# Design Systems Evolution and the Need for a Standard Data Model

John Darringer
IBM T J Watson Research Center
Yorktown Heights, NY
and
Joseph Morrell
IBM Microelectronics
East Fishkill, NY

**Abstract**
*Design systems have evolved over the years from batch processing systems to highly integrated suits of tools sharing design data in memory. To support today's and future chip designs, a user must have such a tightly-integrated system and is forced to purchase the entire system from a single vendor. Small EDA companies must support many proprietary interfaces to market their new tools. To enable choice in purchasing tools, to promote innovation by lowering the barrier to entering the EDA market and to accelerate the transfer of innovative research into production use, the industry needs a standard data model and a common API for accessing design data in memory. The requirement is more than a "nice to have", it is essential for designers to keep pace with technology and will only be met with strong user support.*

## 1 Introduction

Silicon technology has had a truly remarkable record of progress over the last fifty years from chips with a single gate to today's system-on-a-chip designs with a capacity of nearly 100 million gates. Design systems too have evolved from initially just keeping track of component interconnections to automatically generating gates for chips that satisfy constraints on area, power, performance, noise and yield [1]. IBM began developing design systems in the 1950's to support its designer's use of its most advanced technology. As technology advanced and designs became more complex, design systems needed to consider the growing number of issues confronting designers. Moreover, the historically sequential process of optimizing one factor at a time has been replaced with *tightly-coupled* applications that work together to simultaneously consider multiple factors.

Today, IBM still builds its own design systems, but tries to take advantage of vendor tools when ever possible. In the past the task of evaluating and integrating a vendor tool took time, but was helped by the "standard" file formats. But, we can no longer afford to have files between key design system applications. They must work with shared design date in memory through an application program interface (API). The lack of a standard here means that we can rarely afford the time and effort to evaluate a new tool from an EDA vendor or a prototype from a university.

For design systems to keep pace with technology they must continue to evolve. The need to tightly couple critical analysis and optimization applications through shared memory will continue to grow. Without a standard way of connecting these new applicators there will be no way to combine the best algorithms from different suppliers. The result will be a market with a few very expensive design systems each with a subset of the best applications.

**Design Systems in IBM - The Early Years**
IBM built its first design system in the late 1950s [2]. It

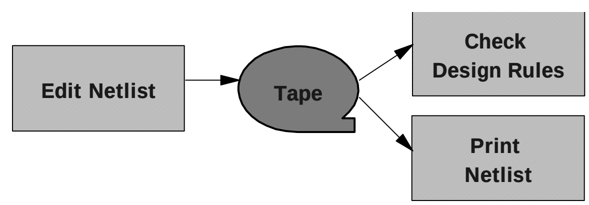



Figure 1 - 1950s Design System

used a central data base, stored on tape, to manage the logic diagrams for its 9000-circuit 7094 computer.
In the 1960s the 3033 machine had a CPU of 90000

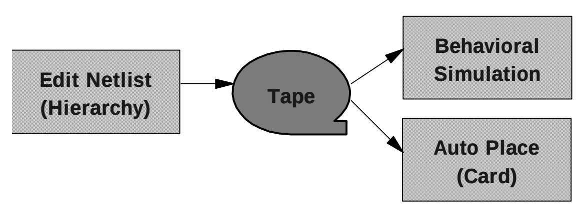



Figure 2 - 1960s Design System

circuits and hierarchy was introduced to handle the growing design information. A behavioral simulator was added to help confirm correct operation and an automatic placement tools was added to position chips containing a few gates on cards. The data base was still maintained on tape.

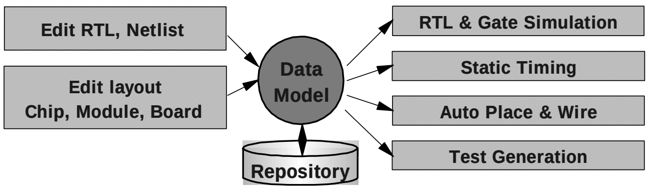



Figure 3 - Engineering Design System

In the late 1960s and early 1970s a more integrated Engineering Design System emerged (EDS) [3]. It provided a random access data base stored on disk along with an early data model that allowed key applications to share data in memory.

EDS maintained both logical and physical data including layouts for chips, cards, boards and cables. Register-transfer level simulation was introduced to handle larger design, such as the 3081 with its 460K circuit CPU. Automatic placement and wiring was provided for the 704 circuit chips as well as the 100-chip water-cooled modules and the 8-module boards. Static timing analysis was used to verify the systems performance, and automatic pattern generation was used to produce vectors from test in manufacturing. EDS was expanded and used throughout the 1970s and 1980s to support all of IBM's advanced hardware products.

**Tight-Coupling Begins**

In the 1980s, IBM pioneered the production use of logic synthesis by automatically generating most of the chips for its large mainframe systems. Hundreds of chips were processed using a sequential flow with full-chip synthesis followed by static timing analysis. The output of timing was used to revise constraints for another full-chip synthesis. This process usually converged in a few iterations with a satisfactory implementation. But as chips became larger and more complex, runtimes became a concern. At this point *Incremental Timing Analysis* was developed and integrated with synthesis [4]. Synthesis transformations could then consider a small change to the chip logic and calculate the impact to timing and then decide to commit the change or back it out. The new timing analyzer would only recalculate timing for the effected components. This advance not only greatly improved the time required for closing timing, but served as an example for future design system evolution.

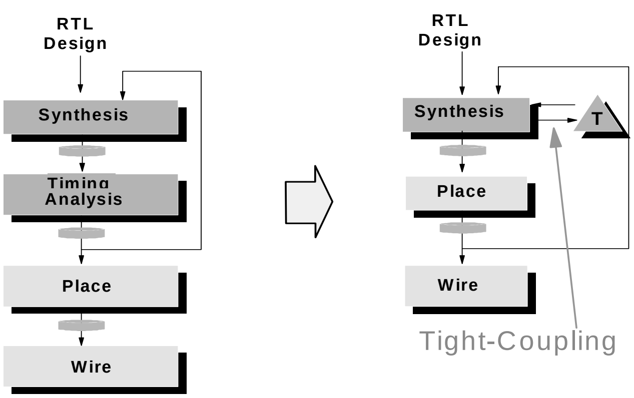



Figure 4 - Initial Tight Coupling 1980s

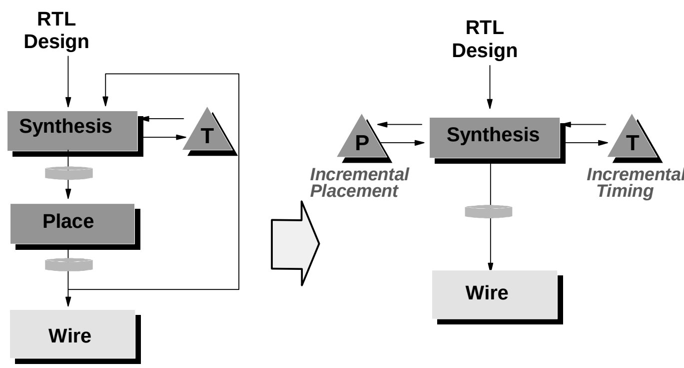



Figure 5 - More Tight Coupling

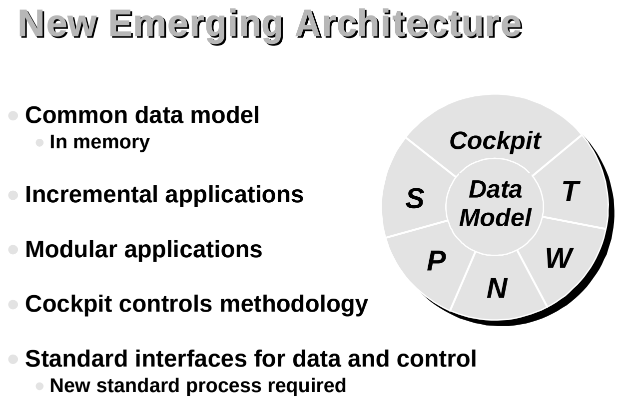
During the 1990s, as chip designs continued to grow in size and interconnect delays became a larger portion of the system cycle times, placement started to have a larger effect of performance. For some million-gate designs the iteration between synthesis and placement became a concern and *Incremental Placement* was integrated [5]. Now a change could be considered to the logic and its placement with excellent prediction of the impact on final chip performance.

These new incremental tools were completely new applications written to operate on a representation of the design stored in memory. Extensions were required to the data model to make each advance possible and the design of the data model was key to the overall improvement in overall processing times.

The movement of more analysis and optimization tools into the tightly-coupled cluster continues as new design issues emerge. For large complex chips placement alone is not sufficient to accurately predict performance. Wire routing information is needed to determine path lengths more precisely and to identify potential noise problems. Thus we now see Incremental Global Routing joining the cluster with Incremental Noise

Analysis close behind and manufacturability in the wings.

**New Design System Architecture**




It was evident that a new design system architecture was emerging. Advancing technology and more complex designs were forcing designers to deal with a ever increasing list of issues. For design systems to keep up, they too would have to deal with these issues and could not afford to treat them sequentially with files separating analysis and optimization applications. We have had a lot of experience developing central data models with supporting repositories and knew the dangers of too much centralization. Most well know algorithms depend on a specific data representation for optimum performance and no one representation is best for all. In the development of IBM's Integrated Data Model (IDM) its designers strived to share the minimum information between applications, while avoiding unnecessary conversions. There are still many cases where an analysis application will copy data from IDM to its internal data representation, perform its analysis and then put the results back in IDM. But, in these cases the benefits of looking at only the changed data and of avoiding file reads and writes overwhelm the cost of data copying. This careful analysis and justification is required in deciding which applications to tightly couple.

Many other applications, such as simulation and test generation access data from IDM, but are not tightly-coupled. They benefit from direct access to the most current design information but do not need to participate in design optimization - yet.

**The Need for a Standard Data Model API**

No one company can develop all the ideas and tools needed to keep design automation on track to support the continued advancement of silicon technology. To remain competitive, designers must have ready access to the latest advances, no matter what university or company they come from. The movement to tightly-coupled design systems and today's use of proprietary data models greatly compounds the task of connecting and evaluating a new tool. IBM began advocating a standard data model interface in the early 1990's. We supported Sematech's efforts to define CHDStd and have been actively participating in the OpenAccess Coalition. In particular, we have been contributing to the OpenAccess architecture and planning for an OpenAccess interface on IBM's Integrated Data Model.

**Engagement with OpenAccess**

When Sematech cancelled it's Design Thrust activities in the late 90's, it thankfully recognized the potential significance of the CHDStd effort that it had been supporting and arranged for the transfer of this effort to Silicon Integration Initiative (Si2) [6] in an attempt at keeping this momentum of establishing an industry standard data model alive. Most of the companies that had been supporting CHDStd also understood its potential and agreed to transfer that support as well. At about the same time, Cadence Design Systems recognized the strong customer demand for a standard and made an "open source" offer of its emerging database technology, known as Genesis. This was the birth of what is now known as OpenAccess (OA) and the OpenAccess Coalition (OAC) [7]. IBM, as a significant player in the CHDStd effort, participated in this transfer and has therefore been involved with the OAC since its inception. In fact, it shares the two "joint chief architect" positions with Cadence on the OAC Change Team, the technical body changed with overseeing the evolution of the standard.

Still, IBM recognizes that simple participation in the OAC will not achieve its objective of being able to "plug and play" its internal tools with commercial and university tools. This level of interoperability is required to provide the "best of breed" support of its semiconductor technologies, which is the fundamental reason that IBM has been supporting these activities for so long. Therefore, we have established a two-phase plan for migrating our internal tools towards the industry standard.

The first phase is actively under way and involves building in-memory translators between OA and IDM. There are three objectives for this phase of development:

1. We must determine how well OA supports IBM technologies. It is clear that IDM supports IBM's technologies today and during the implementation of these translators, we can also determine how well OA supports the IDM constructs and, as a by-product, infer how well OA supports the underlying IBM technology modeling requirements.
2. We must understand how well IDM is aligned with OA. This is a pre-requisite to the next phase of the migration. Our experience has shown that the best

way to address this at a detailed level is by implementing such a set of translators.

3. Although the end goal of our migration is to enable tight integration with commercial and university tools, the OA⇔IDM translators will at least allow the launching of the IBM capabilities as “point tools” within an OA based flow.

Depending on what we learn during the first phase of the migration, we will move on to the second phase of the migration which is to implement an IDM “compatibility layer” on top of the OA interfaces. This is also a technique that has been successfully used within IBM in the past for similar purposes, that is to migrate separate collections of tools to a single data model by providing a thin layer of code which dynamically translates the data presented by the underlying interfaces to match the interfaces used by the tools being migrated, on an object by object basis. Obviously, the outright feasibility, as well as the end performance, of providing such a layer will be dependent upon how closely aligned the two sets of interfaces are. The expectation is that, for the most part, this will not be a significant concern. For those cases where it is a concern, we have three options:

1. We can propose OA extensions through the OAC Change Team process to enable features not currently supported by OA.
2. We can use the robust OA extensibility features to extend OA for our own use. (As an aside, these extensibility features may also be useful in combination with the first option to “prototype” proposed extensions before making them a formal part of the standard.)
3. We can actually change the IDM modeling to better align with OA which, of course, may require an en masse migration of our tools.

Regardless, we may continue to maintain the native IDM implementation for a period until running with native OA tools becomes the normal mode of operation.

**Opportunities for EDA Companies and Universities**

Given the growing adoption of OpenAccess, it makes sense for EDA companies and Universities to develop their tools and prototypes on the OpenAccess API. First, they can take advantage of a highly efficient production data base to jump start their development. The extensibility features of OpenAccess enable experimentation with new data types where needed. Second, when their tools or prototypes become ready, it will much easier to integrate them into a customers environment for early evaluation and feedback. Neither the EDA company nor the customer need to waste time developing translators that also decrease effectiveness.

Finally, given the availability of source code for the reference implementation of the API, means that the users of Open Access have much more control of their product development or research direction.

**Summary**

Design systems have evolved over the years from batch processing systems to highly integrated suits of tools sharing design data in memory, all to support high-performance chip designs. Today, the many vendor-specific proprietary data models force a user to purchase such a tightly-integrated system from a single vendor and they make it difficult for small EDA companies to enter the market. The industry needs a standard API for design data to enable choice in purchasing tools, to promote innovation and to accelerate the transfer of innovative research into production use. The requirement is more than a “nice to have”, it is essential for designers to keep pace with technology and will only be met with strong user support. The OpenAccess coalition provides a forum for developing such a standard to meet the needs of chip designers and to enable innovative tool development by EDA companies and Universities for the future.

**References**

[1] J.A. Darringer, John Darringer, Evan Davidson, David Hathaway, Bernd Koenemann, Mark Lavin, Joseph Morrell, Khalid Rahmat, Wolfgang Roesner, Erich Schanzenbach, Gustavo Tellez, Louise Trevillyan, “EDA in IBM: Past, Present and Future”, IEEE Trans. On CAD, Vol. 19, No. 12, Dec. 2000, pp. 1476-1497.

[2] P.W. Case, H.H. Graff, M. Kloomak, “The Recording Checking and Printing of Logic Diagrams”, Proceedings of the Eastern Joint Computer Conference, Philadelphia, PA, 1958, pp. 108-118.

[3] P.W. Case, M. Correia, W. Gianopulos, W. R. Heller, H. Ofek, T. C. Raymond, R. L. Simek, C. B. Stieglitz, “Design Automation in IBM”, IBM Journal of Research and Development, Vol 25, No5, Spt. 1981, pp 631-646.

[4] R.P. Abato, A.D. Drumm, and D.J. Hathaway, L.P.P.P. van Ginneken, “Incremental Timing Analysis”, U.S. patent 5,508,937, 16 April, 1996.

[5] Hojat, Villarrubia, "An integrated synthesis and placement approach for timing closure of PowerPC microprocessors", International Conference on Computer Design, 1997.

[6] si2.org

[7] si2.orb/openaccess