areas is good for placement, because placement algorithms are better at handling uniformly sized objects. If the clusters are too large then the deviation between the cluster placement used for the optimization and the final detailed cell placement can nullify the accuracy advantage. Similarly if the clusters are too small then the runtime gets adversely impacted.

Since the clusters are small, the shape of the cluster is not very important, hence it has been chosen to be a square with an area proportional to the area of the cells contained in the cluster. For the purpose of placement, all of the cells contained in a cluster are assumed to be at the center of the cluster. The exception is very large cells, such as macro's, which are the only cell within a cluster. Such clusters have the area and shape of the macro itself.

Clusters follow the logical hierarchy, meaning that they only group cells that have the same parent in the hierarchy. If the logical hierarchy is deep and there are modules of all sizes in the logical hierarchy, then it is often advantageous to use the modules of the logical hierarchy as clusters. Since these have names that are meaningful to the RTL designer, the resulting placement is much easier to read for the RTL designer. Sometimes timing constraints are attached to the boundaries of the modules, and therefore their preservation is more important. The RTL designer is already used to using the logical hierarchy as a tool to influence the implementation of the design. Hence, keeping a logical module together is in line with the intentions of the RTL designer, even if it is not optimal with respect to wire length. Keeping elements in logical hierarchy together gives the RTL designer the opportunity to make physical proximity decisions as architectural decisions.

The clusters form a separate hierarchical data structure, which represents the physical hierarchy. This second hierarchy coexists simultaneously with the logical hierarchy. The clusters are allowed to contain a hierarchical cell, which is interpreted as if they contain every leaf cell in the hierarchy tree rooted by the hierarchical cell. In this manner the physical and logical hierarchy may be interwoven. The physical hierarchy has a second, higher level, the level of the partition. The partition is a grouping of clusters in the physical hierarchy.

c. Cluster Placement Technology

The clusters are placed using a force directed placer, which uses a second order non-linear optimization algorithm. While clustering reduces the number of placeable objects, the number of nets decreases sub-linearly. After clustering, there are many nets that connect exactly the same set of clusters. A single net with a higher weight can replace these nets.

After the placement is first performed, a timing verification can be performed. This is the main feasibility check on the implementability of a very large design. The placement of the clusters shows where the long wires are. This information is used in the delay calculation of the long wires.

d. Fast optimization technology

Fast optimization technology is used to implement the logic with the required accuracy. For combinational logic, the speed advantage is obtained through synthesis methods based on the concept of logical effort. The concept of logical effort has a simple and elegant relevance – the delay depends on the gain of the gate, and not on its exact parasitics. The resulting breakthrough is that the ratio of capacitances can be chosen within limits beforehand, and can be kept constant by gate sizing during the design implementation stages. By adopting this methodology, delay can be controlled without advanced knowledge of parasitic capacitances. More importantly, circuit evaluations are performed without guessing at or fixing cell sizes before the actual routing exists.

As the design progresses the gain of each gate is carefully tuned, delays are spread over paths, and the timing is maintained. This methodology has a unique advantage that the gain-based approach results in slew equalization where every cell in the design is sized such that it has exactly the drive strength for that path to meet timing. This is done for all paths whether they are critical or not and due to this approach SI issues are greatly reduced as there are no unnecessarily overdriven or under driven nets which would have played the role of aggressor or victim respectively.

All of this means that there are a number of very important outcomes from the use of gain-based synthesis:

- Synthesis times are dramatically reduced compared to traditional synthesis techniques. In one typical real-world example, a multi-day evaluation with a traditional synthesis tool was reduced to less than 5 hours with gain-based synthesis tool (Blast Create).
- The relative simplicity of Blast Create's gain-based calculations means that it has a far greater capacity than other synthesis solutions. In turn this means that Blast Create has the capacity to handle multimillion-gate designs without resorting to artificial partitioning.

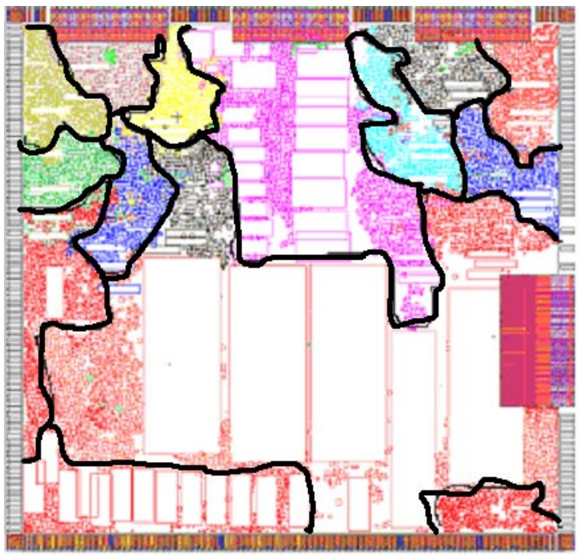
The end of the synthesis step completes all timing optimizations, and all of the circuit delays are determined and frozen.

e. Partitioning

The partitioning methodology works as follows: When the clusters are first placed, the modules in the net list hierarchy form amorphous areas. The system includes graphical means to visualize the placement of the various levels in the logical hierarchy (Figure 3). This forms a guide to the user to determine the choice and final shape and location of top-level partitions of the hierarchical floorplan. Figure 1 shows clusters in a full chip. Each cluster is represented by a different color (shown here by marking the borders explicitly).

The user will have to decide on the appropriate partitions based on the placement of the clusters. Modules in the logical hierarchy are candidates for partitions. Hierarchy manipulation can be used to arrive at good partitions. A heuristic will assign shapes and positions to each of the partitions. The user can intervene at this point and shape the partitions and give them a position.

Figure 3: Cluster representation



Since the partitions are usually rectangular, the clusters will need to be placed into these rectangles. This is accomplished in a second cluster placement stage. In this stage each of the partitions is represented by a region constraint or “fence”, which forces the clusters to remain within their assigned regions.

f. Time budgeting

For the front end RTL designer, the most important aspect of the prototype is timing verification and timing budgeting. From timing verification the RTL designer desires an accurate forecast to determine if timing closure of the RTL design can be achieved. If not, the RTL designer needs feedback as to what measures can be taken to improve the timing of the design. An RTL designer works on a design to improve the timing by changing the architecture of the design.
Timing budgeting uses gain trimming approach which adjusts the delays of the cells to get zero slack at all points in the design. Gain trimming works as follows: a factor (the gain) is adjusted on outputs of cells depending on the slack. Outputs with a positive slack are made slower and outputs with a negative slack are made faster, proportionally to the slack and the number of levels of the path. Since there are a lot of reconvergent paths, which influence each other, this has to be repeated to converge to zero slack. This requires multiple timing analysis runs. Once zero slack has been achieved at all points, the timing constraints can be derived by characterization. As the required and arrival times are the same, either one, or any combination can be used to form the timing constraints if the design will be implemented hierarchically or it can be incrementally refined for final flat implementation.

g. Pin assignment

To implement each of the partitions, it is necessary to determine the pin positions on the boundary of the partitions. This must be done carefully, taking into account the requirements of antenna rules, power grid, as well as congestion. The pin assignment depends directly on the cluster placement. Because clusters are smaller than partitions, and clusters have been placed within the partitions, they give detailed placement information of the cells within each of the partitions. This information is used in pin assignment by routing from cluster to cluster.
Two methods for pin assignment are available. The first method is a simple heuristic based on point to point connections. This method is suitable for interactive work, as pins can be assigned one at a time, or by group. The second method is based on global routing. This method is more sophisticated, as it takes congestion into account.

h. Testability analysis technology

The RTL design can have structural testability issues that would require an iteration to modify the source. The structural issues can be identified early with net list checking tools. The SVP identifies the implementation issues. As the repair for testability issues modifies the netlist this also has to be embedded in the flow that produces the final optimized netlist. As we have stated before that logic synthesis is an integral part of the SVP flow, we need to include DFT analysis and repair as a key required technology.
Failure by logic designers to adequately apply scan design rules early in the design cycle can lead to poor test coverage and testability problems later in the product development cycle. Hence DFT checks including checking for gated clock, bi-directional and tristate bus contention, set/reset inputs help identify and fix, if needed, such problems early in the design cycle.

i. Correlation to implementation technology

A key requirement for a SVP is that it correlates with the actual implementation results. The SVP tools in the market today face a challenge that building interfaces between disparate tools has put most of the RTL-to-GDSII flows together. These typically have multiple analyses engines like extractor and timers and have correlation issues within the flow so it is impossible for an external SVP tool to not have inaccuracy issues. Moreover, the use of different placement and global routing algorithms can lead to a substantial miscorrelation with the actual implementation.
High performance or high-density designs require accuracy of optimization. Clock tree, power structure and scan contribute significantly to the use of routing resources. Also the impact of placement quality cannot be neglected. Poor placement increases overall wire length, and reduces performance, and worse, it leads to invalid assumptions for optimization. Moreover,

reducing optimization in the early stages of the design methodology can lead to larger increases later in the flow.
Magma has the unique advantage that it has common analyses & optimization engines in its integrated flow from prototyping to final implementation. Magma also has a unified data model enabling seamless sharing of all data throughout the complete flow. This ensures that most of the algorithms and all of the analysis engines are shared, guaranteeing close correlation between the prototype and the actual implementation.

j. Glass box Technology

Assembling a design, which has been created using a hierarchical methodology, requires a strategy to reduce the amount of data. The entire design consists of the partitions plus the routes to connect the partitions. In order to assemble the design, it is necessary to have certain data for each of the partitions; in particular, it is necessary to have the pin positions and IO timing, but it is not sufficient. For accurate delay calculation, it is necessary to have a detailed delay model for the input and output wires of each of the partition inputs and outputs. To do antenna rule fixing, a detailed model for each of the wires is required.
One approach is to develop model for each of the many different applications that require analysis of the partitions: delay calculation and antenna rule check are just two of many types of analysis. Other types of analysis are noise analysis, clock skew analysis, detailed routing modeling, design rule checking and timing analysis. This however, requires special data representation, often with special data formats, to be developed for each model, together with the data abstraction methods. This constitutes a considerable code development, test and maintenance burden.
The other approach is not to require the development of special purpose models, but to inspect the design of each of the partitions itself. This has the advantage that the actual design data is inspected in the same manner as the design data is inspected during flat design. After extraction, the actual routing geometries become the "model" for delay calculation, while the same geometries can be inspected for antenna rule violations. While this is considerably easier to implement, the amount of data for each of the partitions is prohibitive.
The solution is to reduce the actual design data, by selective deletion of features, which are not relevant for the chip assembly. In particular, for timing purposes, cells that are contained between register boundaries, with no combinational path to a partition IO, cannot influence the top level timing. Similarly, for antenna rule checking, it is only necessary to inspect the routing geometries between an IO pin and the cells within the partition. Preserving these geometries allows the entire route to be checked using the same antenna rule checking code that is used for a flat design. Other routes can be deleted, assuming that they have been implemented correctly at the partition level.
We have developed data reduction techniques along these lines. The data reduction techniques delete all design elements (cells, nets, routing geometries) which cannot affect any global analysis. While there are many different global analysis methods to take into account, experience shows that in most cases, a large amount of data can be removed. The amount of reduction tends to grow with the size of the partition, so this methodology will scale very well with future technology generations.
Since there is a single, comprehensive file format to represent all design data, no new format for abstraction needs to be invented. Moreover, this single format can support all present and future analysis requirements. No separate formats or "views" are required to support different analysis requirements.

6. Product Description

Blast Create allows logic designers to check, visualize, evaluate and improve the quality of the RTL code and design constraints. A logic-level prototyping or structural analysis is useful in the detection of large muxes, snaking timing paths, fanout violations, combinational loops or deep logic levels. Early detection of such problems leads to good quality RTL, hence a good quality netlist that finally leads to a better implementation. This facilitates a clean ASIC handoff model. The checking is enhanced by visualizations of the RTL functionality that lets designers view schematic representation of their code, view registers and latches, logic-clouds as well as allows them to directly cross-probe into the source RTL. Blast Create also facilitates testability checks via *Blast DFT* that is seamlessly integrated within Blast Create.
Blast Create delivers the only viable prototyping solution for today's complex designs with fast, high-capacity synthesis and fully integrated analysis and implementation engines that operate on a common datamodel. Figure 4 shows a chart of experimental data using Blast RTL on various designs. The chart is shown for the size of design vs runtime in hours. Table 1 shows some of the experimental results of Blast Create usage on real designs.

Figure 4: Experimental results from Blast RTL

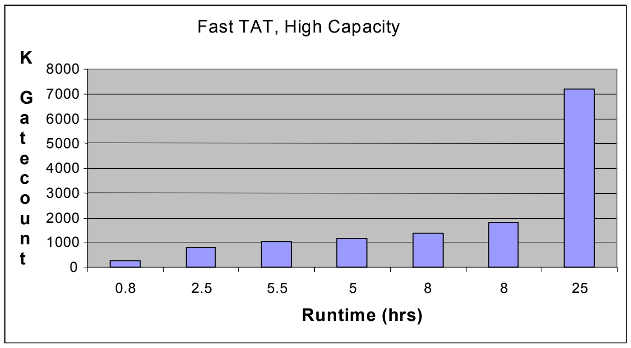


TABLE 1: Experimental results from Blast Create

	A	B	C
Design Size (Gates)	1.3M	6.0M	23.2M
Macros	104	69	--
Cluster Size	200	10	1000
Cluster Placement	6 min	2.5 hrs	2 hrs
Prototype	2.8 hrs	7 hrs	20 hrs
Memory	1.5 GB	317 MB	10GB
Platform	32bit Linux	32bit Linux	64bit Solaris

It is important that the prototype correlates well with the actual implementation tool i.e. the quality of results should not be compromised. Blast Create is implemented on the same code base as *Blast Fusion*; it uses the same timing analysis and shares the same high quality placement and optimization engines.

Blast Create allows you to rapidly place the entire design to create a prototype of the chip. It allows you to place the design including the macros. Floor planning constraints and manual decisions can be quickly evaluated by iterating the placement interactively. To speed up the placement, clusters of cells are placed.

Blast Plan is the portion of the product which uses SVP to create a floor plan. This tool is targeted for the back end designer, who, faced with a very large design, needs to create a floorplan for the hierarchical implementation of a design. Blast Plan allows the selection and creation of partitions, and most importantly, it allows the creation timing constraints and pin positions for the separate implementation of the partitions. Blast Plan also contains the abstraction technology for the bottom up hierarchical assembly of a large chip.

7. Conclusions

We have presented a methodology for the design of next generation of multi-million gate chips. The methodology is based on the construction of a "Silicon Virtual Prototype", which allows both an early timing feasibility check as well as the construction of a globally optimized floor plan.

8. References

[1] Modarres, H. and S. Raam, J.-H. Lai: "Floorplanning of hierarchical layout in ASIC environment" in: Proc. IEEE Custom Integrated Circuits Conf., sec. 7.1, Rochtester, May 1988.

[2] Mallela, S. and L.K. Grover: "Clustering based simulated annealing for standard cell placement" in: Proc. 25th ACM/IEEE Design Automation Conf. pp.312-317, Ahaheim, June 1988.

[3] Chen, X. and M.L. Bushnell: "A module area estimator for VLSI layout", in *Proc. 25th ACM/IEEE Design Automation Conf.* pp.54-59, Anaheim, June 1988.

[4] Pedram, M. and B. Preas: "Accurate prediction of physical design characteristics for random logic", in: Proc. IEEE Int. Conf. Computer Design, pp.100-108, Oct. 1989.

[5] McFarland, M.C.: "A fast floor planning algorithm for architectural evaluation" in *Proc. IEEE Int. Conf. Computer Design*, pp. 96-99, Cambridge, Oct. 1989.

[6] Lengauer, T. and R. Muller: "A robust framework for hierarchical floorplanning with integrated global wiring", in: Proc. Int. Conf. on Computer-Aided Design, pp.148-151, 1990.

[7] Lee, Y.W. and Y.S. Cheung, C.S.K. Yeung: "A flexible clustering and floor planning approach to standard cell placement using hierarchical simulated annealing", in: *Proc. Int. Conf. Circuits and Systems*, pp.882-885, Shenzhen, 1991.

[8] Pedram, M. and E. Kuh: "BEAR-FP: A robust framework for floorplanning", in: Int. J. High-Speed Electron., vol.3, no.1, pp.137-170, 1992.

[9] Ding, C.L. and C.-Y. Ho, M.J. Irwin: "A new optimization driven clustering algorithm for large circuits" in Proc. Eur. Design Automation Conference, pp.28-32, Hamburg, Sept. 1993.

[10] V. Narayananan and D. LaPotin, R. Gupta, G. Vijayan: "PEPPER – a timing driven early floorplanner", in Proc. IEEE Int. Conf. Computer Design. pp.230-235, Austin, 1995.

[11] Mecha, H. and M. Fernandez, F. Tirade, J. Septien, D. Motes, K. Olcoz: "A method for area estimation of data path in high level synthesis", in IEEE Trans. Computer Aided Design, Vol 15, no. 2, pp. 258-265, Feb. 1996.

[12] Chen Chunhong, and Tang Pushan: "Cluster-based placement for macrocell gate arrays", in Proc. 2nd Int. Conf. on ASIC, pp.46-49, Shanghai, Oct. 1996.

[13] Bushroe, R. G. and S. DasGupta, A. Dengi, P. Fisher, S. Grout, G. Ledenbach, N. S. Nagaraj, R. Steele: "Chip hierarchical design system (CHDS): A foundation for timing-driven physical design into the 21st century", in: Proc. Int. Symp. Physical Design, pp.212-217, 1997.

[14] Alpert. C.J. and Jen-Hsin Huang, A.B. Kahng: "Multilevel circuit partitioning" in: IEEE Trans. Computer Aided Design, vol.17, no. 8, pp.655-667, Aug. 1998.

[15] Xianlong Hong and Hong Yu, Changge Qiao, Yiel Cai: "CASH: a novel quadratic placement algorithm for very large standard cell layout design based on clustering", Proc. 5th Int. Conf. Solid-State and I.C. Technology, Beijing Oct. 1998.

[16] Su, Hsiao-Pin and A.C.-H. Wu, Y-.L. Lin: "A timing-driven soft-macro placement and resynthesis method in interaction with chip floor planning", in Proc. 36th Design Automation Conf. pp.262-267, New Orleans, June 1999; Also in: IEEE Trans. Computer Aided Design, Vol 18, no. 4, pp.475-483, April 1999.

[17] Salek, A.H. and Jinan Lou, M. Pedram: "An integrated logical and physical design flow for deep submicron circuits". IEEE Trans. On Computer Aided Design, Vol. 18, no. 9, pp.1305-1315, Sept. 1999.

[18] Sarrafzadeh, M. and M. Wang: "Can fast algorithms be used as good predictors?", in: Syst. Level Interconnect Prediction, pp. 125-1999.

[19] Ranjan. A. and K. Bazargan, M. Sarrafzadeh: “Fast hierarchical floorplanning with congestion and timing control”, in: Proc. 2000 Int. Conf. Computer Design, pp.357-362, Austin, Sept. 2000.

[20] Ranjan A. and K. Bazargan, S. Ogrenci, M. Sarrafzadeh: “Fast floorplanning for effective prediction and construction”, in. IEEE Trans. VLSI Systems, Vol. 9, no. 2, pp.341-351, April 2001.