Test Generation for the JPL Viterbi Decoder Chip

BY

M.A.Breuer, Margaret Driscoll, Rajesh Gupta, Rajiv Gupta, Shen Lin, and Rajagopalan Srinivasan

Technical Report No. CENG 90-08

February 1990

Electrical Engineering - Systems Department
University of Southern California
Los Angeles, CA. 90089-0781

Test Generation for the JPL Viterbi Decoder Chip

M.A. Breuer, Margaret Driscoll, Rajesh Gupta, Rajiv Gupta, Shen Lin, and Rajagopalan Srinivasan

February 6, 1990

This work was supported in part by the Jet Propulsion Laboratory, by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract No. N00014-87-K-0861, and in part by the Semiconductor Research Corporation under Contract No. 88-DP-075. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Acknowledgement

We want to acknowledge the contribution of Sandip Nerkar, Jung Chen Lien, as well as Kuen-Jong Lee who developed TGS. Erland Olsen of JPL helped on modeling the chip.

Abstract

This report documents the results of our efforts in generating a set of test vectors for the JPL Viterbi decoder chip using the CRETE software system developed at USC. A brief discussion of the Viterbi decoder chip and CRETE system is first given. The modeling of the chip and manually generated functional tests are then described. These tests are used to test non-scan parts of the chip. The results produced by CRETE for the scan portion of the design are then presented.

1 Introduction

The Jet Propulsion Laboratory (JPL) is currently designing a new Long Constraint Length VLSI Viterbi Decoder to be used in many future NASA missions. To enhance testability, much of this design uses a scan based architecture. The Test Group at USC is developing several software packages to support design-for-test and test generation of VLSI chips. This report describes the use of some of this software in the generation of tests for the Viterbi Decoder. The tests developed are for the classical single stuck-at fault model.

The goals of this project were the following.

1. To develop a set of tests for the Viterbi Decoder.

2. To debug the test software using a complex test case.

3. To gain experience in modeling and developing tests for a complex circuit.

4. To determine what new features should be added to the test software.

All the goals of this project were successfully met. Detailed analysis fo the Viterbi Decoder gave us insight on extending our system capabilities in the areas of cloud generation and equivalence identification.

2 The Viterbi Decoder

The decoder consists of 8,192 Viterbi butterfly processors. A Viterbi decoder processor IC contains 16 Viterbi butterfly processors, resulting in over 20,000 gates per chip, with each individual butterfly processor having a complexity of about 1800 gates. More details on the design of the decoder can be found in [1]. Aspects of testing a single butterfly processor are discussed in [2].

3 Test Software

Figure 1 shows part of the TEST software system being developed at USC. Cbase is an object oriented database system which has a graphics schematic capture system on its front-end. TGS is a test generation system for combinational logic, and includes a PODEM ATPG package, a good circuit simulator, a fault simulator, and a set of routines for carrying out fault collapsing. CRETE, which stands for Clouding, Reorganization, Equivalence, Test generation and Editing, is the primary system used to process a chip design and produce a test set for the chip. CRETE assumes that the circuit to be processed is a full scan design.

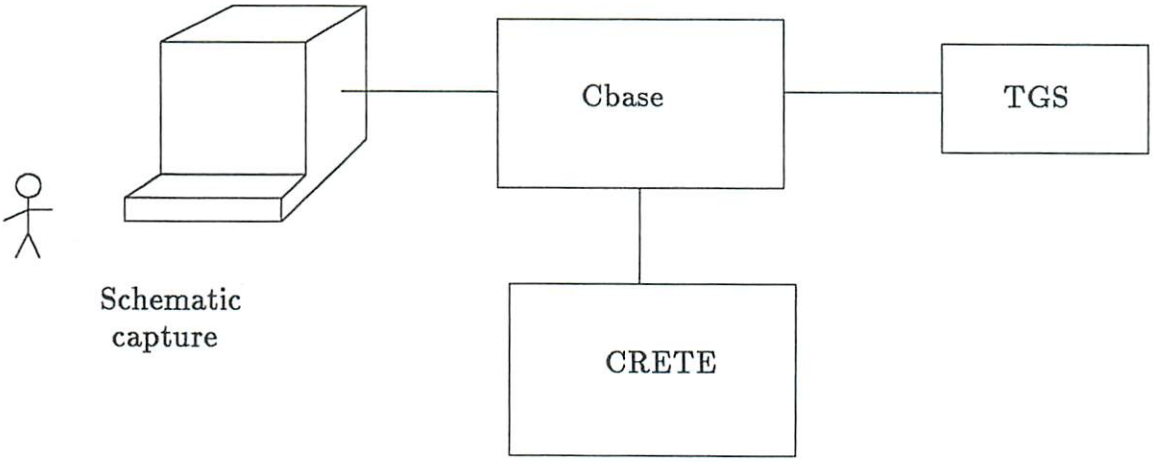


Figure 1: Part of the TEST system

The flip-flop model used in CRETE is shown in Figure 2. A D flip flop is assumed.

No clock, reset, preset or mode control are allowed. The flip-flop is assumed to be clocked between each input vector. Only a Q output is used; a $\overline{Q}$ output must be derived external from the flip-flop by using an inverter. An implicit mode control line is assumed to exist. This line is used to select either the D or S_i input.

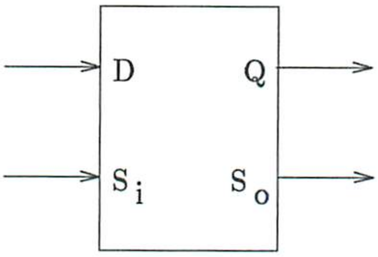


Figure 2: Flip-flop model used in CRETE

CRETE employs the concept of clouds. A *cloud* is a maximally connected block of combinational logic such that all of its inputs are either primary inputs or outputs of flip-flops, and all of its outputs are either primary outputs or inputs to flip-flops. A *replicated cloud* (r-cloud) is a block of logic C_r which can be used to create a cloud C as follows: cloud C consists of n copies of $C_r, n \geq 1$, where some inputs to C_r are common to all copies of C_r. Figure 3 indicates the general structure of a cloud made up of r-clouds. Here A represents outputs from an r-cloud; B represents inputs to an r-cloud; E represents global cloud primary inputs; and D represents global cloud inputs which are outputs from flip-flops.

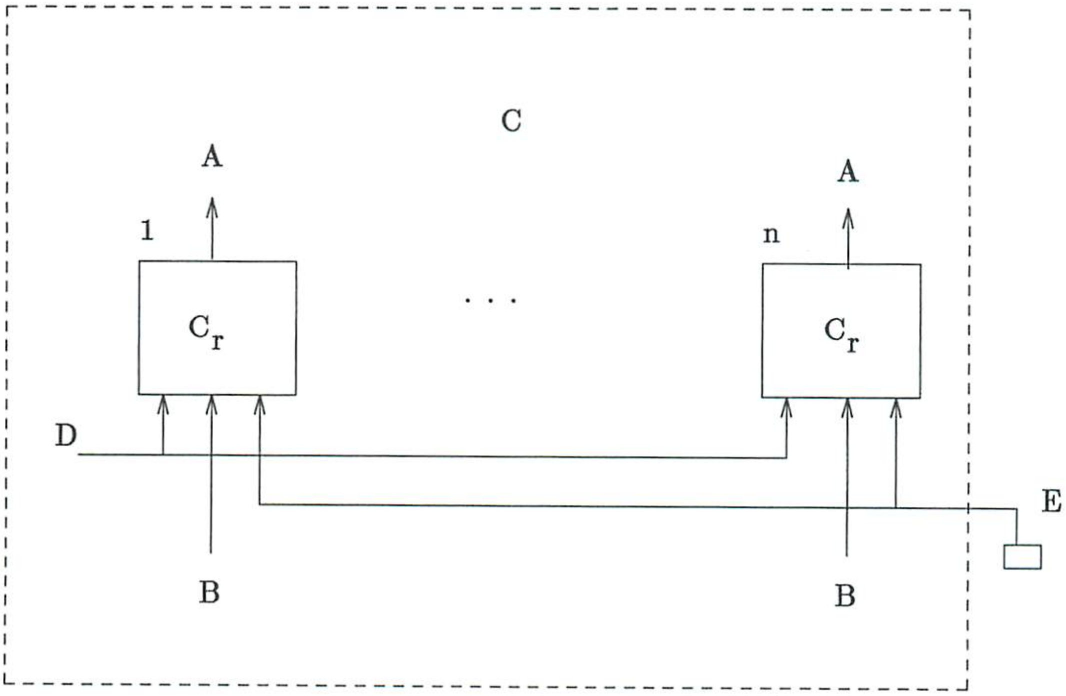


Figure 3: A cloud C consisting of r-clouds C_r

The first process carried out by CRETE is to reorganize the user input hierarchical description of a circuit and produce a new hierarchical description consisting of clouds and flip-flops. A procedure for identifying equivalent coulds, i.e. multiple instances of a cloud has been developed but has not yet been implemented. Tests vectors for each cloud are obtained using TGS. These tests are then combined to create a test set for the entire chip. The combined tests are output to files representing test vectors to be applied to the primary inputs and test vectors which must be loaded into the scan chains. More than one scan chain is allowed. The system automatically determines the order of the flip-flops in the scan chains and orders the bits in the test vectors according to this ordering. CRETE assumes the following standard test methodology is used in testing the combinational logic.

Step 1: Load the scan chains with a test vector while in the test mode.
Step 2: Apply a test vector to the primary inputs .
Step 3: Observe the primary outputs.
Step 4: Issue one system clock while in the normal mode.
Step 5: Scan out the scan chains while in the test mode.

This process is repeated for each test. Steps 1 and 5 can be carried out concurrently. More details on CRETE can be found in [3].

4 Modeling of Viterbi butterfly processor and manual test vectors

Since a Viterbi process consists of 16 identical Viterbi butterfly processors, only one butterfly processor was processed by CRETE. Since the processor is not a full scan design, special models for parts of the processor had to be used so as to force CRETE to produce results which would be applicable to the processor. CRETE could not generate test for some non-scan parts of the logic. For these parts, manually generated tests were derived. In this section we will describe the various modeling assumptions made and describe the manually generated tests. Page numbers referenced below refer to the title of the schematic drawing which define the Viterbi Decoder chip. These schematics are not included in this report.

1. Scan flip-flops were modeled with a D input, a Scan-in input, a Q output, and a Scan-out output only. Clock, clear (when applicable) and T/R lines were left out for test generation purposes. When a Q bar output was needed (registers 5P and 9P on page BFLY) an inverter was added to the Q output in the model. Also, the scan chain in the model is formed using the Scan-in and Scan-out lines, even though the actual circuit sometimes used the Q output instead of the Scan-out port.

2. TGS requires a gate level description for a circuit. The gate level models used for a full adder, multiplexer, XOR and XNOR circuits are shown in Figure 4.

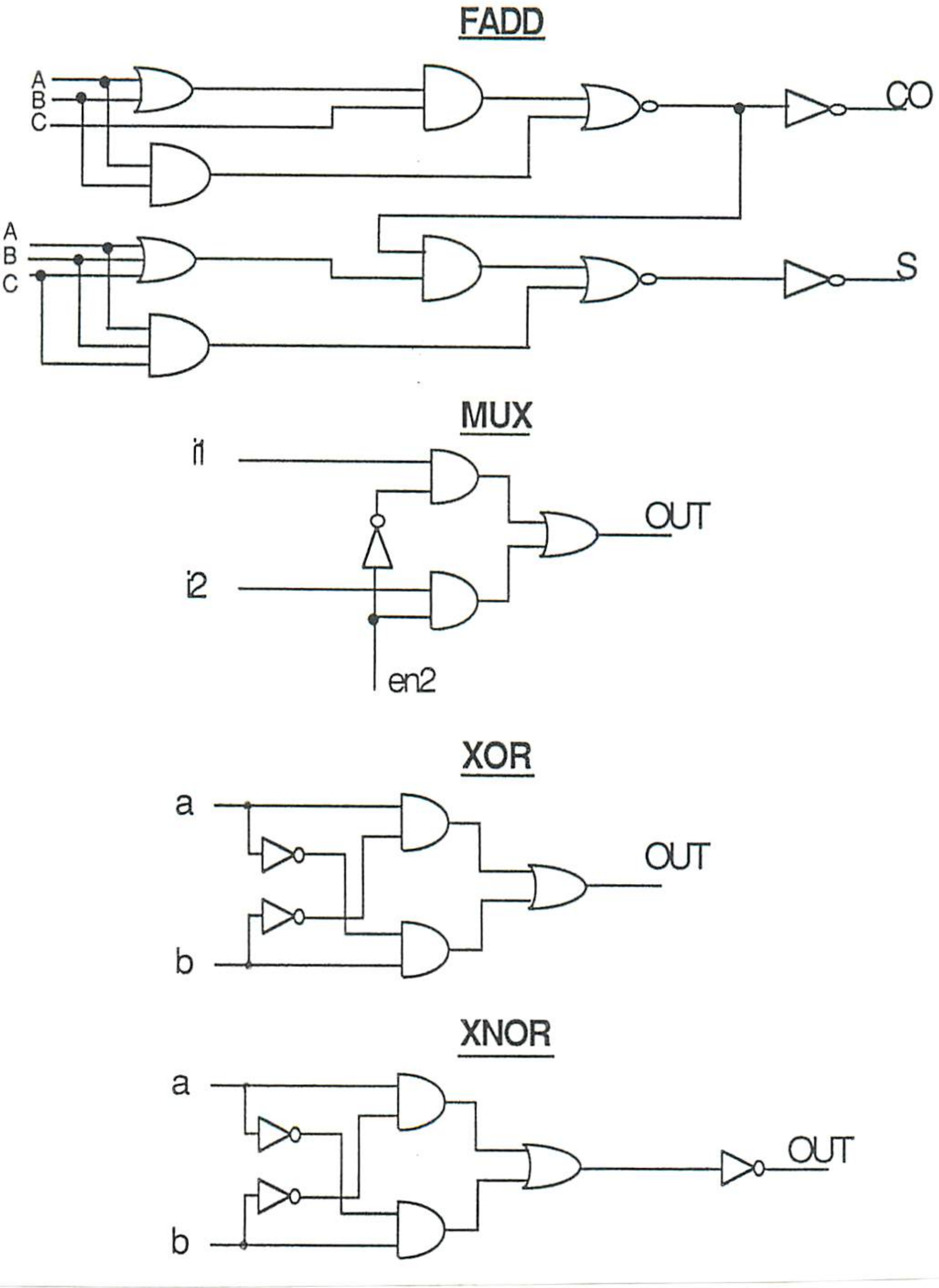
3. The 16 Bit Shift Register (page 16SR) was modelled as 16 scan flip-flops chained together. The Mux (20P) and NAND gate (1P) will be tested with a functional test. During the scan test process, RESET* is held high and WORD_SYNC is held low. This sets up a path from DATA_OD through MUX 20P to MSDATA.

4. In the model of the COMPSEL unit (page COMPSEL), scan flip-flop 11P and Mux 24P were left out. This flip-flop/Mux pair becomes transparent during scan operation due to the fact that the register is being clocked on the falling edge of the clock (due to inverter 16P in the clock path). This part of the circuit will be tested with a functional test.

Some parts of the butterfly processor were not included in the computer model. They are enumerated below along with a discussion of how this logic is to be tested.

1. Buffers and inverter pairs at BFLY level, clock lines and T/R lines.

   Resolution: These elements get tested “for free” when using primary input pins to provide data for other tests, clocking in scan data, and switching between test and normal mode.



FADD

MUX

XOR

XNOR

Figure 4: Gate level models

2. Resets on V1496 scan flip-flops.

   Resolution: The following scan flip-flops are equipped with an asynchronous reset: 1,2, 16, 53, 86, 87, 88.

   Shift into the scan-chain a vector of all 1s while in test mode. Bring RESET* low and then high, and then shift out the vector in the scan-chain while in the test mode. Check for 0s in the aforementioned positions in the vector.

3. Reset on V1496 scan flip-flop 7P in Piped A-B (page PA_B).

   Resolution: This reset is controlled by signal M_Word_Sync*, a buffered version of Word_Sync. Shift in a vector of all 1s in test mode. Bring Word_Sync high and transition the clock from high to low twice. Scan out the test vector and verify that position 15 in the vector is now a 0.

4. KILL* circuitry, specifically V1220 NAND gate 96P (BFLY page), V1220 NAND gate 1P, and V1925 MUX 20P (both on page 16SR).

   Resolution: This circuitry appears twice in Compare-Select. There are two Compare-Select Units in the BFLY, therefore there are 4 instances of this circuit that can be tested simultaneously.

   See Test Sheet 1 for a diagram of this circuitry and a functional test.

5. Inverter V1916 part 16P, V1496 scan flip-flop 11P, and V1925 MUX 24P (all on COMPSEL page).

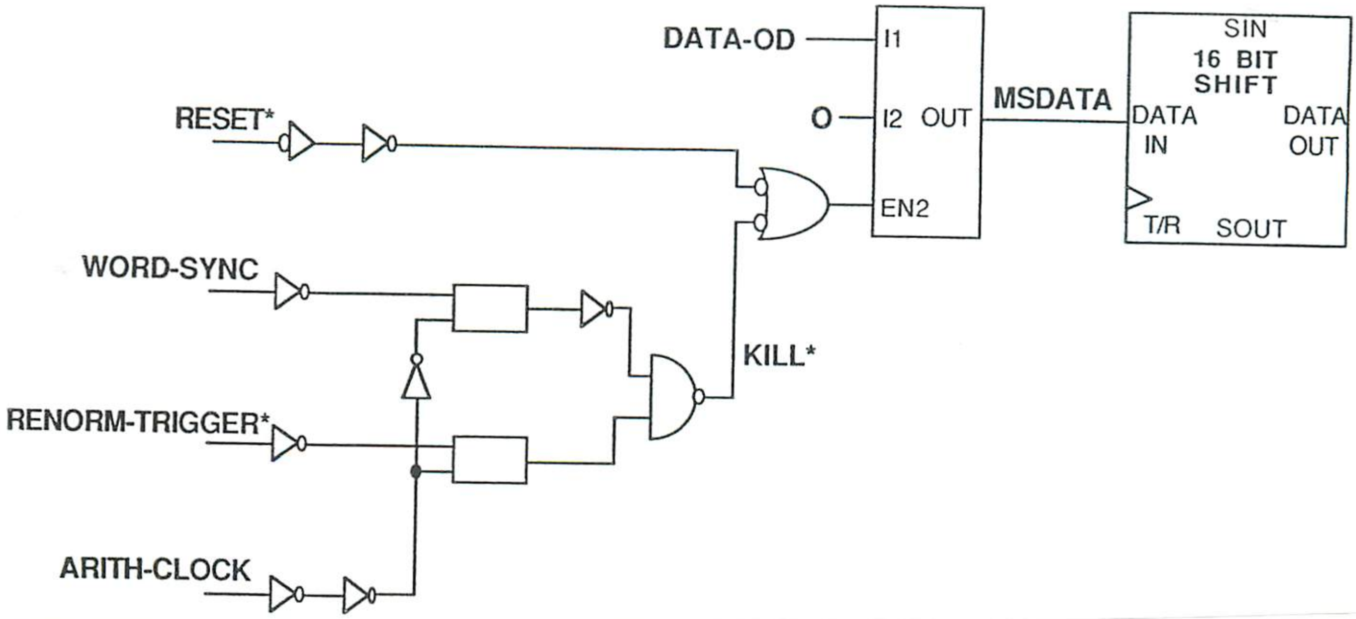
   Resolution: This circuitry appears twice in BFLY, once in each Compare Select. Both instances can be tested simultaneously.

   See Test Sheet 2 for a diagram of this circuit and a functional test.

6. The D inputs to scan flip-flop 11P and scan flip-flop 7P at the BFLY level, and the Init_Bus_In input of Metric Computer do not have valid test values in the scan vectors. This is because there is no logic driving these flip-flops, therefore the cloud associated with these inputs is vacuous and hence no tests are generated for these lines.

   Resolution: These inputs will have been assigned a value of X in the scan test vectors. Alter the test vectors such that they assume a value of 0 in one vector and 1 in another. The expected output vector must be altered accordingly to match the input vector. (See Section 5.3.1 for further details.)

Test Sheet 1



Tests

1. In run mode bring KILL* high by setting WORD_SYNC low and RENORM_TRIGGER* low. Apply one system clock to latch these signals. Bring DATA_OD high (see Test Sheet 3). Set RESET* low - this will select the I2 mux input and force MSDATA to a low. Apply one system clock to load the low into 16 BITSHIFT. Change to test mode, clock out the scan chain and look for a low in positions 21, 37, 54, and 70 of the scan chain. Note that this circuitry appears twice in each COMPARE_SELECT, and there are two COMPARE_SELECT circuits.

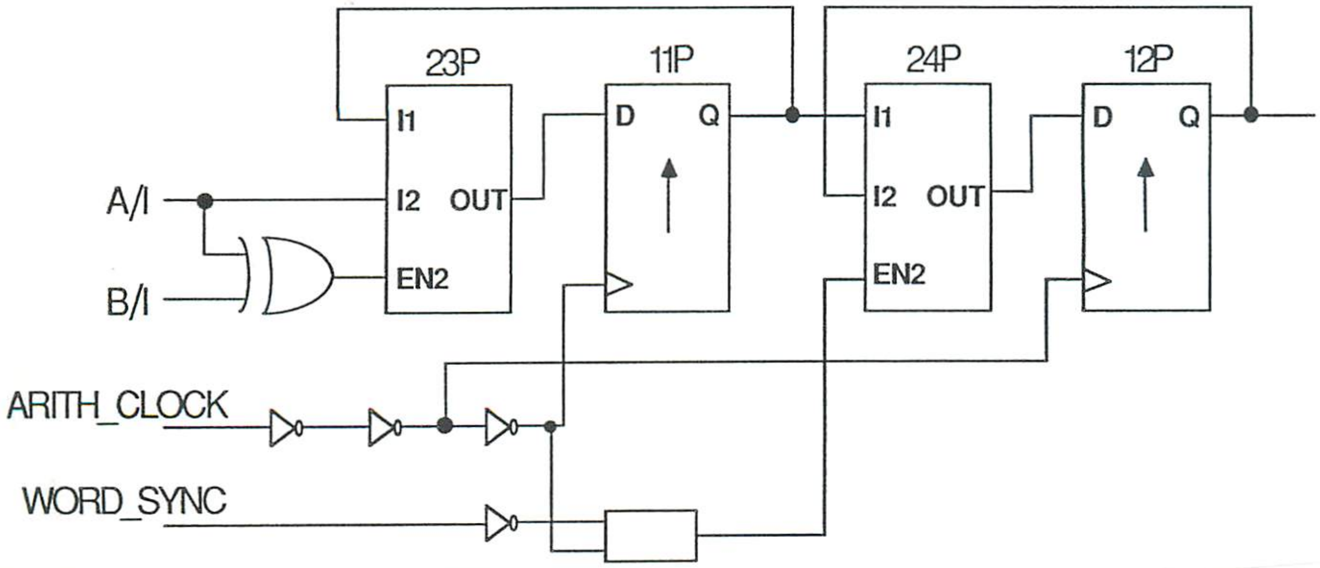
2. Repeat test 1, this time setting WORD_SYNC high and RENORM_TRIGGER* high to generate the high on KILL*.

3. In run mode bring KILL* low by setting WORD_SYNC high and RENORM_TRIGGER* low. Apply one system clock to latch these signals. Bring DATA_OD high (see Test Sheet 3). Set RESET* high - once again selecting the I2 mux input and forcing MSDATA low. Apply one system clock to load this low value into 16 BITSHIFT. Change to test mode, clock out the scan chain and look for a low in positions 21, 37, 54, and 70 of the scan chain.

4. In run mode bring KILL* high by setting WORD_SYNC low and RENORM_TRIGGER* low. Apply one system clock to latch these signals. Bring DATA_OD high (see Test Sheet 3). Set RESET* high - this will select the I1 mux input force MSDATA high. Apply one system clock to load this high value into 16 BITSHIFT. Change to test mode, clock out the scan chain and look for a high in positions 21, 37, 54, and 70 of the scan chain.

5. In run mode bring KILL* high by setting WORD_SYNC high and RENORM_TRIGGER* high. Apply one system clock to latch these signals. Bring DATA_OD low (see Test Sheet 3). Set RESET* high - this will select the I1 mux input and force MSDATA low. Apply one system clock to load this low value into 16 BITSHIFT. Change to test mode, clock out the scan chain and look for a low in positions 21, 37, 54, and 70 of the scan chain.

Test Sheet 2



Tests

1. Bring A/I low and B/I high (see Test Sheet 3), leaving the clock in a high position. In run mode bring WORD_SYNC high and transition the clock from high to low - this will latch a low into scan register 11P and propagate it through mux 24P. Bringing the clock back high again will latch a low into scan register 12P. Change to test mode, clock out the scan chain and look for a low in positions 53 and 86 of the scan chain, as this circuitry appears once in each COMPARE_SELECT, and there are two COMPARE_SELECT circuits.

2. Repeat test 1, this time with A/I high and B/I low. (see Test Sheet 3)

   Check for a high in positions 53 and 86 of the scan chain.

3. While in the test mode, scan in a vector with a high in positions 53 and 86, a low in positions 52 and 85, and leave the clock in a high position. Change to run mode, set WORD_SYNC low and transition the clock from high to low - this will put a low on the I1 input of mux 24P, a high on the I2 input of mux 24P, and will select the I2 input. Bring the system clock high - this will latch the output of mux 24P into scan register 12P. Change to test mode, clock out the scan chain and look for a high in positions 53 and 86 of the scan chain.

4. Repeat test 3, this time with a low in positions 53 and 86, and a high in positions 52 and 85. Check for a low in positions 53 and 86 of the scan chain.

Test Sheet 3

How to access A and B inputs to 16 Bit Shift in COMPARE_SELECT:

The A input of COMPARE_SELECT 6P is the sum of ACC_METRIC_IN_0, BR_METRIC_0, and an internal carry.

The B input of COMPARE_SELECT 6P is the sum of ACC_METRIC_IN_1, BR_METRIC_1, and an internal carry.

The A input of COMPARE_SELECT 7P is the sum of ACC_METRIC_IN_0, BR_METRIC_1, and an internal carry.

The B input of COMPARE_SELECT 7P is the sum of ACC_METRIC_IN_1, BR_METRIC_0, and an internal carry.

ACC_METRIC_IN 0 and 1, and BR_METRIC 0 and 1 are driven by the flip-flops in the scan chain - in positions 1, 2, 7, and 16 respectively. The internal carry signals are driven by the flip-flops in the scan chain in positions 17, 18, 19, and 20 respectively. For simplicity we will hold the ACC_METRIC and BR_METRIC signals at zero and use the carry signals to control the sum bit. The resulting test vectors are shown below.

For Test Sheet 1

scan vector position								Data-OD
1	2	7	16	17	18	19	20	(all 4 instances)
0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0

For Test Sheet 2

scan vector position								Compare Select Inputs			
1	2	7	16	17	18	19	20	A (6P)	B(6P)	A(7P)	B(7P)
0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	1	0	1
0	0	0	0	1	0	1	0	1	0	1	0

5 Results from CRETE

5.1 Schematic capture

Figures 5-11 indicate the circuit schematics of a butterfly processor as entered into Cbase.

5.2 Cloud generation

CRETE processed this circuit and identified 9 clouds which are listed in Table 1. The following clouds are equivalent.

Clouds 1 and 2

Clouds 3 and 4

Clouds 6, 7, and 8

Cloud 9 is actually an r-cloud since it shares common primary inputs (FORCE and FORCECTRL) with replicated versions of this logic existing in the other butterfly processors.

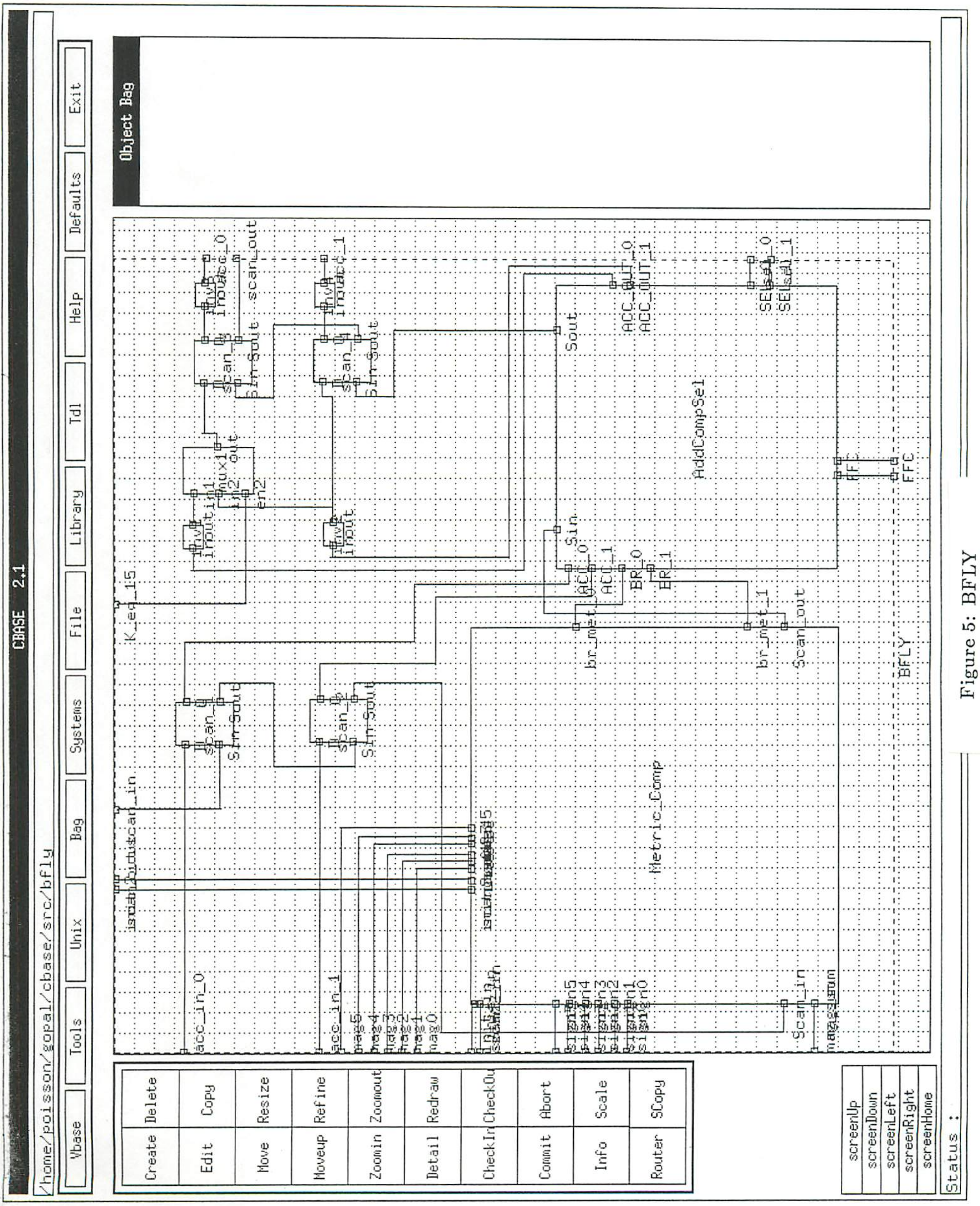


Figure 5: BFLY

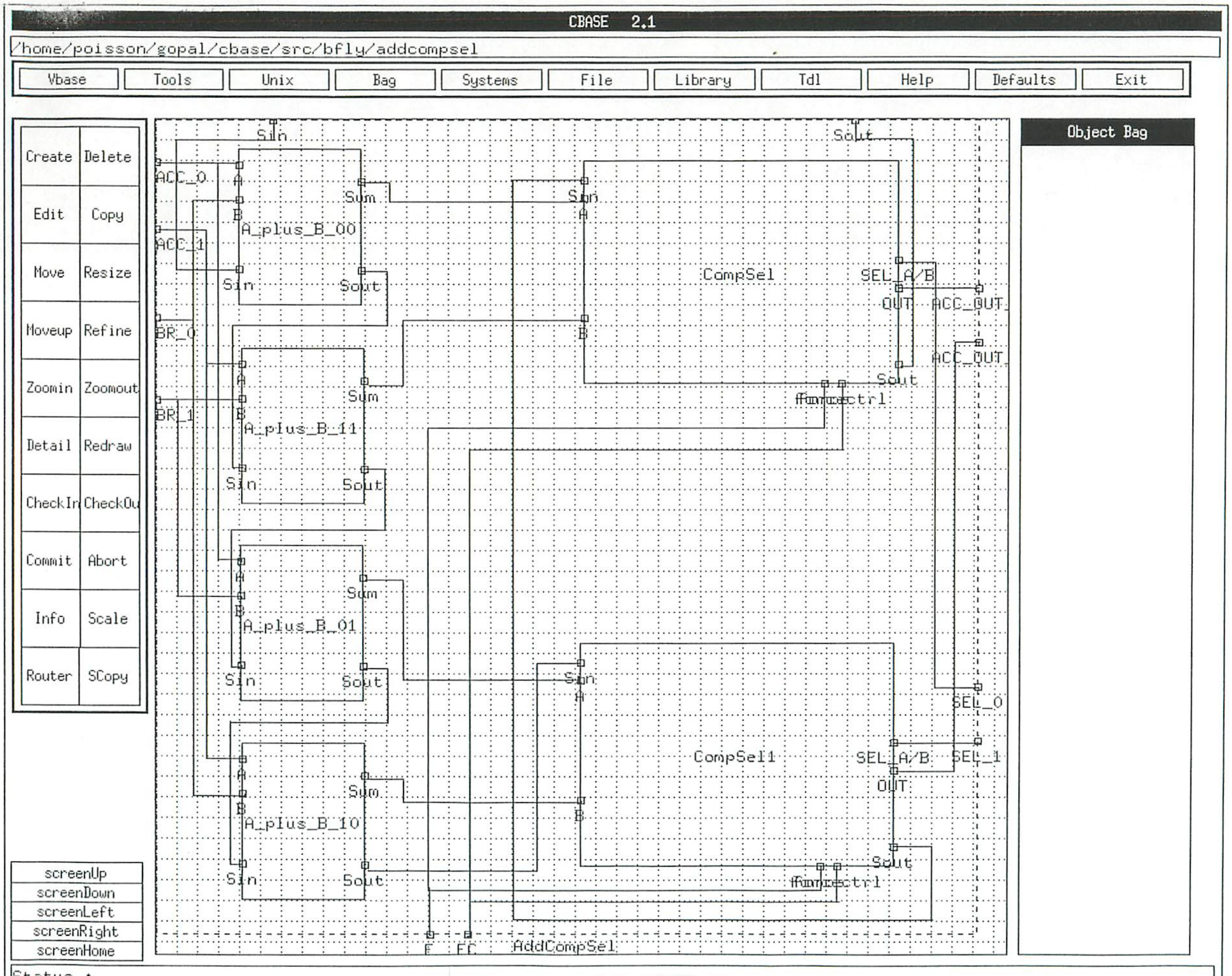


Figure 6: AddCompSel

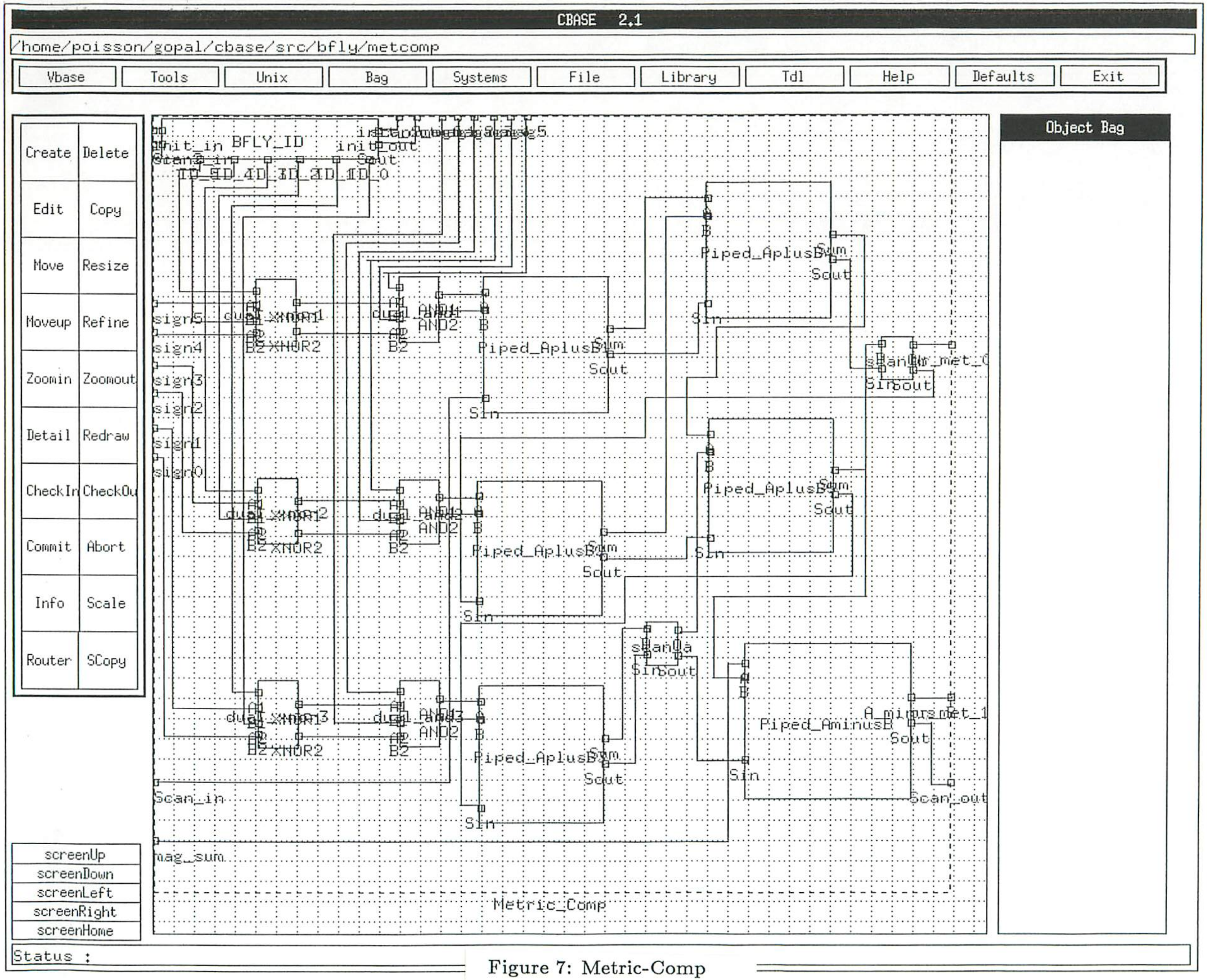


Figure 7: Metric-Comp

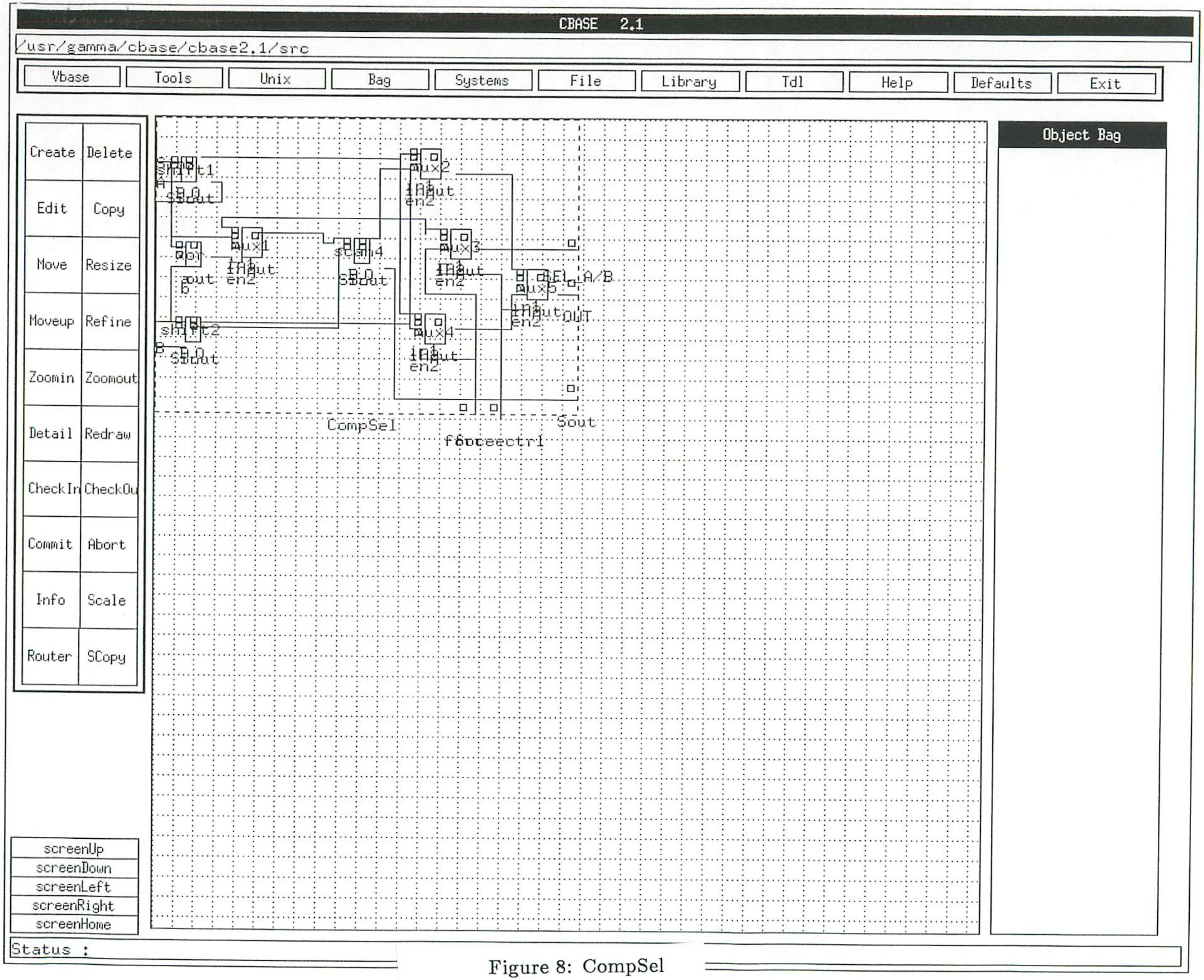


Figure 8: CompSel

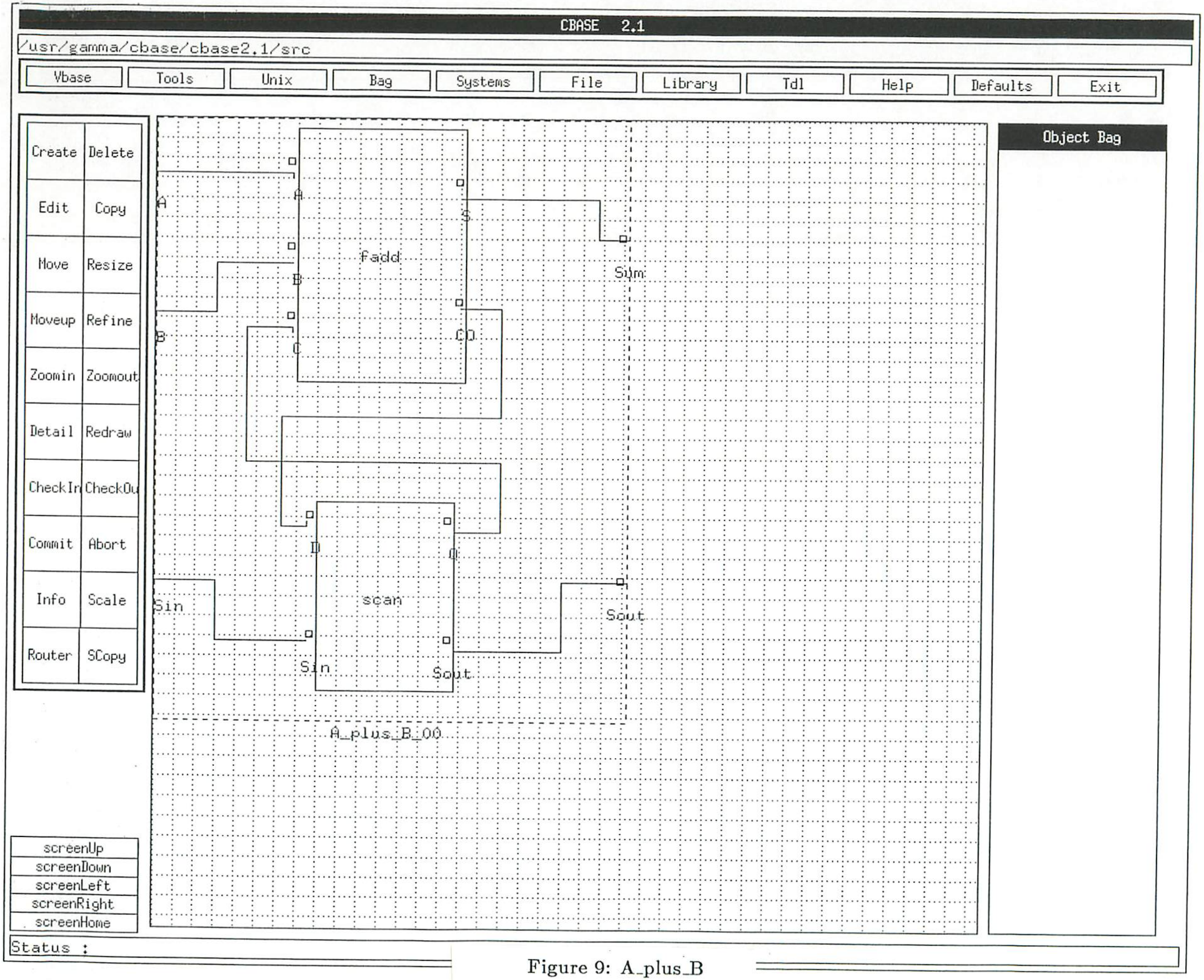


Figure 9: A_plus_B

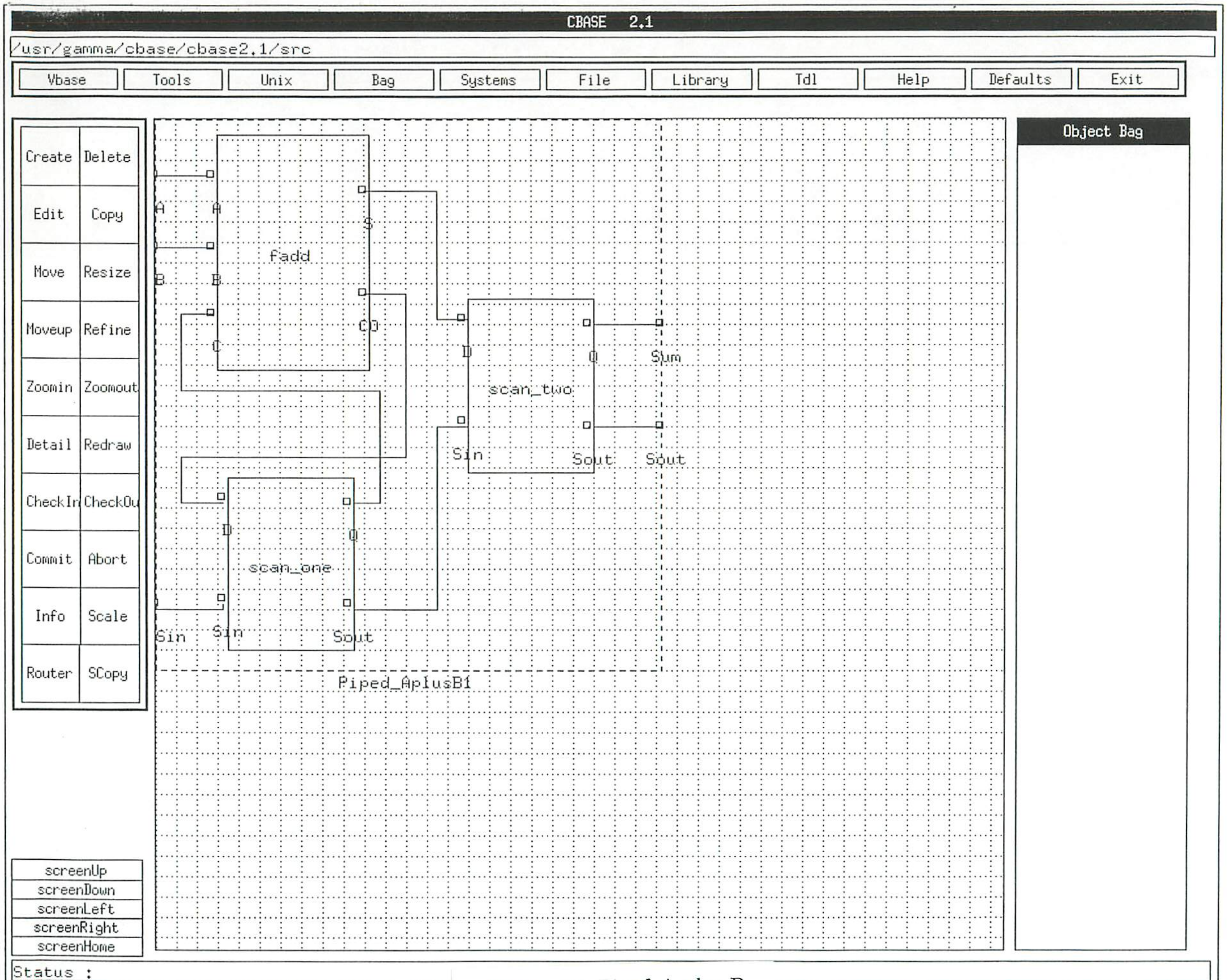


Figure 10: Piped A plus B

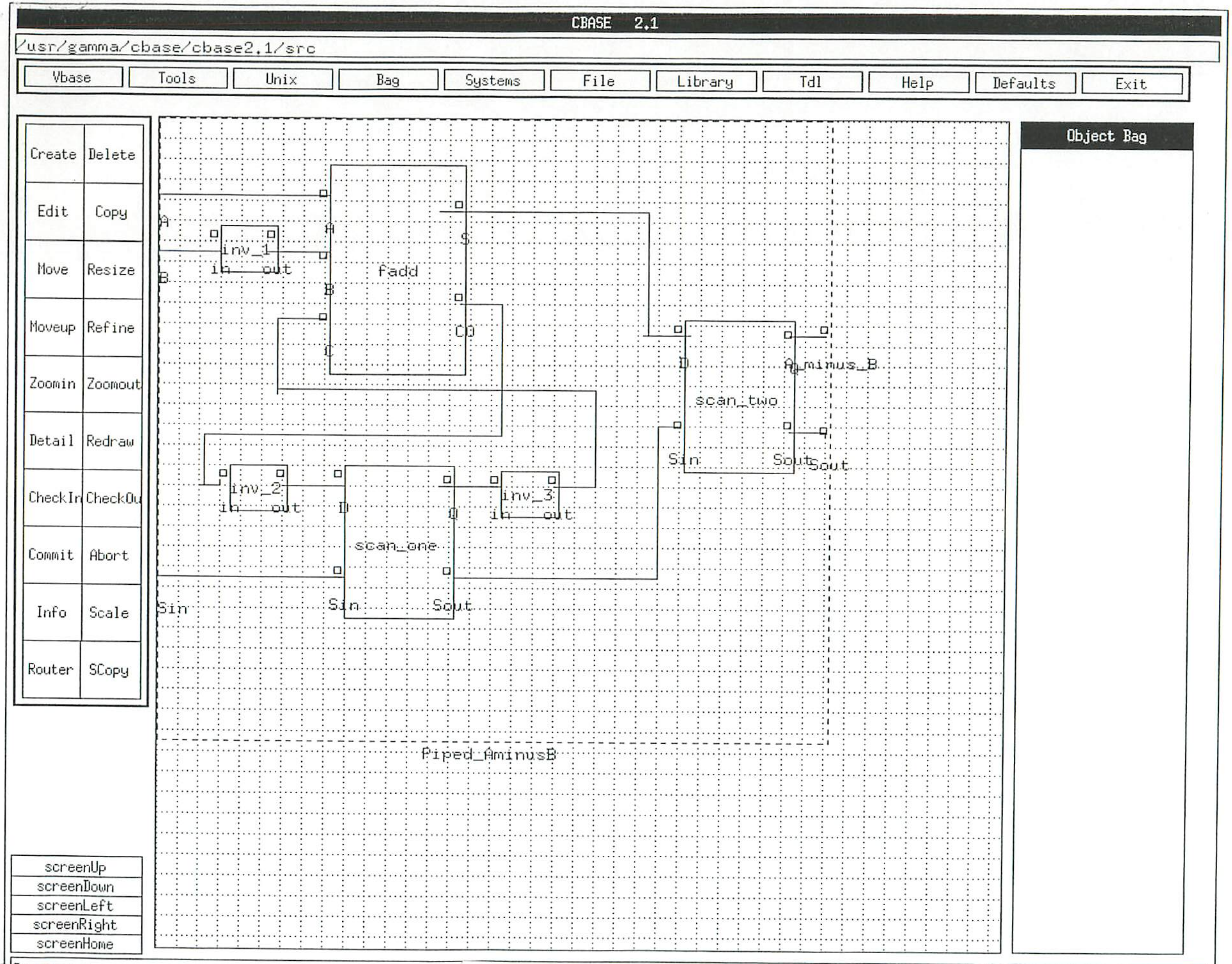


Figure 11: Piped A minus B

CLOUD #	CLOUD NAME	CONTENTS	CONTAINED IN CELL
1	inv3	inv	INV (QBAR of FF (9P)) / BFLY
2	inv4	inv	INV (QBAR of FF (5P)) / BFLY
3	fadd1357	fadd	PIPED_A+B (12P) / METCOMP / BFLY
4	fadd1670	fadd	PIPED_A+B (11P) / METCOMP / BFLY
5	Cluster1860	fadd	PIPED_A-B / METCOMP / BFLY
		inv	PIPED_A-B / METCOMP / BFLY
		inv	PIPED_A-B / METCOMP / BFLY
		inv	PIPED_A-B / METCOMP / BFLY
6	Cluster2574	fadd	PIPED_A+B (7P) / METCOMP / BFLY
		dual_and	AND (30P & 32P) / METCOMP / BFLY
		dual_xnor	XNOR (26P & 31P) / METCOMP / BFLY
7	Cluster3569	fadd	PIPED_A+B (9P) / METCOMP / BFLY
		dual_and	AND (27P & 28P) / METCOMP / BFLY
		dual_xnor	XNOR (22P & 23P) / METCOMP / BFLY
8	Cluster4556	fadd	PIPED_A+B (8P) / METCOMP / BFLY
		dual_and	AND (21P & 29P) / METCOMP / BFLY
		dual_xnor	XNOR (24P & 25P) / METCOMP / BFLY
9	Cluster156	mux	BFLY
		inv	INV (41P) / BFLY
		inv	INV (42P) / BFLY
		fadd	A+B (2P) / ACS / BFLY
		fadd	A+B (3P) / ACS / BFLY
		fadd	A+B (4P) / ACS / BFLY
		fadd	A+B (5P) / ACS / BFLY
		xor	COMPSEL / ACS / BFLY
		mux	COMPSEL / ACS / BFLY
		mux	COMPSEL / ACS / BFLY
		mux	COMPSEL / ACS / BFLY
		mux	COMPSEL / ACS / BFLY
		mux	COMPSEL / ACS / BFLY
		xor	COMPSEL1 / ACS / BFLY
		mux	COMPSEL1 / ACS / BFLY
		mux	COMPSEL1 / ACS / BFLY
		mux	COMPSEL1 / ACS / BFLY
		mux	COMPSEL1 / ACS / BFLY
		mux	COMPSEL1 / ACS / BFLY

Table 1: Clouds in a butterfly processor

5.3 Test vector files

CRETE generated three test vector files for the butterfly processor, consisting of a file for the primary I/O, the first scan chain and the second scan chain.

The organization of these files is given below.

5.3.1 Test vector file for the primary I/O

Tables 2 & 3 list the port names of the primary inputs and outputs of the cell BFLY. Table 4 lists the test vectors associated with these I/Os. For each test vector pair, the first row corresponds to an input test vector and the second row gives the expected response for the outputs. There are 19 primary inputs, 5 primary outputs, and 34 test vectors. The leftmost bit of an input vector corresponds to input signal F, and the leftmost bit of an output vector corresponds to output signal sel_1.

The following primary inputs pass through a clocked buffer at the BFLY level:

Renorm_Trigger
Symbol_Mag $< 5..0 >$
Symbol_Sign $< 5..0 >$
Symbol_Mag_Sum
Force
Forcectrl

For each test vector, signals on these primary inputs must be applied one clock period before other primary input signals in this test vector are applied.

Primary outputs Sel0 and Sel1 pass through a 16 bit shift register at the V16BP level. Signals on these primary outputs can be observed by scanning out the contents of the two 16 bit Memory Interface units.

For each input test vector, an X entry indicates that the corresponding primary input is only feeding a scan flip-flop. For each output response vector, an X entry in the second row indicates that the corresponding primary output is fed by a scan flip-flop. CRETE does not generate tests for these lines because the cloud associated with them is vacuous. These lines can be tested as follows:

The Xs in one or more test vector pair must be replaced by 1s to test for s-a-0 faults on these lines; the Xs in one or more test vector pair must also be replaced by 0s to test for s-a-1 faults on these lines.

5.3.2 Test vector file for the first scan chain

The test vectors pairs to be scanned in and the corresponding response to the be scanned out of this chain are given in Table 5. For each vector pair, the first row corresponds to scan-in test vector, the second row specifies the expected scan-out response.

The right most bit in the input test vector is the first bit to be entered. Once the scan chain is fully loaded, this bit will be in the last flip-flop in the scan chain. There are 88 flip-flops in the scan chain of the modeled butterfly processor (See Table 6). The primary input signal to this scan chain is labeled ISCANIN.

The rightmost bit in the response vector is the first bit to come out of the scan chain. The primary output signal name for this scan chain is labeled OSCANOUT.

For each test vector pair, an X entry in the first row indicates that the corresponding scan flip-flop is directly feeding another flip-flop or a primary output, and an X entry in the second row indicates that the corresponding scan flip-flop is directly fed by another flip-flop or primary input. These lines can be tested as follows:

The Xs in one or more test vector pair must be replaced by 1s to test for s-a-0 faults on these lines; the Xs in one or more test vector pair must also be replaced by 0s to test for s-a-1 faults on these lines.

5.3.3 Test vector file for the second scan chain

The format for this file is the same as for the first scan chain. The results are shown in Tables 7 and 8. There are 6 flip-flops in this scan chain which makes up the butterfly ID register. The primary input and output signal names for this scan chain are IINITIN and OINITOUT, respectively.

Cbase Name		Schematic Name
F	—	FORCE
FC	—	FORCECTRL
mag_sum	—	SYMBOL_MAG_SUM
acc_in_0	—	ACC_METRIC_IN_0
acc_in_1	—	ACC_METRIC_IN_1
K_eq_15	—	K_EQ_15
sign5	—	SYMBOL_SIGN_5
sign4	—	SYMBOL_SIGN_4
sign3	—	SYMBOL_SIGN_3
sign2	—	SYMBOL_SIGN_2
sign1	—	SYMBOL_SIGN_1
sign0	—	SYMBOL_SIGN_0
mag5	—	SYMBOL_MAG_5
mag4	—	SYMBOL_MAG_4
mag3	—	SYMBOL_MAG_3
mag2	—	SYMBOL_MAG_2
mag1	—	SYMBOL_MAG_1
mag0	—	SYMBOL_MAG_0
init_in	—	INIT_BUS_IN

Table 2: Primary inputs

Cbase		
Name		Schematic Name
sel_1	—	Select_1
sel_0	—	Select_0
acc_0	—	ACC_METRIC_OUT_0
acc_1	—	ACC_METRIC_OUT_1
init_out	—	INIT_BUS_OUT

Table 3: Primary outputs

```
T1:  011XX1101010111111X
     0000X

T2:  001XX1101010111111X
     0111X

T3:  001XX1000000111111X
     01XXX

T4:  010XX0010101111111X
     00XXX

T5:  000XX1010101111111X
     00XXX

T6:  000XX1000000111111X
     01XXX

T7:  011XX1000000010101X
     00XXX

T8:  010XX1000000101010X
     00XXX

T9:  00XXX1101010000000X
     10XXX

T10: 01XXX0000000111111X
     00XXX

T11: 00XXX1000000010101X
     11XXX

T12: 11XXX0010101101010X
     11XXX

T13: 00XXX1111111000000X
     00XXX

T14: 00XXX0XXXXXXXXXXXXX
     11XXX

T15: 10XXX0XXXXXXXXXXXXX
     10XXX

T16: 00XXX0XXXXXXXXXXXXX
     11XXX

T17: 10XXX0XXXXXXXXXXXXX
     00XXX

T18: 11XXX0XXXXXXXXXXXXX
     11XXX

T19: 11XXX0XXXXXXXXXXXXX
     11XXX

T20: 01XXX0XXXXXXXXXXXXX
     00XXX

T21: 11XXX0XXXXXXXXXXXXX
     11XXX

T22: 11XXX1XXXXXXXXXXXXX
     11XXX

T23: 10XXX1XXXXXXXXXXXXX
     00XXX

T24: 01XXX0XXXXXXXXXXXXX
     00XXX

T25: 00XXX1XXXXXXXXXXXXX
     01XXX

T26: 01XXX1XXXXXXXXXXXXX
     00XXX

T27: 10XXX0XXXXXXXXXXXXX
     11XXX

T28: 10XXX0XXXXXXXXXXXXX
     10XXX

T29: 01XXX1XXXXXXXXXXXXX
     00XXX

T30: 11XXX0XXXXXXXXXXXXX
     11XXX

T31: 11XXX0XXXXXXXXXXXXX
     11XXX

T32: 00XXX0XXXXXXXXXXXXX
     01XXX

T33: 00XXX1XXXXXXXXXXXXX
     10XXX

T34: 01XXX0XXXXXXXXXXXXX
     00XXX
```

Table 4: Test vectors and output responses for primary I/O

```
T1:  001010110111X0001110XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX0111
     XX1101X110111X0010001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX000

T2:  011111010101X0001011XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX0100
     XX1010X101010X0100011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX100

T3:  111010011101X1101011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XX1010X101010X0010110XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T4:  111101111011X1000010XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XX
     XX1110X111011X1110110XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX000

T5:  001000111001X1000101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XX
     XX1001X100110X0000010XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T6:  111101010001X0100101XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XX
     XX1001X100110X1101011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX100

T7:  011000010011X0110110XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XX
     XX1000X100010X1101100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T8:  101111111111X1101001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XX
     XX1011X101110X1010011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX100

T9:  111XXX11XXX1XXX10001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XX01XXX01XX01XXX11110XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX011

T10: 010XXX10XXX0XXX11100XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XX
     XX10XXX10XX10XXX11011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX010

T11: 100XXX10XXX0XXX11011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XX01XXX01XX01XXX10111XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX111

T12: 100XXX00XXX0XXX10110XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XX01XXX01XX01XXX01101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T13: 100XXX00XXX0XXX11011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XX
     XX00XXX00XX00XXX10101XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T14: 11XXXX0XXXXXXXX11000XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX11100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T15: 00XXXX1XXXXXXXX11100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX11001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX000

T16: 10XXXX1XXXXXXXX00101XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XX
     XXXXXXXXXXXXXXXX10011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX000

T17: 10XXXX1XXXXXXXX01100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX10001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX001
```

Table 5: Test vectors for scan chain 1

```
T18: 11XXXX1XXXXXXXX11011XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XX
     XXXXXXXXXXXXXXXX11111XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX110

T19: 00XXXX0XXXXXXXX01001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX00000XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T20: 01XXXX1XXXXXXXX11111XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX11110XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T21: 10XXXX0XXXXXXXX10010XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XX
     XXXXXXXXXXXXXXXX00101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX011

T22: 10XXXX0XXXXXXXX11001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX10100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX000

T23: 10XXXX1XXXXXXXX01000XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XX
     XXXXXXXXXXXXXXXX10001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX100

T24: 10XXXX0XXXXXXXX10010XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX00101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX001

T25: 00XXXX0XXXXXXXX00001XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX00000XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T26: 00XXXX0XXXXXXXX00000XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX00000XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T27: 00XXXX0XXXXXXXX00011XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX00001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX101

T28; 00XXXX0XXXXXXXX00100XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX00000XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX001

T29: 10XXXX1XXXXXXXX01001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX10011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX100

T30: 10XXXX0XXXXXXXX11010XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XX
     XXXXXXXXXXXXXXXX10101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX000

T31: 10XXXX1XXXXXXXX01001XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XX
     XXXXXXXXXXXXXXXX10011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T32: 10XXXX0XXXXXXXX11110XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX01XX
     XXXXXXXXXXXXXXXX11101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX101

T33: 10XXXX1XXXXXXXX01001XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX01XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX10011XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX11XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX111

T34: 10XXXX0XXXXXXXX11110XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX00XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XX
     XXXXXXXXXXXXXXXX11101XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX10XXXXXXXXXXXXXXX0XXXXXXXXXXXXXXX011
```

Table 5: Con't.

Flip-Flop Position	Flip-Flop Name
1	11P/BFLY
2	7P/BFLY
3	4P/PAPB7P/METCOMP/BFLY
4	3P/PAPB7P/METCOMP/BFLY
5	4P/PAPB12P/METCOMP/BFLY
6	3P/PAPB12P/METCOMP/BFLY
7	13P/METCOMP/BFLY
8	4P/PAPB8P/METCOMP/BFLY
9	3P/PAPB8P/METCOMP/BFLY
10	4P/PAPB11P/METCOMP/BFLY
11	3P/PAPB11P/METCOMP/BFLY
12	4P/PAPB9P/METCOMP/BFLY
13	3P/PAPB9P/METCOMP/BFLY
14	10P/METCOMP/BFLY
15	7P/PA_B/METCOMP/BFLY
16	3P/PA_B/METCOMP/BFLY
17	1P/APB2P/ACS/BFLY
18	1P/APB3P/ACS/BFLY
19	1P/APB4P/ACS/BFLY
20	1P/APB5P/ACS/BFLY
21-36	16 BITSHIFT1P/COMPSEL7P/ACS/BFLY
37-52	16 BITSHIFT9P/COMPSEL7P/ACS/BFLY
53	12P/COMPSEL7P/ACS/BFLY
54-69	16 BITSHIFT1P/COMPSEL6P/ACS/BFLY
70-85	16 BITSHIFT9P/COMPSEL6P/ACS/BFLY
86	12P/COMPSEL6P/ACS/BFLY
87	5P/BFLY
88	9P/BFLY

Table 6: Positions of flip-flops in scan chain 1

```
T1:  101010
     XXXXXX

T2:  000000
     XXXXXX

T3:  101010
     XXXXXX

T4:  010101
     XXXXXX

T5:  000000
     XXXXXX

T6:  010101
     XXXXXX

T7:  000000
     XXXXXX

T8:  000000
     XXXXXX

T9:  010101
     XXXXXX

T10: 000000
     XXXXXX

T11: 101010
     XXXXXX

T12: 000000
     XXXXXX

T13: 000000
     XXXXXX

T14: XXXXXX
     XXXXXX

T15: XXXXXX
     XXXXXX

T16: XXXXXX
     XXXXXX

T17: XXXXXX
     XXXXXX

T18: XXXXXX
     XXXXXX

T19: XXXXXX
     XXXXXX

T20: XXXXXX
     XXXXXX

T21: XXXXXX
     XXXXXX

T22: XXXXXX
     XXXXXX

T23: XXXXXX
     XXXXXX

T24: XXXXXX
     XXXXXX

T25: XXXXXX
     XXXXXX

T26: XXXXXX
     XXXXXX

T27: XXXXXX
     XXXXXX

T28: XXXXXX
     XXXXXX

T29: XXXXXX
     XXXXXX

T30: XXXXXX
     XXXXXX

T31: XXXXXX
     XXXXXX

T32: XXXXXX
     XXXXXX

T33: XXXXXX
     XXXXXX

T34: XXXXXX
     XXXXXX
```

Table 7: Test vectors for scan chain 2

Flip-Flop Position	Flip-Flop Name
1	5/6 BIT SHIFT/METCOMP/BFLY
2	4/6 BIT SHIFT/METCOMP/BFLY
3	3/6 BIT SHIFT/METCOMP/BFLY
4	2/6 BIT SHIFT/METCOMP/BFLY
5	1/6 BIT SHIFT/METCOMP/BFLY
6	0/6 BIT SHIFT/METCOMP/BFLY

Table 8: Positions of flip-flops in scan chain 2

5.4 Processing Unclouded Butterfly

The circuit was also processed by combining all the logic into one cloud and processing this cloud through TGS. The cloud has 49 inputs and 28 outputs. The result consists of 39 test vectors. Since the clouded version of the circuit required only 34 test vectors, the clouding process led to a 12.5% savings in test vectors. These same 34 test vectors are used to test the Viterbi decoder chip which contains 16 butterfly processors. If all of this logic were processed as a single cloud, we esitmate that over 60 test vectors would be required.

References

1. J. Statman, G. Zimmerman, F. Pollara and O. Collins, "A Long Constraint Length VLSI Viterbi Decoder for the DSN," TDA Progress Report 42-95, Vol. July-September 1988, Jet Propulsion Laboratory, Pasadena, California, pp. 134-142, November 15, 1988.

2. M.A. Breuer, "Test Aspects of the JPL Viterbi Decoder," TDA Progress Report 42-96, Vol. October-December 1988, Jet Propulsion Laboratory, Pasadena, California, pp. 59-79, February 15, 1989.

3. "CRETE", USC Technical Report, (draft form), 1989.