110055

N95- 16470

The Design Manager's Aid for Intelligent Decomposition DeMAID

James L. Rogers
Ext. 42810

Many engineering systems are large and multi-disciplinary. Before the design of new complex systems such as large space platforms can begin, the possible interactions among subsystems and their parts must be determined. Once this is completed the proposed system can be decomposed to identify its hierarchical structure. DeMAID (A Design Manager's Aid for Intelligent Decomposition) is a knowledge-based system for ordering the sequence of modules and identifying a possible multilevel structure for the design problem. DeMAID displays the modules in an N x N matrix format (called a design structure matrix) where a module is any process that requires input an generates an output. (Modules which generate an output but do not require an input, such as an initialization process, are also acceptable.) Although DeMAID requires an investment of time to generate and refine the list of modules for input, it could save a considerable amount of money and time in the total design process, particularly in new design problems where the ordering of the modules has not been defined.

The decomposition of a complex design system into subsystems requires the judgment of the design manager. DeMAID reorders and groups the modules based on the interactions among the modules, helping the design manager make decomposition decisions early in the design cycle. These interactions can be deleted interactively. The modules are grouped into circuits (the subsystems) and displayed in a design structure matrix format. Feedbacks, which indicate an iterative process, are minimized and only occur within a subsystem. Since there are no feedback links among the circuits, the circuits can be displayed in a multilevel format. Thus, a large amount of information is reduced to one or two displays which are stored for later retrieval and modification. The design manager and leaders of the design teams then have a visual display of the design problem and the intricate interactions among the different modules.

The design manager could save a substantial amount of time if circuits on the same level of the multilevel structure are executed in parallel. DeMAID estimates the time savings based on the number of available processors. In addition to decomposing the system into subsystems, DeMAID examines the dependencies of a problem with design and behavior variables and creates a dependency matrix. This matrix shows the relationship among the independent design variables and the dependent constraint and objective functions. DeMAID is based on knowledge base techniques to provide flexibility and ease in adding new capabilities. Although DeMAID was originally written for design problems, it has proven to be very general in solving any problem which contains modules (processes) which take an input and generate an output.

The user begins the design of a system by determining the level of modules which need to be ordered. The level is the "granularity" of the problem. The design manager may wish to examine disciplines (a coarse model), analysis programs, or the data level (a fine model). Once the system is divided into these modules, the user determines each module's input and output, creating a data file for the main program. DeMAID is executed through a system of menus. The user has the choice to plan, schedule, display the design structure matrix, display the multilevel organization, examine parallelism, examine the dependency matrix, or trace the effects of a change in the design process. The main program calls a subroutine which reads a rule file and a data file, asserts facts into the knowledge base, and executes the CLIPS inference engine. All DeMAID code is in C for portability.

There are several new capabilities planned for DeMAID. Currently, interactions either exist or not with no quantification as to their strength. Sensitivity analysis is to be used to determine whether or not the interface between two modules is strong or weak. Weak interfaces may be deleted or suspended, thereby reducing iteration times. A second capability will allow the user to breakdown the output of a module into several important pieces so individual pieces can be traced as opposed to the entire output. Currently, DeMAID orders modules within a circuit based on minimizing the number of feedbacks. Since several different orderings may produce the same number of feedbacks, a genetic algorithm is being considered to find the optimal ordering based on a user-defined cost function. Finally, a graphical user interface is being added to make DeMAID more user-friendly.

THE DESIGN MANAGER'S AID FOR INTELLIGENT DECOMPOSITION

(DeMAID)

James L. Rogers

Presented at the Workshop on
The Role of Computers in LaRC R&D
June 15-16, 1994

OUTLINE

The problem

Design structure matrix

Overview of DeMAID

Current capabilities

Design trade-offs

New capabilities

Summary

THE PROBLEM

Designing a new and complex, multidisciplinary system such as the conceptual design of a high speed aircraft requires:

(1) Determining the interactions among the computational and data modules

(2) Reordering the sequence of the modules to minimize feedback

(3) Dividing the design work among the design teams

Very few tools available to aid the design manager in making early design decisions

QUOTES FROM DESIGNING THE DESIGN PROCESS BY DR. DANIEL E. WHITNEY

"Design is 100 people in a room arguing."

"Right now product designers have all the fancy computer tools. This gives them an unfair advantage when negotiating with manufacturing people."

"We don't have a problem. It's those dummies in manufacturing."

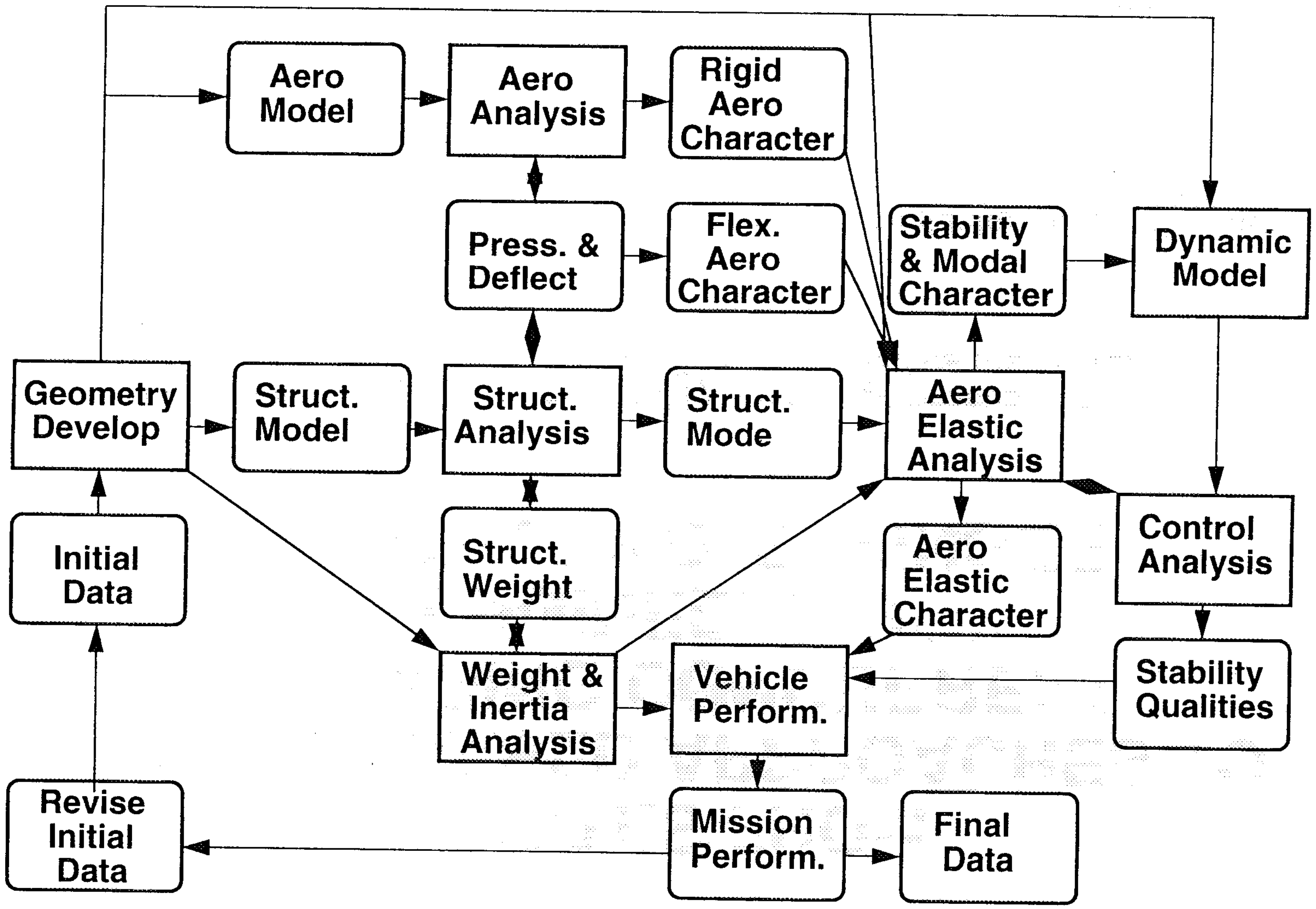
"Engineering proposes designs. Manufacturing counters with problems. There is a lot of conflict, but it is creative."

"Nissan designs all of the parts of an assembly at once while (US auto maker) designs them all in series. How do they ever finish?"

"Word came down from headquarters that we were to eliminate screws. So we redesigned it with all snap fits. Then shipping told us it had to pass a drop test. So we dropped it and it fell apart."

"Just because something can be made doesn't mean it can be manufactured"

PROCESS CHART OF ANALYSES



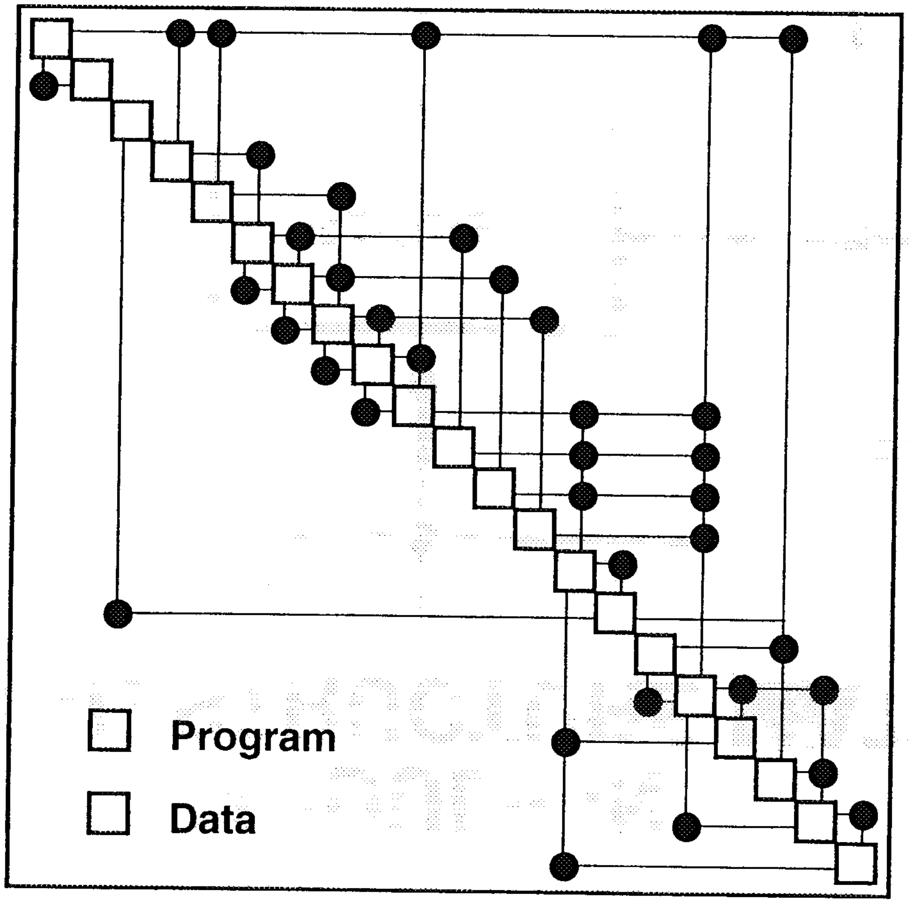
QUOTES FROM "MODEL-BASED APPROACHES TO MANAGING CONCURRENT ENGINEERING" BY STEVEN D. EPPINGER

"Concurrent engineering does not simplify the design process: rather it adds a tremendous amount of inter-task coupling which makes the overall job considerably more difficult."

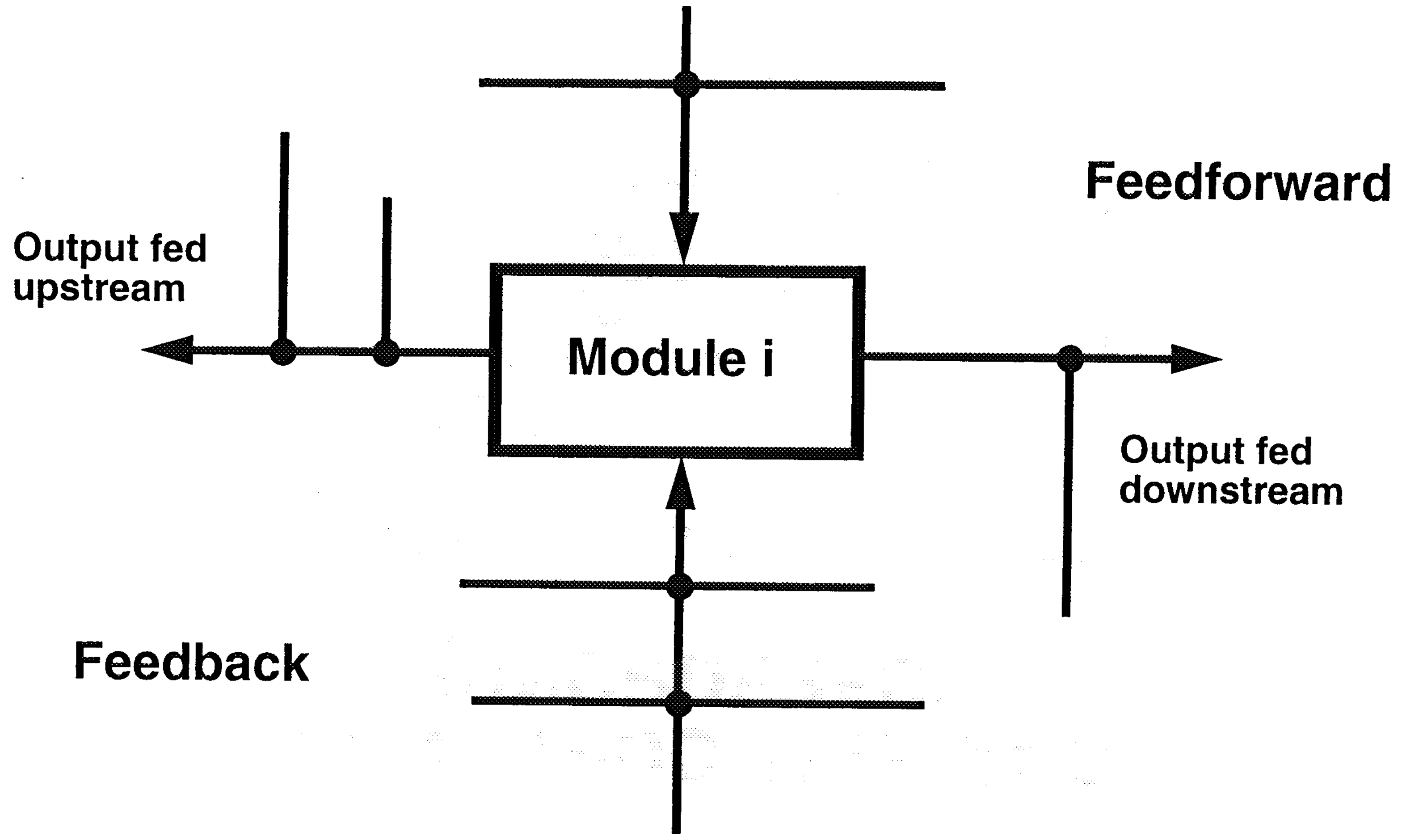
"Concurrent engineering succeeds in reducing overall design time <u>only when adding earlier iteration eliminates later iteration</u> which would have taken even longer."

"The design structure matrix is a tool which allows groups to visualize the relationships among their various activities and reach consensus regarding which feedbacks are to be allowed. However, this tool does not actually show how to alter the process, it merely provides a framework for analyzing alternatives."

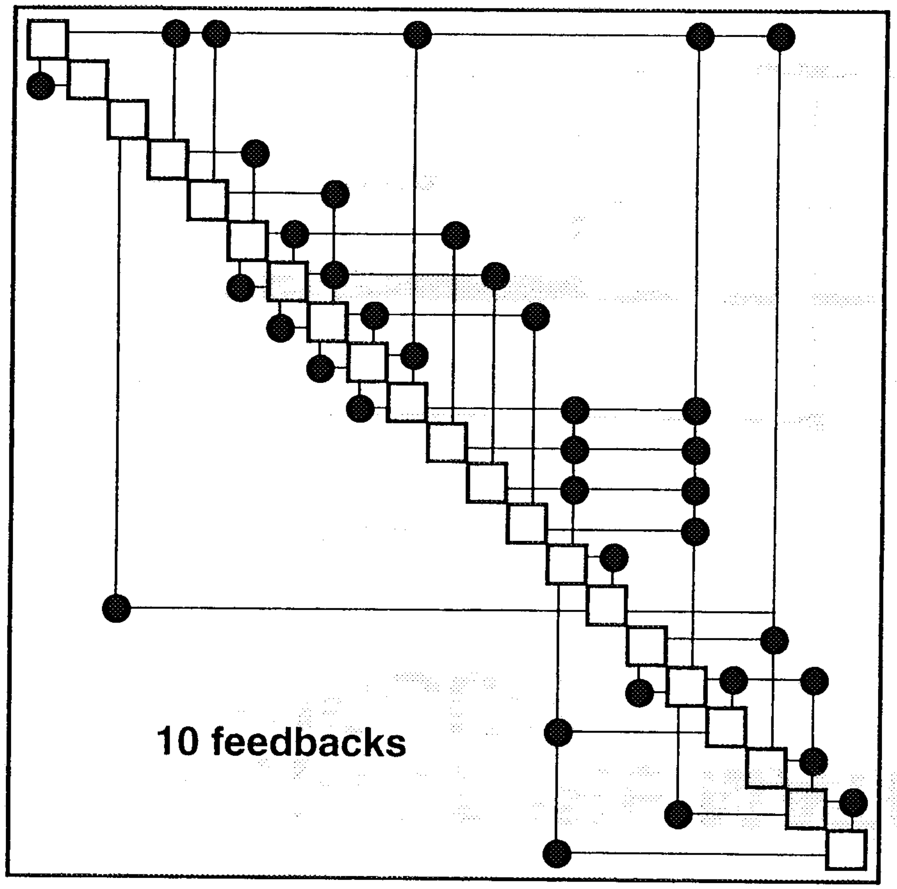
DESIGN STRUCTURE MATRIX (UNORDERED)



MODULE IN DESIGN STRUCTURE MATRIX



DESIGN STRUCTURE MATRIX (UNORDERED)



DESIGN STRUCTURE MATRIX CIRCUIT

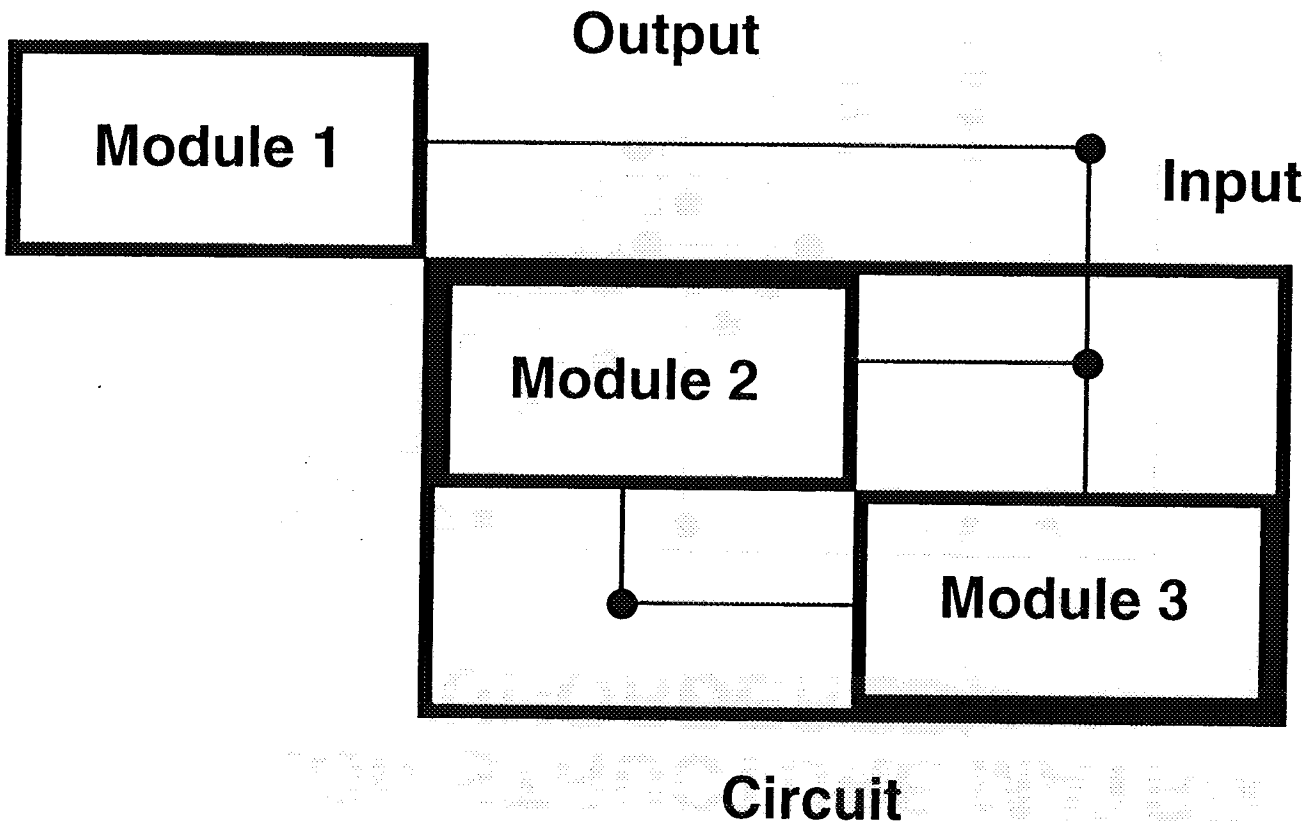
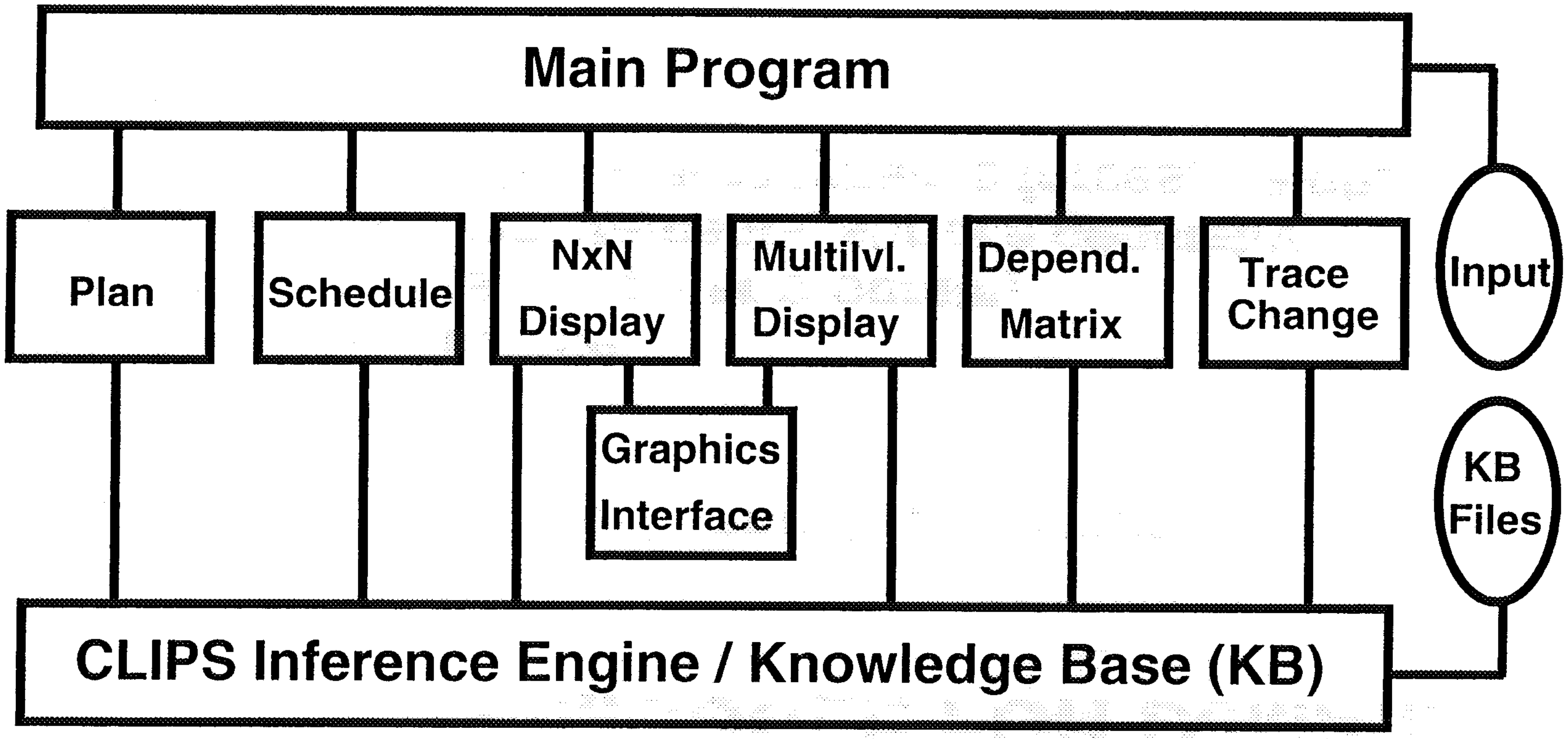


DIAGRAM OF DeMAID

(Design Manager's Aid for Intelligent Decomposition)



DEFINING A MODULE FOR DeMAID

Output = Process (Input1, Input2, . . . , Inputn)

Example:
Output - Performance
Process - ASP (a computer program)
Input - Mission requirements, surface geometry, flap definitions, aerodynamic forces, weight
Time - 34 minutes
Type - 2
Status - uk (unknown)

Sample input:

```
module 1 ASP 2 34 PERF uk MR SG FD AF WT
```

PLANNING FUNCTION

Remove modules not contributing to the solution (modules with output not required as input by another module)

Add modules not in original list to satisfy input requirements

SCHEDULING FUNCTION

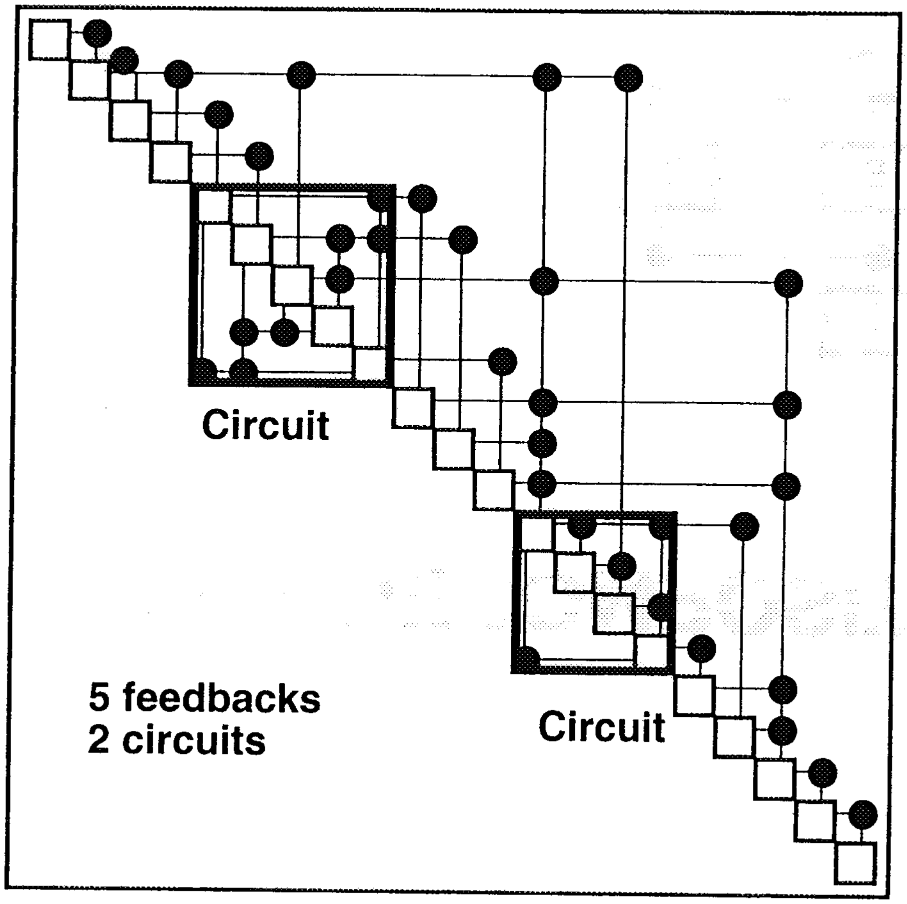
Order modules based on input / output requirements

Minimize feedbacks

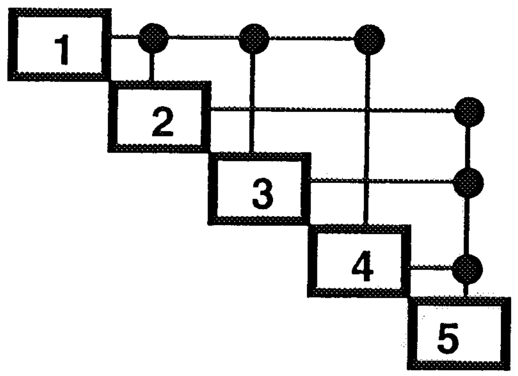
Group modules into circuits

Place modules into design structure matrix
format for display

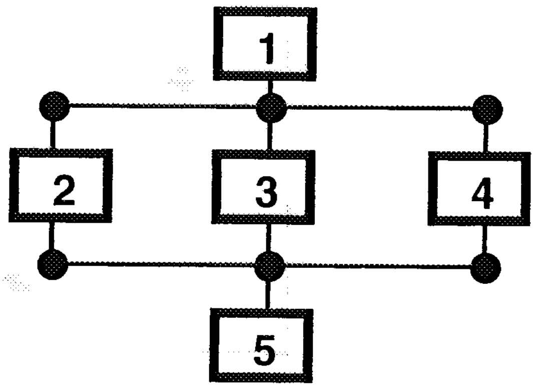
DESIGN STRUCTURE MATRIX (ORDERED)



PROBLEM DECOMPOSITION



Design Structure Matrix of Circuits



Hierarchical Display with Parallelization

DEPENDENCY MATRIX

Design Variables

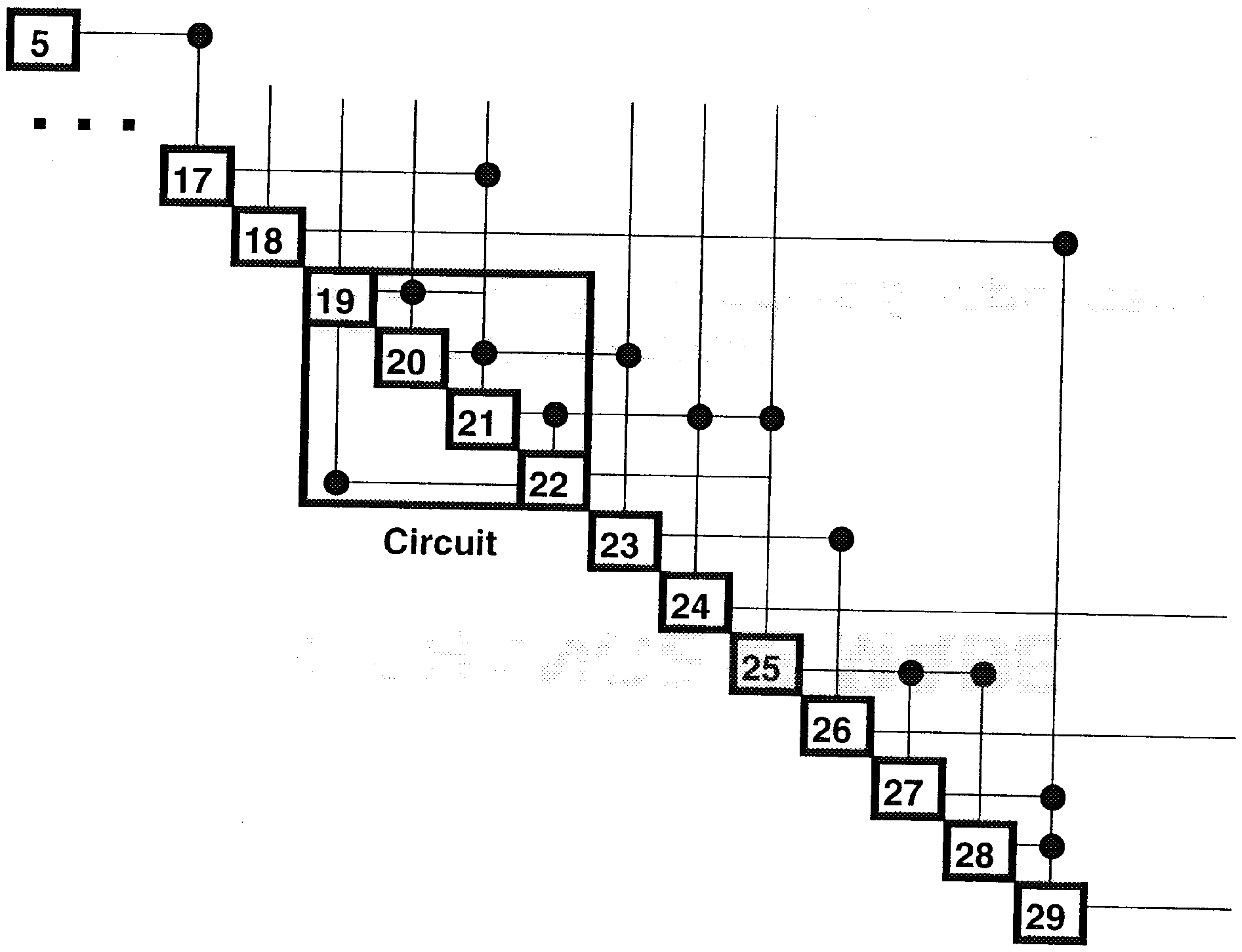
	Module	2	3	4	5	10	11	13	14	15
Constraints	7	X	X	X	X					
	8	X	X	X	X					
	9		X	X	X					
	12		X	X	X	X	X			
	16	X						X	X	X
	17	X						X	X	X

MAKING DESIGN CHANGES

Just because a change is made to some data
in the design process,

this does <u>not</u> mean that all the modules in the
system must be re-executed

DESIGN STRUCTURE MATRIX

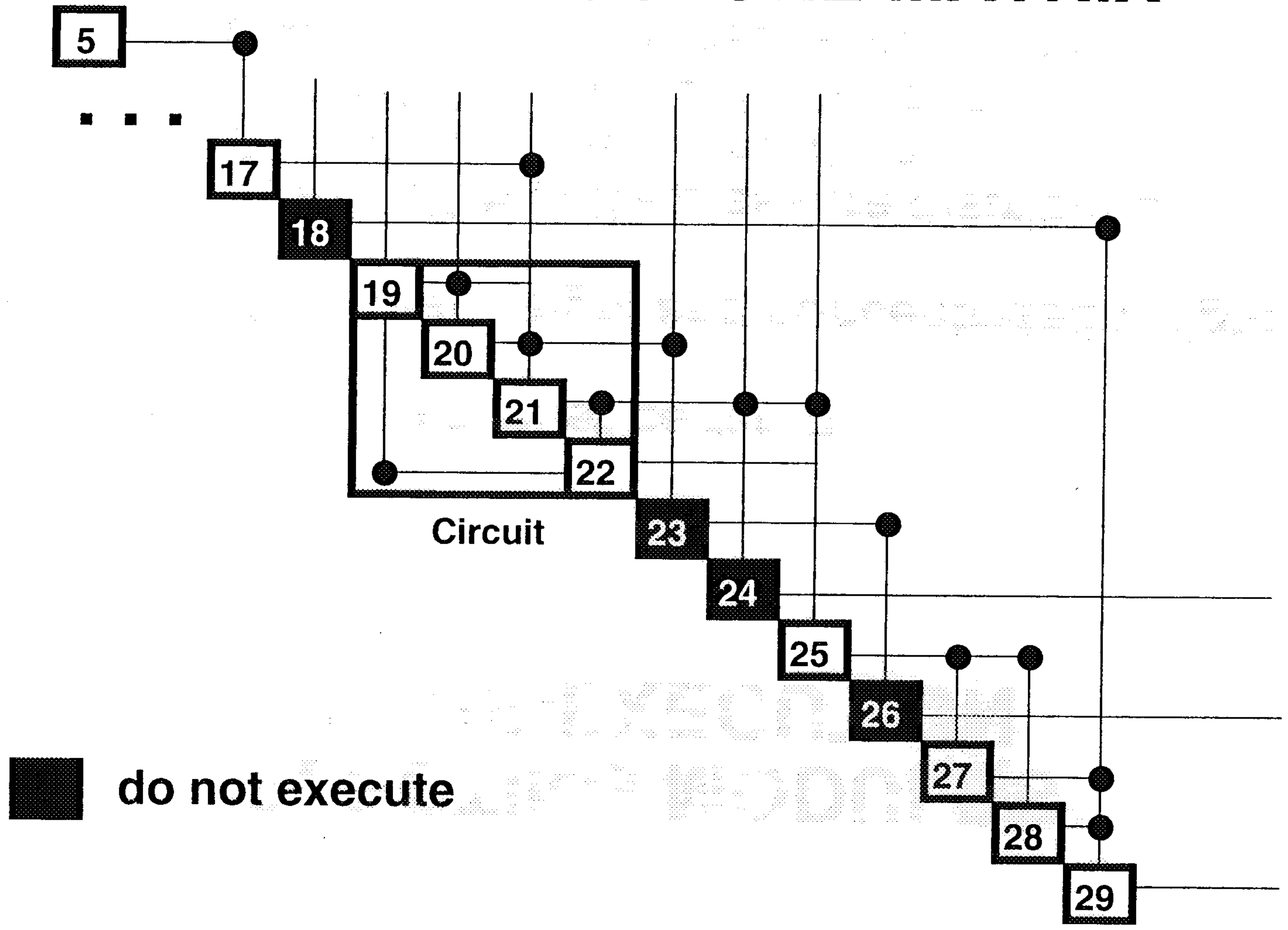


A CHANGE IS MADE

A change is made in the variable
gross weight - module 5 (input data)

What modules must be re-executed to determine
the effect this will have on the variable
elastic polar curves - module 29

MODIFIED DESIGN STRUCTURE MATRIX



SELECTING MODULES FOR RE-EXECUTION

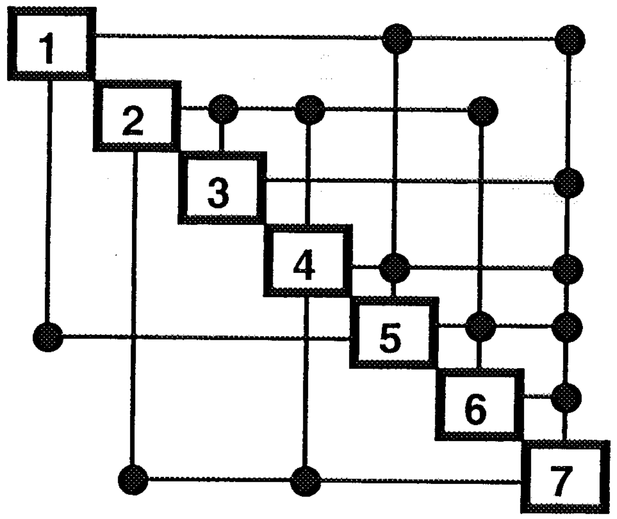
The following modules must be rerun:

TPWEIGHTb (17) - fuel weight and concentrated weight

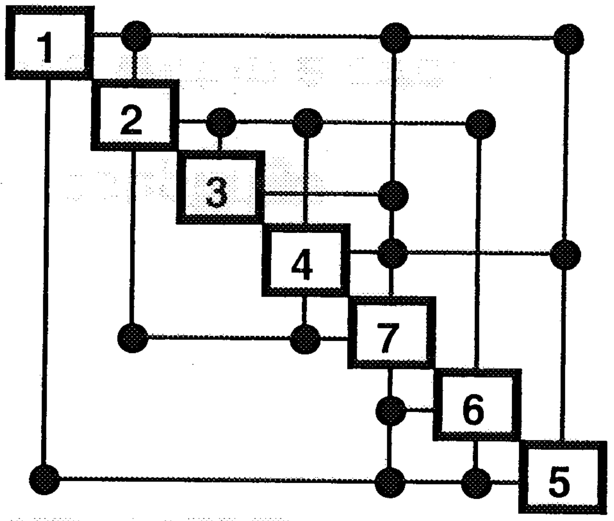
- - - loop - - - SUPERPOS (19) - pressure distribution
- - - loop - - - TRIM (20) - load distribution
- - - loop - - - ELAPSb (21) - wing deformations
- - - loop - - - CAMBER (22) - pressure due to deformations

WDMOD3a (25) - aircraft geometry with elastic deformations
WINGDES (27) - induced drag
AWAVE (28) - wave drag
POLAR (29) - elastic polar curves

EXAMINING DESIGN TRADE-OFFS WITH DEMAID



3 feedbacks
with crossovers



6 feedbacks
no crossovers

NEW CAPABILITIES

Determine "weak" and "strong" couplings

Optimize the ordering of modules within a circuit

Breakdown of output values

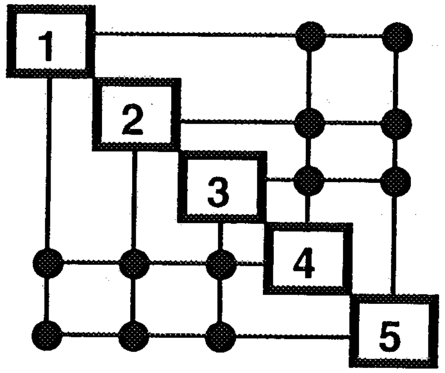
STRENGTH OF COUPLINGS

Determine strength of couplings through sensitivity analysis (Bloebaum SUNY Buffalo)

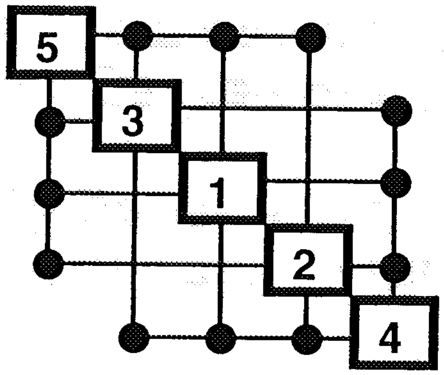
Suspend these couplings (remove process or coupling) without sacrificing system accuracy

Develop rules to determine relationship between a module and those mdoules coupled to it

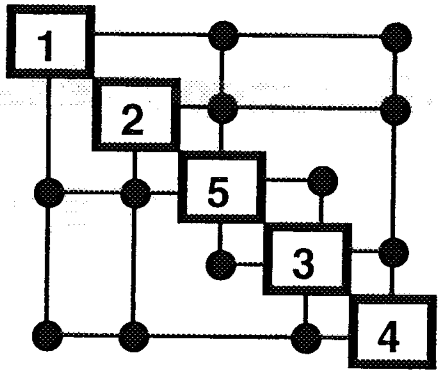
OPTIMIZE THE ORDERING WITHIN A CIRCUIT



Option 1



Option 2



Option 3

Timings

Module 1 - 5	Module 2 - 10	Module 3 - 15
Module 4 - 20	Module 5 - 25	

OPTIMIZE THE ORDERING WITHIN A CIRCUIT

```
Cost function = c1 * # feedbacks      +
                c2 * # crossovers     +
                c3 * timing function  +
                c4 * sensitivity function
```

where c's are user-defined weight coefficients

TIMING FUNCTION

Count number of times a module appears in an iteration

Multiply number of appearances by the execution time of the module

Sum to find the total time

Option 1 = 335   Option 2 = 255   Option 3 = 295

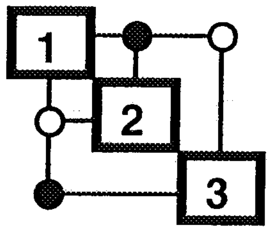
Option 2 has crossovers

SENSITIVITY FUNCTION

Strong coupling = 1 and weak coupling = 0
(actually there will be a wider range)

Add the numbers for feedforward

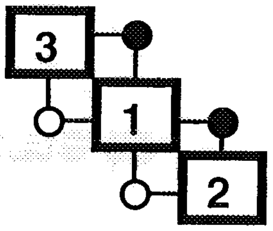
Subtract the numbers for feedback



Option 1

● Strong couplings (1)
1 to 2 and 3 to 1

○ Weak couplings (0)
1 to 3 and 2 to 1



Option 2

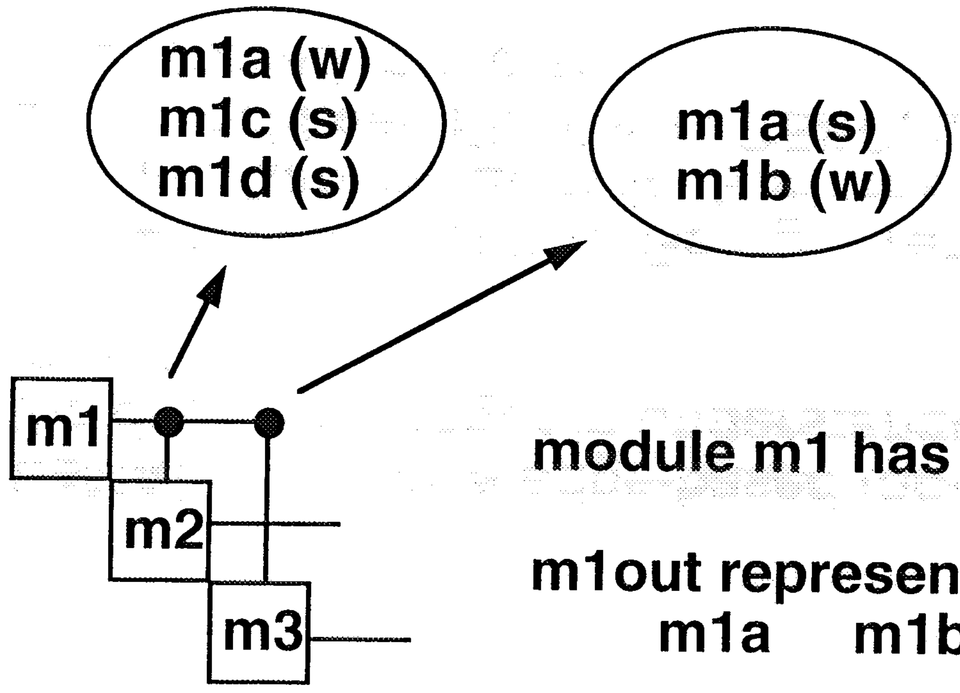
Option 1 = (1+0) - (1+0) = 0
Option 2 = (1+1) - (0+0) = 2

BREAKDOWN OF OUTPUT

Current approach uses only a single name to represent the output of a module

New approach allows single name to reference several output names

EXAMPLE



module m1 has output name m1out

m1out represents four outputs:
m1a m1b m1c m1d

SUMMARY

DeMAID is a knowledge-based tool developed to aid the design manager in understanding a complex design problem

DeMAID is available at Georgia Tech, MIT, SUNY Buffalo, Boeing, General Electric, Northrop, McDonnell Douglas, DEC, Lockheed Ft. Worth, CERC, and Rockwell (widespread interest)

New capabilities to include more optimum sequencing and more coupling information

Several areas for research in design