Information Processing in Japan Vol. 11, 1971

OLBA—An Experimental On-Line Block Analysis System

SETUO OSUCA* AND HIROYUKI YAMAUCHI*

1. *Introduction*

This paper describes an experimental computer aided design and decision system called OLBA (On-Line Block Analyser), which we can use to solve a variety of problems describable in the form of block diagrams.

The requirement for OLBA has been as follows:

(1) It must be applicable to a wide variety of problems represented by the block diagrams, where the definition of the blocks is such that an output of any block is uniquely determined by inputs and current state of the block.

(2) Problems must be solved interactively with OLBA.

(3) Graphical input/output services should be permitted, besides typewriter input/output services.

(4) It must be extensible; In considering that the vast areas of applications are intended, it must be extensible.

2. *Program Description*

2.1. *Block Diagram and its Representation*

A simple block diagram of an analog feed-back system is shown in Fig. 1 as an illustration for OLBA, where every block has its function name and connective relations to other blocks, both of which must be stated by one statement of the language of OLBA. We shall show the characteristic of blocks and connective relations of them below.

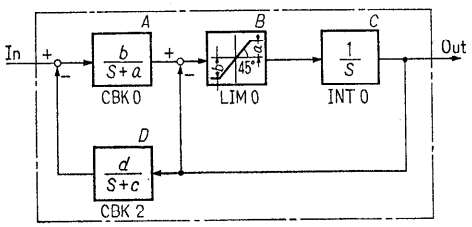


Fig. 1. Example of a block diagram.

In order to model a block diagram, OLBA contains a set of routines called Block Routines or simply Blocks, and uses a special data structure to combine some of Blocks into a whole block diagram. Three kinds of blocks are ava-

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 12, No. 2 (1971), pp. 103–113.

* Institute of Space and Aeronautical Science, University of Tokyo.

ilable; Basic Blocks (BBK), Compound Blocks (CBK) and Special Blocks (SBK). Basic blocks are elementary blocks with corresponding routines in the system that have the basic functions and are available to construct Compound Blocks. Meanwhile, Compound Blocks are representatives of the set of blocks. In Fig. 1. B and C are Basic Blocks and A and D are Compound Blocks in the sense that the ycan be decomposed into the Basic Blocks such as an integrater, a constant multiplier and an adder. The block shown in broken lines consisting of A, B, C and D may also be represented by one larger Compound Block. OLBA usually preserves only Basic Blocks in the system and we can define any Compound Blocks by them whenever necessary. Once we have defined a new Compound Block, we can use it freely to construct a larger block as it were a Basic Block. There are three types of Special Blocks; (a) input/output service block, (b) a data transfer block used only when a CBK is defined, and (c) subroutine blocks. As an input block several function generators which produce pre-specified functions are available, while, for the output service, data display blocks which display data to the specified output devices are available. A block called OUT block is utilized for (b). Since the output of a CBK must be supplied by a BBK or a defined CBK which is a constituent of the former CBK, we specify it by the OUT block setting it at the last of the constituents (see Fig. 4). Subroutine blocks call subrouitnes available in the system and are similar to the CALL statement of FORTRAN.

It must be stressed that the most blocks in OLBA were designed to receive two kinds of inputs. A block with the integral function, for example, must have two kinds of inputs, one is an integral variable and the other is an integrand. In the other blocks, they may be control signals and data inputs, or they may be equivalent but different inputs such as set and reset signals to a flip-flop. We refer to these paired inputs in terms of independent variable X and dependent variable Y respectively for the sake of simplicity. Moreover, each block may be able to receive multi-dimensional input pair or vector inputs. If a block receives several inputs of the same dimension from several blocks, they are combined to one by the operation defined to each block and expressed, for convenience, as $X=\Sigma X_i$, $Y=\Sigma Y_i$. This may be an arithmetic sum or a logical product (for a logical AND block) or a logical sum (for an OR block), etc. On the other hand, we decided that each block should produce only one kind of output which is retained in this block till it will be changed later and used by the other blocks.

Blocks in OLBA are put into the free storage and linked with each other by the pointer structure called Patch List.

2.2. *Program Language*

The OLBA program language consists of two parts, one for Block Declara-

tion and the other, the Commands.

2.2.1. *Block Declaration*

Block Declaration is used to describe the problem, that is, to create blocks and determine their inter-relationship. As explained later, blocks can be declared in the arbitrary form, but as far as BBK and defined CBK are concerned, they are described in the standard form. A general format of expression with the typewriter is as follows;

BLKn (Y_1*X_1) $(Y_2*X_2)\cdots(Y_n*X_n)P_1, P_2, \cdots, P_k$.

where BLK is a block name written with character strings, and n is an identification number (0-255) for the block. An input pair of independent and dependent variables is shown by (Y_i*X_i) where Y_i is a dependent variable and X_i is an independent variable, each of which is represented by a block name whose output is to be an input to BLKn. If BLKn has more than one input, they are joined by '+' or '−' sign. If an independent variable common to all or most of the blocks is used, it is shown by the symbol I. P_1, P_2, etc. are parameters possessed by BLKn. In Fig. 1, the block B which is a BBK called LIM (limitter) is then written in the form,

LIM0 (CBK0-INT0*I) a, b.

In this way, a problem is written by giving successively its constituent block names and their inter-relationships. (A few examples are given in the section 4).

There are about 25 BBK routines provided in OLBA. The details of them are omitted because of the limitation of the space. In the case of graphical inputs, we draw directly a block diagram on the CRT. There is one-one correspondence between graphical and typewriter inputs. But when we use graphical representation, we are allowed to represent blocks and their inter-relationships separately, while, in the case of typewriter inputs, we must exhibit one block and its connective relation by one statement. Because of it, graphical inputs are transformed into a conversion table and then sent to the interpreter in the same order as typewriter inputs. An example of graphical inputs is given in Fig. 2 (at the intermediate stage).

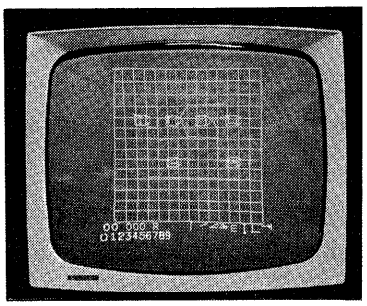


Fig. 2. Example of graphic inputs.

2.2.2. *Commands*

The commands are instructions given to the system by the operator and are shown in the appendix.

3. *Outline of OLBA System*

The outline of OLBA system is given in Fig. 3, the point of which is explained below.

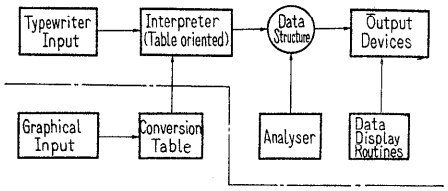


Fig. 3. Outline of the OLBA system.

3.1. *Data Structure*

The data structure is constructed by the blocks and Patch List which associate blocks in a certain order.

Any block is represented by a table called Block Table (BT) in which all the information of the block is contained, i.e., its block name, connective relations with other blocks, parameters, working registers and so forth. The Patch List (PL) defines the order in which each block is processed. An element of a PL consists of two pointers; one to a corresponding block and the other to the next element. As already described, a CBK must perform its function by activating its constituents in certain order, that is, it must have its own lower level PL associating its constituents. As a CBK may contain another defined CBK as its constituents a multi-level structure can be constructed (see Fig. 4).

By the way, there may be much discussion about the definition of the set of the Basic Blocks. It would be most preferable that the set of them are as small as possible and closed with respect to the applicable blocks. But as the vast areas of applications are considered, it would become very difficult to determine such a set. Moreover, in consideration of the simplicity of usage and the efficiency of processing problems, it would be preferable that they will be defined freely. Therefore, we left it open and made it possible to add basic blocks newly defined to the set whenever necessary.

3.2. *Interpreter*

The interpreter of the OLBA language is a table oriented routine and transforms character strings or graphical inputs into the above mentioned structure. This table specifies the format of each statements used and is called Statement Format Definition Table (SFDT). SFDT defines the kinds of symbols used in a ceatain statement and specifies subroutines corresponding to the symbols.

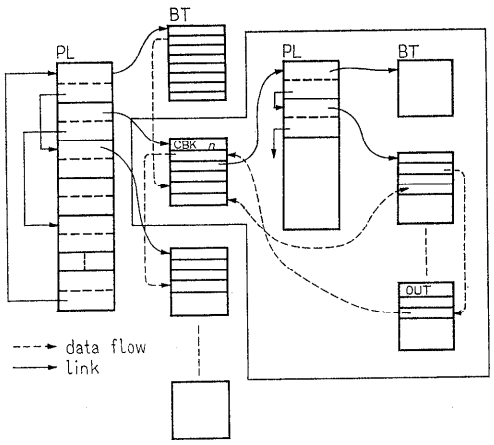


Fig. 4. Data structure.

The same symbols may be used in the different meaning in the different Commands or Block Declarations and the corresponding subroutines are distinguished by the number (see Table. 1). Then, it is possible to extend the system by changing the contents of this table or adding newly defined command or block names with their format specification.

In the following, we exclusively concern with inputs from a typewriter because the essential part of the interpreter lies in the manipulation of the typed inputs.

The interpreter reads inputs until the End Code (.) appears. When misstyping has been done, we correct it by Back Space and (/) or make one statement noneffective by Carriage Retuen and Line Feed. When the End Code has been found, the interpreter stops reading and examines the input sequence just read by refering to the SFDT, and if it is a correct one, its definition format is read out from the SFDT to the Format Register (FR). Then the arguments are processed with reference to the FR; in the case of a command, the specified subroutine is called and in that of a Block Declaration the BT is created in the buffer storage and is copied in the free storage available, linking it to the PL previously created.

To make it clear, we now illustrate simple examples. A command, for example, DEF is used as follows;

DEF CBK1.

Here blank (♭) and (.) are used and the interpreter calls subroutines associated with (♭) and (.) defined in the column of DEF of the SFDT, i.e., that indicates that #1 routine out of a set of routines associated with (♭) and also #5 out of (.) routines should be used and they direct the system to define the Compound Block named CBK1 by the successive Block Declarations until the CLS command

Table 1. Statement Format Definition Table (SFDT).

Statement identifier	♭	/	.	,	−	+	)	(	*	#
DEF	1		5							
CLS			8							
PARAM	2	1	4	1	2	2				
COPY	1	2	3							
CNCT	1	3	12	2	3	3				
INT	1		1		1	1	1	1	1	
LIM	1		6	1	1	1	1	1	1	
STOP	1		6		1	1	1	1		

has appeared.

In the case of a Block Declaration, for example,

LIM5 (INT1+INT2−INT3*I) 10, −10.

where a blank, a right and left bracket, a plus, a minus, an asterisk, a comma and a period are used, the interpreter creates the BT according to the definition of LIM in the SFDT and copies it in the free storage available, linking it to the PL previously created. In this way, the user can define an appropriate format and function of a statement by combining the symbol manipulation routines prepared in the system. If there appears the syntactic error or an undefined block during interpretation, the error message is printed out, otherwise, '*R*' is printed indicating to be ready for reading the next statement.

3.3. *Analyser*

When the data structure has been completely created and the START command is given, the analyser is called to begin computation. The analyser consists of the Patch List Follwer (PLF) and a collection of block subroutines each of which performs the specified function of a block. PLF follows PL and finds the BT to be processed in the pre-determined order, calling the corresponding subroutines, and executes computation necessary to each block, the results of which is put into the Output Register. In the case of a CBK, the analyser stacks its current PL pointer and enters to process the low level structure through the PL pointer in the BT of a CBK. The OUT block in a CBK stores its output in the OUT-reg. of a CBK and then restores the stacked pointer.

3.4 *Data Display Routines*

The data display routine displays the contents of the block to the output devices in numeric or graphic form. This routine is called by a Data Display Block.

4. *Examples*

4.1. *Simulation of Analog System Control*

The experiment of parameter adjustment of feed back control system was dealt with. As an example, a position servo system represented by the simple quadratic equation

$$\frac{d^2y}{dx^2}+\frac{1}{2}\frac{dy}{dx}+y=f(t),\ f(t)=1(t) \qquad (1)$$

is given and expressed by the block diagram shown in Fig. 5*A*, using the basic blocks such as integrators (INTn) and constant multiplier (DVIn). To improve follow-up characteristic of the system, auxiliary circuit with two parameters is added (Fig. 5*B*) which detects position and velocity error of the original system, and linearly combines and feeds back them. The purpose of the experiment was to adjust interactively two multipling parameters to optimize the whole system. The result of the experiment is given on the CRT (DSPL in Fig. 5*C*). The program written with OLBA is given in Fig. 6. By the START command, the experiment is begun resulting the output shown on the CRT and the process is iterated so as to attain the optimum value. Two examples are shown in Fig. 7.

4.2. *Simulation of Digital Circuit*

In this experiment, a 4-bits parallel adder was modelled as in Fig. 8 in which not only logical relations but physical characteristics of the element with random noise were considered, and the behaviour of the carry propagation was simulated. The program is shown in Fig. 9, in which the FOE block of the CBK10 which accepts the normal input and random noise as well represents the linear characteristic of the element. CBK10 which represents carry propagation characteristic is copied to CBK11, CBK12 and CBK13; CBK20 which performs logical

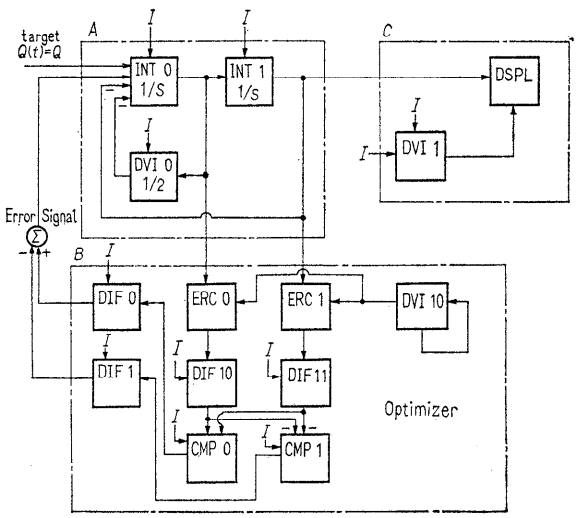
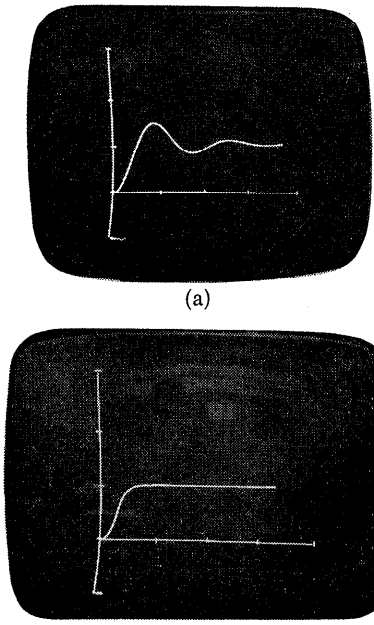


Fig. 5. Optimal control.

```
R  PATCH.
R  INT0 (-DVI0-INT1+DIF0-DIF1*I).
R  INT1 (INT0*I).
R  DVI0 (INT0*I)2.
R  DVI10 (DVI10*DVI10)1.
R  ERC0 (INT0*DVI10).
R  ERC1 (INT1*DVI10).
R  DIF10 (ERC0*I)1.
R  DIF11 (ERC1*I)4.
R  CMP0 (DIF10+DIF11*I)0.
R  CMP1 (-DIF10-DIF11*I)0.
R  DIF0 (CMP0*I)2048.
R  DIF1 (CMP1*I)2048.
R  DVI1 (INT0*I)1.
R  DVI2 (INT1*I)1.
R  DVI3 (I*I)16.
R  PLTX (DVI1*I).
R  PLTY (DVI2*I).
R  STP (DVI3)800.
R  ENDP.
R  INIT.
R  PARAM INT0 YA256/DIF0 PA0/DIF1 PA0.
R  START.
   END
R  INIT.
R  PARAM ERC1 R256/DIF10 PA0/DIF0 PA2048/DIF1 PA2048.
R  PARAM DVI1 PA2/DVI2 PA2.
R  START.
   END
R  INIT.
R  PARAM ERC1 R256/DIF10 PA1.
R  START.
   END
R
```

Fig. 6. Optimal control.



(a)

(b)

Fig. 7.

addition digit by digit is copied to CBK21, CBK22, and CBK23; CBK30 and its copied CBK32 and CBK33 generate a carry from each digit; two operands and the result are stored in FF10~13, FF20~23 and FF30~33 respectively. In this case, the operands are 0111 and 0001 and the summation of them is sampled at the pre-determined time interval and is shown after the START command. The RNG block supplies random noise to FOE.

4.3. *Simulation of Queuing Model*

In this experiment, it was assumed that the time intervals between arrivals and the service time are approximately uniformly distributed and the service is done on a first-come first-served basis, the number of service channels is nine in series, the permitted queue-length is four units. The result are omitted here for the limitation of the space.

Conclusion

OLBA is an experimental interactive problem solving system restricted to the problem describable in a block diagram. This system has been implemented by the proto-type mini-computer with 16-bits 8K memory capacity with approximately 150 blocks available. Our major objective at this experiment has been, at first, to clarify experimentally the matter related to the program making and data structure, and then, to acquire a clue for the more sophisticated system after examining this system by various applications. The program was written by an assembler.

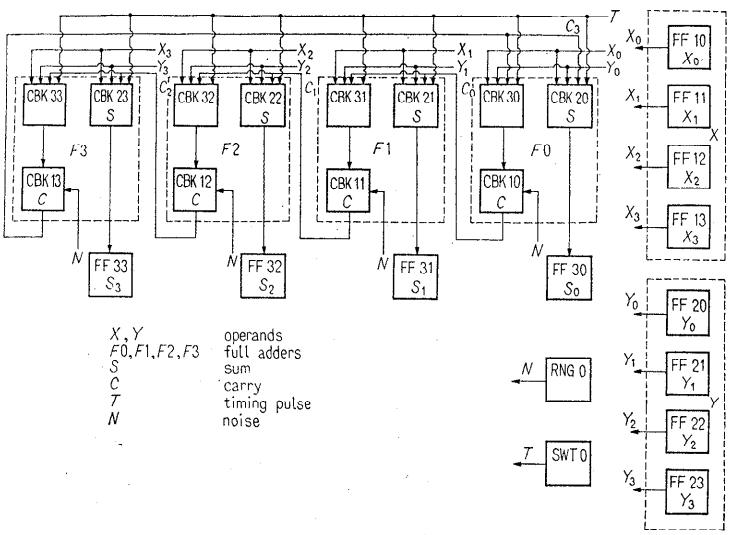


Fig. 8. 4-bits parallel adder.

```
R  PATCH.
R  DVIC (#1)32.
R  DVI1 (DVI0#1)4.
R  SWTC (DVI1#1).
R  RNGC (1).
R  FF10 (FF10#1).
R  FF11 (FF11#1).
R  FF12 (FF12#1).
R  FF13 (FF13#1).
R  FF20 (FF20#1).
R  FF21 (FF21#1).
R  FF22 (FF22#1).
R  FF23 (FF23#1).
R  DEF CBK10.
R  DIFC (CO#1)4C.
R  FOEC (DIFC+C1#1)3,0.
R  CMPO (FOEC#1)30.
R  OUT (CMPC#1).
R  CLS.
R  DEF CBK20.
R  ANDO (CO-C1-C2+C3#1).
R  AND1 (-CC+C1-C2+C3#1).
R  AND2 (CO+C1+C2+C3#1).
R  AND3 (-CC-C1+C2+C3#1).
R  ORO (ANDO+AND1+AND2+AND3#1).
R  OUT (ORO#1).
R  CLS.
R  DEF CBK30.
R  ANDO (CO+C1+C3#1).
R  AND1 (CO+C2+C3#1).
R  AND2 (C1+C2+C3#1).
R  ORO (ANDO+AND1+AND2#1).
R  OUT (ORO#1).
R  CLS.
R  COPY CBK11/CBK10.
R  COPY CBK12/CBK10.
R  COPY CBK13/CBK10.
R  COPY CBK21/CBK20.
R  COPY CBK22/CBK20.
R  COPY CBK23/CBK20.
R  COPY CBK31/CBK30.
R  COPY CBK32/CBK30.
R  COPY CBK33/CBK30.
R  CBK20 (FF10#1)(FF20#1)(FF100#1)(SWT0#1).
R  CBK30 (FF10#1)(FF20#1)(FF100#1)(SWT0#1).
R  CBK10 (CBK30#1)(RNGO#1).
R  CBK21 (FF11#1)(FF21#1)(CBK10#1)(SWT0#1).
R  CBK31 (FF11#1)(FF21#1)(CBK10#1)(SWT0#1).
R  CBK11 (CBK31#1)(RNGO#1).
R  CBK22 (FF12#1)(FF22#1)(CBK11#1)(SWT0#1).
R  CBK32 (FF12#1)(FF22#1)(CBK11#1)(SWT0#1).
R  CBK12 (CBK32#1)(RNGO#1).
R  CBK23 (FF13#1)(FF23#1)(CBK12#1)(SWT0#1).
R  CBK33 (FF13#1)(FF23#1)(CBK12#1)(SWT0#1).
R  CBK13 (CBK33#1)(RNGO#1).
R  FF30 (CBK20#1).
R  FF31 (CBK21#1).
R  FF32 (CBK22#1).
R  FF33 (CBK23#1).
R  FF100 (CBK13#1).
R  ANDO (DVI0+SWT0#1).
R  TYPEZ (FF13+FF12+FF11+FF10*ANDO).
R  TYPEZ (FF23+FF22+FF21+FF20*ANDO).
R  TYPEZ (FF33+FF32+FF31+FF30*ANDO).
R  TYPEZ (FF100*ANDO).
R  STP (DVI1)2.
R  ENDP.
R  INIT.
R  PARAM FF10 Z1/FF11 Z1/FF12 Z1/FF20 Z1.
R  START.
 0000  0001  0001  0001
 0000  0000  0000  0001
 0000  0001  0001  0000
 0000
 0000  0001  0001  0001
 0000  0000  0000  0001
 0000  0000  0000  0000
 0000
 0000  0001  0001  0001
 0000  0000  0000  0001
 0001  0000  0000  0000
 0000
 0000  0001  0001  0001
 0000  0000  0000  0001
 0001  0000  0000  0000
 0000
 0000  0001  0001  0001
 0000  0000  0000  0001
 0001  0000  0000  0000
 0000
  END
R
```

Fig. 9. 4-bits parallel adder.

We recommend that we must fully consider how to cope with the tendency of increasing the complicated interactions between a man and a computer when we use graphical inputs to represent directly a block diagram on the CRT.

Appendix: OLBA Commands

1) OLBA. Call for the OLBA system
2) PATCH. Begin the programming
3) ENDP. End the programming
4) DEF CBKn. Define the compound block CBKn with the successive Block Declarations till CLS
5) CLS. End of the definition paired used with DEF
6) START. Begin the computation
7) PARAM CBKn* BLKm a_1, a_2, ···/BLKk b_1, b_2, ··· Change the value of variables or parameters of the block specified
8) INIT. Initialize all the variables and parameters used in the programming
9) INST CBKn* BLKm. Insert the specified block to the structure
10) DLT CBKn* BLKm. Delete the specified block from the structure
11) CNCT CBKn* BLKm/BLKk. Connect the block BLKm to the block BLKk
12) DISC CBKn* BLKm/BLKk. Kill the block BLKm from the block BLKk

* This is omitted when the specified block is not a constituent of a CBK.

13) COPY CBKm/CBKn. Copy CBKn to CBKm
14) (QUIT). Interrupt the computation with the Switch
15) CNTN. Recover the interruption
16) STORG X, Y. Locate the origin of the CRT to (X, Y)

Reference

[1] Sutherland I. E.: Sketchpad, A Man-Machine Graphical Communication System, *Proc. SJCC* (1963), 329.
[2] Ross, D. T.: The AED Approach to Generalized Computer Aided Design, *Proc. A. C. M. National Meeting* (1967), 367.
[3] Ross, D. T. and I. E. Rodriguez : Theoretical Foundations for the Computer Aided Design System, *Proc. SJCC* (1963), 305.
[4] Gray, J. C.: Compound Data Structure for Computer-Aided Design: A Survey, *Proc. A. C. M. National Meeting* (1967), 355.
[5] Lang, C. A. and J. C. Gray : ASP—a Ring Implemented Associative Structure Package, *Comm. A. C. M. 11.* 8. (1968), 550.