

# A Natural Model for EDA Sensible IP Behavior

Mark Fredrickson, IBM Corporation  
Dan Notestein, SynaptiCAD  
Melanie Yunk, Silicon Integration Initiative, Inc.

## Introduction

The Timing Diagram Markup Language (TDML) is being developed by an Electronic Component Information eXchange (ECIX) Working Group under the management of the Silicon Integration Initiative (Si2). Members of this Working Group include Chronology Corporation, Denali Software, Inc., IBM, Mentor Graphics/Interconnectix Business Unit, and SynaptiCAD.

In November of 1996, SynaptiCAD approached SI2 and asked to be part of the ECIX program in order to develop a timing diagram format as part of the ECIX electronic data sheet standard. Subsequently, SI2 issued a Request for Technology (RFT) in Spring, 1997 to address this issue. Responses were received from both Chronology Corporation and SynaptiCAD. As a result, the TDML Working Group was formed in December 1998, with the goal of developing an interchange standard for timing diagram information that would:

- Allow the customer to choose a timing diagram browser or other support tool,
- Allow the component information (and Intellectual Property) provider to choose a timing diagram editor or other generation tool, and
- At the same time, this new standard would support standalone application use while enabling adoption of timing diagram data as a strategic element of the ECIX Pinnacles Component Information Standard (PCIS) standard.

TDML is an open, industry standard language for the exchange of interactive timing diagrams. TDML allows persons to exchange interactive timing diagram information in a standard form. This standard form promotes the sharing and interchange of information between different organizations and allows interested parties to develop tools for generating, editing, and browsing, these diagrams.

TDML is currently at version 0.9.5, with TDML 1.0 scheduled for release in October 1998.

The standard resulting from this process will be designed for use by ECIX PCIS-compliant datasheets for representation of timing and waveform diagrams that describe component and intellectual property (IP) characteristics. This interchange format will be promoted as an industry-wide standard first through Si2 membership, then formally through the IEEE/IEC. The standard will be openly available to companies worldwide for adoption and use with both commercial and proprietary software tools.

## Interactive timing diagrams - the intuitive way to work with timing relationships

### *What are interactive timing diagrams?*

Interactive timing diagrams look like traditional timing diagrams, showing signal transitions with respect to time and relationships between signal transitions. However, as the name indicates, interactive timing diagrams interact with the designer in a variety of ways. For example, tools that interact with timing diagrams allow the designer to select a signal transition and move it in time. Signal transitions that are dependent on the moved edge may also move automatically. Setup, hold, and pulse-width tests are automatically recalculated and drawn to show how the timing relationships change when the transitions move.

The timing diagram shown in Figure 1. is not an interactive diagram, but if it were, a user could, for example, select the valid-time edge of the DATA signal and move it to the right, modeling a later data arrival time. This would automatically cause the margin on the setup test

to decrease. The data at output Q would become valid at a later time, moving to the right by the same amount as the input DATA.

Timing diagrams are one of the most intuitive ways to visualize and work with any kind of timing relationships. The relationships are easy to see and understand. It is no wonder that databooks, blackboards, and engineer's heads are full of them. Interactive timing diagrams are even better because they make it easy to see and understand how timing relationships change in response to changes in signal arrival-times, path delays, and so forth.

Several EDA companies market timing diagram tools that make use of interactive timing diagrams to help designers understand and control the timing of electronic designs. Some examples of these tools are the "TimingDesigner" family of tools from Chronology (<http://www.chronology.com/>) and the "WaveFormer Pro" set of tools from SynaptiCAD (<http://www.synCAD.com/>).

### ***What's inside an interactive timing diagram?***

Timing diagrams contain information about physical cause-and-effect relationships such as delays and constraints. This information is timing model information that is valid no matter how the part is used. This modeling information is contained in the relationships between signals in the diagrams (see Figure 2.). Diagrams also contain information about when signal transitions arrive at the design inputs, and when the dependent signal transitions occur at other places in the design. These signal arrival-times are usage dependent information that is contained in the transition of the signals in the diagrams.

### ***What are the benefits of interactive timing diagrams?***

Since timing diagrams contain both modeling and usage information, they can be used for both modeling and usage timing tasks. Here are some examples of how interactive timing diagrams are used:

- Using timing diagrams to help designers understand and analyze system timing behavior:

Timing diagrams are a tool used by engineers and designers universally in their attempts to understand and manipulate timing relationships between electronic signals. Timing diagrams are used to show timing relationships, identify timing problems, and analyze potential solutions. Timing diagrams are used because they are easy to understand. They show the timing information in a graphic, intuitive format that allows the user to immediately visualize the important relationships. "Interactive" timing diagrams are especially applicable to timing problem resolution because designers can "try out" different potential solutions and immediately see graphically the effects of those changes. Several popular EDA tools (as noted above) use interactive timing diagrams as a primary interface between person and computer.

- Using timing diagrams to set up timing assertions or simulation patterns:

Timing diagrams are an intuitive way to define the input signal transitions for a static timing analysis or simulation program. In a graphical manner, the user can see the clock and determine the input signals and their transition times and slopes.

- Using timing diagrams to define timing models

As mentioned above, timing diagrams are great for describing input-output relationships and timing constraints between signals. Component databooks contain these types of timing diagrams. **Interactive** timing diagrams are useful for timing models because the designer can change the input signal transitions and see the results immediately. IP vendors are interested in using interactive timing diagrams as models so that potential customers may test the components or technology right in the intended application! Some vendors are already working with EDA tool providers to offer these models.

- Using timing diagrams to visualize results from static timing analysis tools:

Many powerful timing analysis tools provide information to electronic designers in form of textual reports. These reports are loaded with good information but they are hard to digest. Designers often take information from the reports and manually

translate it into timing diagrams in order to visualize the timing relationships being described. It is no exaggeration to say that timing diagrams are the defacto preferred medium for thinking about timing relationships.

## TDML - the Timing Diagram Markup Language

TDML is an industry-backed effort to establish an open, industry standard language for the exchange of interactive timing diagrams. The language is currently under development by the TDML Working Group of the Silicon Integration Initiative (Si2) (<http://www.si2.org/>) as part of the Electronic Component Information Exchange (ECIX) project with members from component and virtual component vendor and EDA companies.

The TDML language allows persons to exchange interactive timing diagram information in a standard form. A standard form for this information promotes sharing between different organizations and allows interested parties to develop tools for generating, editing, browsing, etc. the diagrams.

### How will TDML be used?

Figure 3 and text below describe some of the ways in which TDML will be used.

1. Generating TDML with a timing diagram tool:

Anyone wishing to describe the interface timing behavior of an IP may describe that behavior with timing diagrams by using a timing diagram tool which supports TDML. The diagrams may be edited, printed, etc. as needed, and those diagrams may then be converted into standard TDML for export into other tools.

Semiconductor manufacturers and vendors will exploit this method to display their wares in an easy-to-distribute and easy-to-use method that accurately models the complex timing relationships of their products.

2. Generating TDML from other types of programs:

Companies or individuals that provide IP models of different types will want to see those models represented as interactive

timing diagrams, and will want to export those models in TDML format. These persons may use existing programs that generate models from user input, parameter databases, etc. These programs may be interfaced directly to timing diagram tools, or the programs may generate TDML which may then be imported into a timing diagram tool, so that the users may view and graphically manipulate the models, and export those models in TDML format for use elsewhere.

- A. Browsing IP specifications:

Viewers of the timing/interface behavior of an intellectual property will import TDML describing that behavior into a TDML browser that will display the interactive waveforms. Here is an example of how this scenario will be used: An engineer wants to find out whether or not IP XYZ from company ABC will work in his/her project. So, the engineer surfs over to the company ABC internet website, locates the specifications for the desired IP, and examines them with an online browser which supports TDML.

It is expected that TDML browsers will be available as "plug-ins" to common internet browsers such as Netscape(TM).

- B. Importing TDML IP interface models into timing tools:

Persons using timing diagram analysis tools will be able to download TDML IP timing models into their tools to test the IP with the rest of their design.

- C. Converting TDML IP interface models into other forms:

Tools will be available to convert TDML timing models into many other useful forms for importation into other types of timing and analysis programs.

Scenario mixing: These usage scenarios can be mixed in any desirable manner. For example one practical scenario would be "1-A" in which semiconductor vendors export TDML onto the internet describing their products, and their customers examine the IP specifications using an online TDML browser.

## What does TDML mean to the IP Provider and Customer?

If the user is a design engineer, TDML means that in the near future the design engineer will be able to select among competing, off-the-shelf EDA tools which can read and write interactive timing diagrams. The user will be able to import interactive timing models from IP vendors and from colleagues who may be using the same or different EDA programs. The engineer will be able to export timing models which will not only look good to humans, but will also work well with a variety of EDA computer tools. The user may also be able to use graphic diagram tools to define timing assertions for favorite timing analysis program, or define simulation vectors for a simulation engine. And, after analysis or simulation, the design engineer may be able to take the results into a diagram tool and view them graphically.

If the user is an EDA tool provider dealing with timing or simulation tools, the user should consider adding TDML import and export capabilities to existing tools so that they may communicate with timing models and/or signal transition models with other tools.

Static timing tools such as Motive and PrimeTime (from Synopsys) or EinsTimer (from IBM) have traditionally been functionally very powerful, but difficult to use. Two of the most difficult tasks involved in using these static timing engines are setting up the assertions correctly, and interpreting the output (usually textual reports). TDML-based tools could be used to improve the usability of these tools and add additional capabilities. For example, TDML-based tools could be used to graphically set up timing assertions for a static timing engine. Timing analysis results could be exported into a diagram-based tool for intuitive examination.

If the user (or author in this case) is an IP vendor or a provider of IP models, the author should explore the benefits of making models available in TDML form. Some of those benefits include creating models that are good for both human and computer usage, creating models that are interactive and respond to changing input conditions, and creating models that can be used by a variety of EDA tools.

## A look at TDML syntax

Here is an excerpt from a TDML instance created by SynaptiCAD's WaveFormer Pro timing diagram editor (used with permission). The following code describes the waveform and delay parameter displayed in Figure 4.

Notice when looking at TDML code that each object is contained within matching tags. For example the code excerpt below is one of the first code blocks in a TDML file. The block begins with `<TDML.ADMIN.INFO>` tag and ends with `</TDML.ADMIN.INFO>` tag. All the information between these particular tags describes the type of program used to generate the TDML file.

```
<TDML.ADMIN.INFO>
  <TOOL.INFO ID="X1">
    <TOOL.NAME ID="X2">WaveFormer
    Pro</TOOL.NAME>
    <TOOL.TYPE ID="X3">Timing Diagram
    Editor</TOOL.TYPE>
  </TOOL.INFO>
</TDML.ADMIN.INFO>
```

The next code block entitled `<SIGNAL>` describes SIG0 and its waveform as displayed in the timing diagram. The waveform consists of signal transitions or "edges" that represent a change of state. An edge (E) contains the state (S) of the signal after the transition, and the earliest time and latest time that the transition could happen (TE,TL). Edges can contain an optional ID attribute that enables the edge to be referenced by external objects.

```
<SIGNAL ID="ID2" SHOW="1" SHOW.GRID="0"
GRID.COLOR="0000FF"
  GRID.STYLE="Solid" DRAW.MIN="1"
  EDGES.PER.CYCLE="1">
  <CONN.PTR
  CONN.ID="ID1">SIG0</CONN.PTR>
  <WAVEFORM ID="ID3" LOCKED="0">
    <E ID="ID3V" DRIVEN="1" S="1"
    TE="0" TL="0"></E>
    <E ID="ID4" DRIVEN="1" S="0"
    TE="30000.0" TL="30000.0"></E>
    <E ID="ID5" DRIVEN="1" S="1"
    TE="70000.0" TL="80000.0"></E>
    <E DRIVEN="1" S="0" TE="140000.0"
    TL="140000.0"></E>
    <E DRIVEN="1" S="1" TE="200000.0"
    TL="200000.0"></E>
    <E TE="270000.000000"
    TL="270000.0"></E>
```

```
</WAVEFORM>
</SIGNAL>
```

The delay parameter in the timing diagram is a little more complicated because it references the edges contained in an external signal. The min and max values of the delay are stored in a separate element called a <TDML.CC> so that timing characteristics can be shared by multiple delay parameters.

```
<EDGE.RELATIONSHIPS>
  <RELATIONSHIP TYPE="delay"
  TDML.CC.PTR="ID6" SHOW="1">
    <TWO.EDGE SOURCE.E="ID4"
    TARGET.E="ID5">

<RELATIONSHIP.LABEL>D0</RELATIONSHIP.LABEL>
  </RELATIONSHIP>
</EDGE.RELATIONSHIPS>
<CC.LIST>
  <TITLE ID="X6">Parameter Data
  Table</TITLE>
  <TDML.CC ID="ID6" STATIC="1" SHOW="1">
    <PARAM ID="X7">
      <PARAM.SYMBOL
      ID="X8">D0</PARAM.SYMBOL>
      <PARAM.DESC ID="X9"></PARAM.DESC>
    </PARAM>
    <TDML.VALUE VALUE.TYPE="MIN">
      <TDML.EXPRESSION>
        <EXPRESSION
        ID="X10">40</EXPRESSION>
      </TDML.EXPRESSION>
      </TDML.VALUE>
      <TDML.VALUE VALUE.TYPE="MAX">
        <TDML.EXPRESSION>
          <EXPRESSION
          ID="X11">50</EXPRESSION>
        </TDML.EXPRESSION>
      </TDML.VALUE>
    </TDML.CC>
  </CC.LIST>
```

TDML can be parsed using a SGML/XML parser. Both commercial and free SGML parsers

such as James Clark's SP parser (<http://www.jclark.com/sp/index.htm>) are available online for download.

## TDML and the IP Industry

One of the major challenges to using IP is to determine a method for communicating with the IP block. IP blocks communicate using transaction protocols that describe the rules that must be followed for successful operation. Transaction protocols have long been described using timing diagrams because they form a concise visual description of cause-effect relationships between transaction events. Timing diagrams may also unambiguously describe design constraints such as setup and hold time requirements that must be met by interacting components.

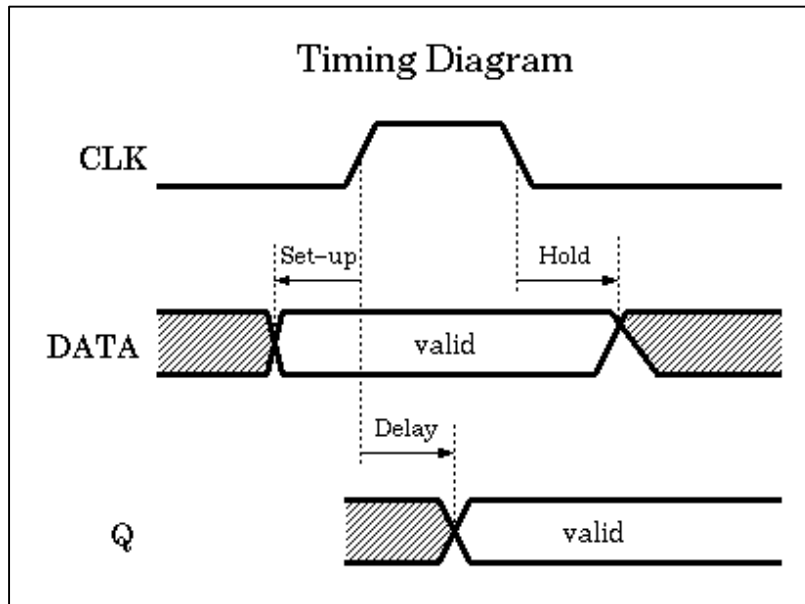
Another important issue in IP usage is substitution of one IP block with another that is behaviorally equivalent either for higher performance, increased functionality, or simply because a design is being ported to a different process. Since timing diagrams focus on describing interface behavior rather than internal operation, they document the design requirements for IP use without dictating implementation details, simplifying drop-in replacement of IP blocks.

TDML provides an open format for distribution of protocol specifications, enabling IP vendors to release complete interface information without locking their customer base down to a specific EDA tool suite. An open format also encourages tool vendors to find new uses for TDML information, increasing the ultimate usefulness of TDML in the design process.

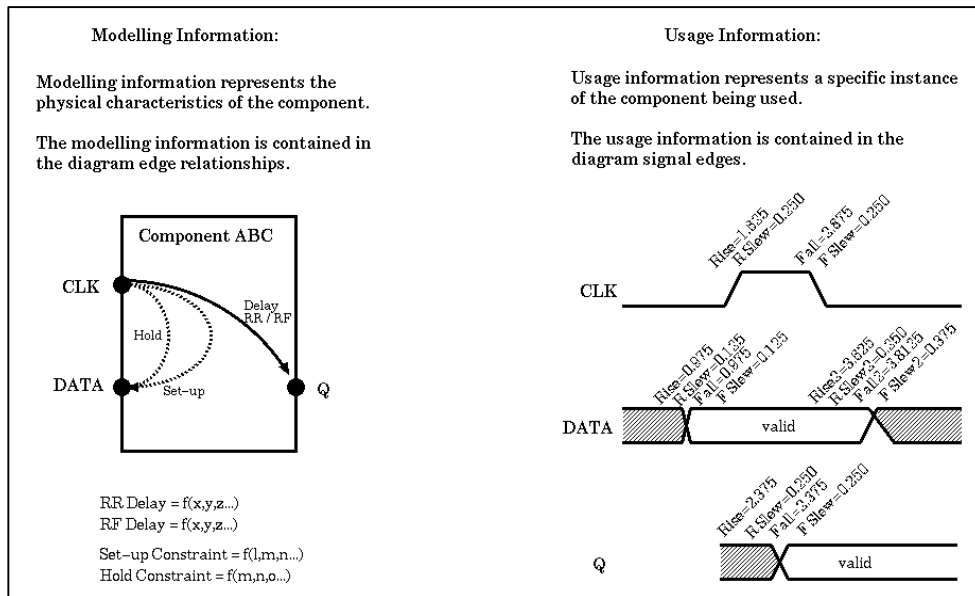
## References

1. Silicon Integration Initiative, *ECIX*, <http://www.si2.org/ecix>
2. Chronology Corporation, *Timing Designer*, <http://www.chronology.com>.
3. SynaptiCAD, *WaveFormer Pro*, <http://www.synCAD.com>.
4. Lewis, Larry, Silver, Kevin, *Virtual components for system-on-a-chip design*, Computer Design, November, 1997, page 52.

# Figures



**Figure 1. Sample Timing Diagram**



**Figure 2. Modeling and Usage Information in a Timing Diagram**

