ISMM

Proceedings of the ISMM
International Symposium

COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS

Honolulu, Hawaii, U.S.A.
February 1-3, 1988

EDITOR: N.F. MARSOLAN

A Publication of
The International Society for Mini
and Microcomputers - ISMM

ISBN 0-88986-136-6

ACTA PRESS
ANAHEIM * CALGARY * ZURICH

TABLE OF CONTENTS

AUTHOR INDEX

Prodeedings of the ISMM International Symposium,
COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS,
Honolulu, Hawaii, U.S.A., February 1-3, 1988, ACTA Press

ON THE DEVELOPMENT OF COMPUTER AIDED DESIGN SOFTWARE FOR MINI AND MICROCOMPUTERS

Richard D. Colbaugh
Department of Mechanical Engineering
New Mexico State University
Las Cruces, NM 88003

Charles R. Dart
Algor Interactive Systems, Inc.
Essex House, Essex Square
Pittsburgh, PA 15206

ABSTRACT

The use of minicomputers and microcomputers for the implementation of computer aided design (CAD) software packages has increased dramatically in recent years. However, the quality of such packages is often quite poor. This paper presents a comprehensive and systematic approach to the writing of high quality, user-friendly CAD software. The approach is discussed in some detail, with particular attention being given to I/O, error diagnostics, documentation, program structure, and portability. Additionally, the roles of interactive graphics and of the computational efficiency of the design algorithms are examined. This approach to the writing of CAD software is illustrated through its application to the CAD of a class of special purpose pivots called flexure pivots. The CAD software developed for these flexure pivots has been implemented on both a minicomputer (DEC VAX 11/780) and a microcomputer (IBM PC), and the results of these implementations are discussed. Finally, consideration is given to the teaching of CAD development techniques to engineering undergraduates.

I. INTRODUCTION

Minicomputers and microcomputers are particularly well suited to the task of hosting computer aided design (CAD) software packages. These computers offer many advantages over mainframes for this application, including the potential for truly interactive programming and for convenient data input and output (I/O). As a result, the use of mini and microcomputers in the area of CAD, and in the related areas of computer aided drafting and computer aided manufacturing, has increased dramatically in recent years [1,2]. Unfortunately, the quality of these software packages is not uniform, and some programs are actually quite poor [1,3]. Part of the reason for this problem has been the lack of availability of a comprehensive and systematic approach to the development of CAD programs. As a result, the development of such systems is often a problem-specific process based on a set of ill-defined heuristics. Another difficulty is the absence of undergraduate instruction in CAD software development at engineering schools. The present paper addresses both of these issues. Specifically, this paper presents a systematic approach to the writing of high quality CAD software, illustrates the effectiveness of this approach through the examination of a specific CAD application, and then reports on progress being made in CAD instruction in the Mechanical Engineering Department of New Mexico State University.

The present correspondence is organized as follows. First, the comprehensive approach to developing CAD software is briefly summarized, with attention being given to I/O, error diagnostics, documentation, program structure, and portability. Additionally, the roles of interactive graphics and computational efficiency are briefly discussed. Next, the approach is illustrated through its application to the CAD of flexure pivots, which are a class of special purpose, limited travel pivots. Flexure pivot theory is briefly summarized, and then the development and implementation of flexure pivot CAD software for both a minicomputer (DEC VAX 11/780) and a microcomputer (IBM PC) is described. Finally, the teaching of CAD development techniques to engineering undergraduates is discussed, and examples of CAD software developed by students are presented.

II. GENERAL CAD SOFTWARE DEVELOPMENT

The development of high quality CAD software is greatly aided by adherence to a comprehensive, systematic approach to CAD programming. Such a procedure for software development has recently been derived by the authors, and shall be briefly summarized here. In what follows, a list of attributes that typify high quality CAD software is given. The integration of these qualities into a comprehensive software design procedure is straightforward and the details will not be given here. A complete description of this approach is presented in [4].

1. The software should be interactive.

While this quality might seem an obvious one in light of the iterative nature of engineering design, many CAD programs are not truly interactive. An interactive program must allow the user to control the program flow, must be consistant in its communication with the user, and should provide clear, concise prompts to promote effective interaction.

2. The software should provide graphics capabilities.

The use of computer graphics is increasing rapidly in many areas of computer application, and the use of graphics in CAD seems essential. The incorporation of interactive graphics in the CAD software allows the engineering design process to proceed naturally, and makes more convenient such tasks as date input and output and user selection of program options.

3. The software should allow clear and efficient input and output of data.

Important I/O characteristics include graphics capabilities, adequate error checking and diagnostics, provisions for easy editing, and features such as input data echoing, automatic file saving, and furnishing all parameter definitions and units.

4. The software should perform complete error checks, and provide complete error messages.

Specifically, the error messages should indicate the cause of the error and also provide suitable diagnostic messages so that each error may be corrected. Observe that good diagnostics is particularly important in CAD programs because of the ill-structured nature of many design problems.

5. The analysis performed by the program must be checked for validity.

It is imperative that the analysis be proven valid for a well defined range of problems and input data, and that the user be made aware of these ranges. Additionally, the software must provide checks that these ranges are not violated.

6. The program should be well documented.

Procedure documentation should be included in the software and should address both the function of the software and the information flow of the program. Both user documentation (prompts, on-screen instructions, HELP functions) and programmer documentation (detailed software documentation to aid future up-dating of the program) should be included.

7. The software should be portable.

The software should be executable on a variety of systems with a minimum of conversion required. Software portability is increased by using code that is in common use, utilizing a modular program structure (with machine dependent functions such as data I/O and graphics separate from the main program structure), and including adequate documentation.

8. The software should be based on numerical algorithms that are robust and computationally efficient.

The desirability of algorithm robustness is clear, but the need for computational efficiency warrants one observation. While efficiency is desirable in any program, it is crucial in an interactive design program since there is a definite limit on the time interval that the user can be expected to sit in front of an inactive screen. Thus, if the program is to be truly interactive, the algorithms upon which it is based must be sufficiently fast so that user interaction is comfortable.

9. The software should be well structured.

Clear program structure is essential if the design program is to be easily maintained and modified. The overall program structure should be developed using a "top-down" process involving the expansion of program modules into progressively more detailed submodules [5]. The program should be modular and possess a clear separation of program functions and a well defined program hierarchy. Additionally, the program should have good structure within each procedure. Program flow should be made as sequential as possible and "back tracking" should be kept to a minimum [6].

10. The program should include a tutorial option and default values for program variables to assist the user.

While the benefits of including "example problems" to acquaint the user with the program's operation are clear, it should be noted that these default values could be updated (either by the user or automatically by the software) as the CAD program solves design problems, thereby providing the software with a (primitive) learning capability.

11. The software should be flexible regarding the user's level of participation.

The software should allow the user to request only a portion of the design program's functions (depending on his needs) and to request a range of program instruction levels (depending on his expertise).

As mentioned earlier, the attributes of high quality CAD software listed above can be readily integrated into a comprehensive and systematic procedure for developing CAD software. While this process is straightforward, the details are somewhat involved and therefore will not be given here [4]. Instead, an example of the application of the procedure to a particular CAD problem will be presented.

III. APPLICATION TO FLEXURE PIVOT DESIGN

The procedure for developing high quality CAD software that was summarized in the preceding section will be illustrated by applying it to the development of design software for the crossed flexure pivot. Crossed flexure pivots of the type shown in Fig. 1 are used extensively in instrumentation applications and in the construction of aeronautical research equipment [7-9]. They are also employed, to a lesser degree, in gyroscopes, governors, regulators, valves, and various limited-deflection linkages [10-12].

The function of a crossed flexure pivot is to permit relative angular rotation between two rigid structural elements. This relative rotation is accomplished through the elastic deformation of the flexure strips that connect the two elements. Flexure pivots enjoy many advantages over conventional pivots, including essentially frictionless travel, excellent properties regarding wear, dirt, and vacuum, and good response to impulsive loads. However, the elastic deformation of the flexure pivot is described by a set of nonlinear second-order differential equations, so that designing an appropriate pivot for a particular application is a formidable task. It is for this reason that the CAD software development method summarized above was applied to flexure pivot design. If an effective CAD program could be written that would allow the mechanical designer to efficiently generate flexure pivot designs, then the numerous advantages of these pivots could be realized.

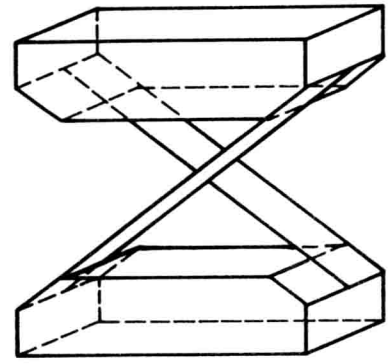


Fig. 1

As indicated in the Introduction section, CAD programs for the crossed flexure pivot have been developed by the authors and implemented on both a mini-computer (DEC VAX 11/780) and a microcomputer (IBM PC). While the problem statement for each of these programs is very similar and the overall program structures are alike, there are sufficient differences between the two CAD programs to make separate presentations desirable.

1. Minicomputer Design Program

The problem statement for the minicomputer CAD program is to design a flexure pivot by determining the values of the following pivot design parameters:

1. flexure strip length, ℓ (in)
2. flexure strip width, b (in)
3. flexure strip thickness, t (in)
4. angle of flexure strip intersection, 2β (degrees)
5. flexure strip material

so that this pivot design will meet user specifications for the following pivot performance criteria:

1. pivot rotational stiffness, k (in-lb/rad)
2. pivot natural frequency, ω (rad/s)
3. precision of pivot rotation, IC (in)
4. maximum permissible pivot stress, σ_{max} (psi)
5. minimum acceptable pivot stability, $\%P_b(\%)$

The numerical algorithms used in the minicomputer CAD program were extremely efficient and quite robust. Computational efficiency presented a major difficulty, and was finally achievedby recasting the flexure pivot differential equation model as a set of elliptic integral equations and by deriving original research results to model the pivot's vibrational dynamics [7]. The overall design algorithm was based on multivariable interpolation theory and was written to considerably reduced the number of times that the various numerical algorithms needed to be executed [7]. Discussion of this theory is beyond the scope of this paper.

The overall structure of the minicomputer CAD program is probably best described by examining the program flow. The program flow is initiated by prompting the user to enter the following inputs:

1. Ranges of acceptable values for the design parameters ℓ, b, t, β plus a list of acceptable materials.

2. Desired values for the pivot performance criteria plus acceptable tolerances on these values.

The following notes regarding this input of data are given:

1. All data is entered in response to graphical prompts.

2. All data input is checked for error and then echoed together with the appropriate units. Additionally, provisions are made so that the input can be readily edited.

3. Default values for all data are available to illustrate program operation to the novice user.

Following this data input, the extreme values of the pivot design parameters are combined appropriately and then used to determine if a physically realizable pivot exists that possesses the specified performance criteria. If such a pivot exists, these results are used to initiate the multivariable interpolation that will yield the desired pivot design. If no such pivot exists, this information is returned to the user together with suggested modifications of the input design parameters that should result in a successful design. Observe that for a novice user it is not uncommon to attempt several iterations of this nature. This fact illustrates the importance of both interactive software and helpful error diagnostics in CAD programs.

Once the flexure pivot design is completed, the design is tested (numerically) to be certain that it will meet the specified performance criteria, and that its response is within the range for which the numerical algorithms are valid. Finally, this pivot design is returned to the user.

The minicomputer CAD program briefly described above was implemented on a DEC VAX 11/780 with excellent results. The program proved to be user friendly and robust. Additionally, most pivot designs were returned in a few seconds, so that computational speed appeared quite acceptable.

2. Microcomputer Design Program

The microcomputer CAD program is very similar to the minicomputer program just described; however, a few important differences need to be mentioned. Most of these differences were a result of the need to improve the computational efficiency of the program so that execution time would be acceptable on the slower microcomputer.

1. The program selects values for the pivot design parameters ℓ, b, t, and β but does not choose a material. This modification results in a significant computational savings, and it was felt that an appropriate material could be selected by the user by using the program iteratively.

2. The option of specifying a desired rotational precision was replaced with the option of maximizing rotational precision. While optimization of a given criterion is typically more expensive computationally, in the present case this change results in increased efficiency [7].

3. The option of including tolerances on the performance criteria was removed.

4. Some of the more elaborate help functions were eliminated.

5. The range of flexure pivot angular deflections over which the numerical algorithms are valid was reduced to $\pm 30°$. It was felt that this range was sufficient for the majority of applications, and this modification allowed some of the system equations to be partially linearized, resulting in increased computational efficiency.

The resulting CAD program was implemented on an IBM PC and tested extensively on a wide range of pivot design problems with excellent results. The program proved to be remarkably robust and accurate, as well as user friendly and flexible. Program execution time was generally acceptable, ranging from a few seconds to approximately 30 seconds. Improved performance was obtained by adding a mathematics coprocessor to the standard PC hardware and running a compiled version of the program. These modifications yielded very acceptable efficiency, with execution time generally less than 10 seconds.

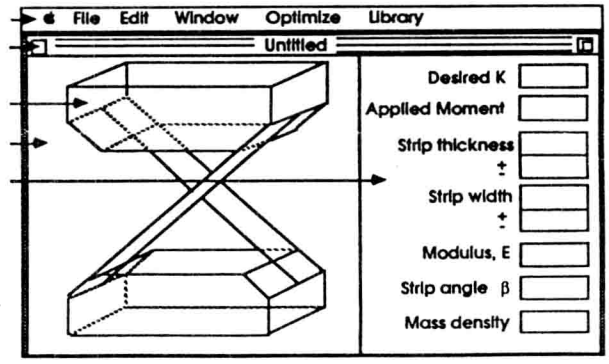
IV. INSTRUCTION OF CAD SOFTWARE DEVELOPMENT TECHNIQUES

The proliferation of CAD program packages in recent years is making the instruction of CAD software development techniques to engineering undergraduates increasingly important. While this importance is clear for students who will one day be writing their own CAD software, it is noted that it is also important for the users of commericially available CAD packages, since knowledgeable students will be better prepared to effectively evaluate and use such packages.

At New Mexico State University, one of the authors (RDC) has introduced CAD software development as a topic in the Mechanical Engineering Department's junior course Mechanical Design I. The instruction consists of a series of lectures followed by the assignment of a CAD software development project. The students are encouraged to work in groups, and are provided with a list of CAD problems drawn from local industry to aid their selection of a project. Student response to this topic has been outstanding, and the quality of work performed by the students is excellent. As an example of the level of student performance, a page from a CAD program User Manual is shown in Fig. 2. This particular CAD program was developed for flexure pivot design and was implemented on an Apple Macintosh Plus computer.

The CAD Desktop

Whenever you open a CAD document or create a new document, CAD creates a new window on the desktop. You can see the following of a typical CAD desktop:



The menu bar at the top of the screen contains the pull-down menus of the CAD comands. You can choose a comand by dragging through the menu and releasing the mouse button when the desired comand is highlighted.

Fig. 2

V. CONCLUDING REMARKS

This paper has presented a summary of a comprehensive, systematic approach to the development of high quality CAD software. Particular attention was given to I/O, error diagnostics, documentation, program structure, portability, graphics, and computational efficiency. The approach was illustrated through its application to the CAD of flexure pivots, and CAD software for both mini and microcomputers was discussed. Finally, some results of an effort to teach CAD software development to engineering undergraduates were given.

ACKNOWLEDGEMENTS

The research reported here was supported in part by the Pennsylvania Transportation Institute.

REFERENCES

1. Eberhardt, A.C. and G. H. Williard, "Coping With the Proliferation of PC CAD", *Mechanical Engineering*, Vol. 109, No. 12, Dec. 1987, pp. 58-62.

2. Middendorf, W. H., *Design of Devices and Systems*, Marcel Dekker, Inc., New York, 1986.

3. Krouse, J. K., *What Every Engineer Should Know About CAD/CAM*, Marcel Dekker, Inc., New York, 1982.

4. Colbaugh, R. D., "A Systematic Approach to the Development of Computer Aided Design Software", *CRL Technical Memorandum*, 1988.

5. Gear, C. W., *Computer Organization and Programming*, McGraw-Hill, New York, 1969.

6. Dijkstra, E., "GOTO Statement Considered Harmful", *Comm.ACM*, Vol. 11, No. 3, 1968, pp. 147-148.

7. Colbaugh, R. D., *An Analysis of the Statics and Dynamics of the Crossed Flexure Pivot*, Ph.D. thesis, The Pennsylvania State University, University Park, PA, 1986.

8. Young, W. E., "An Investigation of the Crossed-Spring Pivot", *J.of Applied Mechanics*, Vol. 11, No. 2, 1944, pp. A113-A120.

9. Wittrick, W. H., "The Properties of Crossed Flexure Pivots", *Aero. Quarterly*, Vol. 11, Feb. 1951, pp 272-292.

10. Eastman, F. S., "The Unique Properties of Flexure Pivots", *Trend in Engineering*, Jan. 1960, pp 5-11.

11. Weinstein, S., "Flexure-Pivot Bearings", *Machine Design*, June 10, 1965, pp. 150-157.

12. Flexure Pivot Reference Manual, Allied-Bendix Aerospace, New York, 1982.

Proceedings of the ISMM International Symposium,
COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS,
Honolulu, Hawaii, U.S.A., February 1-3, 1988, ACTA Press

A PERSONAL COMPUTER BASED PROGRAMMABLE LOGIC DESIGN SYSTEM CASE STUDY

David Mallery
Hewlett-Packard
Roseville, CA 95678

and

Bruce Johnson
University of Nevada-Reno
Reno, Nevada 89557

ABSTRACT

In the late 1970's all electronic computer aided design and simulation was performed on mainframe computers. Since that time, more and more of the mainframe capability has progressed down to the minicomputer and more recently to the desktop or personal computer level. Within the past two years, software has become available for the personal computer to allow the electronic designer to implement a circuit in silicon within hours of the concept.

This paper evaluates the Data I/O - FutureNet approach to the personal computer silicon foundry and compares it to other approaches on the market. The paper will clarify the many different programmable logic devices (PLDS) such as PALs, PLAs, PROMs, and IFLs and the appropriate application for each type of PLD.

The Data I/O - FutureNet System is one of the most complete systems available to the design engineer today. This paper addresses the hardware configuration and the level of support software (e.g., schematic capture and simulation) and the problems associated with interfacing to the programmable logic software and to the type of integrated circuit chosen for the design. Using a simple traffic light controller circuit as an example, the truth table approach is compared to the schematic capture input in arriving at a working silicon integrated circuit. Finally, the paper will address the problems of data file formatting and testing the PLD once it has been fabricated.

INTRODUCTION

Once available only on large mainframes and minicomputers, computer aided design (CAD) software has now been written for personal computers. One of the most unique electrical engineering applications for the personal computer, because of its completeness, is in semicustom logic design. The entire development process from design and fabrication to testing can be performed with a personal computer, CAD software and a peripheral device programmer. The Data I/O - FutureNet system is one such approach to the personal computer silicon foundry. Using semicustom integrated circuits and recently available software, an electronic designer, at his own desk, can design, simulate, implement and test a finished integrated circuit [1].

PROGRAMMABLE LOGIC DEVICES

Programmable logic devices (PLDs) are one member of a family of integrated circuits (ICs) known as semi-custom devices, that provide an alternative to the use of discrete logic gates to implement digital logic design. The PLDs are purchased from the manufacturer in an unprogrammed state; all the gates within the ICs are connected with microscopic fuses prior to programming. CAD software tools perform the function of arranging the pattern of logic gates on the device so that the circuit will perform a specific logic function required by the designer. PLDs, in turn, can be subdivided into Programmable Array Logic (PAL), Programmable Logic Array (PLA), Field Programmable Logic Arrays (FPLA), Programmable Read Only Memory (PROM), Integrated Fuse Logic (IFL) and Generic Array Logic (GAL) among others.

PLDs take advantage of the fact that any Boolean logic equation can be written in a sum-of-products format [2]. These devices consist of an array of AND logic gates connected to an array of OR logic gates. The specific logic function of the PLD is dictated by which logic gates in each array are connected following the programming process.

The PAL is composed of a programmable AND array and a fixed (or permanently connected) OR array. A PAL can replace many discrete logic gates, reduce part count and printed circuit board space and provide security for proprietary designs. Development time for a discrete TTL IC based design could be five to ten times as long as one using PLDs [1].

The PLA and the FPLA have a structure where both the AND and OR logic arrays are programmable. This creates a more versatile device and can contain larger, more complex circuits than a PAL [1]. These devices are also know as IFLs. Other IFLs include Field Programmable Logic Sequencers (FPLS) which contain flip flops with clocked outputs and feedback to implement Mealy State Machines or sequential logic [2].

Most PLDs can be programmed only one time; however, GALs (developed by Lattice Semiconductor) are CMOS PAL replacements that are electrically erasable and reprogrammable. GALS consume low power and can be easily reprogrammed to accommodate design revisions at the engineer's desk.

PROMs are constructed out of a fixed AND array and a programmable OR array. This arrangement creates a look-up table especially suited for memory applications.

MARKET OVERVIEW

A number of companies provide CAD software tools for use with the personal computer. Most of this software is written for the IBM PC, XT, AT and compatibles although there are also CAD tools for the Apple line as well. Many software companies specialize in one area of CAD software such as schematic capture.

Available schematic capture software includes FutureNet (DASH4), OrCAD Systems Corp. (OrCAD), Omation (Schema II), Visionics (EE Designer), Personal CAD Systems (P-CAD) and Case's CT series. Simulation software include FutureNet (CADAT), E/Z CAD Inc. (Plogic), Personal CAD Systems (P-CAD), Aldec Inc. (Susie) Simucad (SILOS), and CASE (Scald), as well as others [3].

PLD development software is supplied by Advanced Micro Devices (PL PL), Assisted Technology Inc. (CULP), Data I/O (ABEL), Monolithic Memories Inc. (PALASM) and Signetics (AMAZE) among others [4]. PALASM1 was the first PLD software released and has been succeeded by PALASM2 for the development of PALs. AMAZE was created to aid in the development of IFLs. The higher level software tools evolved from these such as ABEL and CUPL. These software packages will run on a large variety of different PLDs from several manufacturers.

Device programmers are either controlled by software on a personal computer such as PROMLINK (Data I/O) or are a stand alone unit such as Vatrix and Altec products. The

software approach is more versatile and more easily modified [5].

The electronics industry is moving in the direction of providing integrated software products for all phases of the design process. Most of the products listed provide some means of transferring data files. Schematic capture from one company can sometimes be passed on to a simulator of another. For example, PALASM data files can be run on an ABEL system.

THE DATA I/O - FUTURENET PERSONAL SILICON FOUNDRY

The Data I/O - FutureNet System is one of the most complete PLD development systems available to the design engineer. It is composed of a personal computer and device programmer combined with a complete set of software modules to perform the design, simulation, programming and testing of PLDs. The software set consists of DASH4-STRIDES for hierarchical schematic capture and DASH-CADAT a logic simulating [6]. DASH-ABEL is a software package that interfaces the schematic capture to the Boolean logic translator, known as ABEL. The ABEL translation language is the focal point of the PLD generation process. It is the software that provides the conversion of Boolean logic equations into fuse map data files that contain the necessary information in a standard format for the device programmer [8]. ABEL accepts input in the form of truth tables, state diagrams, logic equations or DASH-ABEL converted schematic diagrams. ABEL provides logic reduction, simulation and documentation of the design. PROMLINK is a

software package that interfaces the personal computer to the device programmer. It controls the programming functions and inputs the fuse map file to the programmer. Programming the PLD completes the manufacturing process and creates a working integrated circuit. The last software module is PLDTEST which is a fault analysis tool. It can create a complete set of test vectors for up to 100 percent testing of the programmed PLD.

CASE STUDY

An example circuit was selected to exercise the development system. The circuit was neatly and easily converted to a PLA implementation in the text "Introduction to VLSI Systems" by Carver Mead and Lynn Conway [9]. This circuit is a simple traffic light controller at the corner of a busy highway and quiet farm road. The logic is selected to maintain the greatest flow of traffic on the highway at the same time accommodating the occasional automobile on the farm road.

The circuit was entered into the CAD system in two ways. The first method was the most direct in that a truth table was generated to describe the possible logic states and then entered directly into ABEL via a text editor. This was by far the more efficient approach. The second method involved working backwards to create a schematic diagram from the truth table and it was entered into the DASH4 schematic editor. The circuit was simulated with DASH-CADAT and translated into ABEL with the DASH-ABEL software module. For ease of conversion from one software module to another the logic gates were written using LSTTL logic -- a technology accepted by all of the software modules. The schematic approach might be favored if a schematic already existed for a discrete logic implementation. However, the complexity of the circuit would be significantly greater if implemented using standard LSTTL logic gates because of the limited numbers of inputs for available AND and OR gates.

Once in ABEL, both approaches were reduced logically and fuse map files generated. The PROMLINK software package controlled the device programmer and the actual integrated circuits were programmed. The common PAL P16R4 was used. Finally, the PALs were tested using PLDtest and the percentage of the chip tested was determined. The truth table approach achieved 100 percent coverage while the schematic approach had 90.4 percent coverage.

DISCUSSION AND CONCLUSIONS

This personal CAD system is composed of an integrated set of six different software modules. Data files describing a logic circuit are passed from one module to another. The files must be of a common format such as all LSTTL logic. A circuit described in generic logic symbols typically cannot be passed from one module to another.

The choice of the particular PLD is a trial and error approach [1]. The ABEL package will indicate if a logic circuit will not fit in a particular PLD but will not help select the ideal PLD for a particular application. A good survey of available programmable logic parts is found in Reference [10]. One weakness in the system design is that ABEL's internal simulator provides only pass or fail information. Also, it is not possible to transfer an ABEL file to the full featured DASH-CADAT logic simulator.

Six different software modules with volumes of documentation need to be covered to fully utilize the system. The system's software modules are menu driven for the most part, however ABEL and CADAT require input in the form of user written programs. These programs are C language derivatives, each with their own syntax. The component libraries supplied with the system are large and contain most standard parts in a large number of technologies (TTL, NMOS, CMOS and ECL among others). For a custom part, a graphical and functional model can be created but the creation of custom components in the DASH editor can be a difficult and tedious process.

The personal silicon foundry resident on a desktop computer is a powerful and capable design tool for creating and testing relatively small logic circuits. Its greatest asset is its availability to the user and its low cost compared to larger, faster and more expensive workstations.

TRADEMARKS

REFERENCES

1. "Smart Tools to Ease PLD Design", Michael Holly, Digital Design, June, 1986
2. "Fundamentals of Logic Design", Second Edition, Charles H. Roth Jr., West Publishing Company, 1979
3. "1987 Survey of Logic Simulators", VLSI Systems Design Staff, VLSI Systems Design, February, 1987.
4. "Programmable Logic with PCs", Teschler, Machine Design, October 25, 1984
5. "A Programmable Logic CAD Station", Walter Bright and Michael A. Mraz, Southcon '84 Electronics Show, 12/3/1-16, 1984
6. "IBM PC-based Software for CAE and CAD", Eva Freeman, Associate Editor, EDN, Sept. 18, 1986, P.162-174
7. "Smoothing the User Interface For PC-Based CAE", Michael Holly, Electro '86 and Mini/Micro Northeast, 8/2/1-9, 1986
8. "Software Tools for Modern Programmable Logic Design", Dave Pellerin, Electro '86 and Mini/Micro Northeast, Conference Record
9. "Introduction to VLSI Systems", Carver A Mead and Lynn A. Conway, Addison-Wesley Publishing Company, 1980
10. "Programmable Logic Overview", E. Meyer, VLSI Systems Design, October, 1987.

Prodeedings of the ISMM International Symposium,
COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS,
Honolulu, Hawaii, U.S.A., February 1-3, 1988, ACTA Press

INTEGRATED CIRCUIT DESIGN AND SIMULATION ON A PERSONAL COMPUTER

James J. Kubinec

Manager - Strategic Development

Advanced Micro Devices

Sunnyvale, California

ABSTRACT: It was only ten years ago that the design of complex integrated circuits required hardware and software that represented between one half and one million dollars in value. Today this task is completely executable on equipment and software the value of which is less than an economy automobile.

INTRODUCTION: Perhaps nowhere in technology does the relationship between two scientific disciplines better demonstrate cynergism. The symbionic relationship between the digital computer industry and the semiconductor industry is a case study as to how each can leverage the growth of the other. The one fact that cannot be disputed is the availability of low cost, high complexity digital logic integrated circuits such as microprocessors, controllers, memories and interface drivers, and this has been a major force in the meteoric advancement of digital computers. The other side of this cynergistic relationship is the fact that design and simulation of these complex integrated circuits is impossible without high performance, economical and available computing equipment. One could construct a valid analogy with the "chicken and egg" paradox. Which came first, the high performance, low cost single chip microprocessor or the computationally powerful low cost digital computer and simulation software? Neither technology could exist without the other, and indeed their very existence is a result of mutual and interlocked evolution. The remainder of this paper will relate the importance, if not the necessity, of the personal computer for the design of integrated circuits. It would be the author's hope that the reader keep in mind as a reference point the importance, and indeed the necessity, of complex integrated circuits to the very existence of personal computers.

DISCUSSION: The design of digital integrated circuits is comprised of three distinct engineering disciplines. To practiced engineers in the field they are known as, 1) circuit design, 2) logic design and 3) topology or mask design.

Now we will examine the "state of the art" in these design activities, both before and after the personal computer era.

1) Circuit Design - This activity is best described as the interconnection of various semiconductor components to perform useful functions in the time and frequency domain. The input to the computer is a network list describing how the components are interconnected, along with mathematical descriptions as to how each responds in time and frequency (models). The object of circuit simulators is to predict with high accuracy the performance of circuits containing two to several hundred semiconductor components. The simulator is computationally extensive, but not generally a "memory hog". As a result this activity found its way early on to the PC base. Today several software companies offer circuit simulators for PCs in the few hundred dollar cost range which are every bit as accurate and useful as their mainframe counter parts.

2) Logic Design - In digital or logic circuits, the time domain variables consisting of volts and current, can be viewed as Boolean variables consisting of 1 and 0. More importantly, the circuits which accept inputs and produce outputs can be viewed as primitive logical operators (or gates) such as "and", "or" and "inverse" (or "not"). It should be clear to the reader that this transformation from circuits with volt and current variables to logic gates with 1 and 0 Boolean variables is a tremendous simplification of the problem -albeit at the expense of accuracy. As the ability to make more and more semiconductor circuit elements on a single chip of silicon grew (currently at over 100,000 transistors per chip), the need for a "logic simulator" became obvious. The first logic simulators became commercially available in the mid 1970's, ran on mainframe computers and cost about $50,000. Today several companies offer PC versions for a few hundred dollars.

3) Topology or Mask Design - All of us have looked at microphotographs of integrated circuit chips or the chips themselves and marvelled at the complexity and small size. A semiconductor process uses from 6 to over 20 mask layers to produce these chips. The graphics design of these masks (topological design) involves tremendously large graphic data bases and the manipulation of them by the designer demands extensive computation. For this reason the design of IC topology was restricted to mainframes and dedicated work stations as late the mid 1980's. However now, as a result of improved graphics in the PC world, at least seven software companies offer complete IC topology design packages on a PC. The mainframe equivalents of these programs range from $25,000 to $250,000, while the PC versions are less than $5,000.

CONCLUSION: The migration of integrated circuit design tools from mainframe computers to PCs, and the resulting cost reductions, has led to a new business form. "Design Boutiques" which consist of a handful of skilled designers with inexpensive PC design tools are springing up throughout the industry. The value which they bring to the product is their skills at innovation and product design which is conveyed through the PC and design software.

Proceedings of the ISMM International Symposium,
COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS,
Honolulu, Hawaii, U.S.A., February 1-3, 1988, ACTA Press

A CAD PACKAGE FOR DESIGN OF VARIABLE STRUCTURE CONTROLLERS

R. Swiniarski*, A. Knafel*, M.B. Zaremba**

* Department of Computer Science
University of Alabama
Tuscaloosa, AL 35487, U.S.A.

** Département d'informatique
Université du Québec à Hull
Hull, Québec J8X 3X7, Canada

Abstract: The paper presents a CAD package for interactive design and simulation of Variable Structure controllers and observers for linear and nonlinear systems with uncertain parameters. The design techniques for VSS control systems are outlined, with particular attention payed to VSS observers. The package also allows to use extended Kalman filter as an estimator of internal states.

Keywords: Variable Structure Systems, CAD, Observers.

1. Introduction.

Variable Structure Systems (VSS), due to their inherent advantages, are recently investigated by many researchers [5.], [8.], [10.], [13.]. Variable structure control is a control scheme in which the feedback gains are discontinuous in time. In such systems, the overall system dynamics is altered by high speed discontinuous switching - similar to the bang-bang time - optimal control. The states of the system tend to a special hypersurface called "switching hypersurface" or "sliding surface". When the sliding surface is reached, the control is changed and the states remain on sliding surface indefinitely (system is in sliding mode).

The following two requirements must be fulfilled in the design of variable structure control:

1. selection of appropriate hypersurface such that the system trajectories will have desirable behavior when confined to the hypersurface,
2. design of appropriate feedback gains, which will guarantee that the system will be attracted to the hypersurface and remain on it.

Theoretically, VSS control possesses excellent robustness properties in the case of uncertain parameters and unknown nonlinearity of the system. VSS applications, however, are limited by chattering of control (high control activity). Some smoothing techniques have to be applied to reduce this disadvantage [3.]. The chattering character of control is not so critical when VSS concept is applied to the design of robust observer for uncertain or nonlinear systems [2.].

Three CAD packages have been written for interactive design of VSS control systems and robust VSS observers for uncertain systems: the first one in FORTRAN 77, whereas the second and the third - in UNIX-C programming environment.

2. Variable Structure Control Systems.

2.1. Theoretical background.

Usually, the variable structure control systems assume that a plant is described by a linear

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \qquad (1.)$$

where : $x(t) \in R^n$, $u(t) \in R^m$,
$A \in R^{nxm}$, $B \in R^{nxm}$

or a nonlinear model

$$\dot{x}(t) = f(x,t) + B(x,t) \cdot u(t) \qquad (2.)$$

where : $x \in R^n$, $u \in R^m$,
$f(x,t)$: $R_+^1 \times R^n \to R^n$, and
$B(x,t)$: $R_+^1 \times R^n \to R^{nxm}$

It is assumed that $B(x,t)$ and $f(x,t)$ are continuous and that $f(x,t)$ is sufficiently smooth (assuring uniqueness of solution of (1.)).

The structure of control in VSS systems is changed upon reaching a switching hyper-surface $\sigma(x) = 0$ on the state space. The switching hypersurface is generally an n-m submanifold of R^n which assures required behavior of system when confined to this hypersurface.

The switching hypersurface can be denoted by

$$\sigma(x) = [\, \sigma_1(x), \ldots, \sigma_m(x)]^T = 0 \qquad (3.)$$

where m is the dimension of control vector $u(t)$. The control has the form

$$u_i = \begin{cases} u_i^+(x) ; & \sigma_i(x) < 0 \\ u_i^-(x) ; & \sigma_i(x) > 0 \end{cases} \qquad (4.)$$

where u_i is the i-th component of u and $\sigma_i(x)$ is the i-th of the m switching hyperplanes which satisfy $\sigma(x) = 0$; $\sigma(x) \in R^m$. The above system with discontinuous control is called a variable-structure system (VSS) since the effect of the switching hyperplanes is to alter the feedback structure of the system. A fundamental aspect of VSS is the sliding motion of the state point on intersection of the switching hyperplanes. It occurs if, at a point on a switching surface $\sigma_i(x) = 0$, the directions of motion along the state trajectories on either side of the surface are not away from the switching surface. The state then slides and remains for some finite time on the switching surface $\sigma_i(x) = 0$.

The condition of sliding motion on its hyperplane may be stated in numerous ways.

$$\lim_{\sigma_i(x) \to 0^+} \dot{\sigma}_i(x) < 0 \qquad (5.)$$

and

$$\lim_{\sigma_I(x) \to 0^-} \dot{\sigma}_i(x) > 0 \qquad (6.)$$

or equivalently

$$\sigma^T(x) \cdot \dot{\sigma}(x) < 0 \qquad (7.)$$

in the neighbourhood of $\sigma_i(x) = 0$.

($\sigma^T(x) \cdot \dot{\sigma}(x)$ is strictly negative).

When in the sliding mode, the VSS satisfies the equations $\sigma_i(x) = 0$ and $\dot{\sigma}_i(x) = 0$ and possesses several important advantages, including behavior like an n-m order dynamic system, high speed, insensitivity to variations in plant parameters and external disturbances, and simplicity of physical realization.

2.2. Design of switching hypersurface.

The design of switching hypersurface should be done regarding the dynamics of the system when it is constrained to the hypersurface. The reason for that is that the system, while confined to the switching surface, behaves like an n-m order dynamic system. The reduction of system order gives the designer the possibility to consider a reduced order linear system instead of a nonlinear one. The design of the switching hypersurface is generally based on the appropriate selection of eigenvalues of the reduced-order system [10.].

2.3. Control design.

The feedback control should guarantee reachability and existence of a sliding mode. A piecewise continuous control u(t) is a feedback type of control and is also a function of sign $\sigma_i(x)$; i = 1, ..., m. In general, u(t) is described by equation (4.).

From the appropriate theorems [4.], [10.] it is known that the control which guarantees $\sigma^T(x) \cdot \dot{\sigma}(x)$ is strictly negative, providing the existence of a sliding mode, and it is reachable on $\sigma(x) = 0$ for the entire state space. The main objective of the control design is to choose u(t) such that $\sigma^T(x) \cdot \dot{\sigma}(x)$ is less than zero. The control is frequently selected in the form [11.]:

$$u(t) = \sum_{i=1}^{n} c_i \cdot x_i(t) \qquad (8.)$$

where

$$c_i = \begin{cases} a_i & \text{for } \sigma(x) \cdot x_i < 0 \\ b_i & \text{for } \sigma(x) \cdot x_i > 0 \end{cases} \qquad (9.)$$

2.4. Variable Structure Observers.

Variable structure observers are a topic of intensive research interest. The influence of chattering - the main disadvantage of VSS - is not so dramatic in VSS observer schemes. At the same time, there are maintained the advantages of VSS systems, like robustness to parameter uncertainties and reduced order of the system when sliding on switching hypersurface. Below given is an introduction to VSS observer design.

Let us consider an observable system in the form

$$\dot{x} = Ax + Bu \qquad x(0) = x_0 \qquad (10.)$$

$$y = Cx \qquad (11.)$$

where $x \in R^n$, $u \in R^m$, $y \in R^1$ and $A \in R^{nxm}$, $C \in R^{1xn}$ are known constant matrices. It is also assumed that rank C = 1. After decomposition of the state vector and transformation [1.]

$$T = \begin{bmatrix} I_{n-1} & 0 \\ C_1 & C_2 \end{bmatrix} \qquad (12.)$$

where $C = (C_1, C_2)$ and $C_2 \in R^1$ are nonsingular matrices, we obtain state equation in the form

$$\dot{p} = A_{11}p + A_{12}y + B_1u \qquad (13.)$$

$$\dot{y} = A_{21}p + A_{22}y + B_2u \qquad (14.)$$

The VS observer is a system defined by:

$$\dot{\bar{p}} = A_{11}\bar{p} + A_{12}\bar{y} + B_1u + LK\,\mathrm{sgn}(x) \qquad (15.)$$

$$\dot{\bar{y}} = A_{21}\bar{p} + A_{22}\bar{y} + B_2u + K\,\mathrm{sgn}(x) \qquad (16.)$$

where $L \in R^{(n-1)}$ and $K \in R^{1x1}$.

Considering the fact, that only a part of state variables is directly available (i.e. y = Cx), the sliding plane can be defined as [10.]:

$$\sigma_i(x) = \bar{y}_i = y_i - \bar{y}_i \qquad (17.)$$

Matrix L should be chosen to guarantee stability of the observer in sliding mode and to specify the transient character of the observation error, i.e. to specify eigenvalue placement in the system

$$\dot{\bar{p}} = A_{11}\bar{p} + A_{12}\bar{y} - LK\,\mathrm{sgn}\,\bar{y} \quad ; \quad \bar{p} = p - \bar{p} \qquad (18.)$$

$$\dot{\bar{y}} = A_{21}\bar{p} + A_{22}\bar{y} - K\,\mathrm{sgn}\,\bar{y} \qquad (19.)$$

There are many possibilities of the choice of matrix K according to various adaptation mechanisms. In order to reduce chattering about the switching plane in sliding mode, the components of matrix K should not take big values, but they must be sufficiently large to guarantee reachability and to make the reaching phase fast. To satisfy the contradictory conditions, we have chosen matrix K in the form

$$K(y) = K_1\,\mathrm{diag}\,(|\bar{y}|) + K_2 \qquad (20.)$$

where K_1, K_2 are constant matrices of dimension 1 x 1 [8.].

For the system which has some eigenvalues greater than zero, there are certain regions in the state space from which the sliding plane cannot be reached [12.]. Since the switching plane is not freely

variable [5.] [6.], it is possible to solve this problem by adopting a "combined observer", i.e. a VS observer which has also components as a reduced observer

$$\dot{\bar{p}} = A_{11}\,\bar{p} + A_{12}\,\bar{y} + B_1 u + LK\,\mathrm{sgn}\sigma + Mv \qquad (21.)$$

where M is an (n-1) x 1 gain matrix and term $v \in R^1$ is defined as $v = \dot{\hat{y}} - \dot{\bar{y}}$. In order to avoid differentiation of system output y, let us define

$$\bar{q} = \bar{p} - M\bar{y} \qquad (22.)$$

Then the observer system takes form

$$\dot{\bar{q}} = (A_{11} - MA_{21})\,\bar{q} + (A_{11}M - MA_{21}M + A_{12} - MA_{22})\,\bar{y} + (B_1 - MB_2)\,u + (L-M)\,K\,\mathrm{sgn}\sigma \qquad (23.)$$

$$\dot{\bar{y}} = A_{21}\,\bar{q} + (A_{22} + A_{21}\,M)\,\bar{y} + B_2 u + K\,\mathrm{sgn}\sigma \qquad (24.)$$

To describe error dynamics, let us define $q = p - My$. Hence, from (17.), (23.) and (24.), error $\tilde{q} = q - \bar{q}$ satisfies

$$\dot{\tilde{q}} = (A_{11} - MA_{21})\,\tilde{q} + (A_{11}M - MA_{21}\,M + A_{12} - MA_{22})\,\tilde{y} + (M - L)\,K\,\mathrm{sgn}\,\tilde{y} \qquad (25.)$$

$$\dot{\tilde{y}} = A_{21}\,\tilde{q} + (A_{22} + A_{21}\,M)\,\tilde{y} - K\,\mathrm{sgn}\,\tilde{y} \qquad (26.)$$

When the system with nonswitchable gain only (M) can be stabilized by choice of matrix M (all eigenvalues less or equal zero), then the switching plane is reachable [12.]. It should be noted that in some cases the trajectory tends to the sliding surface asymptotically.

3. VSS CAD package.

A VSS CAD package has been designed based on the procedures outlined in previous sections. The package is a part of a larger expert system for modeling, simulation and control of uncertain linear and nonlinear dynamic systems. The software has been written in C, FORTRAN 77 and PASCAL for IBM-PC, PDP-11/40 and IBM 3801.

The VSS CAD package, primarily written in FORTRAN 77 for PDP-11/40, allows to interactively design:

- variable structure controllers for linear uncertain systems,
- VS controllers for uncertain nonlinear single-input systems,
- VS controllers for uncertain multi-input systems,
- VS observers for uncertain linear and nonlinear systems.

The updated version (written in C) of the package is enhanced by the possibility of the use of extended Kalman filter as an estimator of internal states. Another main feature offered by the updated version of the CAD package is the possibility of the design of model reference adaptive control systems (MRAC) using the concept of variable structure control [9.].

The packages are fully interactive including graphical display of results. All data are stored in external files, and the graphical part of the software can be easily modified depending on hardware requirements.

The possibility of interactive design and simulation of uncertain nonstationary systems for deterministic and nondeterministic cases is also offered.

The interactivity of the package and a broad range of input devices facilitate analysis of VSS control systems. Plant parameters as well as work conditions can be easily changed during simulation. Filtering of discontinuous control sequences, and the hysteresis region to realistically model the switcher in VSS are also provided. In the case when the plant disturbance statistics are not known, the package allows to apply adaptive Kalman filters or VS observers.

4. Conclusions.

Design techniques for VS control systems and VS observers have been briefly outlined. A VSS CAD package based on the design procedures discussed in the paper has been shortly described. The package allows to interactively design robust VS control systems and VS observers for uncertain plants.

The CAD package has been intensively tested and applied to industrial robot control and electrical dc drive control.

Further work concentrates on extension of the package in the area of nonlinear uncertain system design using global linearization.

5. References.

[1.] Ackerman, J.: Abtastregelung. Springer-Verlag, 1972.

[2.] De Carlo, R.A., Zak, S.H., Matthews, G.P.: Variable Structure Control of Nonlinear Multivariable Systems: A Tutorial. To appear in Proc. IEEE, 1988.

[3.] Dorling, V.M., Zinober, A.S.I.: A comparative study of the sensitivity of observers. Proc. ACF Symposium, Copenhagen, 1985.

[4.] Itkis, V.; Control Systems of Variable Structure. J. Wiley and Sons Inc., 1976.

[5.] Slotine, J.J., Sastry, S.S.: Tracking control of non-linear systems using sliding surfaces with application to robot manipulators. Int. J. Control, vol. 38, No. 2, 1983.

[6.] Slotine, J.J.: Sliding controller design for non-linear systems. Int. J. Control, vol. 40, No. 2, 1984.

[7.] Swiniarski, R., Knafel, A.: Computer aided design of variable structure VSS control systems. Proc. III IFAC/IFIP Symp. CADCE, Copenhagen, 1985.

[8.] Swiniarski, R., Knafel, A.: The problems concerning variable structure observers - reachability, sensitivity, reduced order switching function. (in press).

[9.] Swiniarski, R., Knafel, A., Zaremba, M.B.: Computer Simulation of Model Reference Adaptive Control of Industrial Robot. Proc. Southeastern Simulation Conf., Huntsville, Oct. 1987.

[10.] Utkin, V.I.: Sliding modes in problems of optimization and control (in Russian). Nauka, Moscow, 1981.

[11.] Walcott, B.L., Zak, S.H.: Observation and control of nonlinear uncertain dynamic systems: a variable structure approach. School of Electrical Engineering, Purdue University, 1986.

[12.] White, B.A., Silson, P.M.: Reachability in Variable Structure Control System. IEE Proc. vol. 131, Pt. D, No. 3, 1984.

[13.] Young, K.K.D.: Asymptotic stability of model reference system with Variable Structure control. IEEE Trans. on Automatic Control, vol. AC-22, No. 22, 1977.

Prodeedings of the ISMM International Symposium,
COMPUTER APPLICATIONS IN DESIGN, SIMULATION AND ANALYSIS,
Honolulu, Hawaii, U.S.A., February 1-3, 1988, ACTA Press

MATHEMATICAL MODELING OF CELL GROWTH AND HUMAN EPIDERMAL GROWTH FACTOR PRODUCTION IN RECOMBINANT YEAST

Dr. Steven J. Coppella* and Dr. Prasad Dhurjati
Department of Chemical Engineering, University of Delaware, Newark, DE 19716

* current address: Chemical Engineering Program
University of Maryland Baltimore County, Baltimore, MD 21228
and Medical Biotechnology Center
of the Maryland Biotechnology Institute
University of Maryland, Baltimore, MD 21201

KEY WORDS: model, recombinant, yeast, plasmid, heterologous, protein

ABSTRACT

Experimental data and literature information were assimilated into a mathematical description of recombinant yeast: cell growth, plasmid segregation, and heterologous protein production. Simulation results were verified with batch, fed batch, and hollow fiber bioreactor fermentations.

INTRODUCTION

Despite its long history, growth models for yeast remain incomplete. Past efforts have generally focused on specific aspects of the cell biology: cell cycle and distributions of cell states (Hjortso and Bailey 1984a, 1984b), long term respiration adaptation (Barford 1981), growth lags (Pamment et al. 1978), or the asymmetric dynamic response to step up changes in substrate feed (Lievense 1984). It was the intent of this work to summarize current information on the behavior of yeast into a mathematical model, to utilize the knowledge gained by past efforts, and to extend this model to recombinant production cultures.

A mathematical description representative of the biochemistry and physiology of the yeast *Saccharomyces cerevisiae* was formulated for growth, protein production, and plasmid segregation. The model assimilated both qualitative and quantitative literature information of the cell's metabolism. Data obtained earlier (Coppella 1987) were used to evaluate model parameters and to verify model predictions.

FUNDAMENTALS

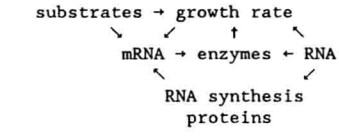
Model fundamentals were four fold. [1] Michaelis-Menten Kinetics described enzyme kinetics:

$$v = \frac{v_{MAX} S}{K + S} \qquad (1)$$

[2] Target values (steady state values) represented the value of a variable at balanced growth (where the cell could be), and were a function of the environment and the physiological state of the cell as proposed by Lievense (1984). For reaction velocity v, $(v)_T$ was the target value, the velocity at balanced growth. [3] Growth lags resulted from the slow response of the macromolecules to synthesize the proteins required to catalyze the utilization of the changing substrate resulting in cell growth. This slow response was a consequence of the transcription and translation processes as shown below:

The rate of change in a species was dependent upon the time constant which was a function of the cell state, and the distance from the target value. [4] Three catabolisms were considered: glucose fermentation, glucose oxidation, and ethanol oxidation.

Mathematical descriptions of cell growth quickly become complicated because of the strong interrelationships between cellular and environmental components as shown below:



Cell growth rate depends on the level of total RNA, enzymes, and the substrate concentrations. In turn the enzyme levels depend upon total specific cellular RNA (including rRNA for protein synthesis) and the level of mRNA from which the protein is translated. mRNA concentration is determined by the substrate levels that regulate transcription, growth rate, and RNA synthesis proteins that depend on total RNA.

The model started from material balances on cell mass, heterologous protein, substrates, and intracellular enzymes. Rate equations for catabolism were required first. Catabolic pathways were simplified and equations for substrate utilization were developed. Expressions for the steady state and dynamic behavior of the specific growth rate and enzyme synthesis were then required. The description of the dynamic behavior of these components started with an expression for RNA, then utilized the interrelationships discussed above to extend the results to the other intracellular species. Production of heterologous proteins were then described. Finally a model was formulated for plasmid segregation to complete the description of recombinant yeast.

DEVELOPMENT

Details on the simplification of catabolic enzyme pools and development of equations presented here are given elsewhere (Coppella 1987). Mass balance equations for both fed batch and hollow fiber bioreactor configurations are summarized in Table 1. The major catabolic pathways in yeast (glucose fermentation, glucose oxidation, and ethanol oxidation) were researched and simplified to develop the rate expressions needed for the mass balances. Simplification was governed by the known biochemistry to achieve the correct functionality of pathway behavior. The enzymes involved in these processes were grouped into four enzyme pools: E_G (glycolysis), E_I (alcohol dehydrogenase I), E_{II} (alcohol dehydrogenase II), and E_T (TCA). The enzyme concentrations were specific to the dry cell weight and normalized by the level at full induction or at full derepression in order to simplify the boundary conditions and to evaluate catabolic rate constants. Mass balances on the catabolic enzymes, stoichiometric equations, gas exchange rates, and rate equations are also summarized in Table 1.

Dynamic lags in cell growth from a step up in substrate concentration, result from delays in the synthesis of required enzymes. This lag results from the dynamic behavior of the synthesis of any enzyme (E_E) which was mathematically described in two steps. First target values (steady state values) of all enzyme synthesis rates (ν_E) and the total specific RNA level (N_X) were related to the steady state specific growth rate $(\mu)_T$. With these relationships, the model for the dynamic behavior of specific RNA level of Lievense (1984) was used to develop the expressions for ν_E and μ.

PROTEIN PRODUCTION

Production of hEGF from the α-factor prepro region was assumed to be a combination of growth associated and nongrowth associated kinetics (Pirt 1985). The initial medium induction lag was described by a saturation kinetic expression with the independent variable chosen to be the number of generations after inoculation (Φ). These equations are summarized in Table 1. n_o equaled the number of generations required for ½ induction of the α-factor promoter and X_R was the recombinant dry cell wt.

PLASMID SEGREGATION

P was the probability of a plasmid free progeny resulting from the division of a recombinant cell. Equal growth kinetics for host and recombinant cells were assumed as found earlier (Coppella 1987). Mass balances on recombinant and host cells for both fed batch and hollow fiber fermentations are presented in Table 1 as well as results of the probability analysis of plasmid partitioning at cell deviation. z was a correction for deviation of plasmid partitioning from random distribution, i.e. z = 1 for random distribution and z < 1 for inhibited plasmid transport to the daughter cell.

Distribution of plasmid copy numbers resulting from unequal partitioning of plasmids during cell division was not considered and is discussed elsewhere (Coppella 1987).

PARAMETER ESTIMATION

Table 2 summarized the parameter values used for simulation. The fed batch model consisted of 37 equations (13 differential equations) and 45 parameters of which 19 were estimated form the literature, 18 calculated from the experimental results presented earlier (Coppella 1987), and 8 adjusted to fit the experimental data due to the lack of available information on the enzyme pools. All the adjusted parameters were fit to the dry cell wt., hEGF, dissolved oxygen, and fraction of recombinant cells data from the batch fermentation. These values were then used to simulate the remaining batch data and the results from the fed batch and hollow fiber fermentations. Of the 8 adjusted parameters, 5 described the yeast catabolism, 1 protein production, 1 dissolved oxygen, and 1 plasmid segregation. Further discussion of parameter estimation is presented elsewhere (Coppella 1987).

MODEL SIMULATIONS

The model was solved with "DELSIM", a simulation program developed by David E. Lamb at the University of Delaware (Newark, DE). Initial and minimum step size was 0.0001 hour, maximum step size 0.05 hour, and the maximum allowed integration error was 0.5%.

Figure 1 presents the batch fermentation experimental and simulated results. Excellent agreement between model predictions and experimental data was achieved for the dry cell wt., glucose, ethanol, and hEGF throughout both growth phases (glucose fermentation and ethanol oxidation) and both lags (diauxic lag and stationary phase). Ethanol utilization was the most difficult to simulate because it resulted from two independently and actively changing enzyme pool concentrations. Similar results were obtained for additional variables (fraction of recombinant cells, dissolved oxygen, carbon dioxide production rate, oxygen uptake rate, and respiratory quotient) for batch experiments (see Coppella 1987).

Verification of simulation of these and additional variables for batch, fed batch, and hollow fiber bioreactor fermentations yielded similar results and was presented elsewhere (Coppella 1987). Also presented were hEGF optimization simulations and results of the model parameter sensitivity study.

CONCLUSIONS

A model was formulated for the growth kinetics, plasmid segregation, and heterologous protein production in *S. cerev.*. Of the 48 parameters required by the model only 8 were adjusted, others were regressed from experimental data or estimated from literature information. The model successfully described the measured dry cell wt., OUR, CPR, RQ, glucose, ethanol, dissolved oxygen, fraction of recombinant cells, and hEGF production behavior for batch, fed batch, and hollow fiber bioreactor fermentations. Simulations were then used to predict optimized hEGF production in a fed batch fermentation with constant specific growth rate. All of the cell growth parameters and initial conditions were important as seen from the sensitivity of model predictions to their value. This demonstrated that the model was well distributed and that all aspects of the catabolism described were significant and require future experimentation to elucidate behavior.

It was hoped that this model would be a useful tool in the future to design experiments to further elucidate cellular phenomenon and to control, design, and optimize industrial yeast processes.

The simplifications made by the model in the catabolic pathways represent a first order approximation. This was required because of the lack of data concerning the behavior of these enzyme pools. It is hoped that this model developed will be used as a tool in the design of experiments to elucidate the needed information. Future models should focus on the junction of the three major pathways: glycolysis, TCA cycle, and ADH-I and ADH-II. The intermediates (pyruvate, acetaldehyde, acetate, and acetyl-CoA) should first be screened to identify the critical constituents. A mass balance on the critical intermediates would allow for separate rate equations to be written for the major pathways. The resulting model should be an improved representation of the actual kinetics. However without the supporting experimentation, such a model would be very presumptuous and of limited value. In addition, information on the synthesis, degradation, and control of the major enzyme pools is also recommended.

ACKNOWLEDGEMENTS

We would like to thank Dr. Jefferson C. Lievense of the Eastman Kodak Co. (Rochester, NY), and Dr. Harold B. White of the University of Delaware (Newark, DE) for their advice. The authors wish to thank the Chiron Corp. (Emeryville, CA) for their generous donation of the cultures used for the previous experimental work. We especially acknowledge the help of Dr. James P. Merryweather, and Dr. Carlos George-Nascimento for their continuous support. Finally, we wish to thank Drs. Andrew Zydney and Michael T. Klein for their advice and encouragement.

NOTATION

$( )_T$	target value of variable, value at balanced growth
$" "_o$	variable value at t = 0, initial condition
a	mass transfer area for hollow fiber bioreactor; cm^2
ADH	alcohol dehydrogenase
a_1	stoichiometric coefficient; g glucose/g dcw
a_2	stoichiometric coefficient; g ethanol/g dcw
a_3	stoichiometric coefficient; g glucose/g dcw
a_4	stoichiometric coefficient; g O_2/g dcw
a_5	stoichiometric coefficient; g ethanol/g dcw
a_6	stoichiometric coefficient; g O_2/g dcw
b_2	constant; g dcw
c	plasmid copy number; plasmids/cell
C	O_2 concentration in liquid medium; mmol/l
C*	O_2 saturation concentration in liquid medium; mmol/l
C^*_o	O_2 saturation concentration in liquid medium at start of fermentation; mmol/l
CPR	carbon dioxide production rate; mmol/l/hr