MODELING DATA TRANSFER SIGNAL TIMINGS IN DAME

B. Huber, K.F. Li, N.J. Dimopoulos, M. Escalante, and E.G. Manning

Department of Electrical and Computer Engineering
University of Victoria, Box 3055
Victoria, B.C., Canada V8W 3P6

ABSTRACT

This paper discusses the timing interface design subsystem in **DAME** (**D**esign **A**utomation of **M**icroprocessor based systems, using an **E**xpert system approach). The timing design subsystem ensures that all signals produced by the interface between components meet the required timing specifications. A main goal of the timing and signal model is the abstraction of common features found in the data transfer timings of most components. This allows relatively few common rules to be used for the interface design which apply to all components. This paper introduces the timing and signal model used to specify the data transfer timing of microprocessor components. Examples of the different timing protocol templates are given, and the design strategy used by the timing design subsystem is discussed.

1. INTRODUCTION

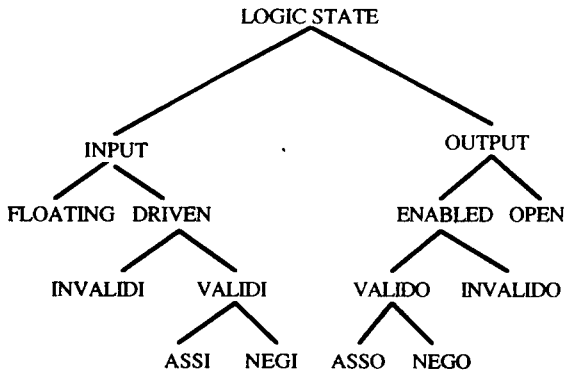
Automatic synthesis tools have been developed over the last few years that help the system designer conceptualizing designs as well as implementing them. These computer aided design (CAD) tools operate at different levels of abstraction spanning from the higher levels, such as the design's overall requirement, down to the implementation gate level [1,2]. High level design languages (HDLs) have been developed employing techniques of abstraction from software design to describe hardware components [3]. HDLs can be used to describe a design at different abstraction levels, but it is often difficult to automatically derive an interface between components described using HDL. This work presents the methods used in DAME [4,5] to generate such interfaces for data transfer signals in microprocessor systems. Specifically, the approach taken to accomplish the functional block design to assure the correct timing of the signals involved in the interfacing of microprocessor system components is described.

2. DATA TRANSFER

In the most general case, data transfer is the exchange of information between two or more devices, accomplished over signal wires that electrically connect these devices together. Information can be carried on the signal wires in the form of the states and the timings of the transitions of the signals. For general information transfer, there are usually several different types of information being transferred simultaneously. For detailed information on the types of information transferred, refer to the accompanying paper on data transfer interface design [6]. To transfer state or timing information, a timing reference is always required. When connecting two devices, this timing reference must somehow be transferred from one device to another without violating any of the timing specifications. This reference can be generated either with the same signal, or with a different signal than the signal carrying the timing information.

2.1 Signal Representation

Devices such as microprocessors and memories have signal pins. Signal pins are used to transfer information to or from a device in the form of electrical voltage levels. At any one point in time, a signal pin has a certain electrical state which corresponds to some *logic state*. The logic state of a signal is a characteristic of the signal that can be propagated through a wire, or through a logic circuit. The following logic states are defined for signals.



LOGIC STATE GRAPH

2.2 Transitions, Events and Timings

A *transition* is the change in a signal from one logic state to another logic state. The state change is assumed to take no time.

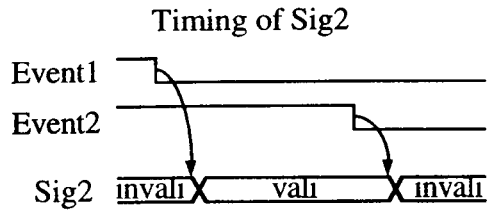
The class of transitions that can be detected physically are termed *detectable*. The only detectable transitions are asserted to negated and negated to asserted.

Symmetrical transitions have the same states but the transition is in the reverse order. If a transition changes from state1 to state2, then its corresponding symmetrical transition changes from state2 to state1.

Linked transitions are two symmetrical transitions of a signal that have no state change between them.

An *event* is the logical combination of one or more detectable transitions of one or more signals. Only AND and OR operators are allowed.

A *timing* is the relationship between the linked transitions of a signal and one or more events. For example, the timing of linked transitions of the signal Sig2 is given below:



A *constraint* in a timing is the relationship between an event and a transition. A constraint gives the timing relationship between an event and a transition that always holds true. The constraint may or may not be causal.

2.3 Data Transfer Timings

In general, there always exist a recognizable event that initiates and another one that terminates the data transfer process. The initiate and terminate events are symmetrically linked transitions of the same signal—the *request*. A device that initiates and terminates an information transfer is called the *master*. The device that accepts this request is called the *slave*.

In practice, the initiation and the termination of a data transfer cycle can always be recognized by events on some of the control signals. The master may wait for an acknowledgment before it generates the terminate event. If acknowledgment is allowed, the overall control of the transfer is *asynchronous*, while if acknowledgment is not allowed the overall control is *synchronous*.

All timing characteristics of each signal are given relative to the initiate and terminate events of the request signal. For the few signals which timing cannot be properly specified using this common reference model, extra specification must be included. There are several advantages in specifying all signals relative to a common timing reference. The most important is the ability to manipulate and modify a signal relative to the request without considering all the other signals (at least as far as the timing is concerned). This makes it possible to write a small number of general signal timing manipulation rules that apply to any signal having a timing relationship with the request. Using a common timing reference also reduces the number of timing relationships that must be entered, stored, and maintained in the system. Instead of having several timing relationships to a multitude of other signals, only one that relative to the common reference is needed.

3. TIMING GRAPHS

To facilitate the synthesis and analysis of timing relationships between signals, *timing graphs* are used. The *nodes* in a timing graph are events or transitions and their interrelationships are the labeled *links* between them. To make it possible to use timing graphs to represent complete cycles such as the read and write cycles for data transfer, which can repeat indefinitely, the graphs should be cyclic and the cycles must be closed. In addition, transitions always come in pairs—every transition must have a symmetrical transition that will change the states of the signals that causes the original transition so that the same transition can occur again.

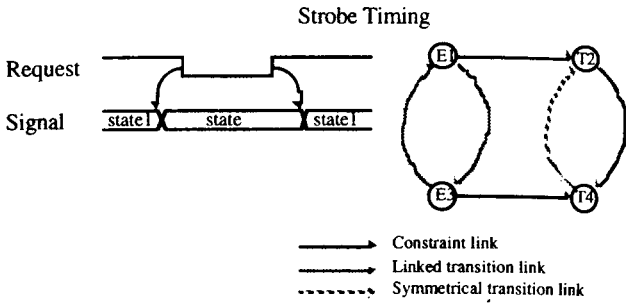
A *symmetrical transition link* is between two symmetrical transitions.

A *linked transition link* between nodes N1 and N2 implies that the state reached when N1 occurs does not change until the state changes to N2.

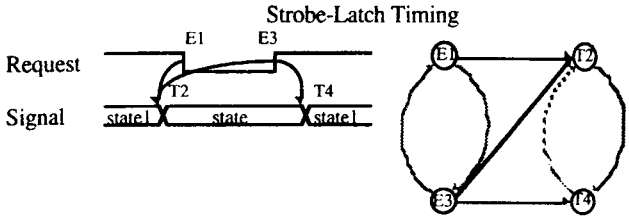
A *constraint link* is a timing specification between any two transition or event nodes in a timing graph. A constraint must be satisfied under all conditions as specified by the timing graph.

3.1 Signal Timing Templates

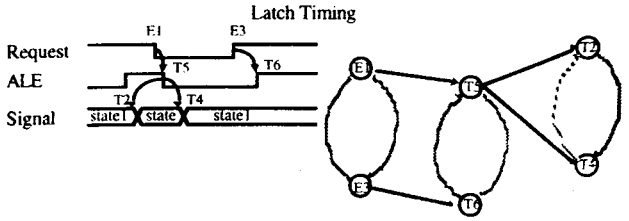
The following timing templates are identified after extracting similarities found in many common microprocessor components, including the 8088, 80286, MC68000, MC68020, Z80, MC6809, EPROMS, static RAMs, and can be used to represent them.



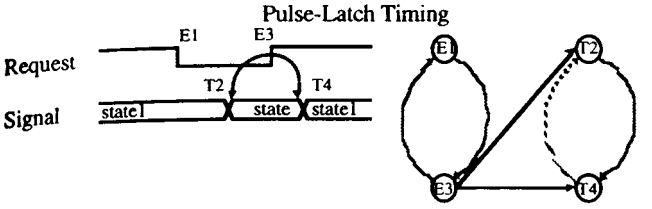
The *Strobe* timing specifies the timing of non-multiplexed signals (e.g., address).



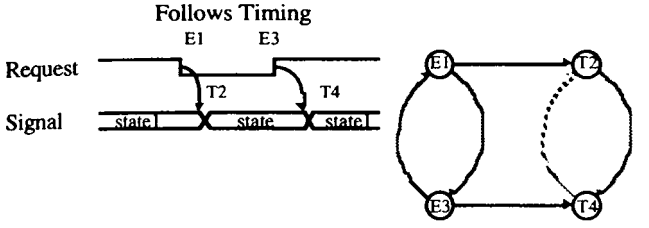
The *Strobe-Latch* timing specifies the timing of non-multiplexed signals of a slave device that has an extra timing constraint relative to the normal strobe timing.



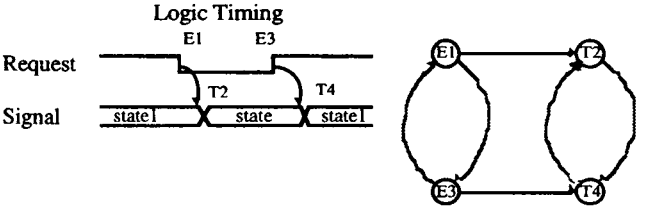
The *Latch* timing specifies the timing of multiplexed signals. Note that an extra signal (ALE) is involved with this timing, which is also constrained to the request.



The *Pulse-Latch* timing specifies the timing of signals that must be latched with the terminate transition (e.g., data input signals for a microprocessor read).

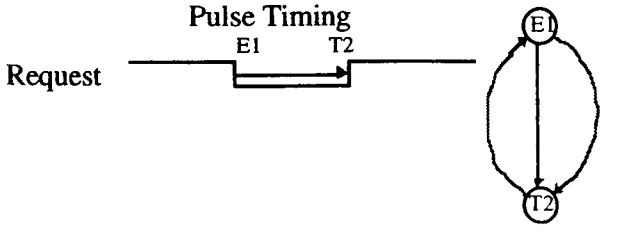


The *Follows* timing specifies the timing of signals that respond to a request signal (e.g., the data signals on a slave during a read). The link between the initiate transition and the first transition of the specified signal is *causal* for this timing.

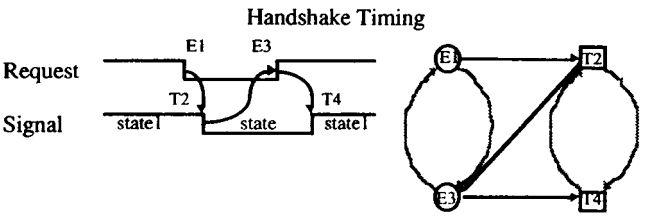


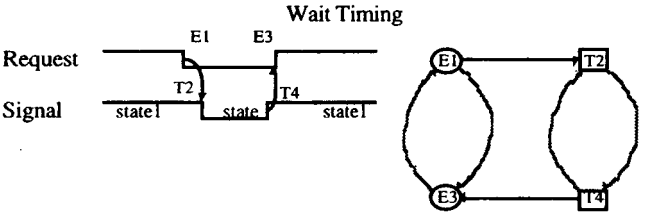
The *Logic* timing specifies the timing of a signal that has the same timing as the request reference timing. (e.g. the timing for the strobe signal on a microprocessor will be Logic timing).

In addition to the above timings that can be used to represent the majority of microprocessor components, the following three templates have been recognized as representations of the overall control timings.



The *pulse* timing specifies the timing of a synchronous request. The delay from the initiate to terminate transition of the request is fixed.





The *handshake* and *wait* timings are used to alter the time between the initiate and terminate transitions asynchronously. For the handshake timing the E1 to T2 constraint determines the delay, while for the wait timing the T4 to E3 *constraint determines the delay.*

4. TIMING INTERFACE DESIGN

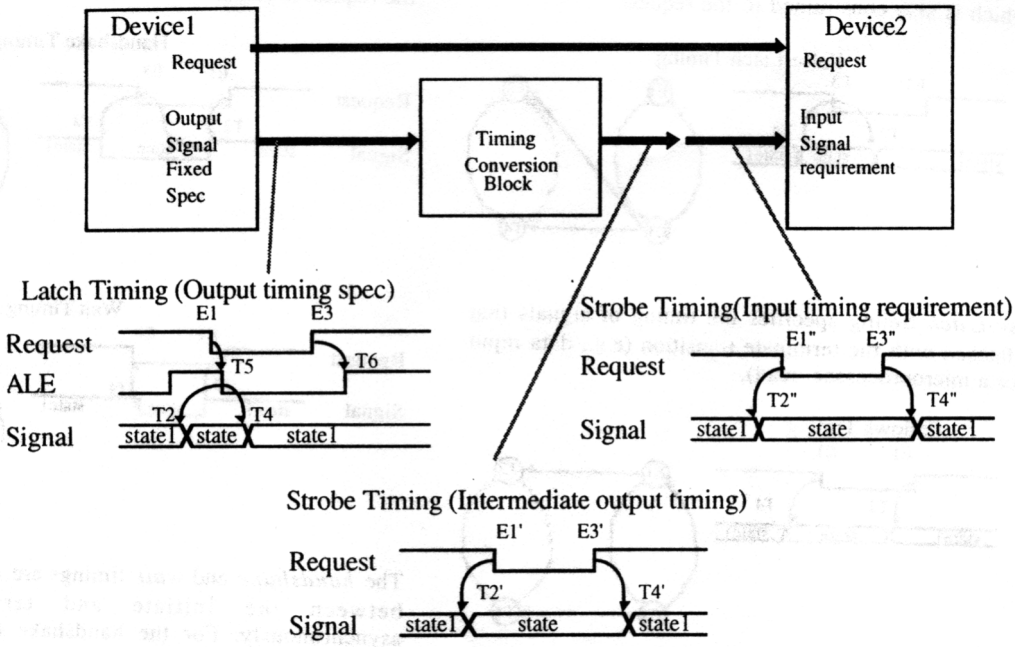
When two information signals have to be connected together, there correspond an output timing from a device and an input timing into a device. The output timing is a fixed timing specification of the output signals, while the input timing is the timing requirement that the input signals must meet. These timings are often incompatible and thus the signals cannot be connected directly. The following discussion concerns with the timing aspect of signal connection. State connection of interface signals is discussed in the accompanying paper [6].

Timing interface design can be viewed as a process that merges the timing graphs of an output specification and an input requirement of the two devices. The timing connection process, or the merging of the timing graphs, involves two steps:

1. An intermediate output signal is generated using a timing converter. The purpose is to generate a timing template that matches the input timing as closely as possible. The actual timing behavior of this intermediate output signal is dependent on the parameters of the conversion block.
2. The intermediate output signal is connected to the input signal. During this process the parameters in the timing converter are constrained to give an implementable design that assures the intermediate output timing meets the input timing requirement.

Dividing the connection process into two tasks allows the designer to first decide on the logical behavior of the timing conversion block, followed by the determination of the parameters of the timing conversion block. Quite often there are more than one block which logical behavior must be determined before the parameters can be constrained.

An example timing connection can be seen in Figure 1. Device 1 produces a latch output timing which feeds into a timing conversion block. The timing conversion block has a strobe intermediate output timing. The intermediate output timing is then connected to the strobe timing input requirement.



Example Latch to Strobe timing connection
Figure 1

4.1 Timing Conversion and Connection

The timing converter produces a timing that is compatible with the input timing. Every timing is compatible to itself, but there are also a few other possibilities—logic timing is compatible with pulse timing, and strobe timing is compatible with strobe-latch timing.

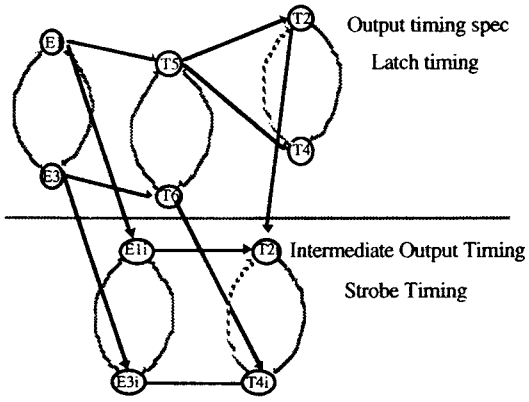


Figure 2

Figure 2 shows the connection of a latch output timing (multiplexed signal) to a strobe input timing (non-multiplexed signal). Analyzing the nodes in the graph it is determined that a transparent latch will accomplish the conversion.

A design rule that checks for latch-to-strobe conversion requirement is used to build a timing conversion block containing a transparent latch. Such conversion rules must be generated for all timing that have the potential of being connected. Since there are a total of 9 timing templates the upper limit for the number of conversion rules is 81.

During the second phase of the connection process, the fact that the two devices do not have an identical timing reference is resolved. Assuming that the request reference has at most simple propagation delays associated with it when connected between the devices, this delay can be incorporated into the constraints on the parameters of the interface block.

Since the output specification of device 1 and the logical behavior of the interface block are known, the intermediate output timing parameters can be determined relative to the request on device 1. If the relationship between the timing references (requests) on the two devices is known, the constraint on these parameters to assure an intermediate output timing that satisfies the input timing requirement can be ascertained.

For example, if $\mathbf{T_{output}}$ is the output specification of device 1, $\mathbf{T_{intermediate}}$ is the intermediate output specification of the interface block, **Parameters** are the parameters of the interface block, and $\mathbf{T_{requestdevice2}}$ is the request timing of device 2, then,

$$\mathbf{T_{intermediate} = F1\ (T_{output},\ Parameters)}$$

$$\mathbf{T_{requestdevice2} = F2\ (T_{requestdevice1})}$$

where F1, F2, $T_{requestdevice1}$ are known. To satisfy these relations, the **Parameters** must be constrained to a range of values.

4.2 Status of the Timing Interface Designer

The timing interface design module in DAME has been implemented in KnowledgeCraft™. The timings are implemented as schemata with slots representing the input signals, output signals, transitions, events and timing constraints. An example timing is given below:

```
(defschema mc68020-address-timing
  (is-a          strobe-timing)
  (signal1       mc68020-ds)
  (signal2       mc68020-address-bus)
  (event1        (asso mc68020-ds))
  (event3        (+ (! nego mc68020-ds))
  (transition2   (! valo mc68020-address-bus))
  (transition4   (! ivalo mc68020-address-bus))
  (time2         (-~ -20))
  (time4         (15 +~)))
```

This timing effectively specifies that the setup time of the address signal is 20 nsec and the hold time of the address signal is 15 nsec relative to the request.

Rules have been written to generate converter blocks from one timing to another. Another set of rules has been implemented to generate the constraints on the parameters of the converter block for each connection. For example, a rule that converts a latch timing (multiplexed bus) to a strobe timing (non-multiplexed bus) will generate a timing converter block that contains a transparent latch. The transparent latch parameters are constrained by a connection rule that connects a strobe intermediate output timing to a strobe input timing, assuring that the parameterized timing satisfies the input requirement.

It is our experience that it was not necessary to provide all the possible rules for conversion from one timing to another, since some timing combinations can never occur. For example, the strobe-latch timing is found as an input timing only, and thus can never be an output. Similarly for the connection rules, a latch timing should never be connected to a strobe input timing without a timing converter, and thus a connector rule is not required for these timings. As an example, below is a simple rule that generates a demultiplexed signal from a multiplexed signal.

```
;rule to fill in a converter block with the
;appropriate logic, if input is latch timing
;and output is a strobe timing.
;i.e., mux to demux signal!
(p modify-latch-to-strobe-block
   (interface-sub-block
     ^schema-name <int-block>
     ^instance 'interface-sub-block
     ^modified-by+inv <superblock>
     ^function
       (member single-signal-converter <>)
     ^status 'new
     ^input-timing <itim>
     ^output-timing <otim>
     ^input-signal <isig>
     ^output-signal <osig>)
   (latch-timing ^schema-name <itim>)
   (strobe-timing ^schema-name <otim>)
   (step ^phase data-xfer-interface-creation)
-->
   (create-new-sig <int-block> <superblock>)
   (create-nonmux-signal <int-block>)
   (adjust-timing
     (get-value <int-block> 'output-timing)
     <otim> <isig>))))
```

5. CONCLUSIONS

The electrical signals between microprocessor components from different manufacturers interact in many different complex ways. To be a functional system, DAME must be capable of designing an interface, complex or not, between any of the components. Expert system rules are used to accomplish the connection process. To keep the number of design rules to a manageable level, commonly used interface protocols were abstracted, so that only the information essential for the design process to proceed is considered.

The model developed for DAME allows extraction of the general properties of timing interactions between signals in the form of a limited number of timing templates. For data transfer, a total of 9 general timing templates have been found, that can be used to represent almost all common data transfer protocols. Such a small number of templates facilitates the maintenance of the component library and the use of a limited number of rules to accomplish the data transfer interface design.

The timing interface designer generates parameterized conversion blocks using simple timing conversion rules. The timing parameters of the conversion blocks are constrained to assure that they satisfy the input timing requirement of the device they connect to. The design of the timing conversion blocks is to be submitted to an implementation subsystem producing a hardware circuit that satisfies the timing parameters specified.

REFERENCES

[1] J.R. Fox, "A Higher Level of Synthesis", *IEEE Spectrum*, pp. 43-47, Mar. 1993.

[2] M. Yoeli, Formal Verification of Hardware Design, *IEEE Comp. Soc. Press*, Los Almitos, California, 1990.

[3] A.C. Parker, "Automated synthesis of digital systems", *IEEE Design & Test*, pp. 75-81, Nov. 1984.

[4] M. Escalante, N.J. Dimopoulos, B. Huber, K.F. Li, & E.G. Manning, "Generic Design Rules for the Design of Microprocessor Based Systems in DAME: Bus Arbitration Sub-systems", *Proc. of the 1991 IEEE Intl. Symp. on Circuit and Systems*, Singapore, pp. 3166-3169.

[5] B. Huber et al., "Microprocessor Components and Signal Behavior Modeling in DAME", in *Proc. of the Can. Conf. on Elec. and Comp. Engineering*, pp. 19.4.1-19.4.4, Sept. 1990.

[6] B. Huber, K.F. Li, N.J. Dimopoulos & E.G. Manning, "Data Transfer Interface Design in DAME", *Proc. of the IEEE Pac. Rim Conf. on Comm., Comp. & Signal Processing*, Victoria, 1993.

ACKNOWLEDGMENT

This research was supported in part by an NSERC Strategic Grant. B. Huber was supported in part by an NSERC post-graduate schloarship.