The Coming of Age of Physical Synthesis

Charles J. Alpert
Manager
Design Productivity Group
IBM Research
Austin, TX 78758

Chris Chu
Associate Professor
Department of ECE
Iowa State University
Ames, IA 50011

Paul G. Villarrubia
Senior Technical Staff Member
Server and Technology Group
IBM Corp.
Austin, TX 78758

***Abstract*— Physical synthesis, the integration of logic synthesis with physical design information, was born in the mid to late 1990s, which means it is about to enter its teenage years. Today, physical synthesis tools are a major part of the EDA industry, accounting for hundreds of millions of dollars in revenue. This work looks at how technology and design trends have affected physical synthesis over the last decade and also how physical synthesis will continue to evolve on its way to adulthood.**

I. Introduction

Physical synthesis is motivated by the technology trend that is driving the increasing magnitude of wire delay as a percentage of total circuit performance. Several technology generations ago, when wire delays were insignificant, the timing of a placed circuit for the most part matched the timing of an unplaced circuit. As wire delays became more significant, there emerged a mismatch in timing between the unplaced and physical implementation of the design. Developers realized early on that wire characteristics could be predicted based on a known placement and looked for ways to combine placement technology with synthesis technology to achieve *timing closure*. This process drove new requirements into placement, synthesis and timing tools.

Today, deep sub-micron effects have necessitated that physical synthesis evolve from timing to *design closure*. Here multi-faceted objectives in routability, area, yield, clock skew, and power must all be managed. This paper begins by outlining the fundamental components of physical synthesis and how they can be a timing closure flow. Next it will explore how various aspects of physical synthesis have matured and morphed as physical synthesis has come of age.

II. Overview of Physical Synthesis

An example fundamental physical synthesis flow is as follows (see [1] for more details):

1. Netlist preparation. Before placement, gates may need to be sized to fit into the image space, buffers may need to be removed, clock trees hidden, etc. In addition, timing information can be extracted from either an unplaced or a previously optimized netlist to generate netweights for timing driven placement.

2. Placement. Besides just traditional minimum wirelength optimization, placement needs to address density targets, designer cell movement constraints, and routability directives.

3. Timing Analysis. After placement, one can run static timing analysis to see how much timing degradation has occurred and identify nets with signal integrity problems.

4. Electrical Correction. After placement, one may find gates which drive load above the allowed specification and long wires for which the signal exceeds the designer specified slew rate. Electrical correction fixes these problems, typically through buffering and gate sizing, thereby getting the design into a reasonably good timing state. One can also employ a logical effort [2] type of approach to get the design into a reasonably good global state in terms of timing closure.

5. Legalization. Fixing electrical violations may insert hundreds of thousands of buffers into the design, and potentially every gate in the design may be resized. This certainly will introduce cell overlaps and make the design illegal. Legalization fixes these overlaps. Ideally, a legalization approach should fix problems without moving cells too far from their intended location since big movements can potentially undo the work from the previous stage.

6. Critical Path Optimization. At this point the design is legal and in a reasonable state to begin identifying the set of most critical paths to work on and fix with incremental synthesis techniques [3]. This phase can be run with incremental timing analysis to give feedback as to whether a given logic transformation really did improve the path in question. These optimizations can also be followed by intermittent calls to legalization.

7. Compression. Critical path optimizations may become stuck rather quickly, when it reaches a certain set of paths that are the most critical yet cannot be fixed without manual design intervention. However, there still could be thousands of failed timing points that exist that could be fixed with light weight optimization (e.g., incremental buffering and gate sizing). The purpose of this phase is to "compress" the remaining negative portion of the timing histogram to leave as little work as possible for the designer. As in the previous step, incremental timing analysis and legalization needs to be incorporated where appropriate.

After these steps, the design may still be far from closing on timing constraints. At this point, a designer could intervene manually or re-run the flow to get a better timing-

driven placement now that the real timing problems have been identified. If the design is close enough to closing, one can enter a phase of more fine grained optimizations, including post-routing timing correction.

Note that this flow is just a template and may be missing several key components of an industrial strength tool. For example, one needs to deploy techniques that recover when the design goes off into a poor state. When certain portions of the design get full, there may be no room left for optimizations to make changes to improve timing. One can deploy area recovery techniques (such as sizing down non-critical gates) to free up space for optimizations to make progress. One can also apply local recovery techniques when certain areas get too hot or too congested for routing.

III. Coming of Age

This section discusses various aspects or components of physical synthesis in the context of how the problems have changed over the preceding decade.

A. Global Placement

When physical synthesis was young, improvements in netlist partitioning emerged to drive new algorithms for min-cut placement [4]. Around the same time, quadratic and force-directed methods started coming onto the academic scene [5] though may have been used in industry long before.

Only recently has the design automation community begun to understand the advantages and limitations of the different algorithms, in large part due to recent placement contests [6] [7]. One trend that emerged is that analytical techniques do a better job in managing the free space for fragmented top-level ASIC designs as shown in Figure 1. Another is that it was shown that high-quality results can be obtained with significantly less run time [8]. There is no question that placement tools have improved greatly in terms of pure wirelength and also speed and scalability. Today there is widespread use of force directed techniques due to the algorithmic efficiency of these approaches as well as the ability to readily adapt to a netlist that is changing during the synthesis process itself.

The fact that now it may take multiple cycles to cross the chip is breaking down traditional placement objectives. Figure 2 shows two placements of a latch with the same wirelength, but only placement (b) will meet its cycle time if it takes two cycles to cross the chip. Current placement algorithms will not distinguish between these solutions, and timing-driven placement may not help either. For example, a lower net weight for the rightmost net could simply force the latch to slam its way to the other side.

B. Legalization

One can represent the placement region as a series of bins, each of which contains its local density information. Then incremental optimization could query a bin to see if it had space before making a change, such as increasing the gate size. After enough optimization had taken place,

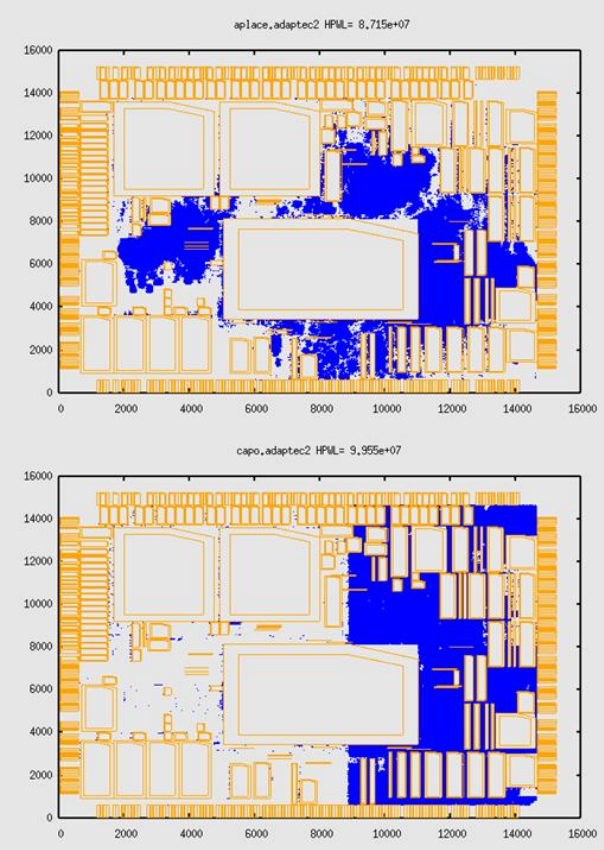

Fig. 1. Solution by APlace (top) and Capo (bottom) for adaptec2 in ISPD 2005 placement contest.

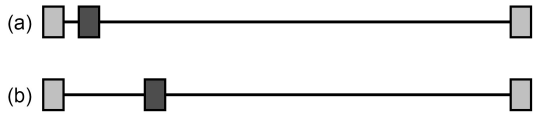

Fig. 2. Placement of latch in (a) has worse timing than the placement in (b) though they have equal wirelength.

legalization could then reconcile the changes in each bin and once again have a legal placement. As long as the bins did not get overfilled, this was not a problem.

Interconnect delays increasing relative to gate delays has caused the number of netlist changes (especially in terms of buffering) to increase dramatically, which causes the bin-based model to break down. Today, a new model for early optimization is to allow cells to be placed at their desired locations and ignore bins entirely. Then, modern legalization techniques that smoothly spread such as diffusion [9] or cell shifting [8] can be used to legalize the design without too much disruption.

Physical synthesis is also moving away from legalization entirely and instead relying on an incremental mode whereby legal locations are found for cells the moment that they are inserted. This way, the design is always legal after any transformation has occurred. Consider increasing the size of a cell that no longer fits in its current space. The "legalization" engine could find a new nearby location

or try and slide its neighbors out of the way, creating a rippling effect that may affect other paths. This kind of pin-point surgical precision is emerging as extremely critical for successful late post-routing optimization and ECOs where one can ill-afford for legalization to disrupt carefully tuned paths.

C. Buffering

In the early days of physical synthesis, topology-aware buffering emerged for load considerations and critical path optimization; the newly inserted buffers could be absorbed into the placement image fairly easily. Today, a large ASIC could require hundreds of thousands of buffers [10] because highly resistive wires create signal integrity problems in just a short distance. This means one must emphasize algorithms that are extremely fast and power efficient, potentially at the expense of delay.

While this trend causes a headache for legalization, it also severely restricts the ability to route. For a long unbuffered net, the router has a lot of flexibility to choose the topology for the net. But if the net has a buffer inserted every millimeter, then the flexibility is eliminated. The router can only choose the routes for the short connections between buffers. If buffers are placed in highly congested routing regions, the router has no choice but to route to the buffers and congestion cannot be alleviated. Consequently, buffering has forced routability to be considered even earlier in the flow.

Crossing the chip in multiple cycles is also forcing buffering and latch placement to become integrated. While algorithmically this is fairly simple, making sure all the latches are globally placed well is certainly a challenge (as discussed in Section III-A).

D. Routability

When physical synthesis was a toddler, global routing was an afterthought. One could perform timing closure and generally route the design without too much difficulty. Of course, this is no longer the case. Not only does global buffering restrict a router's flexibility, but having routes go scenic to avoid congestion can cause nasty timing surprises. Complex design rules and DFM effects for 65 nanometer technologies and beyond only exacerbate congestion problems.

Fast and accurate routing and congestion estimation techniques are emerging to aid with optimization. For example, FastRoute [11] is a new, high-quality global router which is two orders of magnitude faster than previous global routers, thereby enabling it to be embedded into congestion mitigation algorithms. It utilizes an extremely fast and accurate rectilinear Steiner minimal tree algorithm called FLUTE [12] that makes use of pre-computed lookup table to efficiently handle nets with up to nine pins (and uses recursive techniques for larger nets). This router has also been incorporated into FastPlace and is effective at significantly reducing routing congestion without sacrificing much wirelength or runtime [13].

We anticipate seeing other optimizations such as buffering, legalization, and wire synthesis also become more congestion driven in the future. In addition, pre-routing based optimizations will need to keep a closer eye on noise and signal integrity to make sure they do not force the router to place competing aggressor wires next to each other.

E. Wire Synthesis

The wire sizing literature was actually quite active in the 1990s (e.g., [14] [15]); however, industry regards these works as somewhat impractical because routers cannot easily support topologies with multiple and/or continuous wire widths. Instead, one could simply use uniform wire sizing to achieve most of the performance benefit [16]. However, physical synthesis has always had to use wire sizing cautiously since it could have a detrimental effect on the ability to route. For example, sizing a wire up to three routing tracks from one routing track may reduce wire delay significantly, but it also removes two potentially precious routing tracks. Since routing is more difficult than ever, one cannot afford to just size up too many wires.

With 65 nanometer technologies and beyond, there are new "wire sizing" problems that really fall under the category of *layer assignment*. Figure 3 shows a cross section of a routing tile with 14 routing tracks, of which eight belong to a single 1x thickness layer, four belong to a double 2x thickness layer, and two belong to top 4x metal. If one wants to assign a wire in this tile to the thickest metal to get the best performance, there are two tracks of thickest metal available to choose from, and this assignment does not consume any extra routing resources!

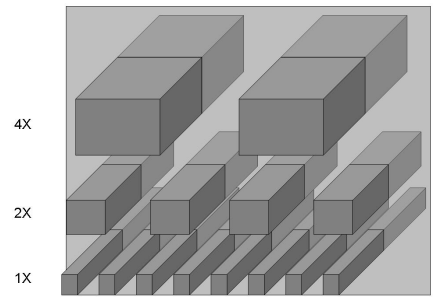

Fig. 3. Layer assignment for 65nm and beyond.

The problem now becomes to perform a global layer assignment of wires to each level of metal, and the delays incurred by this assignment need to be taken into account during physical synthesis, which means they need to happen before routing. Further, there needs to be a mechanism through which this assignment is communicated to the router, e.g., as a hard constraint (this net must be on top metal) or a soft constraint (this net is preferred to be on top metal). One must also be careful not to ignore the high via cost of going up to thick metal, both in terms of delay and its effect on blocking routes on lower metal layers.

Physical synthesis is just beginning to scratch the surface in this space. Moving forward the trend is towards a world

where there are no more wires, just strings and trees of buffers, all motivated by the physical characteristics of wiring as well as the physical topology of the wire sink placement.

F. Gate Sizing

Gate sizing is a core component of physical synthesis. Incremental timing closure becomes much easier if one begins from a reasonably good starting state. One well-known technique for coarse gate sizing is logical effort [2]. In the last decade, this sizing problem has not changed too significantly, except that now it is more important to achieve timing closure with minimum area and power.

Advanced technologies have more than one voltage threshold (Vt) library available. Soon libraries with three or four levels will be common. By changing the Vt assignment of a gate from a high Vt to a low Vt gate, one can sacrifice power for a performance gain. Of course, using all low Vt gates will give the best timing performance while completely blow through any reasonable power budget.

The assignment of gates to different Vt libraries needs to be managed throughout physical synthesis. After electrical correction, one can perform a global Vt assignment but as more paths close on timing, this assignment can become stale. One needs to be able to recover power by moving low Vt cells back to high Vt. There are many different strategies one could use to manage the power budget; at the extremes, one could start by assigning all gates to low Vt and recovering as much power as possible or alternatively starting from all high Vt and only deploying low Vt when absolutely required.

G. Timing Analysis

The underlying timing technology is also driven by deep-submicron trends. While fast incremental timing has always been a core component of physical synthesis, incremental timing needs to work in modes of variable accuracy and variable detail. In the early phases, one can use fast, abstracted models such as table lookup circuit delay models and Elmore delay model. In the later stages of physical synthesis, more detailed timing simulations and interconnect analysis that matches SPICE accuracy are deployed. Optimizations that use extracted information from routing and statistical timing are emerging.

H. Clock Tree Synthesis

In the early days of physical synthesis, clock tree synthesis could occur outside the core flow. It could be inserted after timing closure, followed by incremental optimization to clean up any new paths created via clock skew.

Today, to conserve clock tree power, latches can be placed into local clumps so that there is minimal wire and skew at the final stages of the clock tree. Further, useful skew trees have become commonplace. One could potentially fix critical paths through adjustments on the clock tree as well as on the logic of the critical paths itself. The future will likely see richer sets of incremental clock tree synthesis as well as non-tree clock topologies such as grid-trees to reduce variability.

IV. From Adolescence to Adulthood

Over the next decade, physical synthesis will grow from an adolescent into an adult and continue to evolve. In the next generation of physical synthesis, it will be possible to drive routers to be timing aware, to put critical nets and connections onto preferred wiring planes, and to select preferred wire topologies. Wire synthesis is also motivated by signal integrity issues that can cause timing and functional fails in a circuit. Routers are being driven into an incremental functional footprint so that they can be driven by higher level optimization algorithms. In this new phase of physical synthesis, one can expect to see timing driven routing and re-routing, incremental re-placement and re-routing, and re-routing to avoid signal integrity issues. In summary, we see that the industry has accomplished much in the area of placement based physical synthesis. Going forward we see that there is much to be gained by integrating wiring into the process.

References

[1] C. J. Alpert, S. K. Karandikar, Z. Li, G.-J. Nam, S. T. Quay, H. Ren, C. N. Sze, P. G. Villarrubia, and M. C. Yildiz. Techniques for fast physical synthesis. *Proc. of IEEE*, 95(3):573–599, March 2007.

[2] Ivan Sutherland, Robert F. Sproull, and David Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kauffman, 1999.

[3] L. Trevillyan, D. Kung, R. Puri, L. N. Reddy, and M. A. Kazda. An integrated environment for technology closure of deep-submicron IC designs. *IEEE Design and Test of Computers*, 21(1):14–22, Jan-Feb 2004.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection produce routable placements. In *Proc. ACM/IEEE DAC*, pages 477–482, 2000.

[5] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. ACM/IEEE DAC*, pages 269–274, 1998.

[6] G.-J. Nam, C. J. Alpert, P. Villarubbia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and benchmark suite. In *Proc. ISPD*, pages 216–220, 2005.

[7] G.-J. Nam. ISPD 2006 placement contest: Benchmark suite and results. In *Proc. ISPD*, page 167, 2006.

[8] Natarajan Viswanathan and Chris Chu. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proc. ISPD*, pages 26–33, 2004.

[9] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia. Diffusion-based placement migration. In *Proc. ACM/IEEE DAC*, pages 515–520, 2007.

[10] Pete Osler. Placement driven synthesis case studies on two sets of two chips: hierarchical and flat. In *Proc. ISPD*, pages 190–197, 2004.

[11] Min Pan and Chris Chu. FastRoute 2.0: A high-quality and efficient global router. In *Proc. ASPDAC*, pages 250–255, 2007.

[12] Chris Chu and Yiu-Chung Wong. FLUTE: Fast lookup table base rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE TCAD*, 2007.

[13] Min Pan and Chris Chu. IPR: An integrated placement and routing algorithm. In *Proc. ACM/IEEE DAC*, pages 59–62, 2007.

[14] Jason Cong and Kwok-Shing Leung. Optimal wiresizing under the distributed Elmore delay model. In *Proc. IEEE/ACM ICCAD*, pages 634–639, 1993.

[15] J. P. Fishburn. Shaping a VLSI wire to minimize Elmore delay. In *Proc. European Design and Test Conference*, 1997.

[16] Charles J. Alpert, Anirudh Devgan, John Fishburn, and Stephen T. Quay. Interconnect synthesis without wire tapering. *IEEE TCAD*, 20(1):90–104, January 2000.