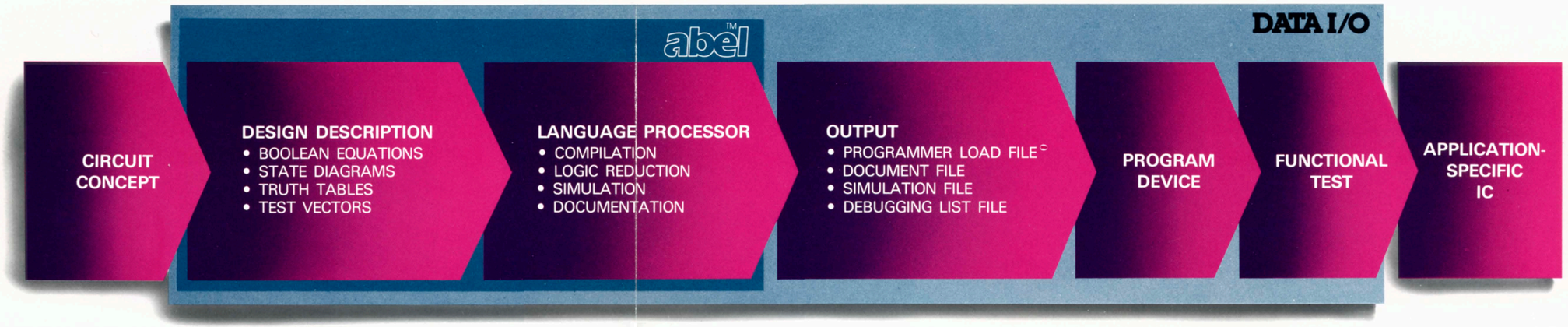
ABEL™–THE HIGH-LEVEL DESIGN LANGUAGE FOR PROGRAMMABLE LOGIC

DATA I/O

NOW YOU CAN CAPTURE ALL THE POWER OF PROGRAMMABLE LOGIC



The engineer's choice of IC design methods used to be one of extremes: standard TTL parts or custom LSI gate arrays. Standard parts are inexpensive and ready for use as soon as they're pulled from the shelf. But they consume lots of board space, and the need for a large variety of them causes large inventories. Custom gate arrays minimize board space, which aids in designing compact circuits. But their high initial cost and long lead times restrict their use.

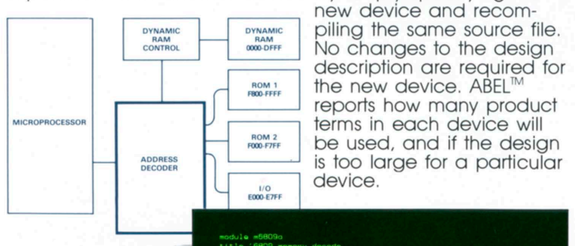
Programmable logic draws on the best of both worlds by allowing the engineer to design compact, inexpensive circuits without high costs and long lead times.

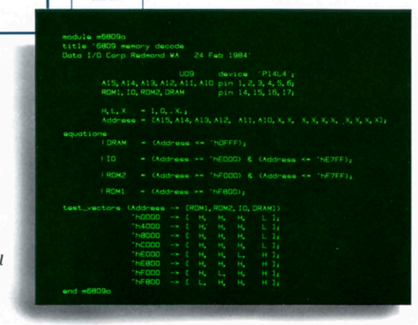
ABEL™ —Advanced Boolean Expression Language

Now there's ABEL™—a programmable-logic software tool that gives designers two capabilities never before available in one package: It compiles logic designs for any logic device—PAL®, IFL or PROM; and it lets you use the most natural form for defining circuit functions—truth tables, state diagrams or Boolean equations.

You Can Choose the Best Device for Your Circuit

Although IFLs, PALs and PROMs are functionally interchangeable in many circuits, one may be more appropriate than another in a particular circuit. ABEL™ makes that determination easy for you. Because ABEL™ doesn't limit you to one type of device, you don't have to force a design to fit a certain device architecture. You can create a design first and then select the best architecture for implementing the design. After compiling the source file that contains the design for your chosen device, you can experiment with another device by simply specifying the new device and recompiling the same source file. No changes to the design description are required for the new device. ABEL™ reports how many product terms in each device will be used, and if the design is too large for a particular device.





In this complete design for an address decoder, you can define outputs concisely by applying relational and logical operators to "Address." Test vectors are integral to all ABEL™ designs.

And, unlike other languages, ABEL™ doesn't force you to rewrite your design description to match the signal polarity of the target device; it simply performs an automatic DeMorgan transformation where necessary.

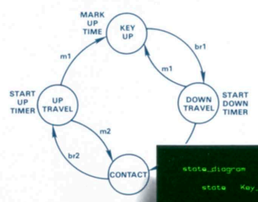
Design the Way You Think

Logic design, like art, has various forms of expression. An integral part of the artist's creation is his form of expression—water colors, oils, charcoal, pen and ink—whatever is most natural for the purpose. Similarly, the engineer's conception of a circuit suggests a "natural" way of expressing a design. State diagrams may be natural for one design, Boolean equations for another, and truth tables for another—or a combination of forms may be best.

ABEL™ is very accommodating. It lets you express your design in any combination of the three forms. No more longhand conversions of truth tables to Boolean equations; no more longhand conversions from state diagrams to Boolean equations to Karnaugh maps. ABEL™ also understands arithmetic operators (+, −, *, /) and relational operators (>, <=, etc.). You simply describe your circuit in the most natural form, and ABEL™ does the rest.

Simulate As You Design

Device simulation is critical for a fully functional design. But other languages treat simulation as an add-on and make it so unwieldy that it becomes easier—although costlier—not to bother with it. The result is that test vectors required for simulation must usually be written as a separate source file and expressed in a different syntax from your design.



State machines translate directly to ABEL™ state diagrams. No more tedious translation of state diagrams to Boolean equations or Karnaugh maps.

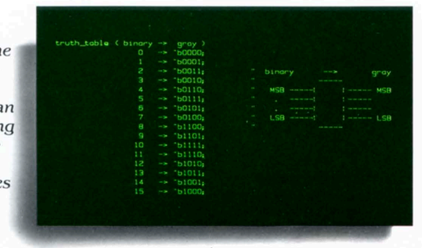
But with ABEL™, simulation is integral to design. You can write test vectors in the same source file and with the same syntax as your design. This interactive design and simulation, or "design loop," lets you create part of a design, write test vectors, and test the design by simulation. In this way you can create and test new design ideas simultaneously, and you debug your design as it evolves.

Because ABEL™ lets you evolve the testing procedure along with the design, you are assured that your final product—the programmed device—is fully testable.

ABEL™ Gives You Design Power

ABEL's high-level language constructs help you minimize design effort and eliminate unnecessary design restrictions. For example, "set" notation lets you group whatever signals you require into sets and treat each set as a separate unit in your design description. Free-format syntax lets you write your design without having to assign design elements to specified line or column numbers. With macros and directives you can instruct the compiler to create blocks of text automatically. And with file inclusion you can merge an existing file with the source file you are writing, which allows you to build a library of common files and avoid redundant keystrokes.

ABEL™ also incorporates a powerful logic-reduction algorithm that reduces the number of gates—and the cost of the hardware—needed to implement your design. This algorithm also makes state diagram design easy—while in other languages it's not even possible.



Truth tables conveniently define many designs, including code converters. You can specify codes using any number base; ABEL™ automatically generates the Boolean equations.

And for thorough record-keeping, ABEL™ produces comprehensive documentation, including reduced equations, fusemaps, labeled IC diagrams, production and service data, and test vectors.

Design Has Never Been So Easy (or Designers So Productive)

ABEL™ also offers powerful, time-saving features that you can rightly expect in a high-level language. For example, it lets you create designs for more than one device from a single source file. You save time by eliminating needless steps, and you streamline your system design by reducing the number of source files.

ABEL's high-level language functions make design easier than with first-generation languages because they allow you to express designs more succinctly. And ABEL™ can easily handle complex designs not possible with first-generation languages.

(Incidentally, ABEL™ also contains a PALASM-to-ABEL converter for existing PALASM files.)

ABEL™ is the first language to let you specify a target device (or devices) either in your design description or before a compile operation. This allows you to experiment with different devices as you develop your design. When you have selected the best devices, you can store them permanently in your design description.

Programmable logic has never been so ABEL™.

abel™

DATA I/O

ABEL™ is Compatible with Most Systems

Chances are that ABEL is compatible with your system right now. ABEL runs on MS-DOS-compatible PCs such as the IBM Personal Computer. It also runs on the DEC® VAX™, and other versions are coming.

And because ABEL load files conform to JEDEC standard JC-42.1-81-62, they can be downloaded to compatible logic device programmers. (Data I/O's Programmable Logic Development System provides design, programming and testing of over 90 logic devices.)

Data I/O Provides Ongoing User Support

ABEL's open-ended architecture makes updates for new devices easy. Joining Data I/O's subscription service assures you of prompt software updates for new devices and enhancements.

Data I/O maintains over 65 sales and representative offices worldwide to answer your questions or problems.

---

*GENERAL SPECIFICATIONS**
Compatible Computer Systems

Personal Computer (distributed on double-sided, double-density 5¼" floppies)

- MS™-DOS or PC-DOS operating systems
- 128k or larger memory
- 192k with two disk drives, recommended minimum

DEC VAX (distributed on 1600 bpi magnetic tape)

- VMS™ operating system
- UNIX™ operating system

Devices Supported

Most 20- and 24-pin PALs
40- and 84-pin PALs capacity in 1984
All IFLs: FPLA, FPLS and FPGA
Most popular logic PROMs

Output Format

For PALs and IFLs: Conforms to Joint Electron Device Engineering Council (JEDEC) JC-42.1-81-62 Logic Device Translation Format

For PROMs: Intel Intellec 8/MDS Translation Format
Motorola Exorciser Translation Format

*For detailed specifications, see the ABEL specification sheet.

ABEL — Tutorial Disk

See for yourself why ABEL is today's state-of-the-art logic-design software. Send $10 with the enclosed order form for an IBM PC-compatible ABEL tutorial disk.

PROMlink™

Now with Data I/O's PROMlink software driver, you can operate Data I/O's programmers from an IBM PC. This new software package gives you the ease of menu-driven operation and data file management. Call your nearest Data I/O sales office or representative for more information.

Data I/O
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
(206) 881-6444

U.S. REGIONAL SALES FACILITIES

Northwest Region
1700 Wyatt Drive
Suite 1
Santa Clara, CA 95054
(408) 727-0641

Southern Region
1810 N. Glenville Drive
Suite 108
Richardson, TX 75081
(214) 235-0044

Western Region
2770 South Harbor Blvd.
Suite K
Santa Ana, CA 92704
(714) 662-1182

Eastern Region
Nashua Road
Route 101A
Amherst, NH 03031
(603) 889-8511

INTERNATIONAL SALES FACILITIES

Data I/O Japan
Ginza Orient Building 6-F
8-9-13, Ginza Chuo-ku
Tokyo 104, Japan
(03) 574-0211

Data I/O Europe
Vondelstraat 50-52
1054 GE Amsterdam
The Netherlands
(20) 186855

Data I/O Germany GmbH
Bahnhofstrasse 3
D-6453 Seligenstadt
West Germany
(6182) 3088/89

CANADIAN REPRESENTATIVE

Allan Crawford Associates, Ltd.
6503 Northam Drive
Mississauga, Ontario
L4V1J2
(416) 678-1500

DATA I/O

0736/40K/784