MICON: A Knowledge Based Single Board Computer Designer

William P. Birmingham
Daniel P. Siewiorek

Department of Electrical and Computer Engineering
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

For some time there has been interest in automatically implementing designs described in a high level representation. This paper presents a workbench that allows a designer to describe a single board computer at the hardware requirements level and have a design produced automatically. The designs produced by the system, MICON, compare favorably to commercial designs. In comparison to manual techniques, the design time has been drastically reduced; a series of three commercial designs was reproduced in a few hours. MICON uses knowledge based engineering approaches in its implementation and is written in the *OPS 5* production system language.

1. Introduction

To help improve the efficiency of the designer, there has been an effort to provide design methodologies that help to abstract away tedious details and to perform as many of the mundane aspects of design as possible. The project described in this paper, MICON (Micro-processor Configurer), provides the designer with a workbench that takes a very high level design description, at the hardware requirements level, and produces a network interconnection list.

There is little software support and few algorithms to help a designer who works at the hardware requirements level. Apparently, traditional design automation approaches are not well suited for design at this level. Also, since designs are specified at a high level, the number of candidate designs can become very large. To overcome these problems, MICON has applied a *knowledge based engineering* approach to design.

Before developing MICON, a model of the task domain was derived. Based on this model, a knowledge based production system was written that duplicates many of the key aspects of the model. Section 2 briefly presents some related work. Section 3 discusses the model, while Section 4 presents its implementation. An introduction to the production system paradigm and a rationale explaining why the system was implemented as a production system is given in Section 5. Section 6 compares MICON generated designs against commercial designs. Section 7 offers some concluding remarks.

2. Related Work

Much of the work done in this project is related to the parent project DEMETER which is concerned with aspects of design above the register transfer level [9]. The CMU-DA project has produced much of the pioneering work in transforming a high level design description, at the Instruction Set Processor (ISP) level, through to physical realization in either integrated circuits or boards [2]. While the MICON project is not directly related to CMU-DA, a co-operative relationship has been developed. An example of higher level design support is the SARA [3] system, which is concerned with architectural aspects of system design. For the synthesis of hardware from high level descriptions, the International Business Machines (IBM) 3081 project has generated some interesting tools and methodologies [10].

Examples of various knowledge engineering systems can be found in [5]. In addition, examples of work done at C-MU can be found in [7] [6]. A somewhat related project [11] selects from a library of commercially available single board computers one that most closely matches the requirements described by a user. Projects underway at Rutgers [8] and MIT [1] are also concerned with the application of expert systems to design problems. However, these projects are primarily concerned with changing or diagnosing existing systems, not synthesizing new designs.

3. Task Domain

The task domain for this project is the construction of single board computers. Single board computers use micro-processors as the primary computational engine. They also contain some local memory, input/output (I/O) devices, and often an interface to a common bussing scheme. Since these computers must, by definition, fit on a single board a premium is placed on board real estate.

There are a large number of different micro processors on the market. While the machines are similar in many respects, it would be difficult to develop the expertise to design with each of them in a reasonable time period. For this reason, the "micro-processor space" was pared down to a manageable size. Several criteria were used in evaluating the candidate micro-processors:

Availability of a local expert on the processor.
: For reasons discussed later (see Section 5), it is critical that an expert designer familiar with the processor be available.

Number of support chips in the processor family.
: The processor must have a family of established peripheral chips. The growth of micro-processor systems is closely linked to the availability and functionality of peripherals compatible with it. While it is true that most processors may use peripherals outside of its family, it takes several glue chips to convert between different protocols; for a single board

*This work was funded in part by the SRC and by Siemens A.G. The views presented here are solely those of the author and do not necessarily represent those of the funding agencies.

0738-100X/84/0000/0565$1.00 © 1984 IEEE

computer this should be avoided since the extra chips consume limited board space.

Usefulness of the processor.

There are two features used to determine the usefulness of the processor. First is the amount of software existing for a processor. Established processor families generally have more software available than new processor families, so established processors would rank higher in this measure. Second, in order to evaluate the quality of designs produced by MICON, a set of metrics is needed. To gain points of comparison for these metrics, a number of designs using the processors is required. The availability and variety of designs available is considered very important.

The construction of single board computers is based upon the integration of subsystems. The subsystems are well known and correspond to functional units:

- processor
- memory array and controller
- peripheral devices and controllers.

Integrated circuit manufacturers have developed chips to realize these functions. Thus, the task of designing single board computers can be pictured as selecting needed functional units and interconnecting them. Although this appears simple it should not be construed as trivial.

MICON is based on a set of tasks and associated subtasks which correspond to the construction of the function units. The two major *tasks* involve design (of a single board computer) and analysis (of the design). Each task is comprised of a set of *subtasks*, to be described later. The philosophy of the system requires the user to specify requirements for each subsystem in turn and then have MICON implement that subsystem. The design of each subsystem is based on knowledge gained through the design of the subsystem that preceded it.

4. The Design Subtasks

The subtasks may be considered incremental steps towards the completion of a task. For the design task, each of the subtasks must be activated at some time during the design process. For the analysis task, it is possible that only some of the subtasks may be used.

There are several basic functions required of each subtask in the design task:

Specification: The user is requested to enter requirements for a particular subsystem. For example, if the user is designing the memory array, it is necessary to specify the type of memory, the amount required, and so forth.

Selection: After the requirement has been ascertained, the required parts are sought. If no part exactly matches the requirement, the closest alternative will be substituted. If there is no alternative (frequently the case for input/output devices) the user is notified and must change the requirement.

Instantiation: Once parts are selected they are instantiated into the design. This is done by creating "logical" copies of the devices from the data base and interconnecting them.

The specific subtasks and their inter-relationships are illustrated in Figure 1. The order shown in the diagram is very important; information determined in one step is built upon in later steps. For instance, the type of controller used in memory selection has an effect on the peripheral device subsystem design since both may share some parts.

4.1. Processor Design

A requirement of MICON is to allow novice designers to successfully build single board computers. It is assumed that the novice knows nothing about specific processors, but has a feel for some of the attributes used in describing processors. Through a set of queries, the requirements of the processing element are determined. The set of queries presented below does not represent a closed set, there are several other imaginable criteria, such as data path width. Additional queries may be added at a later date; however, for the purposes of this project, the queries to follow are considered adequate. If the user would like to use a particular processor, it is possible to circumvent the automatic selection process by naming the processor.

The queries are:

- Processor name.
- Average instruction execution time. This is intended to be a rough measure of processing power.
- Cycle time.
- Application. By a predetermined qualification, each processor has its potential applications stored in the data base as an attribute.
- Cost and Power. The suggested cost and power consumption of the processor.

Not all the queries must be answered. In fact, the designer can answer none of them. If this is done, MICON assumes no processor preferences and selects a processor virtually at random.

Along with each criterion, a weight is required. Weights are used in an evaluation function for choosing the processor which most closely matches the requirements. The evaluation function is essential since the method for choosing a processor for a particular task is not well specified. It is doubtful that a processor will be found that exactly matches the needs of the user. The evaluation function is given in Figure 2.

4.2. Memory Design

The next step in the design task is memory specification and selection. The three types of recognized memory are RAM (static and dynamic) and ROM (blown fuse). The user must specify the type and amount of addressable memory required. If the amount of memory specified exceeds the addressing range of the processor, the user is notified that the amount of specified memory should be changed. The power and area limit for the memory array may also be specified.

The first step in the construction of the memory array is the selection of a memory chip. For dynamic RAM 16K, 64K, and 256K chips are available. For static RAM 1Kx4, 4Kx1, and 16Kx1 chips are available. For ROM 2Kx4, 4Kx4, and 8Kx8 chips are available. The most dense chip that uses the least amount of area will always be chosen. For example, if the requirement was 20K of dynamic RAM, a single row of 64K chips would be used, rather than two rows of 16K. Real estate is considered the single most valuable resource and is always conserved, even for an increase in the cost of some other factors.

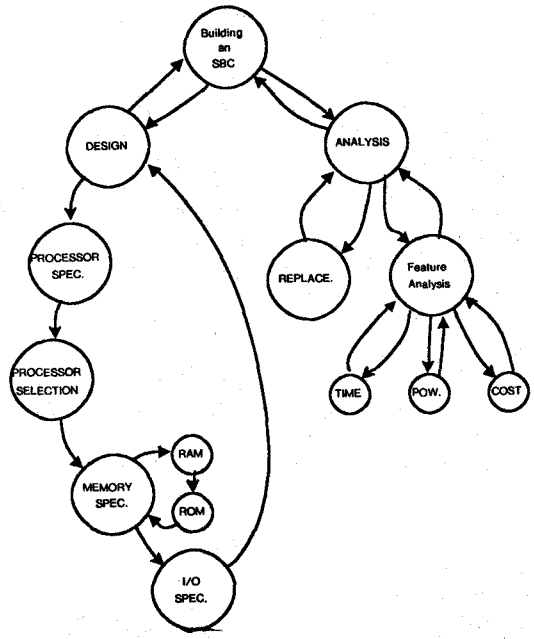

Figure 1: Task and Subtask hierarchy in MICON

After the type of memory chip has been selected, MICON determines the array size. The chips are then logically interconnected. For example, the chip select lines are connected to the address decoder. The inclusion of any support chips, usually in the form of memory controllers, occurs automatically without intervention of the designer. To accomplish this, templates are used (see Section 4.5).

The actual memory chip part number and manufacturer to be used in the design is selected from the data base after the controller has been determined. This order is mandatory because the memory chip speed will be directly related to the speed of the controller. For the initial design, the memory chips selected are fast enough to ensure zero wait states for the processor.

For ROM there are some special requirements. Since many designers use a particular ROM chip, it is possible to specify a specific part number. If the system does not know the part, the user is queried about the part to enhance the system data base. For a designer who has no preference, the system will determine the correct chip in the same manner as used for RAM selection. Additionally, the user must specify where to place the ROM in the address space. For the processors currently described in the data base it is recommended that the ROM be placed at the top of memory: this is where the processors look for boot programs during power-up and reset. It is possible to overlay ROM on RAM so that no RAM space is lost. If the user does not have any idea about where the ROM should be placed MICON will locate it at the top of the address space.

4.3. Peripheral Device Design

The specification of peripheral devices is slightly more complicated than for the previously mentioned subsystems. Peripherals are specified after memory so certain parts can be shared between subsystems. Typically these parts are chip select logic and bus drivers. The user is required to give the type of device needed. The choices consist of:

- timers
- parallel input/output devices (PIO)
- serial input/output devices (SIO).

For those devices that handle a large amount of data quickly, it is possible to connect a direct memory access (DMA) controller to them.

After the appropriate device has been found in the data base, the user is requested to give the device an address. If a device address is not given, MICON will automatically select the address and reserve several words in memory for the device's control register locations. The amount of memory needed for control registers is stored as an attribute of the device in the data base. It is recommended that peripheral device addresses be placed at the top of RAM when no ROM is present. The peripheral devices should be placed just below ROM when it is present. MICON will always select addresses in this fashion. An automatically generated address is displayed to the user.

After all devices have been assigned an address, they are then assigned an interrupt priority (all the peripheral chips used have interrupt capability). If the user does not want to specify a priority for each device, then all are assumed to be equal. The peripherals are instantiated into the design by interconnecting the interrupt lines followed by the data, address, and control lines. Each processor has its own special method of connecting the interrupt lines. To handle each case, MICON invokes an interrupt model for each processor. The models are represented in the form of *rules* which describe in what manner the processor expects its interrupt structure to be built. As in previous subtasks, all the necessary glue chips are included without designer intervention. This includes wait-state generators for slow chips, interrupt prioritizers, and baud rate generators.

Once all the subtasks outlined above have been completed, a design exists that represents the system's best attempt to match the specifications. A list may be obtained that documents the logical interconnections and a memory map of the addresses for the memory and peripheral devices.

```
f(e)=(user-specified-value/stored-value)
        * weight

Where:
user-specified value = value that the
        user supplies to a query.
stored-value = the processor's
        characteristic for a particular
        query. For example, the Z80 would
        have 1.0 watt as its value for
        power consumption.
weight = the user specified importance of
        that particular
        user specified-value.
```

Figure 2: Evaluation Function

4.4. Analysis Task

Analysis provides the user with the capability to verify part of the design, or modify a portion of the design to bring it into closer agreement with the requirements. Modifications take the form of part replacement. Chips which more closely match the requirements of the user (e.g. with respect to cost and power dissipation) may be substituted. If there is a need for a change in the structure of the design the current design should be saved, and MICON be invoked again with different design parameters. The two subtasks forming the analysis task are described below.

The verification subtask enables the user to critically evaluate what MICON has produced along three dimensions. These are:

cost: The cost of each component is given as well as the total cost of the system. The costs are based on estimates and act as a guideline in evaluating the relative costs of different designs.

power: The power consumption of each component and the total system power are calculated.

timing: Several critical timing paths are known to the system. The designer can verify the system has chosen parts that meet certain critical timing paths by examing each path. The paths are**:

- Memory path: processor driver/latch address decode/memory control - memory chip access - data driver.
- Peripheral path: processor - driver - address decode - peripheral chip access - data driver.

The paths listed above are typical for a read operation. The distinction made between peripheral devices and memory stems from the simpler control needed for peripheral devices. Memory typically needs address decode logic, in addition to some form of timing control.

Since there are several parallel paths that exist for each of the subsystems, a timing calculation is performed for each and reported to the user. In the memory path above, the parallel paths are delineated by slashes. Consider the *iAPX-186* as an example. This chip has its data and address bus time multiplexed. The address must be latched, since it only lasts for one bus cycle. After the address portion of the bus cycle, data is applied for the duration of the instruction cycle. The control signals are available throughout the entire instruction cycle and they only need pass through a driver.

It is important to note that the timing subtask uses nominal times for calculations and does not take into account delays in board lines. Instantaneous slew rates are also assumed. Delays due to routing can not be accurately calculated since no routing information is available to MICON (routing occurs at a later time).

The second subtask in the analysis task is replacement. The replacement subtask allows the user to change the parts selected by MICON. It is possible to change certain, but not all, parts in the design. It would not make sense to change the processor family, for instance, since this would significantly change the bus structure. It is assumed that memory is the portion of the design most likely to need redesign.

The following information is required to change a part:

- the number of the current part
- what attribute of the part to change (e.g. cost)
- the type of change requested (e.g. less power).

Based on this information, the type of part is determined and an attempt is made to find a part that will match the new requirements. MICON will "activate" a series of candidates and describe each to the user until one is found or there are no more "activated" candidates.

Once a part has been accepted by the user for change, it must be determined whether the timing of this part is consistent with the rest of the design. A precondition is used to determine if the consistency could possibly be violated. The precondition is:

> If a new part is as fast, or faster, than the replaced part it is concluded that timing will be consistent. If the replacement part is slower, then the design will be inconsistent.

If the precondition is true, the user is notified that system performance may be degraded with the new part. To determine if a design is inconsistent, the access time of the new part is compared with the requirements of the processor*** The number of wait states needed is determined by comparing the delay of the new part with the processor's data ready time and bus cycle length. If no wait states are required, the design is considered consistent. If wait states are required, then the wait state generator is instantiated (if it had not been previously) and all necessary connections are made. The number of wait states needed is reported to the user.

The one exception to the method outlined above deals with dynamic RAM. The controller used in dynamic RAM designs is the Intel 8207 which only allows chips with either 100 ns or 150 ns access times. Therefore, the range of change for dynamic RAM is limited.

4.5. Data Base

The data base is an intergal part of the workbench. It holds two key parts of the system: design templates and chip information. Each part of the data base is discussed below.

All the information about the chips used in the design are held in working memory: this is a vestige of the production system (see Section 5). There are approximately 100 chips currently in the data base, with more expected to be added. Not all these chips are used in a single design. For efficiency those chips essential to the design are activated. Activation can be accomplished in two ways. The first way keeps the attention of the system focused on those chips associated with the selected processor. The second way forms a list of part instantiations used in the current design. MICON makes use of both ways; attention focusing for template selection and list generation for glue chips.

Three major chip types were identified and are represented with their own specific type of working memory element. Working memory elements can be thought of as Pascal records. Each field in the working memory element represents an attribute of the chip. The chip types and their attributes are:

- memory chip element- name, manufacturer, cost, power-static, power dynamic, access-time, cycle-time, type (dynamic RAM, ROM, etc.), number of columns, number of rows, number of pins, width, area, power supply required, and several flags.
- I/O chip element- name, function (parallel input/output device, etc.), manufacturer, address space needed, processors compatible with, number of interrupts generated, cost, power, access time.
- processor chip element clock cycle time, average execution time, name, cost, power, memory access unit (byte, word), time for zero wait state data available, several flags

MICON has most of its design knowledge stored in templates. Templates save computational time by providing large pre-designed sections, so MICON does not have to re create the basics of each design. The templates may be considered generic designs which are modified for each processor.

The major templates are used to describe the relationship between the processor, the board level bus, and the generation of

** Please note that the symbol "/" indicates that both devices are active in parallel.

*** Each processor's description has an attribute telling when it assumes data will be ready in a zero wait state machine cycle.

a few extraneous signals needed by various parts, e.g. data transmit signal for bus transceivers. The *iAPX 186* one of the processors which MICON uses- template is tailored to construct templates for the other processors used by MICON. Due to the *iAPX 186* bus structure (multiplexed data and address) this processor requires a superset of parts compared with those needed for the other two processors, which have separate data and address busses. If a processor other than the *iAPX 186* is being used in a design, MICON knows how to modify a template for proper *customization*. An important point to keep in mind: the templates represent a "logical" design, dealing with such entities as busses or control lines. Templates do not contain actual pin information. The mapping to chip pin information occurs during post processing.

The templates are represented as amalgamations of parts and ports. All the parts needed for a template are represented as working memory elements. Representing the connections between these parts are *port descriptors*. Port-descriptors are working memory elements that detail port linkages across chips. Ports may correspond to actual chip pins, although in some cases they represent a group of pins, e.g. a bus. Templates typically contain about 10 parts and have about 50 connections.

5. Production System Implementation

Expert systems are rapidly becoming an important approach in tackling many engineering tasks. Expert systems should perform as well as human experts. Generally, expert systems are build on the extracted collective knowledge of "human" experts. One of the acknowledged weak points of expert systems is the limited domain of discourse, or knowledge; they are "experts" in one field and usually only in a small subset of that field. Many expert systems are written as production systems: a collection of knowledge represented as a large number of IF-THEN clauses.

5.1. The Production System Paradigm

Knowledge is explicitly represented in production systems as a set of rules, or productions. The terminology used to refer to these productions is *situation action* pairs. The left hand side (LHS) of a production, corresponding to the IF part, defines some situation or pattern to match. The right hand side (RHS), corresponding to the THEN part, describes some action that is to take place. Working memory elements are used in evaluating the left hand side. In essence, productions look for specific patterns in memory and upon recognizing them, carrying out some action. Often the action will change working memory causing another production to fire. In the course of program execution, the pattern-match/production fire steps occur repeatedly until there is an explicit halt or no LHS can find a matching pattern in memory.

An example *OPS 5* [4] production taken from MICON is given in Figure 3. The first line is the name of the production. The two statements before the "-->" constitute the left hand side (the working memory pattern to match), the statements after are the right hand side (the actions to be carried out). The English translation for the production in Figure 3 is:

> If there exists a working memory element named *state* whose attribute *user mode* has the value "design phase" **AND** there is a working memory element named *token* whose *type* attribute has the value "design command string" and attribute val has the value "help" or "?"
> **THEN** execute the statement.

5.2. Contexts

A method of organizing the productions is needed, especially when a large number of productions are required. Sometimes, a seemingly insignificant change in one part of the program can have catastrophic effects in another part of the program when there is no apparent connection. One of the simplest and most effective ways to organize the productions is to partition them according to the different tasks and subtasks constituting MICON. In MICON's implementation task and subtasks correspond to *contexts*. Each production is specified to run only in a certain context, with the exception of a few special productions.

Switching between contexts is controlled by the user. Whenever the user decides to build a specific subsystem, or verify a portion of the design, MICON's context is implicitly changed. In many expert systems, much effort is devoted to deciding when to switch contexts or to generating new contexts. Since MICON allows the user to switch the context, some effort was saved in the control structure implementation.

```
(p design-phase-help
{ <s> (state ↑user-mode design-phase) }
{ <t> (token ↑type design-command-string
             ↑val << help ? >>) }
-->
(write (crlf)The following subsystems are)
(write recognized:)
(write (crlf)memory processor I/O)
(write (crlf)status- shows state of design)
(remove <t>)
(modify <s> ↑user-mode design)
)
```

Figure 3: Rule Example

5.3. Search

The job of building a single board computer can be thought of as a search through a large space, which can be quite costly in both computer and human time. With the chips in MICON's current data base, over 1600 different combinations of processors, memory and peripheral chips are possible. Search in MICON is directed by knowledge gained from the construction of previous subsystems. For this reason it is important that the order of the subtasks be preserved, otherwise the search may become unbound, or nonsensical designs may result.

Once the decision is made to build a certain subsystem, it is constructed in a linear fashion without backtracking across contexts. If the design techniques of the subsystems were not well defined, it would be necessary to provide a mechanism for allowing design decisions to be undone. A limited form of backtracking is permitted within a context to recover from a bad specification. The attention of MICON remains focused on the particular path it is currently following, performing a depth-first search.

The directed search paradigm works effectively for the situations tested. For the selection of a processor, an evaluation function was developed and is used in an A*-like search paradigm. It should also be noted that the working memory is organized to help in search. The working memory elements have been sorted, usually slowest chip first, so that the search path is shortened.

6. Performance

Expert systems should perform as well as a human expert in the domain of expertise. For the purpose of this paper, the following criteria are presented on which to judge the performance of MICON.

Quality of design: The number of chips, relative performance and cost are a good basis of comparison with other systems.

Design Time: Design time is defined to be the time it takes to bring a design from the hardware

requirements to net list generation. For the designs to which this program is to be compared, design time is estimated as the overall run time of MICON.

For each of the processors currently used by MICON- the *iAPX 186*, *Z80*, and *TI 9900*- several designs have been chosen that bound the design space of each processor. The variables perturbated are cost and performance. Through general consensus of experienced designers it was concluded that power is not a major criteria in this domain as long as it does not become excessive.

The results in terms of cost, performance (rated in the number of wait states inserted), and design time are given for each design later in this section. As points of reference, commercially available single board computers based on each of the processors, with the exception of the *iAPX 186*, are included in the results tables. Since the *iAPX 186* is a recent entry into the microprocessor market, no designs based on this processor could be found. For comparison, designs based on the 8086 were used: this precludes direct comparison, but can give a rough estimate of the quality of designs that MICON is building with the *iAPX 186*.

6.1. Capabilities

One of the interesting points bourn out in the trial runs is the computation expense of MICON. The CPU time is fairly constant between high performance and low cost designs for a given processor; for the trials presented, the standard deviation (σ) ranges from 3.43 to 1.47 seconds. The load independent CPU time is for a VAX 11/780[*****] as measured by the **time** function provided by *UNIX*[******]. The small deviation is related to the way the system is written; the floor plan for each processor is fixed and only the chips change. As MICON expands and more design styles are incorporated into the system, it is expected that CPU time will not remain independent of the type of single board computer being developed. This is based on the belief that the design space will enlarge and more computation will be required to fully explore the reasonable choices. Some examples of execution time are given in Table 1.

Since both experts and novices perform the same basic design functions, e.g. selecting a processor and memory, the design time for each class of designer is roughly the same. The novice, assumed here to not know the type of processor that the application requires, will incur a slight incremental expense in determining the processor to use. This is about 3.3% to 31% based on several trials. These percentages were determined by

Processor	Performance	Execution Time(s)
iAPX-186	low	101.2
	medium	107.5
	high	98.6
Z80	low	103.0
	medium	101.1
	high	101.5
TI-9900	low	103.0
	medium	99.4
	high	96.0

Table 1: Performance vs. Execution Time Examples

comparing the run time required for MICON to select the processor to designing a system with a processor the user had chosen. It is expected that total execution time for experts will probably be longer than for novices: experts will perform more parts replacement and analysis functions, a result of exploring the design space to a greater degree.

From analysis of the production traces, selecting and customizing templates require the majority of the execution time. This includes the subtasks of parts selection and instantiation. Of these two activities, the instantiation subtask is more time consuming.

MICON provides benefits to both the experienced designer and the novice designer. The novice is presented with a system that hides all the obscure and confusing details of single board computer design. Thus, the novice can state application requirements in an unambiguous and meaningful way that is easy to understand. In essence, the designer is isolated from learning curve effects, as the system has absorbed this effect itself.

Exploring the design space is one of the most time consuming tasks for a designer. Benefits to the expert include the ability to traverse the design space quickly and efficiently. Most of the mundane calculations and data base searching are done for the expert. All that is required is to substitute different components and compare the results to various requirements. A single design is processed in approximately thirty minutes.

6.2. Discussion of Experimental Runs

Several experimental trial results are given below. The runs are attempts to replicate the designs produced by commercial manufacturers. It appears that when the manufacturers produce a family of designs based on a processor, the amount of memory is the only variable. All of the manufacturers, except Texas Instruments, provided the same peripheral devices throughout the range of designs.

All of the design examples came from the processor manufacturers. Intel has by far the greatest variety of designs; three designs were used to bound the full range of the design space. The Intel designs are based on the 8086 and therefore will show MICON in a better light[*******]. Texas Instruments has two designs in what was considered the low and middle points of the design space. Zilog has only one design, which represented the low end of the design space. The missing middle and high points for Zilog and Texas Instruments were generated for this paper. The results of the experimental runs are summarized in Table 2.

The points of comparison are the estimated cost, as given by MICON, and number of chips used. The designs have the cost dimension normalized; i.e. all designs of a specific type, for example high performance, and based on the same manufacturer are given the same cost. Cost is normalized so that the price break the manufacturers receive would be factored out. As the tables show, MICON has done exceeding well using about 1/4 to 1/3 the chips used in the commercial designs. The underlying reason is that MICON uses a higher degree of integration than the commercial designs. For example, in the dynamic RAM designs, MICON uses a single chip controller, the Intel 8207. The commercial designs use several chips. MICON also uses higher density memory chips. One point should be made: in mapping from "logical" design to the physical representation, it is possible that several chips will have to be added. Since enitities, such as busses, are allocated one driver regardless of their width, during

[*****] Registered Trademark of Digital Equipment Corporation.

[******] Registered Trademark of American Telephone and Telegraph.

[*******] In comparing the Intel designs, it is necessary to note that the *iAPX-186* is a highly integrated processor. The processor has on chip both timers and DMA controllers. Depending on the user's specification, this can save several chips. More importantly, the *iAPX-186* has chip select and wait state generation logic on chip, saving two to four additional chips.

the mapping into the physical representation several drivers must be added. Estimates project that the final chip count will increase by roughly 25%. The performance of the designs produced by MICON and the manufacturers is expected to be about the same. Both sets of design use the same peripheral and processor families.

Processor	Performance	Chip Count
iAPX-186	low	105, 29
	medium	92, 29
	high	43, 27
Z80	low	43, 28
	medium	NA, 28
	high	NA, 20
TI-9900	low	42, 24
	medium	52, 23
	high	NA, 22

NOTE: NA (not applicable) means that the manufacturer provides no design for this performance level.

Table 2: Estimated chip counts for MICON and commercial designs. Under the chip count column the chip count for the commercial design is first followed by MICON's chip count.

7. Conclusions

The designs that have been produced compare well with commercial designs. In terms of part count, MICON designs need roughly 50% fewer chips than the commercial designs: this number is optimistic since the mapping from "logical" to "physical" design representation will require additional chips. The part count increase is due to the underestimation of the width of certain data paths. Projections estimate that the number of chips actually required to implement a MICON design will increase by 25%, at most, bringing the overall chip savings to around 25% over the commercial designs. Much of this gain is based on the use of newer, more highly integrated parts, whereas the commercial designs may be using older technology. MICON effectively tracked the family of designs given by Intel for its 8086 boards, and Texas Instruments for its TI-9900 based boards. It is interesting to note that the amount of execution time does not appear to be related to the complexity of the design.

The results shown in this paper show that designs can be successfully represented at a very high level (the requirements level) and both the time savings and design error reduction make it worthwhile to design at this level. Additionally, the knowledge based approach used in MICON has been shown to be successful, not only in its results, but also in structuring the knowledge used in the domain of single board computer design.

MICON itself is a complete program. It has the ability to produce a "logical" net list and bill of parts. A post-processor for mapping the "logical" net list to a "physical" net list is being built. Also, a graphics interface for producing schematics is under development and partially operational.

References

[1] R. Davis, H. Shrobe, W. Hamscher, Karen Wieckert, M. Shirley, S. Polit.
Diagnosis Based on Description of Structure and Function.
In *Proceedings of the National Conference on Artifical Intelligence*. AAAI, AAAI, 1982.

[2] S.W. Director, J.P. Shen, D.P. Siewiorek, D.E. Thomas.
The CMU DA/CAD Project.
Technical Report CMUCAD 82 2, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1982.

[3] E.G. Estrin.
Modeling for Synthesis The Gap Between Intent and Behavior.
In *Proceedings of The Symposium on Design Automation and Microprocessors*. IEEE, IEEE, 1979.

[4] C. L. Forgy.
OPS5 User's Manual.
Technical Report CMU-CS 81 135, Department of Computer Science, Carnegie-Mellon University, 1981.

[5] F. Hayes-Roth, D.A. Waterman, D.B. Lenat.
Building Expert Systems.
Addison Wesley Inc., 1983.

[6] J. Kim, J. McDermott.
TALIB: An IC Layout Design Assistant.
In *Proceedings of The National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, William Kaufmann Inc., 1983.

[7] J. McDermott.
R1: A Rule based Configurer of Computer Systems.
Technical Report CMU-CS-80 119, Department of Computer Science, Carnegie Mellon University, 1980.

[8] T.M. Mitchell, L.I. Steinberg, S Kedar-Cabelli, V.E. Kelly, J. Shulman, T. Weinrich.
An Intelligent Aid for Circuit Redesign.
In *Proceedings of the National Conference on Artifical Intelligence*. AAAI, AAAI, 1983.

[9] D.P. Siewiorek, D. Giuse, W.P. Birmingham.
Proposal For Research on DEMETER: A Design Methodology and Environment.
Technical Report CMUCAD-83 5, Department of Electrical and Computer Engineering, Carnegie-Mellon University, 1983.

[10] G L. Smith, R.J. Bahnsen, H. Halliwell.
Boolean Comparison of Hardware and Flowcharts.
IBM Journal of Research and Development 26(1):106 116, January, 1982.

[11] M.F. Smith, J.A. Bowen.
Knowledge and Experience Based Systems for Analysis and Design of Microprocessor Applications Hardware.
Microprocessors and Microsystems 16(10):515-519, December, 1982.