UIUCDCS-R-72-539

Realization of the WEB Language

By

William D. Bond and Edward S. Davidson

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

This work was supported in part by Contract AT(11-1)-2118 with the U.S. Atomic Energy Commission.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

1. INTRODUCTION

The purpose of this file note is to specify the various subprocesses involved in processing a WEB description of the logical elements in a digital assembly. A knowledge of WEB, the language in which an assembly may be described, is assumed. This language is specified fully in the following file notes and reports:

Report:

- 243 Hazelhurst, J.H., McCormick, B.H., Bond, W.D.,
"The WEB System, Part I: The Utilization of WEB,"
September 11, 1967.
- 232 Bond, W.D., Davidson, E.S., McCormick, B.H., "The
WEB System, Part II: A Formal Description of the
WEB Input Language," June 19, 1967.
- 233 Bond, W.D., "The WEB System, Part III: WEB Change
Orders," June 19, 1967.

File Number:

- 719 Bond, W.D., "A User's Guide to a Program Which
Generates Wiring Lists," October 31, 1966.
- 642 Lieman, D.S., "Description of the WEB Wiring Table,"
February 4, 1965.

2. GENERAL DESCRIPTION: PASS I

The WEB processor incorporates two separate passes of the input description to fulfill its task.

Pass I has as its input the WEB logic description, and its function is to expand all logic blocks down to unit level and to generate missing block calls as indicated by the "..."¹ instruction. Two sets of output are generated by this pass: the Packaging Skeleton and an Intermediate Description.

2.1 The Packaging Skeleton

The Packaging Skeleton is a listing of the expanded logical description with appropriate replacement of signal names. As described in File 625, a unique identifying number, called the index number, is assigned to each unit and is printed as a part of the Packaging Skeleton. Contrary to the description in File 625, signal names are printed instead of their internal numbers, and the expanded logic listing discussed there has been incorporated here as a part of the Packaging Skeleton (e.g., see Figure 1).

2.2 The Intermediate Description

For trouble shooting it is desirable to have a description of signal paths, complete with specific pin information. However, rack-connector assignments are not available until the second pass, and the input logic description afforded by the WEB language is not well suited for sequential processing by signal path. Thus, an intermediate description of the assembly being processed is generated in pass I. This description consists of two parts: an Internal Description which is machine-oriented, and a Signal Set Listing provided for users.

2.2.1 The Internal Description

The Internal Description is made necessary at this stage as the

¹Alternatively the WEB input description can by extension explicitly indicate iteration by Algol 60-type "FOR" statement.

INDEX	LOCATION	NAME	SUBSCRIPT	SIGNAL 1	SIGNAL 2	SIGNAL 3	SIGNAL 4	SIGNAL 5	
0100		CC	(0)	CSX(0) CC0(0)	EXT(0) CC1(0)	CSX(24)	EXT(1)	CS(1)	CONTINUED
01		NAND1	(1)	C	CC1(0)	CC0(0)			
02		NAND1	(2)	CC0(0)	D	CC1(0)			
03		NAND1	(3)	CSX(0)	E	C			
04		NAND1	(4)	CSX(0)	F	D			
05		NAND2		EXT(0)	A	F	E		
06		NAND1	(5)	E	B	F			
07		NAND1	(6)	CSX(24)	EXT(1)	A			
10		NAND1	(7)	EXT(1)	CS(1)	B			

0200		CC	(1)	CSX(0) CC0(1)	EXT(0) CC1(1)	CSX(25)	EXT(1)	CS(1)	CONTINUED
01		NAND1	(1)	C	CC1(1)	CC0(1)			
02		NAND1	(2)	CC0(1)	D	CC1(1)			
03		NAND1	(3)	CSX(0)	E	C			
04		NAND1	(4)	CSX(0)	F	D			
05		NAND2		EXT(0)	A	F	E		
06		NAND1	(5)	E	B	F			
07		NAND1	(6)	CSX(25)	EXT(1)	A			
10		NAND1	(7)	EXT(1)	CS(1)	B			

Figure 1. Packaging Skeleton

rack-connector assignments must be made manually, requiring two separate runs to be submitted to the WEB processor. The Internal Description is a machine oriented set of information which is necessary to communicate the results of pass I to the pass II processor.

In a later version of WEB, only one run will be necessary, as the rack-connector assignments will be automated. In this case, it will not be necessary to output the Internal Description.

2.2.2 The Signal Set Listing

The other part of the Intermediate Description is the Signal Set Listing (see Figure 2), consisting of the block and position within the terminal list of that block in which a given signal occurs. For example,²

CSX(0): CC(0)-1, CC(1)-1, ..., CC(9)-1, S-1

tells one that the signal path CSX(0) consists of the first terminal of each of blocks CC(0), CC(1), ..., CC(9) and S.

It must be noted that a terminal of a block corresponds to a particular pin only when the block itself is a unit. If a block is not a unit, then a given terminal may correspond to more than one pin. For example, CC(0)-1 really corresponds to the pins associated with NAND1(3)-1 and NAND1(4)-1.

² Compare Hazelhurst and McCormick, *ibid*, p. 18.

WEB - PASS 1 SIGNAL SET LISTING		OUTPUT FROM SAMPLE DATA INPUT TO INTERIM WEB					PAGE DRAWING D-3153	1
SIGNAL NAME	INDEX	TERMINAL LIST						
CSX(0)	010000	CC(0)-1	CC(1)-1	CC(2)-1	CC(3)-1	CC(4)-1		
		CC(5)-1	CC(6)-1	CC(7)-1	CC(8)-1	CC(9)-1		
		S-1						
EXT(0)	010000	CC(0)-2	CC(1)-2	CC(2)-2	CC(3)-2	CC(4)-2		
		CC(5)-2	CC(6)-2	CC(7)-2	CC(8)-2	CC(9)-2		
		S-2						
	030000	AAR(1,2)-1	AAR(1,1)-1	AAR(1,0)-1	AAR(2,2)-1	AAR(2,1)-1		
		AAR(2,0)-1	AAR(3,2)-1	AAR(3,1)-1	AAR(3,0)-1	AAR(4,2)-1		
		AAR(4,1)-1	AAR(4,0)-1					
	040000	AAR(5,2)-1	AAR(5,1)-1	AAR(5,0)-1	AAR(6,2)-1	AAR(6,1)-1		
		AAR(6,0)-1	AAR(7,2)-1	AAR(7,1)-1	AAR(7,0)-1	AAR(8,2)-1		
		AAR(8,1)-1	AAR(8,0)-1					
CSX(24)	010000	CC(0)-3						
		020000	IR(0)-1	IR(1)-1	IR(2)-1	IR(3)-1	IR(4)-1	
			IR(5)-1	IR(6)-1	IR(7)-1	IR(8)-1	IR(9)-1	
			IR(10)-1	IR(11)-1	IR(12)-1	IR(13)-1	IR(14)-1	
			IR(15)-1					

Figure 2. Signal Set Listing

3. GENERAL DESCRIPTION: PASS II

After the rack-connector assignment for each unit indicated in the Packaging Skeleton has been made, the Packaging Description and the Intermediate Description are input to pass II of the WEB processor. Pass II then generates the signal path description with pins assigned such that wire length is minimized. Wire lists are produced from which a technician may wire the assembly.

This pass is accomplished in three phases:

- 1) Phase I generates a signal path description, but the interconnection of pins associated with this description is only a first approximation to the final result.
- 2) Phase II consists of the wire length minimization algorithm which operates on the result of phase I, permuting signal names in the description of a signal path until a minimum wire length to interconnect these pins is found. This minimal signal path description is then stored.
- 3) Phase III then prints the wire lists either with reference to a specific connector, or alternatively listed by signal path name.

4. IMPLEMENTATION OF PASS I: PARTIAL SPECIFICATIONS

4.1 Iterative Instructions

It is convenient to do a prescan of the logic description in order to expand the iterative blocks as indicated implicitly by the "... " instruction and explicitly by the "FOR" instruction. In each case card images missing due to the use of the more concise iterative instruction are generated and written onto a scratch tape. The tape number and record number of any particular block of card images on a scratch tape are planted in the card image of the particular occurrence of the iterative instruction (in the source input) which caused its generation.

When a PACKG or END instruction is sensed, the prescan is completed and the input tape should be rewound to the start of the source input. The main scan can then obtain the information necessary to process an iterative block by referring to the tape record specified by the prescan, the identification of which is now located in the card image of the occurrence of the iterative instruction.

4.2 Expanding the Input

The UNIT and BLOCK declarations are essentially macro definitions. We adopt the convention that unnamed blocks are given the name NONAME and a sequence number, so that they also may be considered as macros. Thus the logic description is just a set of macro definitions, with an all-encompassing macro whose body of definition consists of NONAME (1), NONAME (2), ..., NONAME (n). In the logical description, however, no macro is actually called--only definitions are given. Accordingly, we may consider the END operation as a call of the all-encompassing macro, which in turn causes all expansion to take place.

This implementation requires that the compiler-compiler have subroutines available which will make it simple to write a macro-compiler. With the macro-compiler written, the expansion of the input is then a simple matter.

4.3 Unit Identifying Numbers

A unique identification of any unit in an expanded input is the succession of block names from the outermost block to the unit itself. For example, in Figure 1, unit number 0103 (part of the Index Number 010103) is really OUTERMOST:NONAME(1):CC(0):NAND1(3) and 0204 is really OUTERMOST:NONAME(1):CC(1):NAND1(4). Using this succession of names as an identification would require quite a large amount of processing just to find the identifying number of a unit given the succession of block names, and so it is not acceptable.

The problem is entirely analogous to the problem of encoding the terminal nodes of a tree which may have an arbitrary number of branches emanating from any (non-terminal) node. If one is willing to place an upper bound on the number of members of a sibling set, the following notation could be used:

We define a "left-to-right" order among members of a sibling class, and in so doing number the members of such a class sequentially according to this order. For example, we have.

Then a particular node at tree level n may be uniquely specified by the $(n + 1)$ -tuple

$$p = (s_0, s_1, \dots, s_n)$$

where s_n is the sibling number of the node in question, and s_i , $0 \leq i \leq n-1$, is the sibling number of the node at tree level i along the unique path determined by the root node and the node in question. For example, node A above would be designated (1, 3, 3). Since s_0 is always identically 1, the n -tuple $q = (s_1, \dots, s_n)$ is sufficient for identification of a node at tree level n . For implementation this much information is actually necessary, for in pass II the only way one can point to a specific unit is by an

appropriate encoding of a succession of block names.

When the input description is being expanded, one is doing a prefix scan of the eventual tree, so it is necessary to number block names sequentially at each level of the tree as they are encountered so as to have the numbering scheme as proposed. Thus it will be necessary to retain the name, level, and sequence number of each block. Then, given the succession of block names

$$N_m^1: N_{m-1}^2: \dots: N_1^m,$$

one may search level i block names until name N_i^j is found and obtain its sequence number s_i , and doing this for $i = m, m-1, \dots, 1$ one obtains the m -tuple $(s_m, \dots, s_1)$. The unique unit number $s_m s_{m-1} \dots s_1$, called the index number, is then constructed, and from this the assigned packaging for this unit may be obtained.

We adopt the convention that $m \leq 6$, and $0 \leq s_i < 64$, so that an index number corresponds to the 36 bit word of the IBM 7094. Thus, in the example above, the index number for A is 010303.

4.4 The Intermediate Description

4.4.1 The Signal Set Listing

The lists of signal paths, e.g.,

CSX(0): CC(0)-1, CC(1)-1, ..., CC(9)-1, S-1

can be derived from the block declarations themselves, so the actual units involved are not necessary at this stage. We must impose the convention that signal paths begin with a given block declaration and end with the termination of that block's body of definition. This allows one to specify the wiring between blocks. The Signal Set Listing then simply consists of the lists of these signal paths output in the format specified in Figure 2.

4.4.2 The Internal Description

4.4.2.1 Named Block Table

An entry is made in the Named Block Table each time a named block is defined. An unnamed block is given the name NONAME (n) when it is the nth unnamed block, so that all blocks may be considered as named blocks. The entry essentially describes the structure of the block in terms of signal paths rather than functionally. An entry is of variable length; its structure may best be explained through the use of an example. To this end, in Figure 3, we give the entry to be constructed with the occurrence of the definition of block CC as in File 625.

CC	EXTERNAL	INTERNAL	AUXILIARY INFORMATION
G	Δ Δ 8	C	Δ Δ 7
R	Δ Δ 10	D	Δ Δ 9
Y	Δ Δ 11	E	Δ Δ 11
K	Δ Δ 12	F	Δ Δ 14
X	Δ Δ 14	A	Δ Δ 17
CCO	Δ Δ 15	B	Δ Δ 19
CC1	Δ Δ 17	Δ	Δ Δ 21
Δ	Δ Δ 19	NAND1	Δ 1 1
NAND1	Δ 3 1	NAND1	Δ 3 3
NAND1	Δ 4 1	NAND1	Δ 2 2
NAND2	Δ Δ 1	NAND1	Δ 4 3
NAND1	Δ 6 1	NAND1	Δ 3 2
NAND1	Δ 6 2	NAND2	Δ Δ 4
NAND1	Δ 7 1	NAND1	Δ 5 1
NAND1	Δ 7 2	NAND1	Δ 4 2
NAND1	Δ 1 3	NAND2	Δ Δ 3
NAND1	Δ 2 1	NAND1	Δ 5 3
NAND1	Δ 1 2	NAND2	Δ Δ 2
NAND1	Δ 2 3	NAND1	Δ 6 3
		NAND1	Δ 5 2
		NAND1	Δ 7 3

Figure 3. Named Block Table Entry

As shown, a signal name subentry is of the form:

name	subscript 1	subscript 2	relative pointer
------	-------------	-------------	------------------

where "Δ" is used for "no-entry".

The relative pointer points to a list of entries which are called the constituent parts of the signal path. These entries are of the form:

name	subscript 1	subscript 2	position in terminal list
------	-------------	-------------	---------------------------

The relative pointer of the next signal name subentry minus the current relative pointer gives the number of constituent parts of the current signal name. Thus, as in Figure 3, signal name K has two constituent parts, namely NAND1(6)-2 and NAND1(7)-1

The external signal names are those given in the terminal list if a terminal list is given in this block declaration. In this case, internal signal names are those which are used in the block definition, but not given in the terminal list. If no terminal list is given, e.g., in the case of an unnamed block, all signal names in the definition of the block are treated as external signal names. The specification of this entry thus determines the algorithm to be used for processing a named block.

4.4.2.2 The Internal Representation of the Signal Set Listing

After the Named Block Table has been constructed, one further entry is made for the "outermost block".

If we consider an unnamed block to have a pseudo terminal list, i.e., a list of signal names occurring in the unnamed block, then the additional entry to be made to the Named Block Table consists of a list in lexicographical order of the signal names occurring in the pseudo terminal lists of the unnamed blocks, with references to the position within each unnamed block's pseudo terminal list at which a given signal name occurs. As described in the previous section, each of these signal name subentries points to the constituent parts of the signal name.

The algorithm to generate the Signal Set Listing is as follows:

If there are m signal names in the "outermost block" entry of the Named Block Table, then we start with the stack:

Now suppose the stack has been processed to the point:

We now describe the processing, and include as an example, the derivation of the CSX(0) entry of Figure 2:

- 1) Replace the top of stack by its equivalent in the "outermost block" entry.

- 2) Push into the stack the constituent parts of the top of stack.

- 3) Now the top of stack is a signal name of the form BLOCK NAME - TERMINAL LIST POSITION.

Replace the top of stack by this BLOCK NAME, and push into the stack the constituent parts of the signal name which was the top of stack.

- 4) Pop the top of stack into an output stack until a block name is the top of stack.

output stack:

CC(0) - 1	CC(1) - 1
-----------	-----------

 ...

CC(9) - 1	S - 1
-----------	-------

- 5) Pop the top of stack into the output stack.
- 6) If the top of stack is now a NONAME reference, that is, NONAME(i)-j, then go to (3).

If the top of stack is a signal name other than NONAME(i)-j, then pop this signal name into the output stack.

- 7) If the stack is now empty, go to (8). Otherwise, go to (1).
- 8) Save the output stack on magnetic tape. This is the internal representation of the Signal Set Listing, and is to be used in phase I of pass II.

After saving (but not destroying) the output stack, the information in this stack may be popped to the Signal Set Listing print routine, transforming any occurrence of an unnamed block to its appropriate index number.

4.4.2.3 The Fill Table

As a part of the internal description, a Fill table must be output. An entry is made in this table for each distinct unit in the assembly being processed. Such an entry is:

Index Number	Rack	Connector	Unit
--------------	------	-----------	------

The index number is the n-tuple discussed in Section 4.3. Pass I must complete this entry except for rack-connector-unit information which is provided by the Fill instruction in the Packaging Description processed by pass II.

5. IMPLEMENTATION OF PASS II: PARTIAL SPECIFICATIONS

The input to pass II consists of the Internal Description generated by pass I, the Packaging Description, as specified in File 625, and the filled Packaging Skeleton, the latter two being provided by the user.

5.1 Phase I

From the Internal Description, phase I uses the internal representation of the Signal Set Listing and the macro compiler tables. Phase I generates the set of pins assigned to each of the units in a given signal path, the algorithm being as follows:

Consider the internal representation of the following signal from the Signal Set Listing

CSX(0): CC(0)-1, CC(1)-1 ---, CC(9)-1, S-1

processed to the point

CSX(0)	NONAME(1)	CC(0) - 1	CC(1) - 1
--------	-----------	-----------	-----------

We now describe the processing involved:

CC(1) - 1 is an abbreviation of block name CC(1), position 1, and thus is a signal name -- the first signal name in the declaration of block CC. That is, CC(1) - 1 is just another name for the signal G within block CC(1).

Accordingly, we replace CC(1) - 1 in the stack by its equivalent: the block name CC(1), and the signal name G (obtained from the Named Block Table, in the entry for CC).

CSX(0)	NONAME(1)	CC(0) - 1	CC(1)	G
--------	-----------	-----------	-------	---

Then from the Named Block Table of the Internal Description, in the entry for CC, we find that G is the signal path:

G: NAND1(3) - 1, NAND1(4) - 1

We then replace G in the stack by its constituent parts:

CSX(0)	NONAME(1)	CC(0) - 1	CC(1)	NAND1(3) - 1	NAND1(4) - 1
--------	-----------	-----------	-------	--------------	--------------

(Actually, the insertion and replacement of the signal name G itself in the stack is for descriptive purposes only, for by the construction of the Named Block Table, the signal path of the first signal of block CC may be obtained just by referring to CC(1) - 1).

Since the top of stack is a signal name of the form UNIT NAME - TERMINAL LIST POSITION, we now reference the macro table entry for CC, the first block name from the top of stack, to obtain the sequence number of NAND1(4), which is 04 -- this is s_1 of the index number of the unit currently being processed.

Similarly, we find that the sequence number of CC(1) in the macro table entry for NONAME(1) is 02 -- this is s_2 of the index number of the unit currently being processed. NONAME(1) has sequence number 01, so that the index number of the unit being processed (that is, the top of stack) is 010204.

With this index number we do a table look-up on the Fill Table, and find that this unit has been assigned to D4/9, that is, rack D, connector 04 and unit position 9.

From the board assignment table we find that rack D connector 04 has board A-180 in it.

Finally, searching the description of board A-180, we find that the pin associated with the first terminal (the first terminal, because we are working on NAND1(4) - 1) of unit 9 is pin 16.

Thus, we may output the pin number D4-16.

CSX(0)	NONAME(1)	CC(0) - 1	CC(1)	NAND1(3) - 1
--------	-----------	-----------	-------	--------------

output stack:

D4-16

Similarly, since the top of stack is again a signal name of a unit, i.e., a signal name of the form UNIT NAME - TERMINAL LIST POSITION, we

repeat the process yielding:

Since top of stack is now just a block name and not a signal name, we remove it and begin anew with the signal name CC(0) - 1. Processing this gives us

Now an unnamed block "name" is the top of stack, and so we just remove it and continue.

If the top of stack is an "outermost block" external signal, then the phase I processing of this signal is complete. Thus, the output stack is now transferred to phase II for optimization of the interconnection of these pins, and this being completed, phase II returns control to phase I to start anew with the next signal path in the stack.

Otherwise, the top of stack is a signal name of the form BLOCK NAME - TERMINAL LIST POSITION, and processing must continue as previously specified.

5.2 Phase II: Wire Minimization

5.2.1 Backboard Configuration

The geometrical configuration of the backboard in terms of the allocation nomenclature is derived from the "PINS" and "CELLS" instructions of the packaging description (of File No. 625). Actually since it is unlikely that WEB will be used in the immediate future for processing machines other than ILLIAC III, it is recommended that interim WEB not require this input. Rather a fixed geometry can be used. Four constants are assumed by the program: the vertical rack to rack spacing, the horizontal cell to cell spacing, and the vertical and horizontal pin to pin

spacing within a cell. Using this information each pin of the Signal Path Description is transformed to a pair of Cartesian coordinates.

5.2.2 Distance Matrix

For each signal of the Signal Path Description the point-to-point direct distance wire length is calculated for each pair of points in the signal path. These distances are compared to the set of available wire lengths and the minimum available wire length greater than the distance (with a slack factor) is selected. These selected lengths are entered in a Distance Matrix.

5.2.3 Minimization of Total Path

Existing procedures for wire minimization based on the Travelling Salesman Problem have serious shortcomings. A new technique, which is an extension of a non-optimal method proposed by Croes, has been developed and does lead to optimum solutions. The details of the method are presented in full in the Department of Computer Science Report No. 201. This method is now being programmed and will be tested on typical problems for reasonableness in running time and memory requirements.

In the interim a method proposed by Karp and Held of IBM has been made available for our use. This program is the fastest method to date and works well for problems of up to 13 points. It can solve any 13 point problem in 17 seconds, and any 8 point problem in about 20 milliseconds on the IBM 7090. The method is unfortunately not well suited for larger problems.

5.3 Phase III

The wire lists generated by phase II are now printed either with reference to a specific connector* or with reference to signal path name as in Figure 4.

* See Lieman, "Description of the WEB Wiring Table", Department of Computer Science File No. 642, February 4, 1965.

SIGNAL NAME	INDEX	PIN	WIRE	PIN	WIRE	PIN	WIRE	PIN	WIRE	PIN	WIRE
CSX(0)	010103	D01-16	RED	D02-16	RED	D03-16	RED	D04-16	RED	D05-16	RED
		D06-16	RED	D07-16	RED	D08-16	RED	D09-16	RED	D10-16	RED
		D11-16	RED	D12-16	RED	D13-16	RED	D14-16	RED	D15-16	RED
		D16-16	RED	D17-16	RED	D18-16	RED	D19-16	RED	D20-16	RED
		D21-16	RED	D22-16							
EXT(0)	040101	B01-01	WHITE	A01- A	RED	A02- A	RED	A03- A	RED	A04- A	RED
		A05- A	RED	A06- A	RED	A07- A	RED	A08- A	RED	A09- A	RED
		A10- A	RED	A11- A	RED	A12- A	WHITE	B12-01	RED	B11-01	RED
		B10-01	RED	B09-01	RED	B08-01	RED	B07-01	RED	B06-01	RED
		B05-01	RED	B04-01	RED	B03-01	RED	B02-01	GREEN	E02-05	BLUE
		E04-05	BLUE	E06-05	BLUE	E08-05	BLUE	E10-05	BLUE	E12-05	BLUE
		E14-05	BLUE	E16-05	BLUE	E18-05	BLUE	E20-05	BLUE	E22-05	

Figure 4. Wiring Lists

6. SUMMARY

The WEB language is designed to describe the interconnection and packaging of the logical elements in a digital assembly. Essentially this language allows one to describe an assembly concisely in terms of the basic functional blocks of the assembly and the nested organization of these blocks. In the implementation of this language it was found that although a functional description was most convenient for the logic designer, the task of producing actual wiring lists for the construction of the assembly could best be performed by using the signal path as the basic entity for processing.

The macro nature of the WEB language with its inherent modularity and iterative structure allows the transition from this language to an intermediate language having signal paths as its basic constituents to be performed most easily by using a push down stack algorithm. By extension, a push down stack algorithm is also used for determining the pins associated with a given signal path described in this intermediate language.

Finally the minimization of the wire length of the interconnection of pins in a given signal path is performed.

The WEB language is the vital means of communication between the logic designer and computational facilities at his disposal. The implementation of this language in a time sharing environment will provide real time information concerning the numerous physical and functional limitations the logic designer must face in the design of a digital assembly.

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.
COO-2118-0036
UIUCDCS-R-72-539

2. TITLE
Realization of the WEB Language

3. TYPE OF DOCUMENT (Check one):

- ☒ a. Scientific and technical report
☐ b. Conference paper not to be published in a journal:
Title of conference _____
Date of conference _____
Exact location of conference _____
Sponsoring organization _____
☐ c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- ☒ a. AEC's normal announcement and distribution procedures may be followed.
☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

William D. Bond and Edward S. Davidson, Research Assistants
Bruce H. McCormick, Principal Investigator

Organization
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature

Bruce H. McCormick

Date

August 1, 1972

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- ☐ a. AEC patent clearance has been granted by responsible AEC patent group.
☐ b. Report has been sent to responsible AEC patent group for clearance.
☐ c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-72-539	2.	3. Recipient's Accession No.
4. Title and Subtitle Realization of the WEB Language		5. Report Date August, 1972	
7. Author(s) William D. Bond and Edward S. Davidson		6.	
9. Performing Organization Name and Address Dept. of Computer Science Univ. of Illinois at Urbana-Champaign Urbana, Illinois 61801		8. Performing Organization Rept. No.	
		10. Project/Task/Work Unit No. ILLIAC III	
		11. Contract/Grant No. AT(11-1)-2118	
12. Sponsoring Organization Name and Address U. S. Atomic Energy Commission		13. Type of Report & Period Covered	
		14. Feb. - June, 1972	
15. Supplementary Notes			
16. Abstracts WEB: a language to describe the interconnection and packaging of the logical units in a digital processor has been fully discussed in previous reports. The purpose of this report is to specify the various subprocesses involved in processing a WEB description of the logical elements in a digital assembly. A knowledge of WEB, the language in which an assembly may be described, is assumed.			
17. Key Words and Document Analysis. 17a. Descriptors descriptors programming language interconnection packaging design automation digital processor			
17b. Identifiers/Open-Ended Terms			
7c. COSATI Field/Group			
18. Availability Statement RELEASE UNLIMITED		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
		20. Security Class (This Page) UNCLASSIFIED	22. Price