Presented at the CUBE Symposium,
Lawrence Livermore Laboratory,
Livermore, California,
October 23-25, 1974.

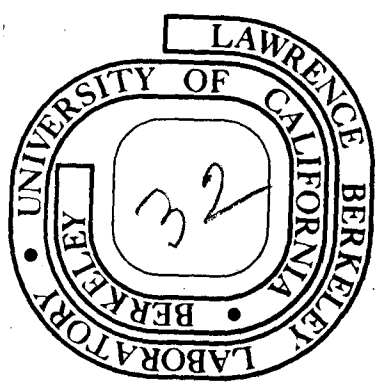
LBL-3392
c.6

AN INTRODUCTION TO LLCAD
(LAWRENCE LABORATORY COMPUTER AIDED DESIGN)

Joseph E. Katz and Van L. Jacobsen

October 1974

Prepared for the U. S. Atomic Energy Commission
under Contract W-7405-ENG-48



LBL-3392
c.2

DISCLAIMER

AN INTRODUCTION TO LLCAD (LAWRENCE LABORATORY COMPUTER AIDED DESIGN)

Joseph E. Katz and Van L. Jacobsen,
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

ABSTRACT

LLCAD is a mostly FORTRAN computer program designed to relieve the user of the tedious jobs associated with the design, production, documentation, and maintenance of digital systems. LLCAD is written to operate on the CDC 7600 computer systems at the Lawrence Berkeley and Livermore Laboratories. This paper will describe the over-all structure and operation of the program. The input processor and report generator portions of the code are described, and a general description of the LLCAD language and an example of its use is provided.

INTRODUCTION

LLCAD, Lawrence Laboratory Computer Aided Design Program, is intended to relieve the user of the many tedious and routine steps in the digital design process. The memory, speed and consistency of a high powered digital computer are used to aid the designer to greatly reduce clerical errors, to do the humdrum table look-ups for pin assignments, and to provide a tireless source of documentation. Designers, by means of LLCAD, may proceed from the system specification and logic diagram stage to the debugging phase via a short detour to the computer center.

The LLCAD program was originally developed at the Stanford Research Institute by Kaye Tomlin, Ralph Keirstad, and John Yarborough. The SRI portion of LLCAD, the input processor, fills a data structure from the input description of a design. SRI also supplied a specification for the Report Generators which were written at the Lawrence Berkeley Laboratory.

This paper will provide the reader with a brief introduction to the program. Included in the introduction are a discussion of the LLCAD code, an example of how it is used, and what kind of results it can produce. The description of the LLCAD program will include a short description of the input processor code. The structure of the Report Generators and their operation are briefly described.

The reader is provided with a general introduction to the language rather than an exhaustive complete discussion of the LLCAD syntax. The introduction is illustrated by means of the input code for a simple design. The outputs produced by LLCAD are described and an example is provided.

PROGRAM DESCRIPTION

LLCAD is divided into two major sections: A syntax processor and a report generator. The syntax processor accepts a format free card image input file and converts it to fixed format output files and tables. The report generator does a sequence of sorts on these files to distribute common information, delete redundant items, and output reports. Extensive error checking is done in both phases: the syntax processor checks for both incorrect syntax and logically inconsistant description (invalid address specification, trying to use an undefined pluggable unit, etc.); the report generator checks for inconsistant design and certain design errors (multiple pluggable units at the same address, excess load on an output signal, etc.).

The syntax processor comprises the bulk of LLCAD. It consists of approximately 10,000 source lines, contains about 200 subroutines, and requires 140k octal (48k decimal) words of core to run on a CDC 7600. The code was initially written entirely in ANSI - 1966 standard FORTRAN IV and a significant effort went into trying to achieve machine independance. Another large effort was invested in making maximum use of symbolic definition with the pleasant result that the code is both easily extensible and readily transportable.

LLCAD is essentially keyword driven. Each major class of statement starts with a unique keyword [CHASSIS, LOGICAL, etc., see tables 1 through 3] which causes LLCAD to initiate a subroutine unique to that keyword. This subroutine is then responsible for the rest of the processing of the particular statement. This is not the standard method of syntax processor implementation and serves to increase both the size and complexity of the code, but it allows for extremely detailed error messages and for the detection of a very large number of possible errors. Presently, about 300 error messages exist in the syntax processor, representing over 1000 errors or anamolous conditions.

The report generator provides the majority of useful output of LLCAD and does further error checking on aspects of the design. There are three types of processing done in the report generator: Most of its time is spent collating (sorting) the primary design description file on its various fields, then checking that impossible or inconsistant combinations haven't been specified. (For example, a sort will be done to group all records with the same first two address levels. For each of the groups of records, the pluggable unit description is checked to make sure that multiple pluggable units haven't been specified for the same address, then the pluggable unit information is put into all records in the group that don't already contain it.) The second type of processing done by the report generator is to produce the various reports requested by the user. This consists of simply sorting the design file into an appropriate order (by signal name for a load-check list, by 3-level address for a pluggable unit directory, etc.), then reading sequentially through the sorted file to output pertinent information from each record. The last, and potentially most interesting, type of report generator processing involves analysis of aspects of the design to supply unspecified information or to optimize some set of parameters. Only two analysis processes are currently implemented: one computes near-minimum length wire-chains from the signal interconnections specified in the design, the other assigns unused circuits on pluggable units to logical elements of the design (the logic blocks defined later in this paper).

INPUT LANGUAGE INTRODUCTION

The user communicates with LLCAD by means of a set of input data statements. These statements are read in a format free fashion which ignores blanks, blank cards, and card boundaries.

Each LLCAD Statement starts with a keyword which specifies the particular type of Statement to be processed. Each Statement is terminated by a semi-colon (;). The following discussion will treat the LLCAD syntax in a very general manner. A much more specific treatment of the syntax is provided in the LLCAD User's Manual.[1]

The language may be divided into five groups of Statements that perform related functions. In order to set the stage for a description of the LLCAD Statement type groupings we shall pause to consider at which point in the digital design process one would use this program.

In order to satisfy a set of system specifications a designer soon finds himself with a sketch or many sketches, of a logic diagram consisting of some combination of gates, flip-flops, adders, shift registers, etc. The implementation of this design is, of course, dependent on some choice of hardware type, indicators, cabling, connectors, etc., and the selection of a family of logic devices (pluggable units). After the above choices have been made, the LLCAD program may be used to relieve the designer of such tiresome and error-prone tasks as the production of wire-lists, the checking of loads on output drivers and the production of required tables of documentation.

STATEMENT TYPES

LLCAD is not limited by any particular family group of logic device nor is it restricted to any particular hardware type. The first group of LLCAD syntax statements are a set of Hardware Statements used to describe the hardware environment. Sets of hardware descriptive statements may be gathered in libraries so the user will not need to regenerate, at each time of use, new code for common hardware types.

To implement any given design, it is possible to choose from a wide selection of logical devices (pluggable units). The LLCAD Pluggable Unit Statements used to describe the pluggable unit environment, are capable of describing logical devices

ranging from an individual integrated circuits in dual-in-line packages (DIPs) to an array of DIPs on a printed circuit card. As in the case of common hardware types, it will be found convenient to assemble libraries for the common pluggable unit families.

Once the user has told LLCAD what kind of hardware and pluggable units are to be used, the task of describing the interconnection of the logical elements remains. This task is done by a set of Statements called Assignment Statements. They are the most important and frequently used statements in the LLCAD syntax. Assignment Statements are used to describe the interconnection of logical elements by use of signal names, to place pluggable units at particular hardware addresses, to specify wire-types, to specify wire-or connections, and to describe connections to input or output connectors.

To assist the user in describing digital designs in LLCAD syntax, there are two Statements in the input language that facilitate the description of repetitive structures. The first is the FOR Statement which is similar to the ALGOL FOR or the FORTRAN DO Statements. The second is the DEFINE Statement which operates like a text replacement macro. These two types of "Shorthand" Statements are more fully described in the LLCAD User's Manual.[1]

The fifth group of LLCAD statements introduced are the Control Statements. They are used to control the operation of LLCAD by selecting default options, output reports, etc.

Instead of attempting a detailed description of each of the Statement types, an example is presented to show how they are used. By means of this simple design example, many features of the language will be illustrated.

EXAMPLE

Consider the simple design shown in Fig. 1. For this discussion, it is decided to implement this design with the Control Logic type of hardware. The Hardware Statements sufficient to describe this environment are shown in Table 1. A three-level addressing scheme is used. The CHASSIS Statement [see code line (1)] specifies a backplane of up to three bins (named A to C), each bin may have up to forty card slots (named 1 to 40), and each card may have as many as 44 pins (with the pins named as shown in the CHASSIS Statement). The SOCKET Statement describes the pin coordinates and identifies the forty-four pins on the connector named 2X22 [see

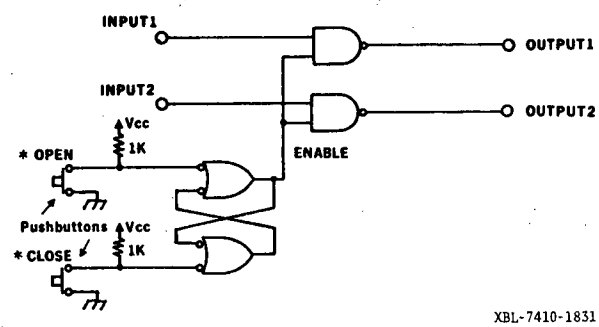


Fig. 1. Simple design.

Table 1: Hardware Statements

```
CHASSIS BIN<A,B,C>, CARD<1 TO 40>,                                    (1)
        PIN<1 TO 22,A TO F,H,J TO N,P,R TO Z>;

SOCKET 2X22, SIZE = (0.14/3.276),                                     (2)
       PIN = <1 TO 22,A TO F,H,J TO N,P,R TO Z>(0.0[22],0.14[22]/
       <0.0 TO 3.276 BY 0.156>[2]);

TEMPLATE BIN TYPE = CL (SOCKET = 2X22,                                (3)
PLACEMENTS = <1 TO 40>(0.0 TO 15.6 BY 0.4/0.0[40]));

BAY 1 = <A TO C>;                                                     (4)

LAYOUT A = CL (0.0/0.0),                                              (5)
       B = CL (0.0/3.276),
       C = CL (0.0/6.552);

LENGTH WIRE = <1 TO 19>(0.315,0.625,0.94,1.2,1.5,2.0 TO 3.53 BY       (6)
                        0.51,4.22,4.89,5.6 TO 9.1 BY .7,10.1,11.1/
                        3.5 TO 4.5 BY 0.25,5 TO 6.5 BY 0.5,7.25 TO
                        12.5 BY 0.75,13.5,14.5);
```

code line (2)]. The TEMPLATE Statement locates the sockets in the bin, while the BAY and LAYOUT Statements complete the specification of the coordinates of the three bins on the backplane [see code lines (3) to (5)].

The LENGTH Statement [code line (6)] is included in this hardware description set but it is not peculiar to the Control Logic hardware. This Statement specifies a table of nineteen pin-to-pin lengths and the corresponding nineteen pre-stripped wire lengths. The information given by the LENGTH Statement is used by LLCAD to produce a numerical control tape for a semi-automatic wire-wrap machine.

The Pluggable Unit Statements necessary to implement the simple design are shown in Table 2. The POWER Statement [code line (7)] defines the power supply bus names. The Control Statement shown in code line (8), selects an option that lets LLCAD assign circuit numbers automatically. The sample code shown in lines (9) and (10) describe a twelve 2-input nand gate logic card, Control Logic type CNG-152T. The ELEMENT Statement [see code line (9)] specifies the output and input connections for a device named N2T. The ELEMENT Statement also specifies: the FANOUT (drive capability of the N2T output); the LOAD (input loading of the N2T); and finally, the PREFERRED CARD (the pluggable unit that contains elements of the name N2T is a CNG-152T). Note, the drive capability of the N2T is 10 but for report purposes, to be shown later in this paper, an artificially reduced value of 1 has been used.

The LOGICAL Statement [see code line (10)] identifies the type of socket that the pluggable unit uses, specifies the power pins, and gives the total power supply drain for the pluggable unit. The LOGICAL Statement also assigns a circuit function to each of the pluggable unit pins.

In this example, the FOR Statement imbedded in sample code line (10) is used as a "shorthand" convenience. The four lines marked (10a) to (10e) are

Table 2: Pluggable Unit Statements

```
  POWER VCC,GND;                                                    (7)
  ASSIGN CIRCUIT-NUMBERS;                                           (8)
// TWELVE 2-INPUT NAND GATES CNG-152T    //
  ELEMENT                                                           (9)
  (OUT) = N2T(INPUT1,INPUT2),
  FANOUT = 1, LOAD = 1, PREFERRED CARD = CNG-152T;
  LOGICAL CARD CNG-152T,                                            (10)
  SOCKET = 2X22, POWER=(Y=VCC, 21=VCC, Z=GND, 22=GND),
  DRAIN = 45,
  FOR↑ I = 1,6,C,D,9,10,K,N,13,16,S                                 (10a)
  AND↑ J = 2,4,A,E,7,11,H,L,14,17,P                                 (10b)
  AND↑ K = 3,5,B,F,8,12,J,M,15,18,R                                 (10c)
  < <      ($↑I.2$) = N2T($↑J.2$ ,$ ↑K.2$), >>                      (10d)
  END FOR   END OF I, J, AND K LOOP;                                (10e)
   (V ) = N2T(T  ,U )
  END CNG-152T CARD ;
```

Expanded Code Produced by FOR Statement, Lines (10a) to (10e)

```
  (01) = N2T(02,03) ,
  (06) = N2T(04,05) ,
  (C ) = N2T(A ,B ) ,
  (D ) = N2T(E ,F ) ,
  (09) = N2T(07,08) ,
  (10) = N2T(11,12) ,
  (K ) = N2T(H ,J ) ,
  (N ) = N2T(L ,M ) ,
  (13) = N2T(14,15) ,
  (16) = N2T(17,18) ,
  (S ) = N2T(P ,R ) ,
```

expanded by the input processor to appear in the input text stream as shown at the bottom of Table 2. Note, that the completed LOGICAL Statement assigns each of the twelve outputs and their corresponding inputs to pins of the pluggable unit.

The sample code shown in Tables 1 and 2 constitutes a description of the Control Logic hardware and a common pluggable unit. The code shown in lines (1) to (10) may be used many times over to implement other designs with this type of hardware. Of course other pluggable units can be added to the library as required.

Assignment and Control Statements to complete the description of the simple design are shown in Table 3. The LLCAD syntax to describe the interconnection of the logical elements is in the form of logic-like equations. Simple statements are used to assign pluggable units and to describe input and output connections. From these simple statements [lines (11) to (17)] all of the pin assignments and documentation necessary to describe this simple design is produced by LLCAD.

A pluggable unit containing twelve 2-input nand gates is placed at Bin A, Card slot 1 by code line (11). Code lines (12) to (15) describe all of the wiring shown in the simple design [see Fig. 1]. All input and output connections to the push buttons and to connector PG-1 are described by code lines (16) and (17).

The Control Statements [code lines (18) to (20)] direct LLCAD to produce two outputs. One is a paper tape to control a semi-automatic wire-wrap machine and the other is an output listing called the Load Check Listing, [see Fig. 2]. The Load Check Listing is an example of one of the many types of output reports that are available. The User's Manual[1] describes these reports in more detail.

The Load Check Listing [Fig. 2] is sorted on the signal names that appear in input design description [see Table 3]. A separate output line is provided for each occurence of the signal name. The signal source address will be on the first line of output for any signal name. Many attributes of the signal name are listed for each address. The column labeled F, for function, will contain an S for a source, or an L for a load. The column labeled F/L, for fanout or load, contains the value specified in the pluggable unit description, or the arbitrary value of 99 for the fanout of an external source. LOGIC BLK, refers to the Logic Block identifier which LLCAD automatically assigns to each element. The Logic Block identifier is used by LLCAD to differentiate among identical elements. This simple design, for example, uses four N2T elements named ENABLE, *ENABLE, OUTPUT1 and OUTPUT2 [see the Load Check Listing, Fig. 2]. A discussion on the naming of Logic Blocks is provided in the User's Manual.[1] ELEMENT, CKT, and USE all are referenced back to the pluggable unit description. REMARKS are a reprint of comments provided in the Assignment Statements [see Table 3] by the user. The

Table 3: Assignment and Control Statements

```
USE CNG-152T AT A-1;                                                    (11)
(ENABLE) = N2T(*OPEN,*ENABLE) /CROSS COUPLED GATE/;                     (12)
(*ENABLE) = N2T(*CLOSE,ENABLE) /CROSS COUPLED GATE/;                    (13)
(OUTPUT1) = N2T(INPUT1,ENABLE);                                         (14)
(OUTPUT2) = N2T(INPUT2,ENABLE);                                         (15)
EXSOURCE *OPEN AT PB-1-NO,                                              (16)
         *CLOSE AT PB-2-NO /PUSH BUTTON SWITCHES/,
         INPUT1 AT PG-1-A,
         INPUT2 AT PG-1-B;
EXLOAD OUTPUT1 AT PG-1-D,                                               (17)
       OUTPUT2 AT PG-1-E  /CONNECTIONS VIA PG-1/;
//CONTROL STATEMENTS //
MAKE RUN;                                                               (18)
PRINT LOADS;                                                            (19)
PUNCH NC-LIST;                                                          (20)
```

```
      SIGNAL          F  F/L   ADDRESS    LOGIC BLK   ELEMENT   CKT USE      REMARKS                LINE
      ------          -  ---   -------    ---------   -------   --- ---      -------                ----
     *CLOSE           S   99  PB-002- NO  *CLOSE      EXTERNAL                PUSH BUTTON SWITCHES    43 1
     *CLOSE           L    1  A-001-004   *ENABLE     N2T         2 INPUT1    CROSS COUPLED GATE      39 1

     *ENABLE          S    1  A-001-006   *ENABLE     N2T         2 OUT       CROSS COUPLED GATE      39 1
     *ENABLE          L    1  A-001-003    ENABLE     N2T         1 INPUT2    CROSS COUPLED GATE      38 1

      ENABLE          S    1  A-001-001    ENABLE     N2T         1 OUT       CROSS COUPLED GATE      38 1
      ENABLE          L    1  A-001-005   *ENABLE     N2T         2 INPUT2    CROSS COUPLED GATE      39 1
      ENABLE          L    1  A-001-  B    OUTPUT1    N2T         3 INPUT2                            40 1
      ENABLE          L    1  A-001-  F    OUTPUT2    N2T         4 INPUT2                            41 1
****** EXCESS LOAD =  2.0

      INPUT1          S   99  PG-001-  A   *INPUT1    EXTERNAL                                        44 1
      INPUT1          L    1  A-001-  A    OUTPUT1    N2T         3 INPUT1                            40 1

      INPUT2          S   99  PG-001-  B   *INPUT2    EXTERNAL                                        45 1
      INPUT2          L    1  A-001-  E    OUTPUT2    N2T         4 INPUT1                            41 1

     *OPEN            S   99  PB-001- NO  *OPEN       EXTERNAL                                        42 1
     *OPEN            L    1  A-001-002    ENABLE     N2T         1 INPUT1    CROSS COUPLED GATE      38 1

      OUTPUT1         S    1  A-001-  C    OUTPUT1    N2T         3 OUT                               40 1
      OUTPUT1         L       PG-001-  D   *          EXTERNAL                                        46 1

      OUTPUT2         S    1  A-001-  D    OUTPUT2    N2T         4 OUT                               41 1
      OUTPUT2         L       PG-001-  E   *          EXTERNAL                CONNECTIONS VIA PG-1    47 1
```

1 ERROR(S) IN ABOVE REPORT. PAGE NUMBERS OF FIRST 50 PAGES CONTAINING ERRORS FOLLOW.

1,

Fig. 2. Load check listing.

last column, LINE, is a line number reference back to the input code listing printed by LLCAD. The input code shown in Tables 1, 2, and 3, is printed during execution by the input processor of LLCAD with line numbers automatically generated.

Note, this Load Check Listing has an error message included for the deliberately generated error. It is seldom the case that one has to go to such extremes to see error messages or noteworthy conditions reported by LLCAD. Conditions that generate error or noteworthy condition messages are described in the User's Manual[1].

ACKNOWLEDGEMENTS

Many of our associates both at LBL and LLL have contributed to the development of this program. The original funding to purchase the SRI portion of the code was jointly provided by both the LBL and LLL Electronics Engineering Departments.

At LBL, the continued support and encouragement of the Head of the EE Department, Dick A. Mack, is gratefully acknowledged. The engineering advice of Douglas L. Abbott and the programming aid of David Jensen were of great assistance. Both Waldo Magnuson, Jr., and Herschel H. Loomis, Jr., of LLL, participated in the design of the input syntax and contributed to the overall program development.

This work was performed under the auspices of the U.S. Atomic Energy Commission.

REFERENCES

1. LLCAD (Lawrence Laboratory Computer Aided Design) User's Manual, Joseph E. Katz and Herschel H. Loomis, Jr., UCID-3688, Nov. 1974.

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720