Quicklist—The Basis for a Computer-Aided Logic Design System

W. A. Lushbaugh
Communications Systems Research Section

The description of a digital system must eventually include a point-to-point wire list. Previous methods of computer-generated wire lists have required one record of input for each point to be wired. Quicklist is a preprocessor for an existing wire listing program that cuts the description of the wire list by about 67 percent over previous methods. Quicklist is intended to be the basis of a computer-aided logic design system.

I. Introduction

Many digital systems are constructed on wire wrap boards. These boards are most conveniently wired by automatic or semiautomatic wire wrap systems for large quantity productions, but many prototype and small quantity boards are still wrapped by hand. In any event, a computer-generated wire list is advantageous to expedite the work. Most wire listing programs, however, require one record of input for each point to be wired. Quicklist is a preprocessor for an existing wire list program that cuts the description of the wire list by about 67 percent over previous methods. Quicklist is intended to be the basis of a computer-aided logic design system.

II. Review of Wire List Program

H. C. Wilck has written a program that has proved very useful in providing wire listing information for the wire wrap board used by the Digital Projects Group. This program is a very advanced program in that it allows inputs in the form of chip signal names rather than actual chip pin numbers, although the user may call out chip pin numbers if desired. The method by which this is accomplished is to provide a definition deck in front of the wiring information. Some of the highlights of this program follow:

The definition deck has the following format:

D, type, number of pins, number of defined pins, spacing, description

1, type, pin number – pin function, pin number – pin function, . . .

2, type, pin number – pin function, . . .

.
.
.

D, type, . . .

where the terms have the following definitions:

Type: 1 to 8 alphanumeric characters used to designate the dual in-line package (DIP); e.g., SN 7400.

Number of Pins: A decimal number (NP) specifying the number of pins on the DIP. $2 \leqslant NP \leqslant 88$.

Number of Defined Pins: A decimal number (NDP) specifying the number of pin definitions that follow. If NDP < NP then there are pins that the manufacturer has not connected.

Spacing: A decimal number (SP) specifying the spacing between the two columns of pins on the DIP in tenths of an inch.

Description: 0 to 60 alphanumeric characters describing the nature of the package; e.g., DUAL J-K FLIP-FLOP.

Pin Number: A decimal number (N) that designates the pin to be named. $1 \leqslant N \leqslant NP$.

Pin Function (name): 0 to 4 alphanumeric characters. This name describes the electrical function of the pin; such as OUT1, A0, A1, IN3, Q1′, etc. The pin function may be used in lieu of the pin number when the wire listing data is given to the program.

The SIGNAL ASSIGNMENT DECK is the actual wire routing information to the program. This deck is chip oriented, i.e., input is grouped by DIP's, with all signals on one chip entered as a group. The format of this deck is:

P, LOC, Type, Function

1, LOC, pin id(entification) – signal name, pin id – signal name, . . .

2, LOC, pin id – signal name, . . .

.

.

.

P, LOC, Type, . . .

where these terms have the following definitions:

LOC: A five-digit number specifying the location of the corner pin (pin 1) of the chip on the board.

Type: 1 to 8 alphanumeric characters. This must agree with one of the types in the definition deck.

Pin id: Either the actual pin number of the chip *or* the mnemonic pin function assigned to that pin in the definition deck.

Signal name: 1 to 8 alphanumeric characters that identify the signal to be connected to the pin specified by the pin id. Every pin tied to the same signal name will be tied together by the wire listing program.

Special Characters. Both in the definition deck and the signal assignment deck the dash (-) between pin number – pin function or pin identification – signal name may be replaced by other special signals. These are:

& – defines the pin to be an output that can be collector or-ed or tri state.

* – defines the pin to be an output that cannot be tied to another output.

+ – defines the pin as connected to +5V.

> – defines the pin to be connected to ground.

III. Quicklist

Quicklist is a preprocessor program written to shorten the input signal assignments to the program described above. Quicklist uses two different methods to shorten the input data. The first of these is the use of parentheses to allow a compact way of describing signal names which increment (or decrement) by one, i.e., typical signal names in a bus. The other method used is the equal sign. Using this symbol in a signal list says that the rest of the signals on that chip are the same as those on the previous chip described. Inputs to Quicklist look very similar to those described above; i.e., a typical signal assignment deck is:

P, LOC, type, function

1, LOC, pin id – signal name, . . .

except that cards with numbers in col. 1 may take the form

1, LOC, pin id $(N_1 - N_2)$ – signal name $(N_3 - N_4)$, . . . , =

where pin id $(N_1 - N_2)$ is the name (as called out in the definition deck) of a *set* of pins that have names that go in order from N_1 to N_2. For example, for the 2147 memory chip A00, A01, . . . A11 are the 12 address inputs. Similarly, signal name $(N_3 - N_4)$ is the name of a set of signals that run from say $SIGN_3$ to $SIGN_4$; e.g., ADBUS 00 to ADBUS 11. As an example, consider two 2147 chips wired to the same address buss, write enable and chip select. The Quicklist input would then be:

P, LOC, 2147, Memory 1

1, LOC, DIN-DATA0, DOUT-DOUT0

2, LOC, A(0-11) – ADBUS(0)

3, LOC, WE-WRTEN, CS-CHPSEL

P, LOC1, 2147, Memory 2

1, LOC1, DIN-DATA1, DOUT-DOUT1, =

The output from these five input cards would be:

P, LOC, 2147, memory 1

1, LOC, DIN-DATA0; DOUT-DOUT0

2, LOC, A00-ADBUS00, A01-ADBUS01, . . .

3, LOC, A04-ADBUS04, . . .

4, LOC, A08-ADBUS08, . . . , A11-ADBUS11

5, LOC, WE-WRTEN, CS-CHPSEL

P, LOC1, 2147, memory 2

1, LOC1, DIN-DATA1, DOUT-DOUT1

2, LOC1, A00-ADBUS00, A01-ABUS01, . . .

3, LOC1, A04-ADBUS04, . . .

4, LOC1, A08-ADBUS08, . . . , A11-ADBUS11

5, LOC1, SE-WRTEN, CS-CHPSEL

In this case, the 16 nonpower and ground signals of the first chip were described with three short inputs cards, and the second chip was described in only two cards due to the use of the equal sign.

It should be noticed that Quicklist provides automatic double digiting; i.e., if either N_1 or N_2 is a two-digit number, then all the numbers from N_1 to N_2 are put in double-digit format. The same is true of the pair N_3 and N_4. The use of N_2 or N_4 is optional, but at least one of them must appear. To clarify this, the following input strings all produce identical output:

A(00-11) – ADBUS (00-11)

A(0-11) – ADBUS(00-11)

A(0-11) – ADBUS(0-11)

A(0-11) – ADBUS(0)

A(0) – ADBUS(0-11)

Notice that the last two of these are the easiest to enter and the second from last is the one to be recommended; i.e., put the range of the bussed signal on the side where the definition deck can pick up any errors.

Either N_1 or N_3 may be a single letter. This accommodates chips whose outputs are, for example, QA, QB, QC, QD as in the SN74163. N_1 must be smaller than N_2 but N_3 is allowed to be less than N_4 (including letters with $A < B < C \ldots$ etc.). In this case autodecrementing of signal names is supplied, but autoincrementing on N_1 is always done. Thus, the string

A(0) – ADBUS(11-0)

will produce output of the form

A00 – ADBUS11, A01 – ADBUS10, A02 – ADBUS09, etc.

Table 1 shows some typical examples of the forms pin id (N_1 – N_2) - sig name (N_3 – N_4). Note that pin id as well as sign name are optional. Also, one of the pairs N_1, N_2 or N_3, N_4 may be omitted.

IV. Future Expansion

Quicklist can easily be expanded and generalized to be of even more use to the design engineer. The syntax of Quicklist and the associated program should allow dummy variables and loops as in any high-level programming language. To allow this, however, one must, and indeed would like to, allow the LOC(ation) field of the signal list input cards to be mnemonics for locations that would be filled in later by a location deck which would locate the DIPs on the board. This feature would also allow an easy way to leave a design fixed but to transfer it to a different board that might have a different geometry. An example will clarify this. Consider the following set of input cards:

FOR I:=0, 7

P, M(I), 2147, 4K MEMORY

1, M(I), A(0-11) – ADBUS(0)

2, M(I), D IN – DATA(I), DOUT-DOUT(I)

3, M(I), WE-WRTEN, CS-CHPSEL

END

This syntax describes a 4 K by 8-bit memory consisting of eight 2147's all tied to the same address buss, write enable, and chip select. The data input of each of these eight chips is tied to one of a data input bus DATA0 through DATA7 and the data output tied to a data output bus. The locations of these eight chips are unspecified but these locations have been assigned the mnemonics M0 through M7, and will be assigned at a later time. Once dummy variables are allowed, nested loops and arithmetic expressions would be a natural extension.

For example:

FOR I:=0, 3

P, R(I), SN54LS374, 8 BIT REGISTER

1, R(I), D(0-7) – INBUS(0+8*I)

2, R(I), Q(0-7) – OUTBUS(0+8*I)

3, R(I), CL-CLOCK, OS-OUTSEL

END

would describe a 32-bit hold register, consisting of four SN54LS374 DIP's each containing 8 bits.

The constraint that all inputs of a DIP be input as a group should also be dropped. When designing things like counters, many of the inputs can easily be described by the above-mentioned syntax, but certain inputs like carry in and carry out are different for each chip in the design. For instance, a 20-bit counter using SN54163's would most easily be described as

FOR I:=0,3

P, AD(I), SN54163, 4 BIT COUNTER

1, AD(I), (A-D) – INBUS (0+4*I)

2, AD(I), Q(A-D) – AD(0+4*I)

3, AD(I), LD-LOAD, CK-CLOCK, CLR-CLEAR

END

4, AD0, EP-ENABLE, ET-ENABLE, RCO-RIPCRY

4, AD1, EP-RIPCRY, ET-RIPCRY, RCO-CRY1

FOR I:= 2,3

4, AD(I), EP-RIPCRY, ET-CRY(I-1), RCO-CRY(I)

END

V. Conclusion

Quicklist has proven quite useful to those who have used it. The number of keystrokes necessary to describe digital systems has been reduced up to 67 percent in systems that have a highly bussed structure. The future plans for expansion would further increase the effectiveness of the program and make it the basis for a computer-aided-design system.

Table 1. Quicklist examples

Example	Result (first, second, . . . last)
A(0) – BUS(0-11)	A00 – BUS00, A01 – BUS01, . . . , A11 – BUS11
A(0-11) – BUS (0)	A00 – BUS00, A01 – BUS01, . . . , A11 – BUS11
A(0) – BUS(11-0)	A00 – BUS11, A01 – BUS10, . . . , A11 – BUS00
A(5-8) – BUS(0)	A5 – BUS0, A6 – BUS1, . . . , A8 – BUS3
Q(A) – BUS(0-3)	QA – BUS0, QB – BUS1, . . . , QD – BUS3
Q(D) – BUS(D-A)	QD – BUDD, QE – BUSC, . . . , QG – BUSA
A(6-9) – (0)OUT	A6 – 0OUT, A7 – 1OUT, . . . , A9 - 3OUT
A(4-7) – (2(0)B	A4 – 20B, A5 – 21B, . . . , A7 – 23B
IN(A)1 – T(0-4)6B	INA1 – T06B, INB1 – T16B, . . . , INE1 – T46B
A(5-9) – RESET	A5 – RESET, A6 – RESET, . . . , A9 – RESET
A(0-3)>	A0>, A1>, . . . , A3>; grounds four pins
(1-4) – RESET	1 – RESET, 2 – RESET, . . . , 4 – RESET
(1-4) +	1+, 2+, 3+, 4+, ties pins 1-4 to +5V