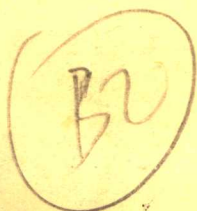
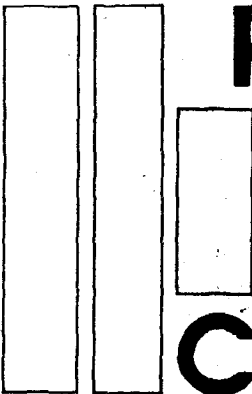


**REAL-TIME
EXPERT
SYSTEMS
COMPUTER
ARCHITECTURE**





REAL-TIME EXPERT SYSTEMS COMPUTER ARCHITECTURE

Robert F. Hodson

Department of
Computer Science
Christopher Newport College
Newport News, Virginia

Abraham Kandel

Department of
Computer Science and Engineering
University of South Florida
Tampa, Florida

CRC Press

Boca Raton Ann Arbor Boston London

Acquiring Editor: Russ Hall
Vice President, EDP: Richard O. Sales
Coordinating Editor: Terri Morris
Cover Design: Miriam Recio

Library of Congress Cataloging-in-Publication Data

Real-time expert systems computer architecture / Robert F. Hodson, Abraham Kandel.

p. cm.

Includes bibliographical references and index.

ISBN 0-8493-4215-5

1. Expert systems (Computer science). 2. Real-time data processing. 3. Computer architecture.

I. Kandel, Abraham. II. Title.

QA76.76.E95H53 1991

006.3'3—dc20

91-21766

CIP

This book represents information obtained from authentic and highly regarded sources. Reprinted material is quoted with permission, and sources are indicated. A wide variety of references are listed. Every reasonable effort has been made to give reliable data and information, but the author and the publisher cannot assume responsibility for the validity of all materials or for the consequences of their use.

All rights reserved. This book, or any parts thereof, may not be reproduced in any form without written consent from the publisher.

Direct all inquiries to CRC Press, Inc., 2000 Corporate Blvd., N.W. Boca Raton, Florida 33431.

© 1991 by CRC Press, Inc.

International Standard Book Number 0-8493-4215-5

Library of Congress Card Number 91-21766

Printed in the United States

PREFACE

Expert systems and real-time systems technology have been developed independently. Expert systems have been successfully implemented in many complex applications traditionally performed by human experts. Real-time systems have been successfully applied in areas requiring interaction with dynamic environments, control and monitoring applications for example. Merging these two technologies will yield intelligent systems capable of interacting with complex dynamic environments, an area in which human operators have poor productivity, due to cognitive overload.

The integration of real-time systems and expert systems impose new requirements on computer architectures. A real-time expert system must be able to process critical events in a timely fashion, handle uncertain data, degrade gracefully, and process knowledge efficiently. There are challenges in meeting the requirements for this new class of systems and they are addressed in the design of a computer system, specifically for real time expert systems.

A multiprocessor system is developed to meet the rigorous system requirements. The individual processors, called intelligent control units (ICUs), are organized in a logically reconfigurable topology to efficiently map onto the problem domain. The ICUs are special purpose processors designed to process rules in an extremely efficient fashion. The design encompasses features to support both the real-time environment and the knowledge processing associated with expert systems.

The real-time expert system architecture is simulated to validate its design and to evaluate its performance. Impressive simulation results confirm the approach taken in this research and encourage further development of this computer architecture.

R. F. H.

A. K.

April 1991

THE AUTHORS

Abraham Kandel is Professor and Chairman of the Computer Science and Engineering Department at The University of South Florida in Tampa, Florida. Previously he was the Chairman of the Computer Science Department at Florida State University as well as the Director of the Institute for Expert Systems and Robotics at FSU and the Director of the State University System Center for Artificial Intelligence. He received his Ph.D. in EECS from the University of New Mexico, his M.S. from the University of California, and his B.Sc. in EE from the Technion—Israel Institute of Technology. Dr. Kandel is a senior member of the Institute of Electrical and Electronics Engineering and a member of the Association for Computer Machinery, as well as an advisory editor to the international journals *Fuzzy Sets and Systems*, *Information Sciences*, and *Expert Systems*. The co-author of *Fuzzy Switching and Automata: Theory and Applications* (1979), author of *Fuzzy Techniques in Pattern Recognition* (1982), co-author of *Discrete Mathematics for Computer Scientists* (1983), co-author of *Fuzzy Relational Databases—A Key to Expert Systems* (1984), co-editor of *Approximate Reasoning in Expert Systems* (1985), author of *Fuzzy Mathematical Techniques with Applications* (1986), co-author of *Designing Fuzzy Expert Systems* (1986), co-author of *Digital Logic Design* (1988), co-author of *Engineering Risk and Hazard Assessment* (1988) and co-author of *Elements of Computer Organization* (1989). He has written more than 180 research papers for numerous professional publications in Computer Science and Engineering.

Robert F. Hodson completed a Ph.D. in Computer Science at Florida State University in 1989. His research focused on innovative computer architectures for artificial intelligence applications. Upon completion of his Ph.D., Dr. Hodson continued to research this area as a visiting assistant professor at Florida State University and later accepted a faculty position at Christopher Newport College. In addition to his academic experience, Dr. Hodson was a computer engineer in the industrial sector for more than six years. His work experiences include the development, integration and test of several complex real-time computer architectures.

CONTENTS

| | | |
|-----------|--|-----------|
| 1. | Real-Time Expert Systems | 1 |
| 1.1 | Expert Systems | 1 |
| 1.2 | Real-Time Systems | 2 |
| 1.3 | Why Real-Time Expert System? | 2 |
| 1.4 | Existing Architectures | 4 |
| 1.4.1 | von Neumann Architectures | 4 |
| 1.4.2 | Dataflow Architectures | 5 |
| 1.4.3 | Multiprocessor Systems | 6 |
| 1.5 | Systems Requirements | 8 |
| 1.5.1 | Timing | 8 |
| 1.5.2 | Procedural & Declarative Representations | 9 |
| 1.5.3 | Uncertainty | 9 |
| 1.5.4 | Environment Interface | 10 |
| 1.5.5 | Hierarchical Organization | 11 |
| 1.6 | Summary | 12 |
| 2. | System Design | 13 |
| 2.1 | Topology | 13 |
| 2.2 | Processor Structure | 15 |
| 2.2.1 | Interference Mechanism | 20 |
| 2.2.2 | Dataflow Methodology | 20 |
| 2.2.3 | Priority | 21 |
| 2.2.4 | Uncertainty Processing | 22 |
| 2.2.5 | External Interface | 25 |
| 2.2.6 | Temporal Data | 26 |
| 2.2.7 | Procedural Data | 27 |
| 2.2.8 | Communication | 27 |
| 2.2.9 | Dynamic Memories | 27 |
| 2.3 | Summary | 29 |
| 3. | System Implementation | 31 |
| 3.1 | ARL Implementation | 31 |
| 3.1.1 | ARL Control | 32 |
| 3.1.2 | ARL Interfaces | 32 |
| 3.1.3 | Memory Address Hardware | 35 |
| 3.1.4 | Rule Memory | 35 |
| 3.1.5 | ARL Memory | 37 |
| 3.1.6 | Memory Register | 38 |
| 3.1.7 | Decrement Logic | 39 |
| 3.1.8 | Link Priority Logic | 40 |
| 3.1.9 | Negation Logic | 41 |

| | | |
|--------|--|----|
| 3.1.10 | Min/Max Logic | 41 |
| 3.1.11 | ARL Microcode..... | 42 |
| 3.2 | APQ Implementation | 45 |
| 3.2.1 | APQ Control..... | 46 |
| 3.2.2 | APQ Interfaces | 46 |
| 3.2.3 | APQ Priority Logic | 48 |
| 3.2.4 | APQ Memory Organization..... | 48 |
| 3.2.5 | APQ Register File | 49 |
| 3.2.6 | APQ Free Register and Pointer Buffer | 49 |
| 3.2.7 | APQ Empty Queue Logic..... | 50 |
| 3.2.8 | APQ Microcode..... | 51 |
| 3.3 | CPQ Implementation | 53 |
| 3.3.1 | CPQ Microcode..... | 53 |
| 3.4 | AP Implementation | 54 |
| 3.4.1 | AP Control..... | 54 |
| 3.4.2 | AP Interfaces | 57 |
| 3.4.3 | Antecedent Memory | 58 |
| 3.4.4 | Antecedent Compare Hardware | 60 |
| 3.4.5 | Wait List Memory | 61 |
| 3.4.6 | AP Microcode | 62 |
| 3.5 | CP Implementation | 64 |
| 3.5.1 | CP Code | 71 |
| 3.6 | WM Implementation | 74 |
| 3.7 | Message Buffer Implementation | 76 |
| 3.8 | External Interface Implementation | 79 |
| 3.9 | Real-Time Clock Implementation..... | 81 |
| 3.10 | System Reset Implementation | 82 |
| 3.11 | Clock Distribution | 82 |
| 3.12 | Summary | 83 |

| | | |
|-----------|---------------------------------------|-----------|
| I. | System Simulation | 85 |
| 4.1 | Simulation Goals | 85 |
| 4.2 | Simulation Approach | 86 |
| 4.2.1 | Discrete-Event Simulation | 86 |
| 4.2.2 | Process Concept | 86 |
| 4.2.3 | Resource Concept..... | 87 |
| 4.2.4 | Process-Resource Example..... | 87 |
| 4.2.5 | Program Structure..... | 88 |
| 4.2.6 | Simulation Model..... | 90 |
| 4.3 | Simulation Results..... | 96 |
| 4.3.1 | Peak & Typical Processing Rates | 96 |
| 4.3.2 | Effect of ICUs | 98 |
| 4.3.3 | Effect of Tasks | 100 |
| 4.3.4 | Effect of Pruning | 102 |

| | | |
|---------------------------|--|------------|
| 4.4 | Conclusion | 103 |
| Bibliography | | 105 |
| A. | Hardware Diagrams | 113 |
| A.1 | ARL Hardware | 113 |
| A.2 | APQ Hardware | 135 |
| A.3 | CPQ Hardware | 145 |
| A.4 | AP Hardware | 157 |
| A.5 | CP Hardware | 177 |
| A.6 | WM Hardware | 197 |
| A.7 | MB Hardware | 221 |
| A.8 | External Interface Hardware | 225 |
| A.9 | Real-Time Clock Hardware | 233 |
| A.10 | System Reset Hardware | 235 |
| A.11 | Clock Generation Hardware | 237 |
| B. | Sequencer and Microprocessor Code | 239 |
| B.1 | ARL Microcode | 239 |
| B.1.1 | Microbit Definitions | 239 |
| B.1.2 | Test Bit Definitions | 244 |
| B.1.3 | Microcode Description | 245 |
| B.2 | APQ Microcode | 257 |
| B.2.1 | Microbit Definition | 257 |
| B.2.2 | Test Bit Definition | 259 |
| B.2.3 | Microcode Description | 260 |
| B.3 | CPQ Microcode | 263 |
| B.3.1 | Microbit Definition | 263 |
| B.3.2 | Test Bit Definition | 265 |
| B.3.3 | Microcode Description | 266 |
| B.4 | AP Microcode | 269 |
| B.4.1 | Microbit Definition | 269 |
| B.4.2 | Test Bit Definition | 274 |
| B.4.3 | Microcode Description | 275 |
| B.5 | CP Code | 286 |
| B.5.1 | CP Address Map | 286 |
| B.5.2 | CP Code Segment Descriptions | 286 |
| Index | | 291 |

Chapter 1

REAL-TIME EXPERT SYSTEMS

1.1 Expert Systems

In the mid-1950's, the field of Artificial Intelligence evolved to study computer systems that exhibited characteristics associated with human intelligence, including, language understanding, learning, reasoning, problem solving, etc. [Barr86]. One of the technologies which emerged from AI is the class of systems known as expert systems. The structure of an expert system is such that there are two major components:

1. a domain dependent knowledge base, and
2. an inference mechanism which may be common to a number of domains [Myers86].

The knowledge base contains facts about the problem domain and relationships between pieces of knowledge. The inference mechanism is a control structure which typically performs a pattern-directed search of the knowledge base to infer information not explicitly stated.

The expert system concept has proven successful in solving problems in many complex problem domains. Early successes with expert systems include a mass

2 *Real-Time Expert Systems Computer Architecture*

spectrogram interpreter called DENDRAL (Feigenbaum, 1971), MYCIN (Shortliffe, 1976), a system which diagnoses bacterial infections of the blood, and PROSPECTOR (Duda, 1978), a mineral exploration system [Forsy84].

1.2 Real-Time Systems

In [Laffe88] two definitions of real-time are given:

1. a system exhibits real-time behavior if it is predictably fast enough for use by the process being serviced, and
2. the system response has a strict time limit, regardless of the algorithm employed.

Both of these definitions refer to interaction with an external environment in a timely fashion. This is the fundamental feature which distinguishes real-time processing from nonreal-time processing, the element of temporal restrictions.

Some common real-time systems include simple controllers, like those found in household appliances, and monitoring systems, used in alarms. Of course, there are many more sophisticated real-time applications, like flight simulation systems, missile guidance, and robotics systems. But all of these systems have the common attribute of time restrictions on processing when interacting with a dynamic external environment.

1.3 Why Real-Time Expert System?

The integration of expert systems and real-time systems is a logical next step in the development of these two technologies. Real-time expert systems could

be used to replace or assist human operators in a wide range of applications [Benne87]. A related reason, which supports the development of real-time expert systems, is to reduce cognitive overload on operators to improve productivity [Laffe88]. In [Hess87], a model of human interaction with complex dynamic systems shows *high* workload correlates to *low* handling qualities. This relationship reflects the overwhelmed feeling and poor response of human operators when presented with many rapidly changing streams of information.

Complex environments that are potentially dangerous or pose threat to human life are domains which will benefit from real-time expert systems. In space and military applications there are many opportunities to utilize real-time expert systems. Some existing real-time expert systems include:

- FMC's Intelligent Mobile Robot [McTam87]
- Expert System for Satellite Orbit [Laffe88]
- Expert Navigator [Laffe88]
- Flight Expert System [Laffe88]
- Resource Allocation System [Benne87] for readiness in a pilot's air-space.

These systems offer a high level of expertise, while interacting with complex dynamic environments. They either replace or assist manned vehicles and thus reduce risk to human operators.

1.4 Existing Architectures

1.4.1 von Neumann Architectures

Many enhancements have been made to the von Neumann architecture to improve performance when processing AI applications. Recently a wave of RISCs (Reduced Instruction Set Computers) have been under investigation as AI processors [Patte85, Hill86, Patte82]. Some of these systems include special high speed cache memories, register windows, and other features to support AI processing and improve performance. Other systems, like LISP machines [Plesz87], Personal Sequential Inference Machine [Rigas85], and the Dorado [Lamps84], are more complex than RISC machines, using microcoded processors, but these systems are also fundamentally a von Neumann architecture.

The von Neumann computer architecture has performed admirably for a large number of varied applications. Many real-time applications currently use von Neumann style microprocessor control systems. Unfortunately, the von Neumann architecture has not adapted as readily to expert system applications. Processing in expert systems is primarily non-numeric and does not run efficiently on the conventional von Neumann machine. Symbolic knowledge representations used in expert systems are fundamentally different from those of numeric processing. Symbolic operations are memory intensive. A von Neumann architecture presents a processor to memory bottleneck when intensive irregular memory accesses are made [Hwang87]. Additionally, the von Neumann architecture is fundamentally sequential in nature and does not utilize concurrency to increase performance.

| Dataflow | von Neumann |
|--|------------------------------------|
| storage embodied in the notion of a variable | shared memory concept |
| data tokens | memory references |
| sequencing based on data dependencies | program counter sequencing control |
| decentralized control | centralized control |

Table 1.1: Dataflow & von Neumann Features.

1.4.2 Dataflow Architectures

The dataflow computer [Denni88, Herat88] is another general purpose system, which is non-von Neumann in design and addresses some of the shortcomings of the von Neumann architecture. The dataflow computer moves away from the idea of centralized sequential control. The execution of a dataflow program proceeds as data becomes available. The dataflow approach has the potential to exploit a high degree of concurrency efficiently [Tiber84].

The following discussion is largely taken from [Hwang84] and considers the merits and shortcomings of dataflow computers. In the dataflow computer the information items are operation packets and data tokens. Operation packets contain the opcode, operands, and destinations of successor instructions. The data tokens contain intermediate and final results and their destinations. The tokens and packets are passed along to the system resources during the execution of a dataflow program. The machine architecture therefore takes the form of a packet-switched distributed multiprocessor organization. Table 1.1 compares some of the features of the dataflow architecture to that of the von Neumann machine.

6 *Real-Time Expert Systems Computer Architecture*

Many of the new ideas introduced in the dataflow computer can reap benefits not found in the von Neumann architecture. However, dataflow designs are relatively recent and are still controversial. Some of the potential problems in the general purpose dataflow computer are:

- a data driven approach at the instruction level can cause excessive pipelining overhead
- dataflow programs tend to waste memory and increase program length
- when there is a large number of instructions and processing elements, the packet-switched network can be cost-prohibitive and a system bottleneck.

In the development of dataflow computers, the architect must carefully consider many of the practical aspects of the design and implementation.

4.3 Multiprocessor Systems

There have been many multiprocessor computer systems developed. A brief list of some of the MIMD architectures are as follows:

- C.mmp (16 PDP-11s interconnected by a crossbar)
- IBM 370/168MP (dual processors with shared memory)
- Tandem 16 (16 processors with dual common busses)
- Cray X-MP (dual processors, shared memory, pipelining)
- SPUR (6 to 12 RISC-like LISP processors, common bus)
- HEP (up to 16 processors, packet switched network, pipelining)

These computer systems all support multiple SISD-type processors in some form of interconnection network. The individual processors within these systems have a von Neumann structure, although some systems like the Cray X-MP and the HEP support a high degree of pipelining. The SPUR processors also have special features to facilitate the execution of LISP programs, but in general, these systems suffer from many of the disadvantages found in the single processor von Neumann computer when processing symbolic information used in expert systems.

All the systems discussed so far have been general purpose computer systems. Each system can be programmed to solve problems in virtually any domain given sufficient time. Particular features of the individual systems enhance performance in a given area, like the Cray X-MP has special features to support vector processing. But even with some special tailoring of the processors, these machines are still quite general. An example of a special purpose processor would be an IOP (I/O Processor). An IOP is specifically designed to process data transfer instructions in a highly efficient fashion. By limiting the scope of applications that a processor is designed to execute, the performance of the processor can be greatly improved over that of a general purpose system. The increased performance can be attributed, in part, to a reduction in the semantic gap between the high level language concepts and the underlying computer architecture [Myers82]. The design of special purpose processing units will be part of the strategy used to meet the special requirements of real-time expert systems.

1.5 System Requirements

1.5.1 Timing

Several factors contribute to the difficulty of bringing expert systems into real-time environments, foremost, the rigorous performance requirements that must be met [Odett87]. In real-time systems the required response time is a major influencing factor on the overall system [Benne87]. In real-time systems, a critical event cannot be overlooked because the system is processing another action; awareness and responsiveness are essential [Kaebl87].

A real-time expert system must be uniquely aware of the timing constraints placed on it. The system should be able to make decisions concerning time and integrate time elements into its planning. The system must be able to handle elaborate plans in sequence and gracefully degrade when the planning process is time limited [Benne87]. Wright [Wrigh86] points out that interviews with experts indicate problems can be reasoned at different levels and response time is different at different levels. Using this type of progressive reasoning allows the system to make the best possible decision in the time available. Hexscon (hybrid expert system controller) uses task priorities to aid in progressive reasoning by scheduling tasks based on available time.

The symbolic processing, characteristic in expert systems, will influence the design of the system. An effective mapping of the expert system's operations onto the computer system will reduce the semantic gap and improve system performance. Special hardware must be designed to efficiently implement the time consuming operations that may limit system responsiveness. For example, the control of dynamic memory (heap management) can be a time consuming

operation in expert system implementations, and therefore special hardware should be considered to support the operation.

Since system responsiveness is essential, a multiprocessor system will be required to achieve effective processing rates by exploiting high level parallelism. Additionally, fine grain parallelism in the processor design should be employed whenever it can improve system performance.

1.5.2 Procedural & Declarative Representations

Declarative representations stress static aspects of knowledge: facts, objects, events and their relationships. Procedural representations focus on capturing knowledge in procedures which show how to use knowledge [Barr86]. Expert systems typically provide knowledge in a declarative form, for example, rule-based, logic, and semantic net representations. In real-time systems there is an abundance of procedural knowledge [Wrigh86]. A real-time expert system is required to integrate these two knowledge representations to effectively utilize both forms of knowledge.

1.5.3 Uncertainty

There are several sources of uncertainty in real-time expert systems, and several techniques have been developed for coping with uncertainty. The major sources of uncertainty in real-time expert systems are inexact knowledge and inexact data.

Uncertainty in knowledge can manifest itself in several forms. Linguistic terms, like *small*, *low*, or *young* can convey information which is not clear-cut or well defined [Leung88]. Another form of uncertainty is related to the