CHDStd - APPLICATION SUPPORT FOR REUSABLE HIERARCHICAL INTERCONNECT TIMING VIEWS

S. Grout
G. Ledenbach
R. G. Bushroe
P. Fisher
Amrich Chokhavtia
SEMATECH
2706 Montopolis Dr,
Austin, TX 78741, USA

D. Cottrell
D. Mallis
Silicon Integration Initiative,
4030 W. Braker Ln, Suite 550
Austin, TX, 78759, USA

S. DasGupta
IBM Corporation
11400 Burnett Rd
Austin, TX 78758, USA

J. Morrell
IBM Corporation, B/334-2,
East Fishkill, New York, USA

ABSTRACT

This paper describes an important new facility for timing-driven design applications within the new CHDStd standard for a SEMATECH design system for large complex chips. We first review EDA requirements for CHDStd hierarchy for large complex leading edge chips and current EDA problems in accurately and efficiently handling complex interconnect. We then describe our approach for fully-reusable hierarchical interconnect timing views in support of timing driven design for 0.25u technologies and below. The result is a method which builds on SEMATECH's new controlled error parasitic timing calculation capability for deep submicron, providing means for compactly storing and reusing accurate hierarchical timing views for 28M to 100M transistor chip designs.

1. INTRODUCTION

SEMATECH is a consortium focusing on advanced semiconductor manufacturing technology for its multi-national Member Companies. As part of its overall activities, SEMATECH is developing a new chip design system to enable large multi-team chip design groups to design large complex chips in 0.25-micron to 0.18-micron technologies and beyond. The system is called the Chip Hierarchical Design System (CHDS). The goal of CHDS is to provide flexible design methodologies for timing driven logical and physical design of complex microprocessors and advanced high-end ASIC's. As part of the CHDS development, SEMATECH in partnership with Silicon Integration Initiative (SI2) is developing a new high-performance hierarchical design representation standard, CHDS Technical Data (CHDStd).

The 1994 National Technology Roadmap for Semiconductors (NTRS) [1] identified the crisis in complexity of products implemented as chip designs caused by shrinking feature size, increasing functionality and complexity, designed by increasingly larger design teams. In order to mitigate this increasing complexity and size, both CHDS and CHDStd have requirements [2, 3, 4, 5, 6, 7] to describe these large chip designs hierarchically, provide significantly-improved timing driven design capability, and provide for greatly improved processing of design by EDA tools.

2. CURRENT PRACTICE AND PROBLEMS

In today's EDA ASIC systems [8], the practice dominantly is to handle parasitics as lumped values of capacitance and trapezoidal time delay. Per the NTRS, for feature sizes of 0.25u and below, effort must be made to be increasingly more accurate in identifying and using both net parasitics and timing. Both R-L-C parasitic models of interconnect and resulting time delay must be calculated from accurate treatment of the fields of both target nets and their surrounding 'threat' nets. This calculations must be accomplished with controlled variable accuracy and is the subject of separate papers.

3. NEEDS AND CHALLENGES

One could simply go ahead and solve these interconnect problems using today's approaches. However, there are several key factors that lead us to develop and productize some new approaches. Without taking care of these factors, it is highly unlikely we will have a viable set of solutions and that the CHDS system will not be useful for SEMATECH Member Company chip designs.

1. Chip sizes are going to 100M+ transistors. We need approaches to make sure design data within an EDA system stays within a reasonable size. Both creating, storing, moving, and reusing design data must be handled so that the data set sizes continue to allow reasonable run times and hardware resource requirements.

2. Signal coupling between nets can no longer be handled via conservative design rules. In fact, we need to move quickly to handle nets that are routed over the top of and through blocks, cells, and macros in fairly arbitrary ways.

3. A number of contemporary approaches to interconnect parasitics continue to require the use of populating completely flattened design data sets prior to, during, and after parasitic extraction and timing calculations. To support parasitic calculations within a large multi-team design group that may be geographically dispersed, we will have to store extremely large data sets and moving them around the world as part of a single overall chip design process. This will be a huge problem in terms of both run time and data storage.

4. Approaches to handling interconnect parasitic calculations are needed that would support the storing of net, netsegment, via, and ports (aka pins of used blocks, cells, transistors, and macros) within a design hierarchy such that parasitics can be calculated and recalculated at an algebra-like computational

speed. This task is typically done today using occurrence level (aka flattened) design data. An approach needs to be identified and verified that supports the hierarchical storage and accurate reuse of deep submicron (DSM) parasitics. This approach will need to accurately and efficiently support rapid and accurate calculation of the most complex coupled interconnect parasitics.

5. To get closure of performance of a large complex leading-edge chip design, the entire design system must be much more timing driven within a central delay architecture. Chip design timing closure is a key requirement of CHDS.

6. While there are other needs that must be met in this area in order to have a viable timing driven design system for large complex 0.25u to 0.18u chips, the above are some of the most important ones. If we can find a solution for properly handling timing views in a reusable way within a folded design hierarchy, we will be able to handle 0.25u and below interconnect coupled parasitics persistently in a folded hierarchy.

The main need for accurate parasitics is to create accurate timing. Once we get that timing, we, to a large degree, have the information we need to move forward with timing-driven design. This means, though, that we may later need to recalculate the parasitics (or bear the considerable cost of storing them fully flattened as occurrence data) so we can later reexamine loading, drive, and related design constraints as we make incremental design changes.

1. APPROACH

We now describe our approach for handling the net delay timing data that results from the above accurate net parasitics. In addition to simply providing persistent storage of accurate timing for later use by EDA applications, we also provide an approach to correctly handle compact storage of such net timing within an instance (aka folded) design hierarchy in such a way that the timing data is fully reusable. That is, once timing values are found for the definition of a block, cell, macro or core within a design hierarchy, that set of timing values must be correctly stored as a hierarchical timing view so it can accurately used and reused in the instantiation of that block (i.e., use of the block within higher level blocks), regardless of the circumstances.

Impact of Hierarchy

A formal approach is used within CHDStd to define and describe a design hierarchy. This approach seeks to completely describe all aspects of a large complex chip design in terms of a folded or instance hierarchy. It is critical that CHDS and CHDStd uses a hierarchical representation approach in order to counteract the increasingly large chip design sizes (28M transistors going to 100+M transistors). That is, the growth of populated CHDStd data must grow no worse than approximately the log of the number of transistors inferred by the design hierarchy. Design hierarchies populated in the past have resulted in 1M to 10M transistor chip designs requiring typically around 60,000 populated instances (i.e., unique number of usages of block definitions within the folded hierarchical design). Applicable definitions of aspects of CHDStd hierarchy are included in the references included with this paper. The intent of those definitions is that they support both complete definitions of chip designs and are consistent with definitions of designs used within other modern hierarchical system representations such as VHDL.

We first focus on properly describing interconnect delay timing within the CHDStd hierarchical representation. To do this correctly, our goal is populate net timing data that is based on appropriately-calculated coupled interconnect parasitics [9] such that we separate timing that is independent of how a block is instantiated or applied from timing that dependent on how a block is instantiated. The following summarizes key concepts of hierarchy and timing we need to use for our approach:

Block Internal Definition

Here we include timing on nets and netsegments (aka 'subnets') which are included entirely within the block. That is, these nets have no connection to ports on the block interface nor coupling to nets on the block interface or to nets outside of the block.

The timing of these nets does not change regardless how the block is instantiated. If the net timing is in any way dependent on how the block is instantiated, then that timing has to be handled as described below. That is, instantiation-dependent nets need to handled as part of the block instance's interface definition.

Block Definition Interface

Here we include the timing on nets and netsegments which are connected to ports of the block definition interface. This means these nets may also be connected to nets outside of the block (into the next level of the design hierarchy.)

The timing on these nets should be populated with timing without regard to their termination or coupling to other nets. Depending on the technology, this timing would normally correspond to the timing of the 'Unterminated' net, i.e., the timing that results without any outside parasitic load or coupling to some other net.

Block Instance Interface

Here we include what we term the 'Differential Net Timing'. This timing is calculated by taking the difference between the timing for the open-circuit unterminated delay and the actual instance-terminated and coupled timing. Both overall interconnect delay and slew should be treated this way. To store this in the design hierarchy properly, this incremental timing is then associated with the port of the block instance interface (aka Port Instance) to which the net or netsegment is connected.

Note that the Block Instance Interface data is populated as part of the next higher level of the design hierarchy i.e., within the parent block's containing block internal definition and its instances, and not as part of the current block's definition

Where there is coupling to nets outside of the block, we propose to add additional ports to the Block Instance Interface that specifically identifies that coupling. We, therefore, refer to these particular ports as 'Coupling Ports'. The efficacy of this last mechanism is still under consideration and debate within the CHDS Program and we will present out results with the final paper. With these mechanisms in place, we now can populate the Block Instance Ports with appropriate Differential Net Timing information.

2. HIERARCHICAL TIMING VIEWS

However, with the above steps of this method, there are netlist interconnect patters for which this approach is not fully accurate (e.g., complex fanout and feedback within a block), so that currently this method is only useful for some cases and is the challenge to be solved later. We therefore turn to the use of storing this complex hierarchical interconnect data in the form of timing arcs of hierarchical timing views.

This same approach has application not only to individual net and netsegment hierarchical timing, but also to creating, handling, and reusing timing views.

Current Practice - The current practice is to create a block timing view for each Occurrence of the block, that is, for each individual use or copy of the block. This means EDA systems typically today either recreate timing views each time they are needed by other EDA applications or timing views are created and stored for each occurrence of each block.

Hierarchical Timing Views - Our approach for reusable hierarchical timing is superior to the above since we create persistent timing views within a folded design hierarchy for each block internal definition and block instance interface. This approach does require creation of a timing view instance interface 'shell' for the actual net timing for those nets of the block which are connected or coupled to nets at the next level of the hierarchy. These instance interface shell's are, therefore, simply a correct description of the timing for next level's block instance ports per the environment in which the block is used.

Block Internal Definition Timing View - The approach we use here is to again make use of the concept of separating instance-specific and instance-independent timing arc information. We therefore create a timing view for the block internal definition with timing arcs for all of the internal nets and block instances.

Block Instance Interface Timing View - We separately then create a timing view for the block instance interface definition, with arcs for each of the paths defined by the interconnect to the boundary of the block internal definition timing view. This latter timing view is attached to the block instance definition as part of the higher-level containing block definition.

Reusable Timing Views - With the above mechanisms defined, in place, and with timing data populated at each level of the design hierarchy, EDA tools needing accurate timing can walk the design hierarchy, and pick up and (re)use timing data regardless how each block within the hierarchy is used.

3. IMPLEMENTATION AND RESULTS

Figure 1 shows a a typical complex mixed logical and physical hierarchy chip design example. The coupled RLC parasitic models and controlled accuracy net and subnet timing are calculated using CHDS capabilities [9] which we will not describe here except to note that those coupled parasitic models and net timing are currently saved persistently within an expanded-hierarchy occurrence model. The nets and subnet effects and timing may include accurate timing data that results from routing over blocks.

In this example, block definition A is used at both location A1 and A2 but with quite different wiring at the parent block level, including a route over the A block which penetrates further down the hierarchy into F1(F). F1 itself has net to net coupling. Note that in block F1(F), the SIG3 subnet will have instance specific timing as it connects up to A1(A) via its port 'a' and on into J1(J) port 'b'. Note also that net, SIG1, within F1(F) is a net within the block internal definition and has coupling with the local net, 'SIG3'. Note that only the net, REG1, is an internel net which is instance independent.

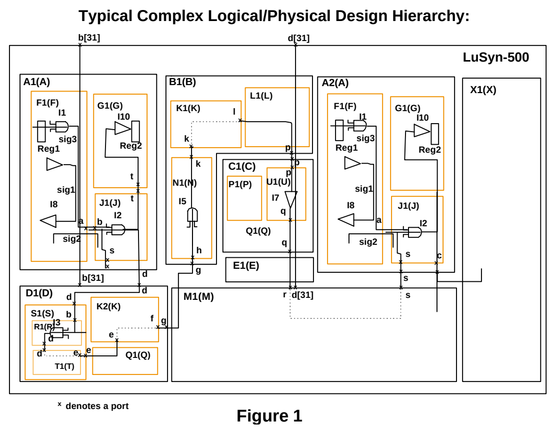
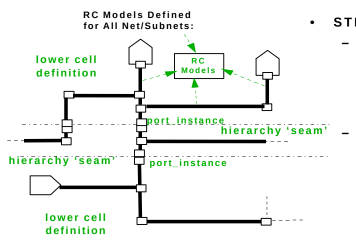


Figure 1

We now show a set of figures for a complex hierarchical net fragment to illustrate our approach.

Figure 2 show the interconnect fragment and the creation of RC parasitic models for the hierarchical pieces of the interconnect.

REUSABLE Subnet Delay

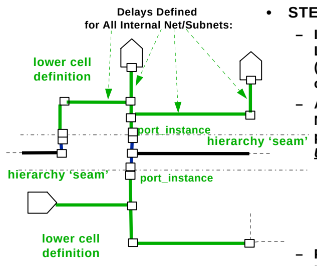


- STEP 1:
  - Establish Accurate RC Parasitic models for each aspect of the entire hierarchical net.
    - Store in CHDStd
  - Then - Identify all the Delays for a Cell's Internal definition that are instance independent.

Figure 2

Figure 3 shows the calculated hierarchical subnet delays for block definition interface. These are the delays which are instance-specific since the hierarchy above in part determines their value. As a key starting point, we also focus on determining the unterminated port delays.

REUSABLE Subnet Delay

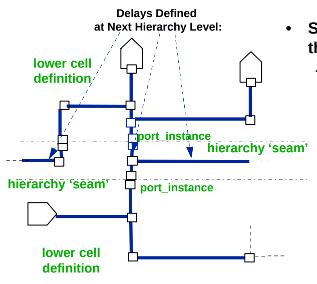


- **STEP 2:**
  - **Identify Delays for Nets that Lead-to (terminate) on a (external) Port_definition leading off the Block/Cell:**
  - **Approach - Calculate the Net/Subnet Delay to the port_definition with the port *UNTERMINATED***
    - **This is the Delay of the Subnets connected to (External) port_definitions of an Uninstantiated cell/block definition.**
  - **Populate CHDStd with those Interface Delays, including only the Unterminated_port Delays.**

Figure 3

Figure 4 shows the delays in the block internal definition of the parent cell. Their values in this case are instance independent. They however do connect to the block instance interface, so that their resulting values must produce the pathwise subnet delay values accurately working from CHDStd persistently stored timing values.
Figure 4 show the process then of calculating the overall hierarchical net timing for both the local instance independent and lower-level instance specific delays.

REUSABLE Subnet Delay

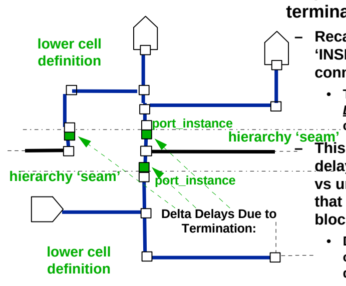


- **STEP 3: Identify the Delays at the Next Hierarchical Level:**
  - **At the Next Higher Level of Hierarchy,**
    - **Identify the Delays of the Nets/Subnets of that higher level Cell's internal_definition**
    - **Including the Delays for the Net/Subnets that Lead-To/Terminate on each Port_instance (ports that are connected to down into the instantiated cell/block.)**

Figure 4

Figure 5 shows the process of dealing with the now terminated delays and populating that timing information with the instance ports. In this way, we have a method for dealing with many instance specific timing situations.

REUSABLE Subnet Delay



- **Step 4: Find the Delta delay due terminating the Instance:**
  - **Recalculate the Delay for Nets/Subnets 'INSIDE' the instantiated cell/block that connect to these ports.**
    - **This terminated delay will normally be *Different* than the unterminated Delay calculated for the cell/block definition.**
  - **This identifies 'Delta' or 'Incremental' delay relative to the instance terminated vs unterminated subnet connected to that port within the instantiated block_definition.**
    - **Difference in the instance-specific delay of the net portion in that local area of the design hierarchy.**
    - **Subnet delay just inside the lower-level Block definition Step**

Figure 5

Figure 6 shows the process of using the accurate subnet timing found by separate CHDS coupled parasitic extraction and timing calculation tools and creating the two types of timing views that make up the overall reusable hierarchical timing views. The rssulting timing view portions are stored persistently in the CHDStd folded design hierarchy for a particular chip design.

- **Step 5 - Construct Hierarchical Timing Views**
  - **First Construct a Timing_View of timing Arcs for Only the Internal_definition timing**
  - **Then Construct Timing View of Timing Arcs for Block Instance Interface**
    - **Part of Instance of Containing Block Definition**
  - **This gives us Accurate ReUsable Timing Views!**

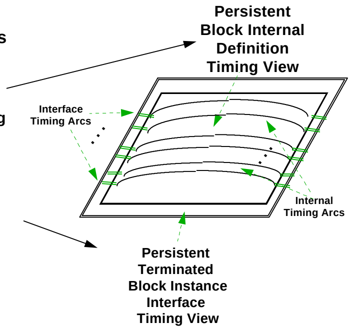
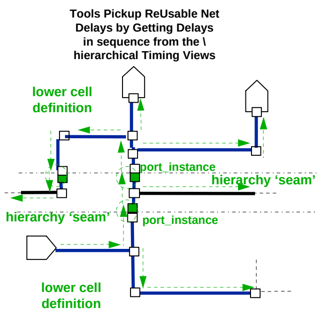


Figure 6

Figure 7 shows then the use of the stored reusable hierarchical timing views to later provide timing values to CHDS and other CHDStd compliant EDA application tools for all of the subnets of a hierarchical net.

REUSABLE Subnet Delay



- **Step 6: Later - Using the Saved Incremental Delay:**
  - **Tools, such as static timing analyzers, picks up, in sequence, each Subnet delay, plus these instance-specific-offsetting incremental delays without Recalculating Delays from the RCs.**

Figure 7

The above approach for reusable hierarchical timing views is currently under consideration by the CHDS development teams for incorporation into the CHDS tools. Several in-depth analyses and team reviews have been made. We expect to also present a far more detailed complex example using data from some early CHDS system, tool, and testing results by the time the paper is finalized for the ISPD-E 98 Conference.

However, we already know from the results and metrics obtained in earlier hierarchical design system implementations and CHDStd-related hierarchical EDA work, that the data population required for this approach is significantly less than handling timing using a fully flattened occurrence approach. That is, for timing views and for the same example chip designs mentioned earlier in this paper, we would populate only 60,000 block-instance timing views as an upper bound. This is in stark contrast to populating timing for all the hierarchical nets interconnecting the 5M to 10M transistors that comprises the actual chip.

4. CONCLUSIONS

It is startling that the key to far greater data capacity and design flexibility in handling design data stems from the simple idea of defining hierarchical timing views in terms of 'open circuit delay' and 'actual-termination delay'! However, that is precisely what this approach is based on. This may be an indication there is difficulty working out how to utilize hierarchy for other design representation requirements and EDA applications. This paper hopefully will provide a key to working out approaches for other key deep submicron problems, including handling of reusable R-L-C complex coupled interconnect parasitic timing within the folded instance design hierarchy, a solution that will soon be needed.

ACKNOWLEDGMENTS

We are grateful for the guidance and comments we received the rest of the members of the CHDS and CHDStd development teams, including members from Synopsys, SI2, IBM, Cadence CCT, Lucent, OEA, and Ultima. In particular, the Lucent CHDS team members have provided several challenging test case examples that helped to fine tune our approach.

REFERENCES

[1] The National Technology Roadmap for Semiconductors, pp 5-43. Semiconductor Industry Association (SIA), 1994

[2] J. Sayah, R.Gupta, D. Sherlekar, P. Honsinger, J. Apte, S. Bollinger, H. Chen, S. DasGupta, E. Hsieh, A. Huber, E. Hughes, Z. Kurzum, V. Rao, T. Tabtieng, V. Valijan, D. Yang. Design Planning for high-performance ASICs, *IBM Journal of Research and Development*, Vol. 40 No. 4, pp. 431-452, July, 1996

[3] R.G. Bushroe, S. DasGupta, A. Dengi, P. Fisher, S. Grout, G. Ledenbach, Nagaraj NS, R. Steele. Chip Hierarchical Design System (CHDS): A foundation for Timing Driven Physical Design into the 21st Century, to be published in the *International Symposium on Physical Design* (ISPD97), April 13-15, 1997

[4] Requirements Document - SEMATECH ECAD Program CHDS Technical Data (CHDStd) - Requirements for a Chip Hierarchical Product Design Specification Needed to Support 0.25-micron Chip Design. SEMATECH, October 6, 1995

[5] Chip Hierarchical Design System - Integrated Hierarchical Forward Timing Driven Electrical and Physical Design Requirements for 0.25-micron Chip Design, Version 1.0. SEMATECH, February 21, 1996

[6] An Integrated Data Model for Hierarchical Design Assembly, May 31, 1996, published jointly by IBM and CFI - www.cfi.org/CHDStd/infomodel

[7] CHDStd Information Model Documentation, April 10, 1996, published by CFI -www.cfi.org/CHDStd/chdsIndex.html

[8] EDA Industry Standards Roadmap for Design and Test, Version 1.0, published jointly by EDAC, CFI, and SEMATECH, January 15, 1996

[9] V. Chandramouli, Ram Swaminathan, Chi-Yuan Lo, Nagaraj NS, Jesse Lu, Wai-kai-sun, Don Cottrell. An Integrated Approach for Net Parasitic Extraction, Tau-97 Interconnect Workshop, December 4-5, 1997

Authors and Affiliation:

S. Grout
G. Ledenbach
R. G. Bushroe
P. Fisher

SEMATECH
2706 Montopolis Dr,
Austin, TX 78741, USA

D. Cottrell
D. Mallis

Silicon Integration Initiative,
4030 W. Braker Ln, Suite 550
Austin, TX, 78759, USA

S. DasGupta

IBM Corporation
11400 Burnett Rd
Austin, TX 78758, USA

J. Morrell
J. Sayah,
R. Gupta,
PT Patel, P. Adams

IBM Corporation, B/334-2,
East Fishkill, New York, USA