DATA TRANSFER INTERFACE DESIGN IN DAME

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

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

ABSTRACT

This paper discusses the component model and interface design procedure for data transfer in **DAME** (**D**esign **A**utomation of **M**icroprocessor based systems, using an **E**xpert system approach), a system in development that will be capable of producing the complete design from original system specification, such as the type of application, cost, processing requirements, power consumption, etc.

Signals connecting microprocessor components are used to transfer information between the components. Several different types of information were found to be involved in data transfer. The model developed for data transfer treats each of the information exchange as a primitive. Each information primitive consists of the timing information of signals involved and the state information about the exact state of these signals during the information exchange. In DAME, design decisions are made by a set of rules that manipulates the information exchange primitives to obtain the final interface design.

1. INTRODUCTION

The rapid evolution of semiconductor technology allows a system designer to build sophisticated microprocessor systems using off-the-shelf components. New components, which must be integrated into new systems, are introduced regularly. There is strong motivation for the automation of the design process since it is becoming increasingly difficult for the designer to keep up-to-date with the ever changing new components [1]. One of the goals of the DAME project is an attempt to minimize the impact of the introduction of new components on system design by automating the low level design [2,3,4]. Thus, the system designer is relieved of dealing with the complex signal interface protocols and instead can concentrate on the higher level functionality aspect of the design.

Another problem with the evermore sophisticated systems is the increasingly difficult task of system verification. As components become more complex, a proper verification procedure for the resulting design becomes necessary since it may be impossible for the human designer to manually verify all the design parameters. By automating the design process some formal hardware verification techniques can be employed to assure a correct design [5].

The lack of comprehensive theory of system integration and design choices has led to a more or less empirical set of rules for system design, which an experienced designer can draw upon to give an optimum solution to a problem. Many efforts have been made to organize this empirical knowledge so systems that are capable of automatic design can be produced. For examples, R1 is an expert system capable of configuring VAX computers [6], and VEXED is a VLSI circuit design consultant [7]. In addition, several systems have been specifically built to automate the microprocessor systems design process such as KDMS [8], and Micon [9].

During the design process of microprocessor based systems, the designer usually goes through a formal design process incorporating the following phases:

1. The Design Specification phase that establishes the system responsibilities and design constraints.
2. In the Configuration phase the gross system architecture is established.
3. The Behaviour Description phase defines the capability of the gross system components.
4. In the Functional Block Design phase, the functions of the subsystems are mapped onto available components.
5. During the Integration phase the functional blocks are connected together with missing parts synthesized.
6. In the Implementation phase an actual circuit corresponding to the design is produced.

This paper discusses the approach taken to accomplish the Integration phase which synthesizes the interface between components.

2. DATA TRANSFER MODEL

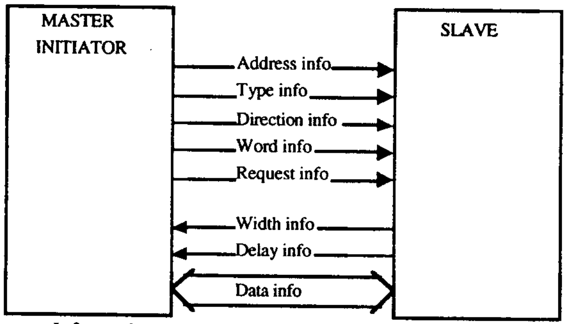
Data transfer is one of the capabilities of microprocessor components. Other capabilities include bus arbitration and interrupt acknowledge. A model is needed to encapsulate these information among others. The conceived microprocessor component model has several levels of abstraction from the general component properties to the detailed timing of the individual signals. This model is implemented in the frame-based KnowledgeCraft^{TM} development environment.

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. Several types of information exchange involved in data transfer have been identified. Aside from the data information itself, the others are:

Request	Events indicating the initiation and termination of the data transfer.
Address	Specific location where the data information can be found /stored (e.g., address 78F3).

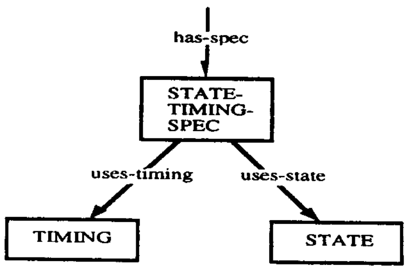
Type	General information where the data information space is (e.g., User, Supervisor, IO, Memory).
Word	Size of the information transfer (e.g., 8, 16, 32 bits).
Direction	The direction of data transfer (e.g., Read).
Width	Which signals will be used to transfer the data information (e.g., D0-D7).
Delay	Informs the initiator how long to wait before sending the terminate transition.

The device that initiates the data transfer is called a *master*. The master may generate the Request, Address, Type, Word and Direction information. The device receiving the request is called a *slave* and it may generate the Width and Delay information.



Information transferred between Master and Slave

Information can be carried on the signal wires in the forms of the states and the timings of the transitions of the signals. This natural division was incorporated into the component model where each information type is represented by state and timing separately.



The request can be represented by a signal that has transitions for the initiate event and the terminate event. Every information timing will be constrained to this request signal. The specification of this constraint is called a *timing*. This type of organization allows the timing of all signals to be treated the same, be it the timing of an address signal, or the timing of an acknowledge signal that controls the delay. For more information on the timing aspect, refer to the accompanying paper on modeling data transfer signal timings [10].

The component model must represent information for the data transfer in a structured and consistent way for efficient manipulation. In KnowledgeCraft^{TM} this is accomplished using *schemata* and *relations*. Inheritance can be used for instances of schemata to inherit properties from their parents automatically. A method for a structured and flexible way for entering components into a component library is provided. This was done by organizing the components structure as shown in Figure 1.

Every component has capabilities represented by *capability schemata*. The data transfer capability has different types of information transfer associated with it. Each information transfer is associated with a state-timing specification that is linked to the capability schema through different types of *has-spec* relations. Each state-timing specification consists in turn of a timing schema and a state schema. All the instances of the schemata just discussed are related through a *is-a* relation to a prototype schema that contains information that is common to all schemata of this type. Each of the prototype schema has a model schema that gives the allowed and recommended slot values for each of the slots of the prototype schema. The model schema is used to assist in the entry of components into the component library.

3. INTERFACE DESIGN

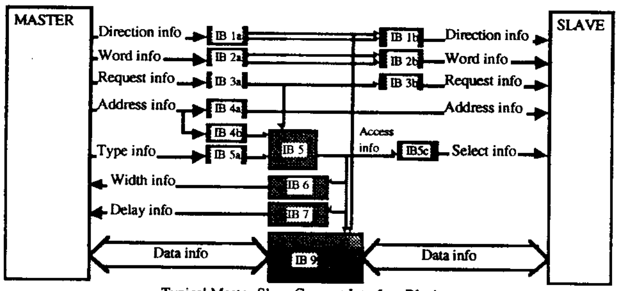
Interface design concerns with how data transfer signals representing similar types of information are connected. All devices have a request signal and all information transfers are synchronized to this request signal. In order for all information transfers to proceed properly between the devices, the request signals will have to be connected together. Connecting the request signal directly makes it possible to synchronize the timing of one type of information signal between the two devices without considering other information signals. In practice the request signal will usually have a delay associated with it which must be taken into account when finalizing the timing design. The detailed timing design procedure is discussed in the accompanying paper [10].

In most cases all the information sent from the master will be received by the slave and vice versa. However, quite frequently both the slave and master do not have receptors or transmitters for some of the information. The only information absolutely required is the request information from both the master and the slave. Therefore, to accomplish device connection, any missing information must be generated and any extra information must be used in a consistent way by the interface logic. For example, often the master has more address information than the slave can handle, such as the master may produce 20 address lines while the slave only requires 12. In this case, a hardware decoder uses the extra master address signals to produce a 'select' signal.

During the interface design process, similar information signals are connected after converting the states to compatible states and the timings to compatible timings. State connection is accomplished by translation circuitry. If the translation involves only asserted or negated signal states, combinatorial logic can be used (the state translation usually involves an n-input to m-output decoder). If other states must be obtained, such as tristate, specialized hardware must be utilized. How to use extra or generate missing information signals is a design choice. Many variations are possible with tradeoffs in complexity, speed, cost and power consumption.

3.1 Interface blocks

In our design approach, all information transfer signals are connected using interface blocks. An *interface block* is a hardware circuit that generates the proper states and the proper timing required by the slave's information input, from the specification of the master's information output. For example, the figure 2 shows interface blocks that could be generated to connect a master and a slave. Note that this is one implementation of many.

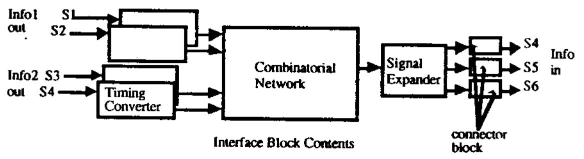


Typical Master-Slave Connect Interface Blocks
Figure 2

In general, information signals are connected using interface block to accomplish the following tasks:

i. Information signals from different sources are combined to generate internal signals for information transfer.
ii. Internal signals are decoded to generate the correct output states.
iii Information input signals are supplied with the correct timing.

A typical interface block is organized as follows:



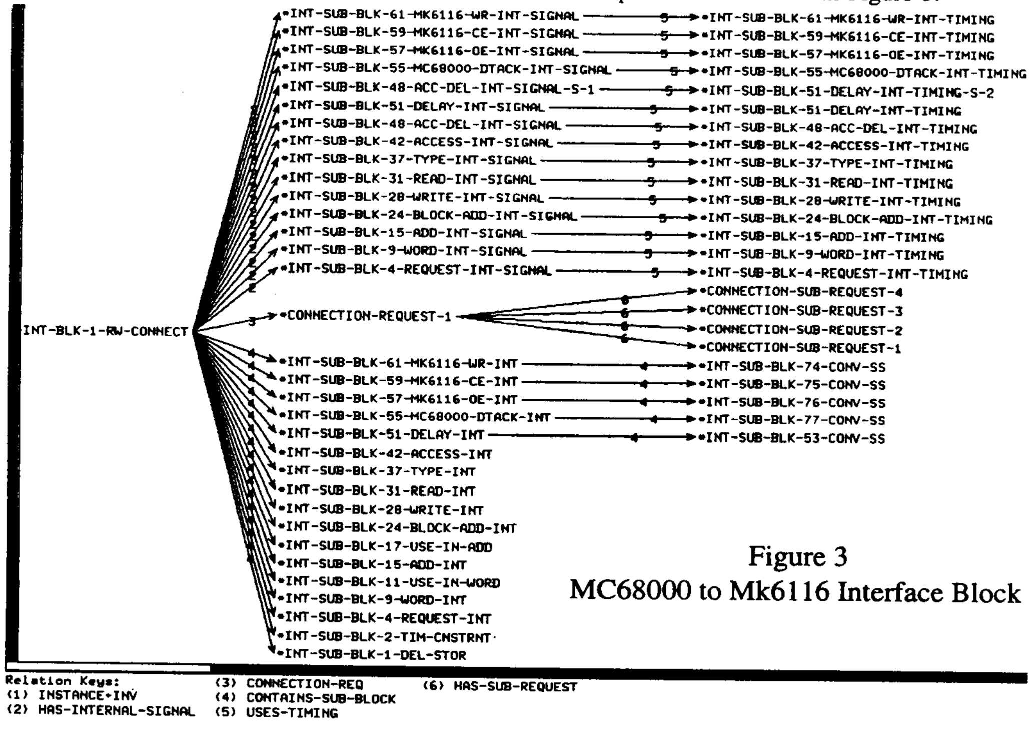
Interface Block Contents

The timings of the individual signals entering an interface block are adjusted using timing converters (e.g., buffers, delays or latches). The states are matched using a combinatorial network. The signal expander supplies signals to several input information locations and connector blocks are used to obtain the final implementation constraints (e.g., propagation delay).

4. STATUS

In DAME's design hierarchy, once components are chosen in the Functional Block Design phase, an interface between the components is to be produced during the Integration phase. This interface can then be fed to the Implementation phase using various existing VLSI and PLD tools to produce the actual hardware. To accomplish the interface design effectively, a component model and rules for the design procedure must be developed. The model for data transfer treats each of the information exchange as a primitive, consisting of both timing and state information. The design procedure must assure that the correct states and timing relationships among the signals are maintained. It must also be able to handle missing or extra information provided by a device.

The component model has been developed and integrated into the KnowledgeCraft^{TM} environment. Several components have been entered into the component library including MC68000, MC68020, Z80, and 8088 microprocessors, and several static RAMs and EPROMs. Rules have been written for the design of data transfer interface. This interface design process is initiated by a higher level connection request that also specifies the components chosen. The interface designer currently produces a network of schemata representing the interface circuit. Such a design describing the control signal connection for a MK6116 2k*8 static RAM and a MC68000 microprocessor is shown in Figure 3.



INT-BLK-1-RW-CONNECT
•INT-SUB-BLK-61-MK6116-WR-INT-SIGNAL
•INT-SUB-BLK-59-MK6116-CE-INT-SIGNAL
•INT-SUB-BLK-57-MK6116-OE-INT-SIGNAL
•INT-SUB-BLK-55-MC68000-DTACK-INT-SIGNAL
•INT-SUB-BLK-48-ACC-DEL-INT-SIGNAL-S-1
•INT-SUB-BLK-51-DELAY-INT-SIGNAL
•INT-SUB-BLK-48-ACC-DEL-INT-SIGNAL
•INT-SUB-BLK-42-ACCESS-INT-SIGNAL
•INT-SUB-BLK-37-TYPE-INT-SIGNAL
•INT-SUB-BLK-31-READ-INT-SIGNAL
•INT-SUB-BLK-28-WRITE-INT-SIGNAL
•INT-SUB-BLK-24-BLOCK-ADD-INT-SIGNAL
•INT-SUB-BLK-15-ADD-INT-SIGNAL
•INT-SUB-BLK-9-WORD-INT-SIGNAL
•INT-SUB-BLK-4-REQUEST-INT-SIGNAL
•INT-SUB-BLK-61-MK6116-WR-INT-TIMING
•INT-SUB-BLK-59-MK6116-CE-INT-TIMING
•INT-SUB-BLK-57-MK6116-OE-INT-TIMING
•INT-SUB-BLK-55-MC68000-DTACK-INT-TIMING
•INT-SUB-BLK-51-DELAY-INT-TIMING-S-2
•INT-SUB-BLK-51-DELAY-INT-TIMING
•INT-SUB-BLK-48-ACC-DEL-INT-TIMING
•INT-SUB-BLK-42-ACCESS-INT-TIMING
•INT-SUB-BLK-37-TYPE-INT-TIMING
•INT-SUB-BLK-31-READ-INT-TIMING
•INT-SUB-BLK-28-WRITE-INT-TIMING
•INT-SUB-BLK-24-BLOCK-ADD-INT-TIMING
•INT-SUB-BLK-15-ADD-INT-TIMING
•INT-SUB-BLK-9-WORD-INT-TIMING
•INT-SUB-BLK-4-REQUEST-INT-TIMING
•CONNECTION-REQUEST-1
•CONNECTION-SUB-REQUEST-4
•CONNECTION-SUB-REQUEST-3
•CONNECTION-SUB-REQUEST-2
•CONNECTION-SUB-REQUEST-1
•INT-SUB-BLK-61-MK6116-WR-INT
•INT-SUB-BLK-59-MK6116-CE-INT
•INT-SUB-BLK-57-MK6116-OE-INT
•INT-SUB-BLK-55-MC68000-DTACK-INT
•INT-SUB-BLK-51-DELAY-INT
•INT-SUB-BLK-42-ACCESS-INT
•INT-SUB-BLK-37-TYPE-INT
•INT-SUB-BLK-31-READ-INT
•INT-SUB-BLK-28-WRITE-INT
•INT-SUB-BLK-24-BLOCK-ADD-INT
•INT-SUB-BLK-17-USE-IN-ADD
•INT-SUB-BLK-15-ADD-INT
•INT-SUB-BLK-11-USE-IN-WORD
•INT-SUB-BLK-9-WORD-INT
•INT-SUB-BLK-4-REQUEST-INT
•INT-SUB-BLK-2-TIM-CNSTRNT
•INT-SUB-BLK-1-DEL-STOR
•INT-SUB-BLK-74-CONV-SS
•INT-SUB-BLK-75-CONV-SS
•INT-SUB-BLK-76-CONV-SS
•INT-SUB-BLK-77-CONV-SS
•INT-SUB-BLK-53-CONV-SS
Relation Keys:
(1) INSTANCE-INV
(2) HAS-INTERNAL-SIGNAL
(3) CONNECTION-REQ
(4) CONTAINS-SUB-BLOCK
(5) USES-TIMING
(6) HAS-SUB-REQUEST

Figure 3
MC68000 to Mk6116 Interface Block

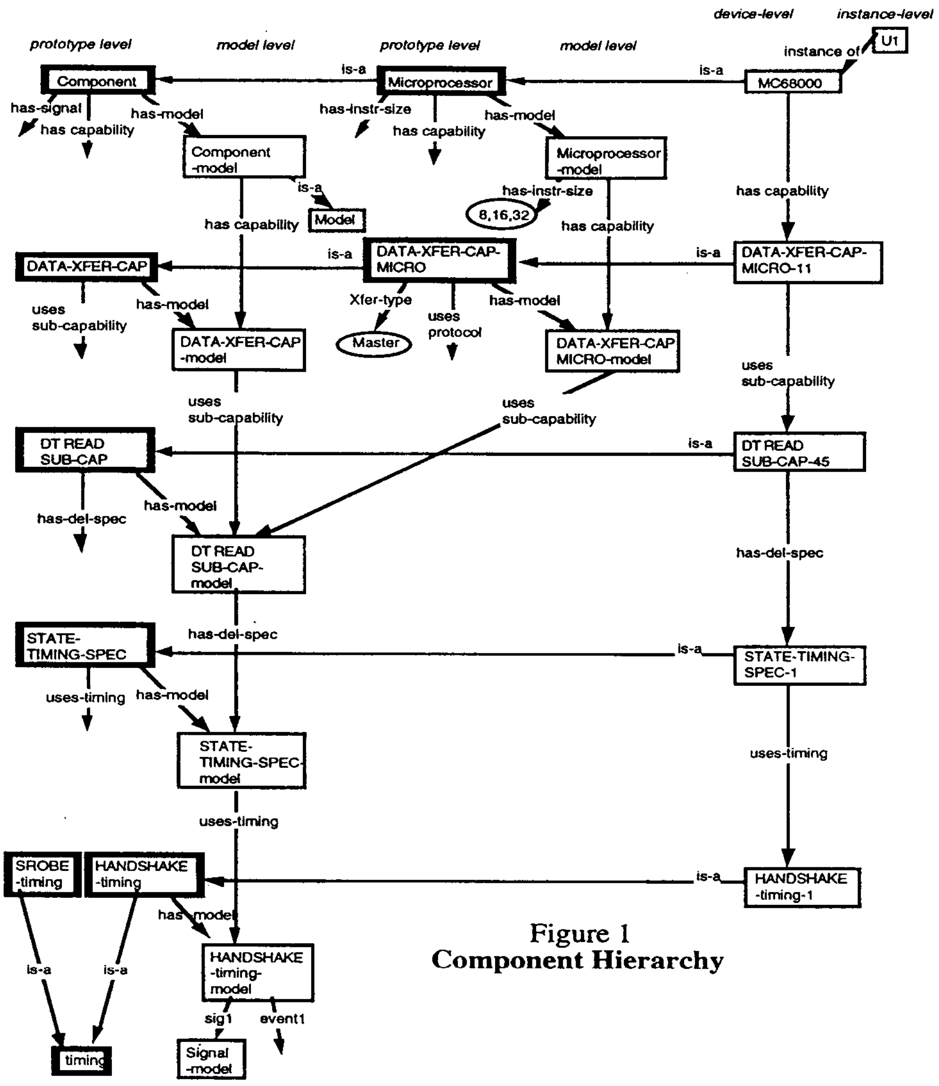


Figure 1
Component Hierarchy

REFERENCES

[1] A.C. Parker, "Automated Synthesis of Digital Systems", *IEEE Design & Test,* pp. 75-81, Nov 1984.

[2] 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 Cir. and Sys.*, Singapore, pp. 3166-3169.

[3] N.J. Dimopoulos, K.F. Li, & E.G. Manning, "DAME: A Rule Based Designer of Microprocessor Based Systems", *Proc. of the 2nd Intl. Con. on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, Tennessee, pp. 486-492, 1989.

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

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

[6] J. McDermott, "R1: A Rule Based Configurer of Computer Systems", *Artificial Intelligence*, vol. 19, pp. 39-88, Sep 1982.

[7] T.M. Mitchell et al., "A Knowledge Based Approach to Design,", *IEEE Trans. Pattern Analysis and Mach. Intelligence,* vol. 7, no. 5, pp. 502-510, Sep 1985.

[8] Y.H. Kuo et al., "KDMS: An Expert System for the Integrated Hardware/Software Design of Micro-processor-based Digital Systems", *IEEE Micro*, vol. 11, pp. 32-92, Aug 1991.

[9] W.P. Birmingham, A.P. Gupta, & D.P. Siewiorek, "The Micon System for Computer Design", *IEEE Micro*, vol. 9, pp. 32-92, Aug 1991.

[10] B. Huber, K.F. Li, N.J. Dimopoulos, M. Escalante & E.G. Manning, "Modeling Data Transfer Signal Timings in DAME", *Proc. of the IEEE Pac. Rim Conf. on Comm., Comp. & Signal Processing*, 1993.

ACKNOWLEDGEMENT

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