INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION

INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/542 - 2

ICTP-INFN
SECOND COURSE ON BASIC VLSI DESIGN TECHNIQUES
18 February - 15 March 1991

SOLO 1400
(Lectures 6 to 10)

Franck BUONANNO
European Silicon Structures (ES2)
72/78 Grand Rue
Sevres 92310
France

ES2 Blocks: Aims & Objectives

Generator Availability

Personality Files

Associated Files

Orientation of Blocks

Memory Enable Concepts

Minimize: State Machines

1

Generators

ROM - max size 128kBits	4 to 128 bits/word 16 to 64 K words
RAM- max size 8 Kbits	4 to 64 bits/word 4 to 1024 words
PLA- maximum number of min terms 128	1 to 64 inputs 1 to 32 outputs
MULT- n*m maximum number 32*32	4 to 32 for n 4 to 32 for m

ROM Personality Files

ROM - personality file *file.rom*

address → | *data* ←

		} directives {
.Address 10		*decimal*
.Data 16		*hex*
00	a7	
01	b3	
03	00	
04	10	
..	..	

blck030

ES2

PLA Personality Files

PLA- personality file *file.pla*

```
.Inputs 6
.Outputs 5
.Minterms 26
.Innames   r g s3 s2 s1 s0
.Outnames    t3 t2 t1 t0 ren
.prog
0------          000000
10000            000010
1100-            000000
10001            001000
.........        .....
```

or comes from Minimize

blck040

ES2

Choice & Orientation of Blocks

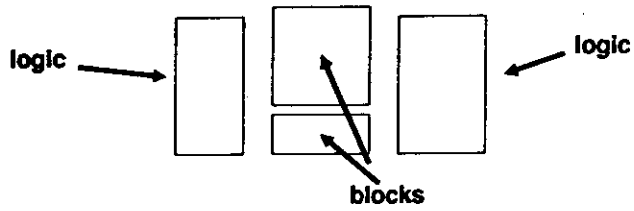
Guidelines for block placement

The block option with the smallest area is not necessarily the most space efficient.

Solo 1400 works best with tall thin columns.

Relate the block size to previously generated blocks.

If you can choose the aspect ratio of two or more of your blocks so that they have the same linear dimension on one of their sides it may be possible to stack them in the same column.

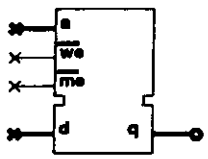


bick060

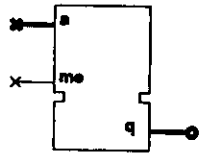
ES2

Memory Enable

RAM



ROM



We recommend in a POSITIVE edge sensitive SYSTEM

choose RAM me to be negative in sense

ROM me to be positive in sense

Connect directly to system clock

bick070

ES2

Running Generate

%generate [filename] [options]

{ interactive program }

file.dft - library file for draft

file.inc - include file for model

file.cif - data base for place,route and draw

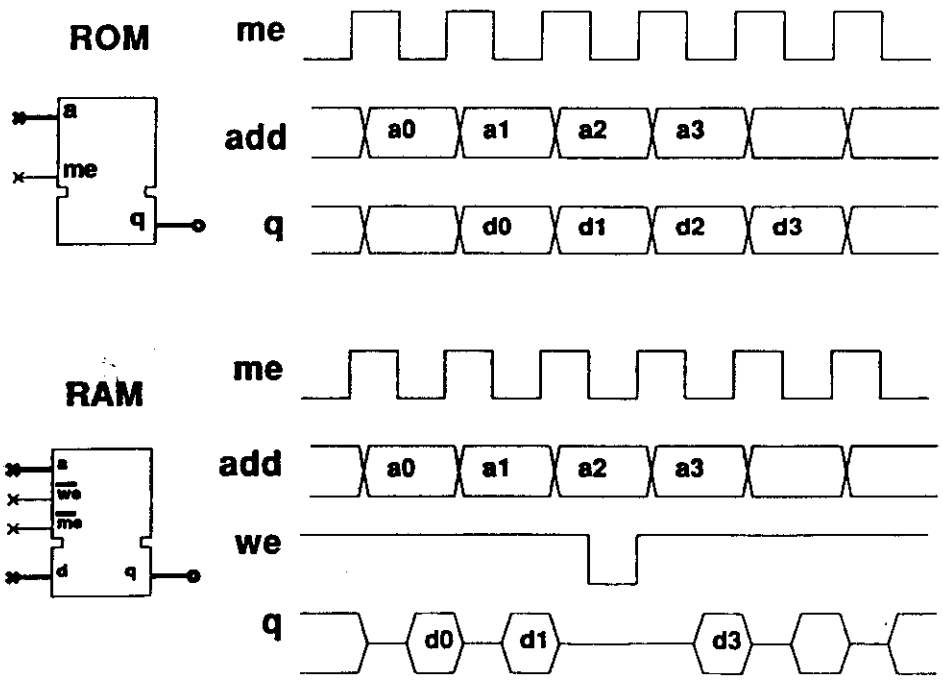
file.emf - exert model file. Change environmental variable EXERT_MODEL_EXTENSION to point to this.

file.ds - electrical design rules (data sheet)

bick050

ES2

Memory Waveforms



bick080

ES2

Minimize

Purpose

bick090

ES2

Minimize

Input formats

bick100

ES2

Minimize

Output formats

bick110

ES2

Minimize

How to run and include into draft

bick120

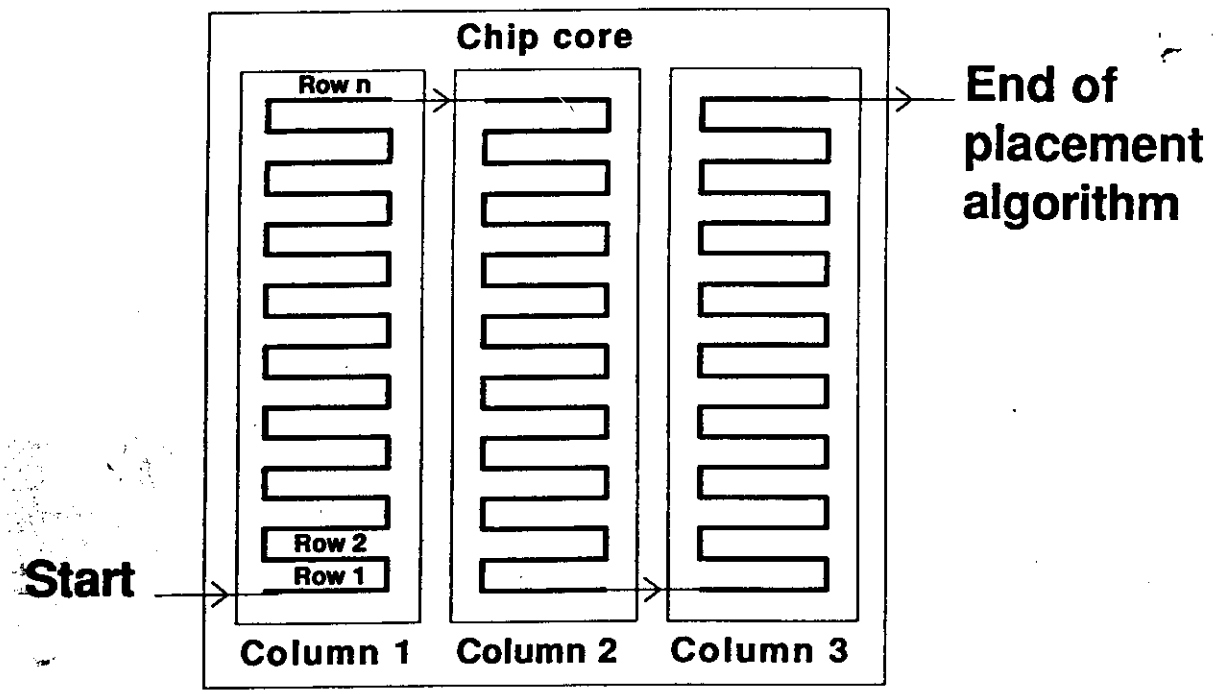
ES2

Layout and Placement of Blocks

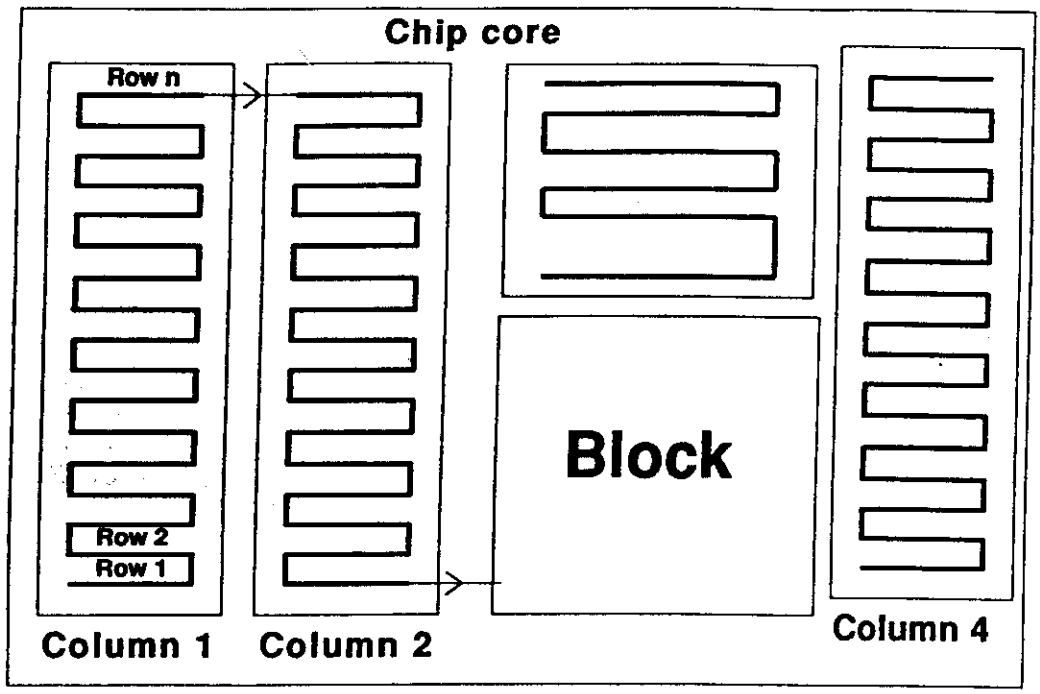
Aims and Objectives

- Physical Hierarchy
- Choosing Stages, Rows and Columns
- Placing Blocks into the Array
- Place with auto, vary and replace options
- Route
- Draw
- Artview

Physical Placement



Placing Blocks



blck150

ES2

Artview Summary of Commands

level	-	set level of abstraction
draw	-	draw entire artview of Chip or package
zoom	-	specify corners, followed by redraw
redraw	-	draw at current zoom setting
layer	-	add and remove process layers
help	-	additional commands available [CTRL-Z to exit from HELP]

blck160

ES2

Chapter 7

MADS

the Simulator

MADS

Contents:

- Environment Variables
- Dirctory Structure
- Batch Simulation Modes:
  - with a .ecf File
  - with a .drv File (Driven)
- WDL (Waveform Description Language)
- Templates
- Constraints and Contentions
- Activity Analysis

MADS 7-2

MADS Overview

Notes:

Running MADS

Used SOLO environment variables with their attributes and possible values (values set by default marked *):

Variable	Attributes	Possible Values
design	name	name of the design
design	process	ecpd15, (ecpd12)
design	load	off*, on (mandatory for shipdes)
design	condition	ind*, mil
design	delay	min, nom, max*
design	blocks	list of generated blocks
exert	probe	on*, off (use .prb file)
exert	constraints	on*, off

MADS 7-3

Running MADS

Notes:

Running MADS (continued)

Used SOLO environment variables with their attributes and possible values (values set by default marked *, not mandatory values marked +):

Variable	Attributes	Possible Values
exert	contentions	on*, off
exert	toggle	on, off*
exert	change	on, off*
exert	halt	integer +
exert	padloadfile	filename +
exert	allpadload	integer +

MADS 7-4

Running MADS (cont.)

With "set exert halt integer", a maximum Simulation Interval in ns may be specified.

Notes:

Running MADS (continued)

Used SOLO environment variables with their attributes and possible values (values set by default marked *, not mandatory values marked +):

Variable	Attributes	Possible Values
mads	use	on, off
mads	drive	on, off
mads	trace	filename +
mads	wdl	on, off*
mads	errorstate	on, off*
mads	keeptrace	on, off*
mads	silent	on*, off

MADS 7–5

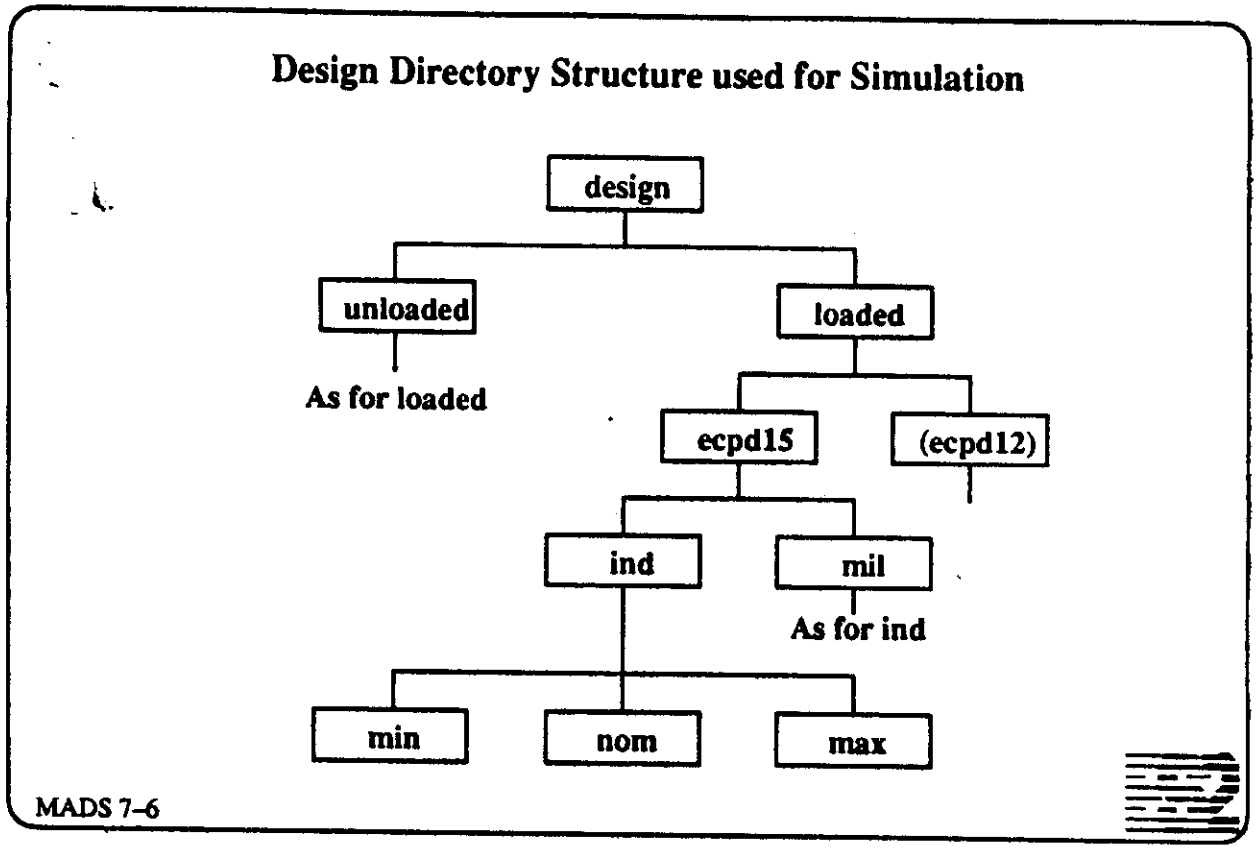
Running MADS (cont.)

In order to run MADS at least the following Variables have to be set:
design name
design process
and if blocks are used:
design blocks
and one of the following 3 stimuli formats:
mads use
mads drive
mads wdl

Notes: _____

_____

_____



Design Directory Structure used for Simulation

At the moment the process ecpd15 (1.5u) is used with its corresponding libraries. The next major release 3.1 will also contain the new ecpd12 process (1.2u).

Notes:

Batch Simulation with a .ecf file

Typical Commands:

Assigning values to nets:

```
set[a(0:3)],16_c => a(3)  a(2)  a(1)  a(0)
                     1     1     0     0
```

Be carful! (Example)

```
set[a,b,c,d],12 => d     c     b     a
                   1     1     0     0
invert[a(0:3)]
```

Setting the default base:

base 8 (alternativly 2, 10, 16)

MADS 7–7

Batch Simulation with a .ecf file

The .ecf file format is a simple transparent way of stimulating the Simulator, but it's lack of forming more complex test patterns (loops and conditions) makes the .drv and .wdl format better suited.

Notes: _____

_____

_____

Batch Simulation with a .ecf file (contimued)

Further Commands:

simulate 200 {Simulate for 200 ns}

simulate 0 {Simulate until stable}

mark [a(0:3)],b] {Include Signals in simulation record (.trc file)}

test [a(0:3)],12 {Compares the value of the specified net with the given value after the last simulation cycle}

map [a,b],(/counter/clk,12)
{Generates aliases for nets:
/counter/clk -> a; net 12 -> b}

MADS 7-8

Batch Simulation with a .ecf file (contimued)

Notes:

Batch Simulation with a .ecf file (continued)

Handling bidirectional nets (Example):

```
mark a(1)      {mark signal a(1) for tracing}
input a(1)     {declare a(1) as input}
set a(1),1     {set input}
simulate 100   {simulate 100 ns}
output a(1)    {declare a(1) as output}
simulate 100   {simulate 100 ns}
```

MADS 7–9

Handling bidirectional Signals in .ecf files

Always allow bidirectional Signals to float for one test cycle, before changing direction to avoid contentions!

Notes: _____

_____

_____

Running MADS with a .ecf file

You must perform at least the following steps:

- **set mads use on**
  tells mads to use the designname.ecf file
- **set design blocks <names of used blocks>**
  tells mads to include the simulation models for the used blocks
- **mads**
  Starts the Simulation

MADS 7-10

Running MADS with a .ecf file

If MADS was formerly run with a Stimuli different from .ecf format (.drv or .wdl) the corresponding variables must first be unset!

e.g.:

```
set mads wdl off
set mads drive off
```

Notes: _____

_____

_____

Batch Simulation with a .drv file (Driven)

Advantages of a driven Simulation against a Simulation with .ecf file:

- **Possibility of repetitive sequences**
- **Relative and absolute times**
- **Expressions available**
- **Simulation can be halted by trigger conditions**

MADS 7-11

Batch Simulation with a .drv file (Driven)

Simulation with drive files is a quick way of generating test patterns of medium complexity.

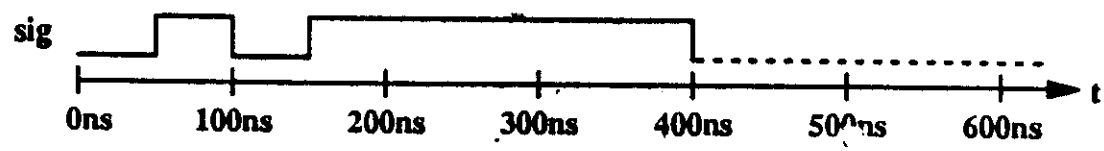
But if timing relationships between different signals become more important WDL is the better approach.

Notes:

Simple Waveforms within Drive Files

Example:

```
sig =l 50 =h 50 =l 50 =h 250 =l
```



That can be shortened to:

```
sig =l 50 50 50 250
```

And by the help of expressions:

```
#t = 50
sig =l t t t t*5
```

MADS 7–12

Simple Waveforms within Drive Files

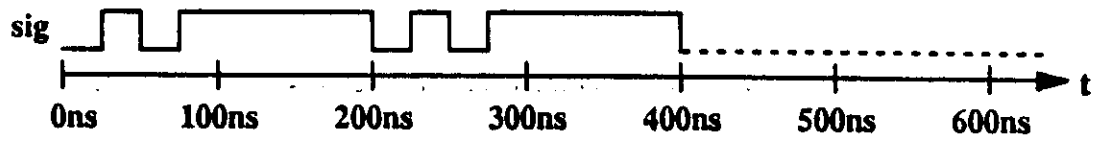
If the signal assignment "=" is missing, the signal simply toggles.

Notes:

Specifying repetitive sequences within Drive Files

Example:

```
#cycle = 50
#t = cycle/2
sig = l [[t]*3 t*5]*2
```

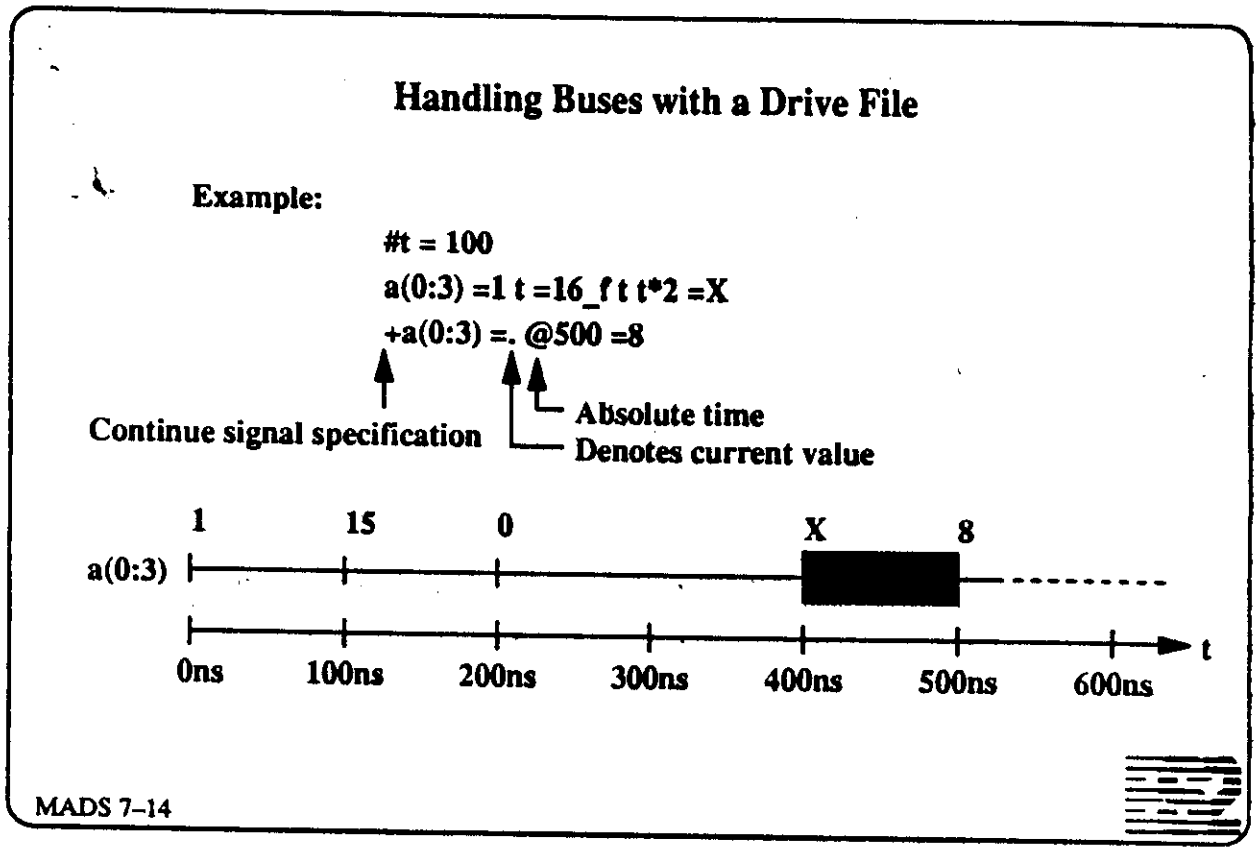


MADS 7-13

Specifying repetitive sequences within Drive Files

Don't confuse the repeatment statement "[ ]",
with it's traditional meaning!

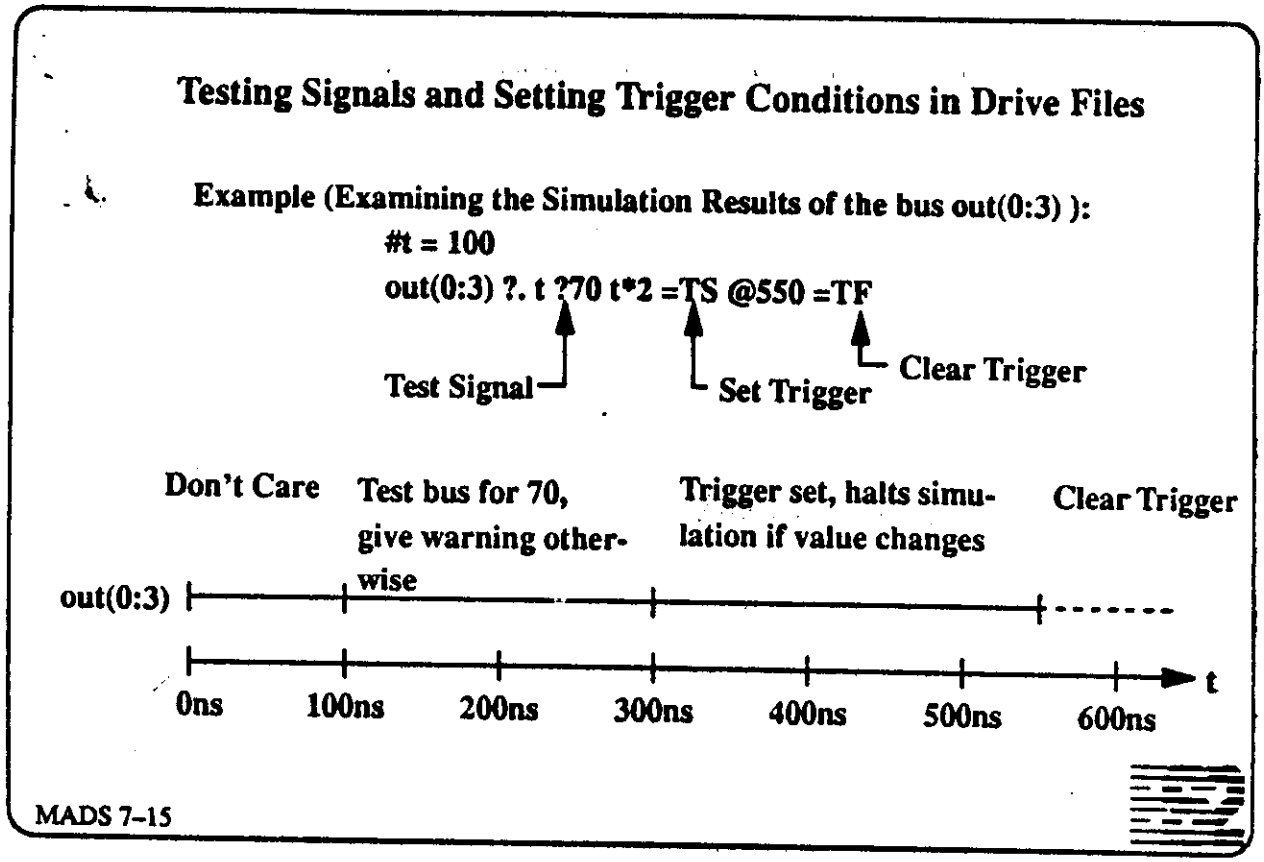
Notes:



Handling Buses with a Drive File

Buses have to be denoted as in DRAFT e.g. a(0:3) **not** a(3:0) with value assignment always MSB...LSB! This applies only to drive files.

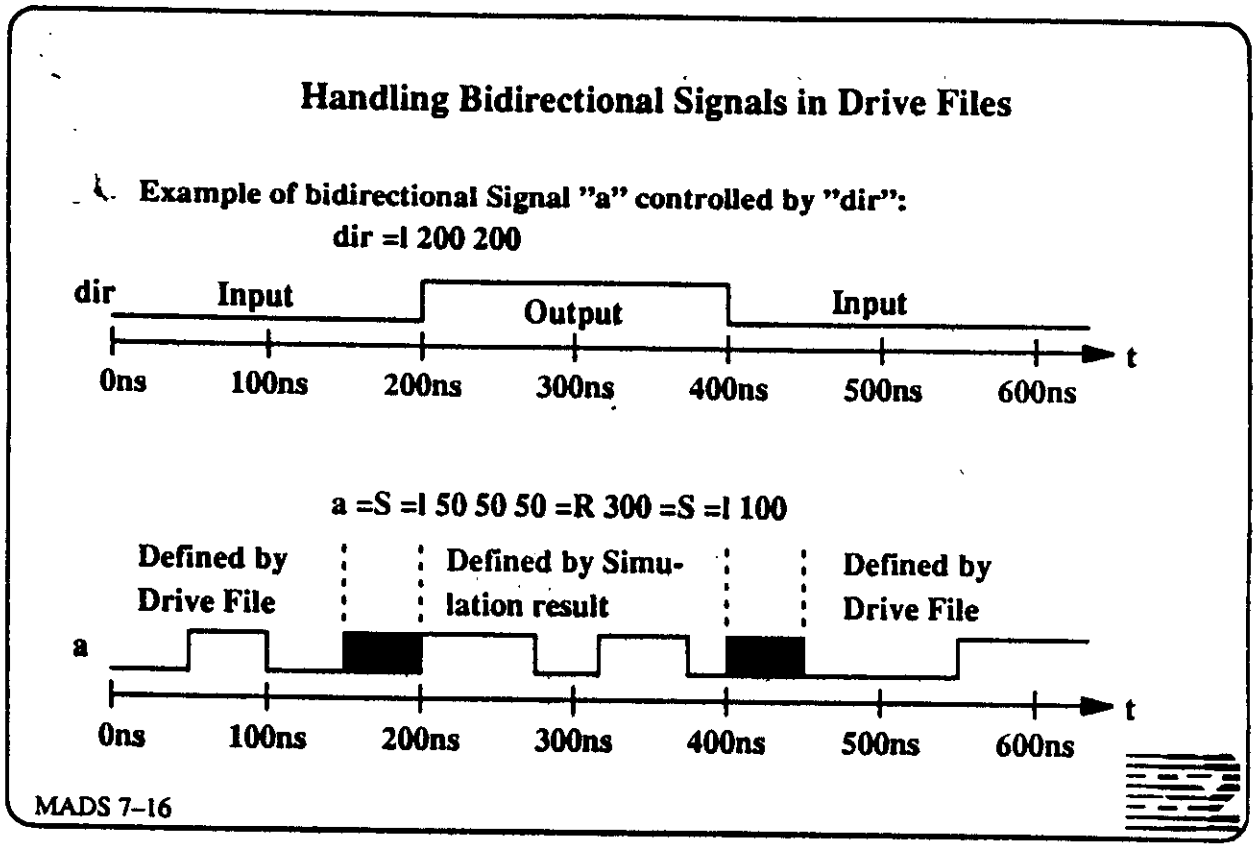
Notes:



Testing Signals and Setting Triggers in Drive Files

These features are especially helpful in examining large Simulation records consistently. The expected result can thereby be predefined in the drive file and need not be visually inspected for the whole record.

Notes:



Handling Bidirectional Signals in Drive Files

Always allow bidirectional Signals to float for one test cycle, before changing direction to avoid contentions!

Notes: _____

_____

_____

Marking Signals in drive files for Inclusion in the Simulation Record

Examples:

a(0:3)	{all Toplevel Signals simply by name}
/counter/clk	{Sublevel Signals by giving the Path}
enable#180	{by mapping net numbers, as given in the expanded listing, to symbolic names}

MADS 7–17

Marking Signals in drive files for Inclusion

Notes:

Drive File Syntax Summary

- l – Low
- h – High
- x – Undefined
- s – Set, bidirectional net defined as Input
- r – Released, bidirectional net defined as Output
- = – Assignment
- ? – Test Signal (Gives a Warning if a mismatch is found)
- # – Constant Declaration and assigning symbols to net numbers
- . – Current Value of net
- @ – Absolute Time
- ts – Set Trigger (Simulation is halted if Signal changes)
- tf – Clear Trigger
- + – Continue Signal Specification after Newline

MADS 7–18

Drive File Syntax Summary

Notes:

Running MADS with a Drive (.drv) file

You must perform at least the following steps:

– set mads drive on

tells mads to use the designname.drv file

– set design blocks <names of used blocks>

tells mads to include the simulation models for the used blocks

– mads

Starts the Simulation

MADS 7–19

Running MADS with a Drive (.drv) file

If you change to a driven Simulation from a format like .ecf or .wdl, don't forget to unset the corresponding variables.

Notes:

Waveform Description Language (WDL)

Contenets:

- WDL Program Structure
- Language Specifics
- Handling Signals
- Operators
- Control Statements
- Functions
- Examples

MADS 7–20

Waveform Description Language (WDL)

WDL is the most powerful way of handling Simulations in the SOLO 1400 Design System. Because of its "C–like" Structure it makes even large sets of test vectors handable in a clear and consistent manner.

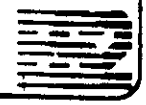
Notes:

WDL Program Structure

Example:


```
Input x;
Input y;
Output z;
main ()
{
<Body of Program>
}
```

MADS 7–21



WDL Program Structure

Notes:

WDL Basic Language Elements

Signals:	**Input, Output, IO, signal**
Variables:	**int**
Functions:	**bool, int, void**
Main Program:	**main ()** **{ Body }**
Statements:	**Integer Arithmetic: a=b+c** **Loops: for, while** **Conditions: if, else** **Special Commands: Set_Cycle, Next_Cycle, (Simulate), Toggle**

MADS 7–22

WDL Basic Language Elements

Note that not all features of WDL can be handled in this chapter, for more information please consult a "C" manual.

Notes: _____

_____

_____

WDL Basic Syntax Features

C like Syntax:

- Case specific
- Comments between // and EOL or between /* and */
- Blanks and Newlines not significant
- All statements terminated by semi–colon

MADS 7–23

WDL Basic Syntax Features

Notes:

Declaring Signals and Assigning Values in WDL

Examples:

```
Input d(7:0);                  // bus d is declared as Input
d(7:0)=2_10111100;             // d is assigned an initial value in bin

Input a(7:0)=0x0A;             /* bus a is declared as Input and
                                  assigned an initial value in hex */

Signal cs("/ramctrl/cs");      /* Declare internal Signal by giving
                                  the pathname, e.g. for tracing */
```

Take care:

```
a(2:0)=2_100;                  // a(2)=1, a(1)=0, a(0)=0
a(0:2)=2_100;                  // a(2)=0, a(1)=0, a(0)=1
```

MADS 7-24

Declaring Signals and Assigning Values in WDL

Note the differences in WDL between a(0:2) and a(2:0). The recommended notation therefore is a(2:0) in WDL for a bus in draft a(0:2).

Neither in draft– nor in .drv files the notation a(2:0) is allowed!

Notes:

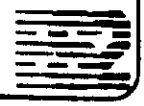
Handling Bidirectional Signals in WDL

Example:

```
IO d(3:0);              // Define bidirectional Data Bus d

d=In;                   // define d as Input
d=2_100X                // Assign value
.
.
.
d=Out                   /* Setting d to be Output, d is no longer
                           defined by the Assignments following
                           the "In" Statement */
```

MADS 7-26



Handling Bidirectional Signals in WDL

Once again: Always allow bidirectional Signals to float for one test cycle, before changing direction to avoid contentions!

Notes: _____

_____

_____

Aliasing and Concatenation in WDL

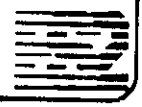
Examples:

```
Input dupper(7:0);          // dupper(7:0) is declared as Input
Input dlower(7:0);          // The same but with dlower

du=dupper;                  // Aliasing
dl=dlower;                  // Aliasing

d=du+dl;                    // Concatenation

deven=d(15:0 by 2);         /* Creates 8 bit subset deven(7:0) of
                               d(15:0), i.e. d(14), d(12),..,d(0) */
```



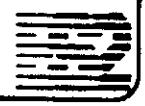
MADS 7–25

Aliasing and Concatenation in WDL

Notes:

WDL Operators

- **Assignment:** =, +=, –=, *=, /=
  a*=3 //a=a*3
- **Arithmetic:** +, –, *, /, .%(remainder)
- **Conditional:** ? :
  a=b ? c : d //if a=b set a=c else set a=d



MADS 7–27

WDL Operators

Just like "C"

Notes: _____

_____

_____

WDL Operators (Cont.)

- Logical/Bitwise: | (or), & (and), ^ (exor)

- Shift: << (Shift left), >> (Shift right)

- Unary: – (negative), ~ (one's Complement), ! (Inversion),
  ++ (auto-Increment), — (auto-decrement)
  a++ //a=a+1

MADS 7–28

WDL Operators (Cont.)

Again like "C"

Notes: _____

_____

_____

Conditional Statements in WDL

Simple Form:

```
if (condition)
statement;
```

More complex Form:

```
if (condition)
{
statement(s);
}
else
{
statement(s);
}
```

The usable Operators in specifying "(condition)" are:

Integer: ==, !=, <, <=, >, >=

Bool: || (or), && (and)

MADS 7-29

Conditional Statements in WDL

The Condition– and Loop Statements make the major differences to .drv and .ecf format.

Notes:

Loops in WDL

The "while" construct:

Simple Form:

```
while (condition)
statement;
```

More complex Form:

```
while (condition)
{
statement(s);
}
```

The "for" construct:

```
for (initialisation; condition; increment)
{
statement(s);
}
```

MADS 7–30

Loops in WDL

Note that both loops and conditions can be nested, what is especially useful in defining complex waveforms.

Notes: _____

_____

_____

Functions in WDL

```
function_type function_name (list of arguments)
{
statement(s);
return value;
}
```

"function_type" can be:
bool – returns 0 or 1
int – returns an integer
void – returns nothing, so the return statement must be omitted

"(list of arguments)" is of the form (type name, type name, ...),
with the possible types: signal or int

MADS 7–31

Functions in WDL

Functions are extremely helpful for keeping a clear and consistent overall structure, so if in doubt always make use of functions!

Notes: _____

_____

_____

WDL Example (Ramtest)

```
Input oe = 1;          Input we = 1;          Input ce = 1;
Input a(7:0) = 0;      IO d(7:0);             int vector = 100;

void Cewr() {
        Simulate;
        we = 0;
        ce = 0;
        Simulate;
        we = 1;
        ce = 1;
        Simulate;
}
void Cerd() {
        Simulate;
        oe = 0;
        we = 1;
        ce = 0;
        Simulate;
        oe = 1;
        ce = 1;
        Simulate;
}
main() {
        Set_Cycle(vector);
        Simulate;
                for (i = 0; i < 256; i++) {
                        a = i;
                        d = In;
                        d = 1 << (i%8);   //Rotate 1 through d
                        Cewr;
                        d = Out;
                        Cerd;
                }
}
```

MADS 7–32

Using WDL for Simulation

There are two possibilities:

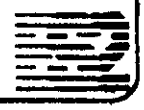
1. Compiling WDL separately:

Create WDL file with name design_name.wdl
Call wdl, the WDL Compiler to compile the .wdl file into .ecf file
set mads use on (to control mads by the .ecf file)
Call mads

2. Tell mads to use a WDL file:

Create WDL file with name design_name.wdl
set mads wdl on (the compilation is done within mads)
Call mads

MADS 7–33



Using WDL for Simulation

The first approach is recommended during development and debugging of the WDL source file, i.e. the Compiler is used stand–alone without the following Simulation. The second is useful when only minor changes are necessary in the source file. MADS always checks if a Compilation is necessary at all, i.e. if the corresponding files have changed.

Notes:

Useful Templates for Simulation

The "extract" Program (normally run after "model") creates the following templates for Simulation in the top level design directory:

- Template drive file (design_name.drt)
- Template .ecf file (design_name.ecf)
- Template .wdl file (design_name.wdl)

These Templates contain all top level signals together with their corresponding declarations, a vector declaration for a default Simulation time step and undefined top level Signal assignments.

Edit these templates to your requirements and save them under the correct name: .ect as .ecf, .drt as .drv, .wdt as .wdl

MADS 7-34

Useful Templates for Simulation

Notes:

Template Examples

.ect

```
MARK a(0:7 by 1)
MARK we
MARK oe
MARK ce
MARK d(0:7 by 1)
SET a(0:7 by 1),
SET we,
SET oe,
SET ce,
SIMULATE 1000
```

.wdt

```
main()
{
        Input     a(7:0);
        Input     we;
        Input     oe;
        Input     ce;
        Output    d(7:0);
        Set_Cycle(1000);
        a = ;
        we = ;
        oe = ;
        ce = ;
        Simulate;
}
```

.drt

```
#vector = 1000
a(0:7) =
we =
oe =
ce =
d(0:7)
```

MADS 7-35

Template Examples

Notes: _____

Verify

Verify helps checking your Simulation Patterns without using the Simulator what helps saving time (mads automatically invokes verify):

- **Define the Test Pattern using .ecf or .drv format (.wdl format must be compiled first into .ecf)**
- **Set the correct options in verify i.e.**
  set verify drive filename, for verifying a drive file
  set verify use [on,off], for verifying a design_name.ecf file
- **Call verify**

In the design directory a file design_name.trc is created, that can be visually inspected with the wave program like a Simulation via;
!wave design_name.trc

MADS 7-36

Verify

MADS always performs "verify" before the actual Simulation if the patterns have changed since the last run, to check its consistency.

Notes:

Probing Signals

Draft allows individual nets to be monitored during Simulation with the "Probe Signal" feature accessible from the Schematic Menu;

- **Probe Signal (Schematic Menu in Draft)**
- **set exert probe on (on by default)**

All probed Signals are included in the Simulation record and can be inspected with wave

To cancel probes, simply reprobe the corresponding Signal in Draft (toggle function)

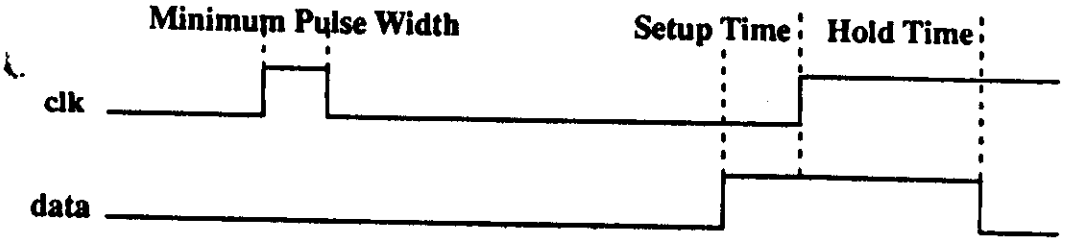
MADS 7-37

Probing Signals

The probed signals are listed in a file called design_name.prb, in the design directory.

Notes:

Timing Constraints



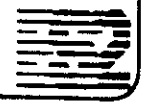
Constraints are checked by the Simulator by default, this may be turned off with "set exert constraints off".

Constraints may be defined:

- **implicitely within libraries (e.g. flip–flops)**
- **within "Model Code" by the user via the "constraint (..)" statement**

Violations are displayed on the screen but are also written to the logfiles "design_name.elf" within the appropriate Simulation subdirectories

MADS 7–38



Timing Constraints

If constraints are violated at flip–flop inputs, i.e. setup and hold time violations, MADS can be advised to propagate X at the corresponding outputs via the errorstate variable, that is normally set off by:
set mads errorstate on

Notes:

Contentions

Contentions arise when there exists more than one driver for a specific net. This typically happens in the following situations:

- **Two tristate drivers are enabled simultaneously (bus conflict)**
- **A bidirectional Pad is switched to Output at the same time the Pad is still driven by the Simulator (IO defined as In) who represents the Tester in the factory**

Contentions are checked by the Simulator by default, this may be turned off with "set exert contentions off".

Violations are displayed on the screen but are also written to the files "design_name.elf" within the appropriate Simulation subdirectories

MADS 7–39

Contentions

Constraint Violations and Contentions in the final Simulation record for the testing equipment must be avoided in any case!!! If this happens the circuit behaves unpredictable during testing stage at the factory, this means it is untestable.

Notes:

Activity Analysis

Activity Analysis is carried out by the Simulator to create the change rate of all nodes, in order to get a quantity for:

- Fault coverage,
  i.e. to check the the Quality of:
  - the test vectors
  - the testability of the design

A high change rate (>95% to pass the final design audit "shipdes") is mandatory.

Activity Analysis is performed automatically during loaded Simulations after physical design, it can be controlled anytime with:
set exert change [on,off]
The results including untoggled nodes are written to the files "design_name.elf" within the appropriate Simulation subdirectories

MADS 7–40

Activity Analysis

Notes:

The Simulation Logfile .elf

Activity Analysis, Contentions and Timing Constraint Violations are logged to the files design_name.elf in the corresponding Simulation Subdirectories.

They can be inspected within the soloshell by means of:

- log mads
  where the referenced logfile is defined by the variables:
  design_load, design_process, design_condition, design_delay
  for ex.: loaded/ecpd15/ind/max/decoder.elf

or by temporarily leaving the soloshell and examining the files with a editor in the corresponding simulation subdirectory:

- Ctrl Z
- vi simulation_dir/design_name.elf

MADS 7–41

The Simulation Logfile .elf

The so referenced nodes can be identified in the schematics or in the model file via the expanded listing file design_name.exl, generated by the expand program, optionally run after "model". To identify these nets in DRAFT use the feature "Look for" from the Schematic Menu.

Notes: _____

_____

_____

Padloading

Padloads can be included during Simulation in two ways:

– As common load for all Pads in pF via:
set exert allpadload integer

– As individual load defined in pF for each Pad in a Padloadfile via:
set exert padloadfile filename,
with the following format:

out1	20
out2	100
...	...

MADS 7–42

Padloading

Including Capacitive Pinloads during Simulation gives the possibility to take physical aspects of the printed circuit board environment into account.

Notes:

Shipping Rules for Simulation

In order to pass "shipdes", the final Design Audit, the following Simulations must have been performed:

– Three loaded Simulations (after physical Design) with min, nom and max delay, including Activity Analysis, Constraint and Contention Checking i.e:

set design load on
(this automatically enables Activity Analysis, so an implicit "set exert change on" is performed)

Min. Delay Simulation: set design delay min, mads
Nom. Delay Simulation: set design delay nom, mads
Max. Delay Simulation: set design delay max, mads

Constaint and Contention Checking are enabled by default

MADS 7–43

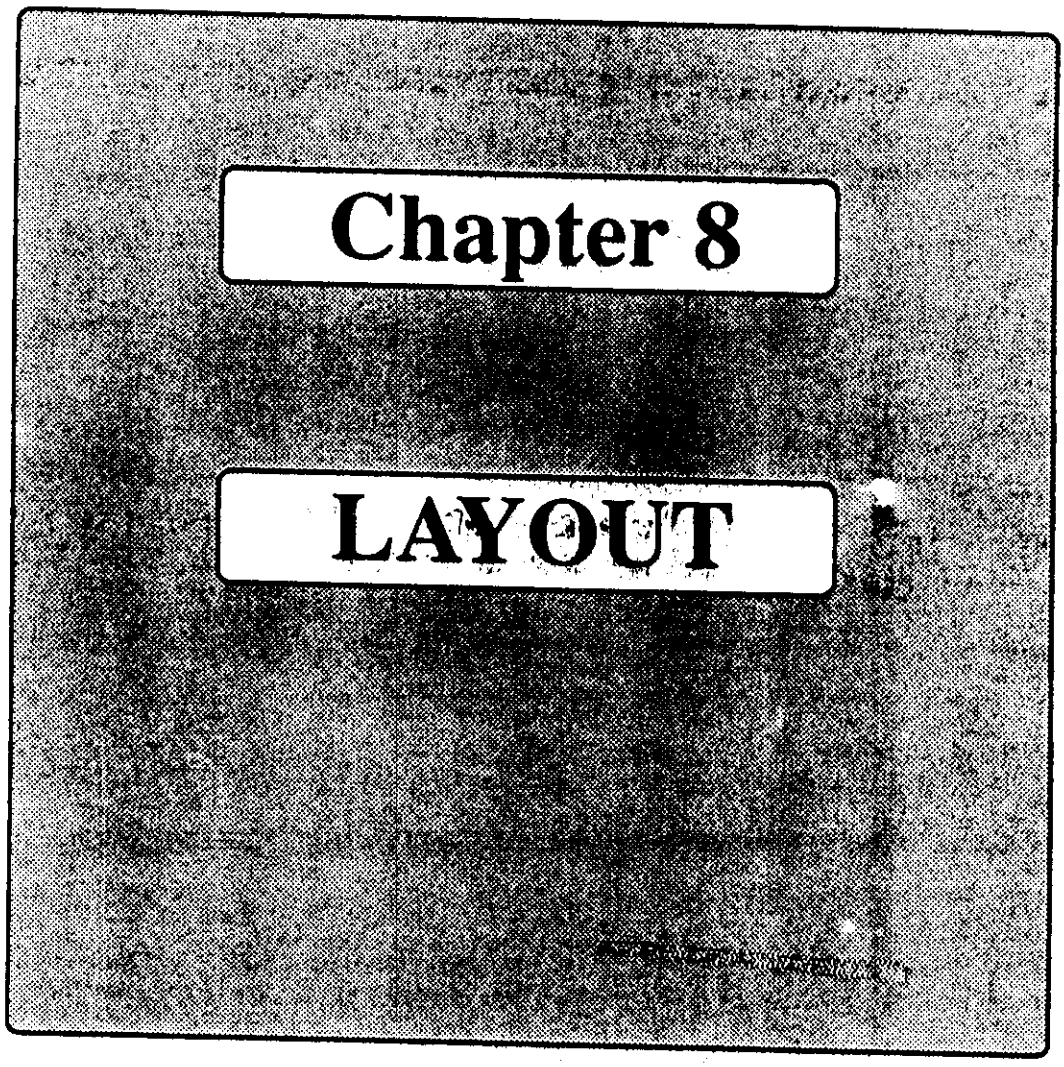
Shipping Rules for Simulation

It's good practice to carry out a provisional Physical Design before fine tuning the design, in order to get data for a loaded Simulation, which can increa : the unloaded delays up to 30%. This will get more and more important with decreasing structures!

Notes: _____

_____

_____


Chapter 8
LAYOUT

LAYOUT (P. G. R. D.)

Contents:

LAYOUT010

ES2

Layout

The four main pieces of software that make up the "layout" suite are PLACE, GATE, ROUTE and DRAW. Together they perform perhaps the most important function of the Solo toolset, that of laying out your design so as to be correct by construction.

Notes

Running place

Used SOLO environment variables with their attributes and possible values: (values marked * set by default)

Variable	Attributes	Possible Values
design	name	name of the design
design	process	ecpd15
design	blocks	list of generated blocks
place	d2	on,off*
place	prompt	on,off*
place	manual	on,off*
place	level	floor*

LAYOUT 8–2

Running Place

The advanced feature "d2 placement", i.e. a twodimensional placement algorithm, may sometimes give slightly improved results in "blockless" designs.

Notes: _____

_____

_____

Running place, continued

Used SOLO environment variables with their attributes and possible values:
(values marked * set by default)

Variable	Attributes	Possible Values
place	auto	off, on*
place	feed	off, on*
place	vary	off, on*
place	position	on, off*
place	modify	on, off
place	level	floor*

Invoking: place

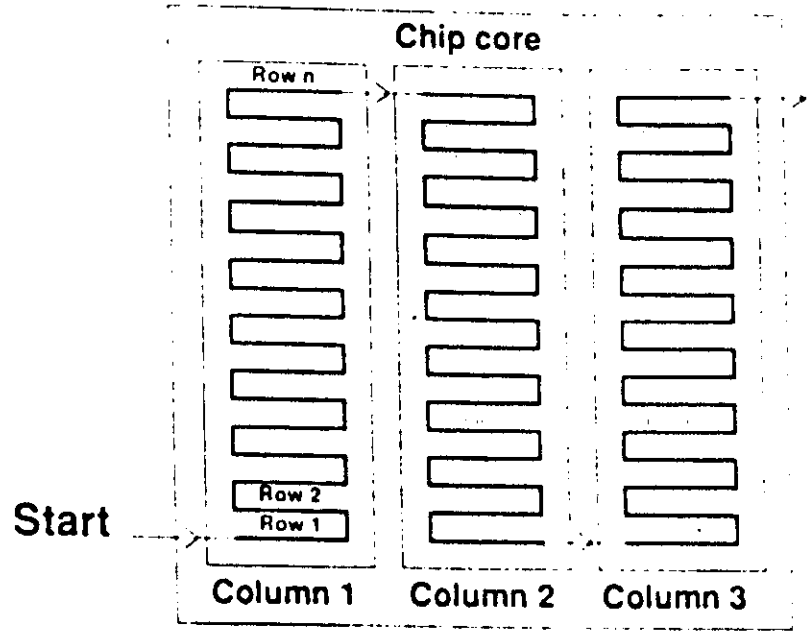
LAYOUT 8–3

Running place, continued

With the level variable set to all, every instance down to baselibrary level is included in the floorplan

Notes:

CHOOSING ROWS



- You may need to increase array size from the default values.

LAYOUT050 ES2

Choosing Rows

Notes

Optimizing Placement

Place terminates always successfully with the default settings!

Tuning may be necessary for the following reasons:
- **to achieve a smaller die, i.e. better usage of silicon**
- **to change the aspect ratio of the chip to fit it in a special package**

To achieve this perform the following steps:
- **set place position on**
- **run place, what creates the additional, so called position file .pos, that includes a chip floorplan of the user defined parts**
- **Edit this file, defining different sizes of rows and columns, and different sequential order of the parts, by simply reordering the lines for the corresponding parts (especially blocks)**
- **Save this so modified file under design_name.rep**
- **set place modify on**
- **Rerun place**

LAYOUT 8–4

Optimizing Placement

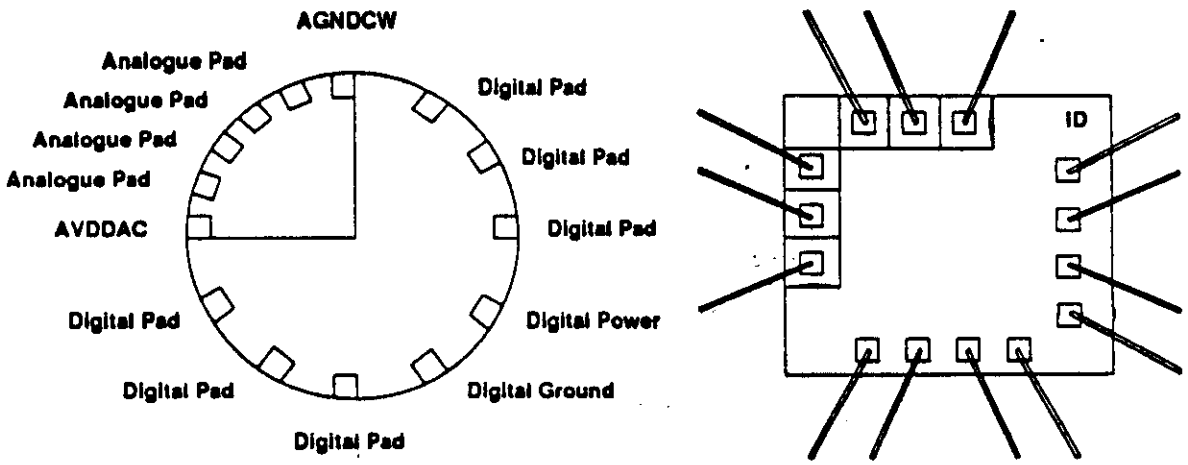
Iterative optimization via the position file is the most effective way for optimizing placement.

Notes: _____

_____

_____

MIXED ANALOGUE AND DIGITAL PADS



LAYOUT080 ES2

Mixed Pads

Notes

ROUTING

- Routing is fully automatic and does not leave any nets unrouted.
- Assigning a CRITICAL condition to a signal in MODEL will give the signal or signals priority when routing.
- Manual routing data can be given for selected nets.

Routing

Usage:

route <filename> -manual <filename 1>

(where <filename 1> is assumed to have the .MRI extension).

For details of manual routing see the manuals.

Notes

Running Route

Used SOLO environment variables with their attributes and possible values:
(values marked * set by default)

Variable	Attributes	Possible Values
design	name	name of the design
design	process	ecpd15
design	max_frequncy	1..99
design	padwidth	5000*
design	usepinout	on,off
design	material	plastic, ceramic*
route	manual	on, off*

LAYOUT 8–5

Running Route

Manual intervention in the routing phase is seldomly worth the spent effort!

Notes: _____

_____

_____

Available Options with Route

Beside routing the chip using metal1 and metal2, the program performs physical Pad Placement with the following options:

- **if "design usepinout" is set to on, a Pad Placement is performed according to the data generated by "pinout", otherwise a default Pad Placement is performed**
- **"design max_frequency" defines the width of the power rails**
- **"design padwidth" defines the width of the "spacer pads" (unit=10nm), available in pinout, to increase the chip size in padlimited designs, to solve bonding problems**
- **"design material" implicitely defines the pad pitch due to different bonding rules**
- **if "route manual" is set to on, a manual routing file .rmf is taken for routing?**

LAYOUT 8-6

Available Options with Route

With this release the bond pitches for plastic and ceramic have become the same, 150u for the new padlib2.

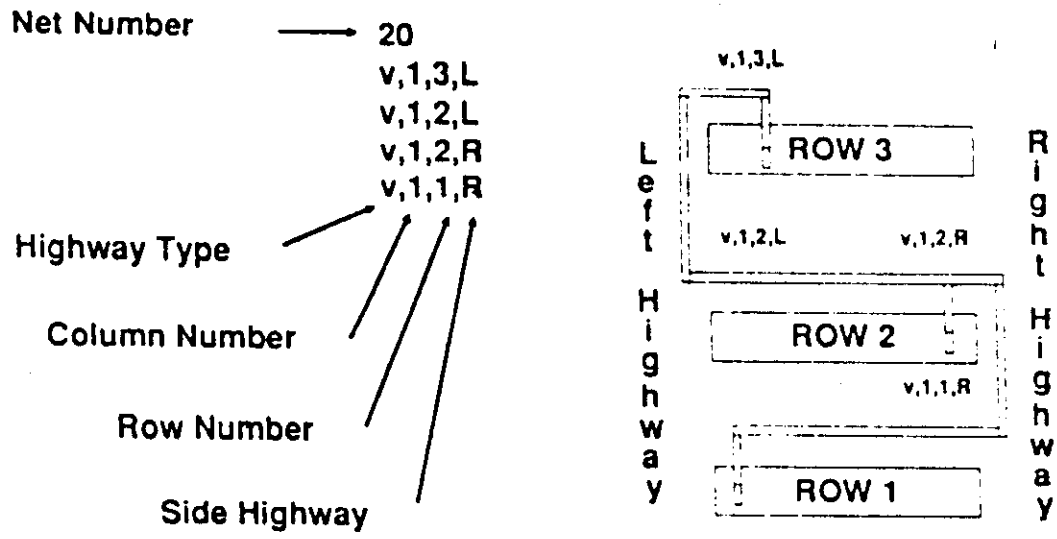
Notes: _____

_____

_____

ROUTING SIGNALS

In the Manual Routing File (.MRF):



LAYOUT110 **ES2**

Routing Signals

Net numbers can be found using EXERT interactively or by EXPAND.

Notes

DRAW

- Produces actual polygon information from PLACE and ROUTE algorithms.
- One signal's CIF data can be output using:

  draw <filename> -signal 10

- This is useful when manually routing and checking signals.

LAYOUT120 *ES2*

DRAW

Notes

ARTVIEW

Contents: Artview Commands
Interrogating the Chip artwork

ARTVIEW010 **ES2**

ARTVIEW

ARTVIEW is not an editor - its purpose is to display the output from DRAW.

Notes

ARTVIEW SUMMARY OF COMMANDS

draw	-	draw entire artview of Chip or package
redraw	-	draw at current zoom setting
analyse	-	bring new .CIF file into ARTVIEW
zoom	-	specify corners, followed by redraw
level	-	set level of abstraction
layer	-	add and remove process layers
help	-	additional commands available [CTRL-Z to exit from HELP]

ES2

ARTVIEW030

Commands

Notes

ARTVIEW

- ZOOM - Zoom in to examine selected areas.
- Measurement can be performed by using GRID or POINT.
- Row and routing symbols can be identified by using LEVEL.
- Colour and shading can be altered using the IDENT command.

ARTVIEW040 ES2

ARTVIEW

Notes

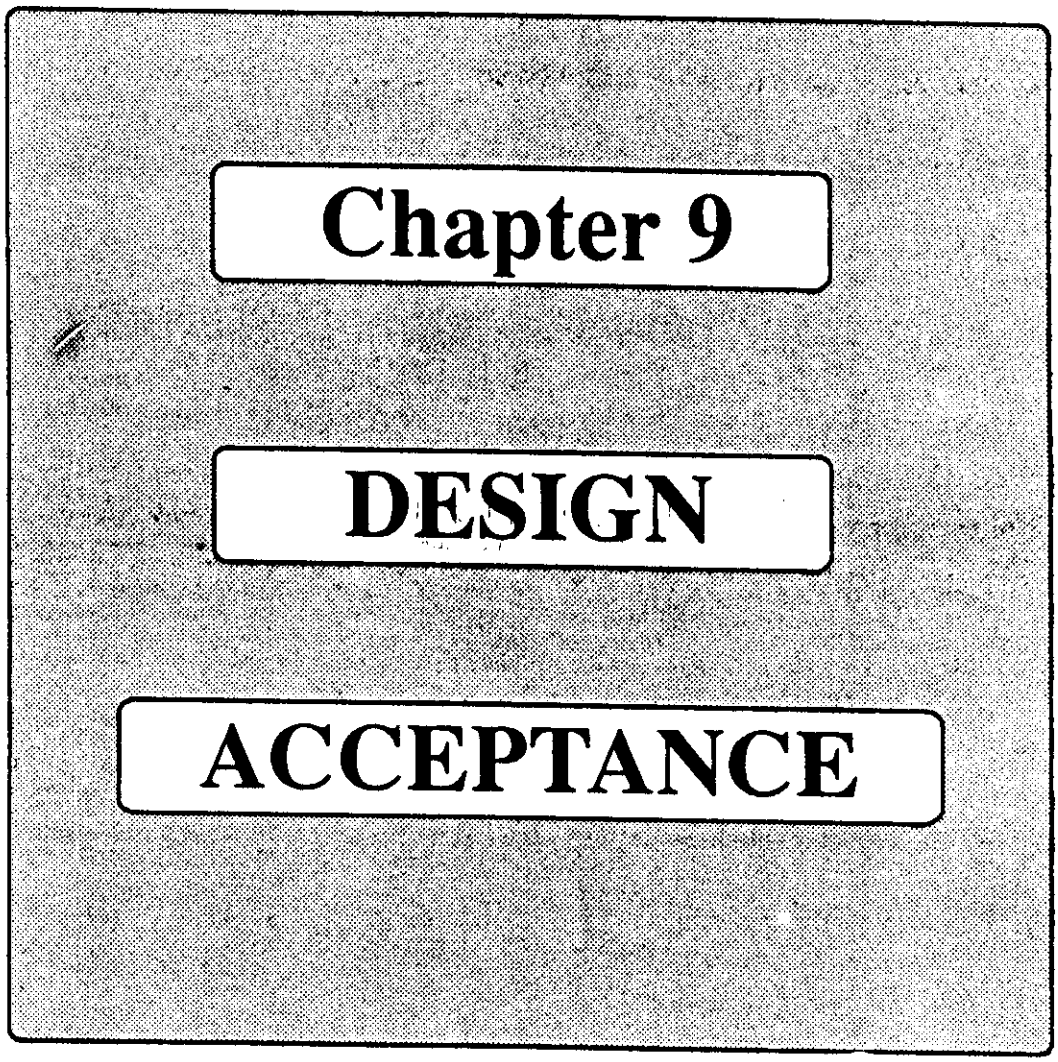
LAYERS

- The layers, styles and definitions are found by the command: layers?
- Layers can be added by "+", followed by the layer name; e.g."layer + cm" adds metal one layer.
- Layers can be subtracted by "-" followed by the layer name; e.g. "layer - cm" removes metal two layers.

ARTVIEW050 **ES2**

Layers

Notes


Chapter 9
DESIGN
ACCEPTANCE

Design Acceptance

audit

padaudit

checkskew

difvec

tis

shipdes

DESIGN ACCEPTANCE 9-2

Design Acceptance

Notes:

Running audit

Used SOLO environment variables with their attributes and possible values:

Variable	Attributes	Possible Values
design	name	name of the design
design	process	ecpd15, ecdm20
design	load	off, on (mandatory for shipdes)

Invoking: audit

Checks for Fanout violations. Creates .fan file for identification of overloaded nets, in a format similar to .exl, created by expand

Running audit

Notes: _____

_____

_____

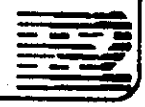
Running padaudit

Used SOLO environment variables with their attributes and possible values:

Variable	Attributes	Possible Values
design	name	the name of the top level part
design	process	ecpd15, ecdm20
design	bond	off, on (mandatory for shipdes)

Invoking: padaudit

Checks for correct pad usage and connectivity between pads and package pins.



Running padaudit

Notes:

Running checkskew

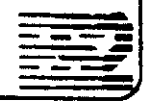
Used SOLO environment variables with their attributes and possible values:

Variable	Attributes	Possible Values
design	name	name of the design
design	process	ecpd15, ecdm20
design	load	off, on (mandatory for shipdes)
design	condition	industrial, military, commercial
design	delay	min, nom, max (mand. f. shipdes)

Invoking: checkskew

Checks if changes of input vectors have the required time distance to active clock edges. For this reason a control file .ckc must be created.

DESIGN ACCEPTANCE 9–5



Running checkskew

Notes:

Format of the checkskew .ckc file

The following two keywords exist and must be both present:

/CLOCK **to define the reference signal**

/SIGNALS **to define the data to be latched with the reference**

Example of a .ckc file:

```
/CLOCK
ck
/SIGNALS
a +     (the +/– indicates the active edges of ck for the lefthand
b –     signal)
c       (if +/– is omitted, checkskew tests for changes on always
        the same edges)
```

DESIGN ACCEPTANCE 9–6

Format of the checkskew .ckc file

Notes: _____

_____

_____

Running diffvec

Used SOLO environment variables with their attributes and possible values:
(mandatory values for shipdes marked *, also set by default)

Variable	Attributes	Possible Values
design	name	name of the design
diffvec	teststep	min:1000, 10000 (for analog)
diffvec	teststep2	min:1000, 10000 (for analog)
diffvec	strobe	90* (required strobe point in %)
diffvec	strobe2	90* (required strobe point in %)
diffvec	load	loaded*
diffvec	load2	loaded*
diffvec	process	ecpd15, ecdm20
diffvec	process2	ecpd15, ecdm20

Running diffvec

Notes: _____

_____

_____

Running diffvec (continued)

Used SOLO environment variables with their attributes and possible values: (mandatory values for shipdes marked *, also set by default)

Variable	Attributes	Possible Values
diffvec	condition	industrial, military, commercial
diffvec	condition2	industrial, military, commercial
diffvec	delay	max, nom, min*
diffvec	delay2	min, nom, max*

Invoking: diffvec
Compares two simulation runs with different attributes for consistency at the strobe points.

Notes:

TESTER INTERFACE SOFTWARE (TIS)

- Run after package

- Vectors to conform to ES2's requirements for static test vectors.

- Input at regular intervals of 1000 ns for digital circuits and 10000 ns for analogue circuits

- Output to be stable at 90% of time stable

DESMN090 *ES2*

Tester Interface Software (TIS)

Notes:

TESTER INTERFACE SOFTWARE (TIS)

Follows a list of checks done :

- Consistency between bonding (.bnd) and trace (.trc) files
- No of outputs switching simultaneously
- Enable control of bidir and tristate buffers
- No of changes per output before strobe time
- Any changes after strobe time
- Any contention on output pins

DESMN110 *ES2*

Tester Interface Software (TIS)

Notes:

Running tis

Used SOLO environment variables with their attributes and possible values: (mandatory values for shipdes marked *, also set by default)

Variable	Attributes	Possible Values
design	name	name of the design
design	teststep	min:1000, 10000 (for analog)
design	strobe	90* (required strobe point in %)
design	load	off, on
design	process	ecpd15, ecdm20
design	condition	industrial, military, commercial
design	delay	min, nom, max

Invoking: tis

Checks that simulation results conform to ES2 Tester requirements.

Running tis

Notes: _____

_____

_____

Running shipdes

Used SOLO environment variables with their attributes and possible values:
(mandatory values for shipdes marked *, also set by default)

Variable	Attributes	Possible Values
design	name	name of the design
design	teststep	min:1000, 10000 (for analog)
design	process	ecpd15, ecdm20
design	condition	industrial, military, commercial
design	material	ceramic, plastic
design	blocks	names of used Blocks

Invoking: shipdes
Checks for correct Design Flow and consitency of used software

DESIGN ACCEPTANCE 9–10

Running shipdes

Notes:

Invoking the Environment

To start the Environment for a specific design perform the following:

Generate a directory with the name of the top level part;
Invoke the Environment with the command solo in that directory. This also creates the additional necessary ES2 directories automatically, if these don't exist already;
The Design Process is then performed completely in that Environment;

Finally you have to set at least the following variables via:

1. set design name <designname>
2. set design process ecpd15

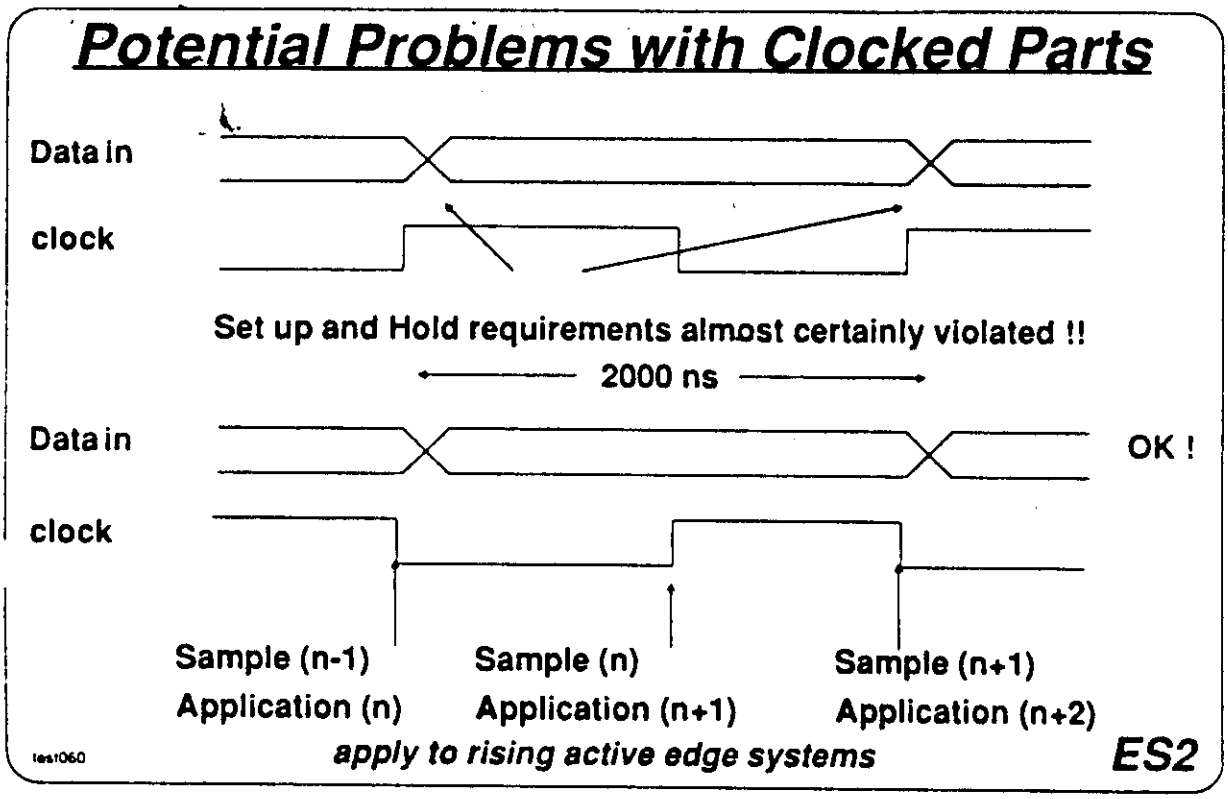
DESIGN ENVIRONMENT 2a–5

Invoking the Environment

Notes: _____

_____

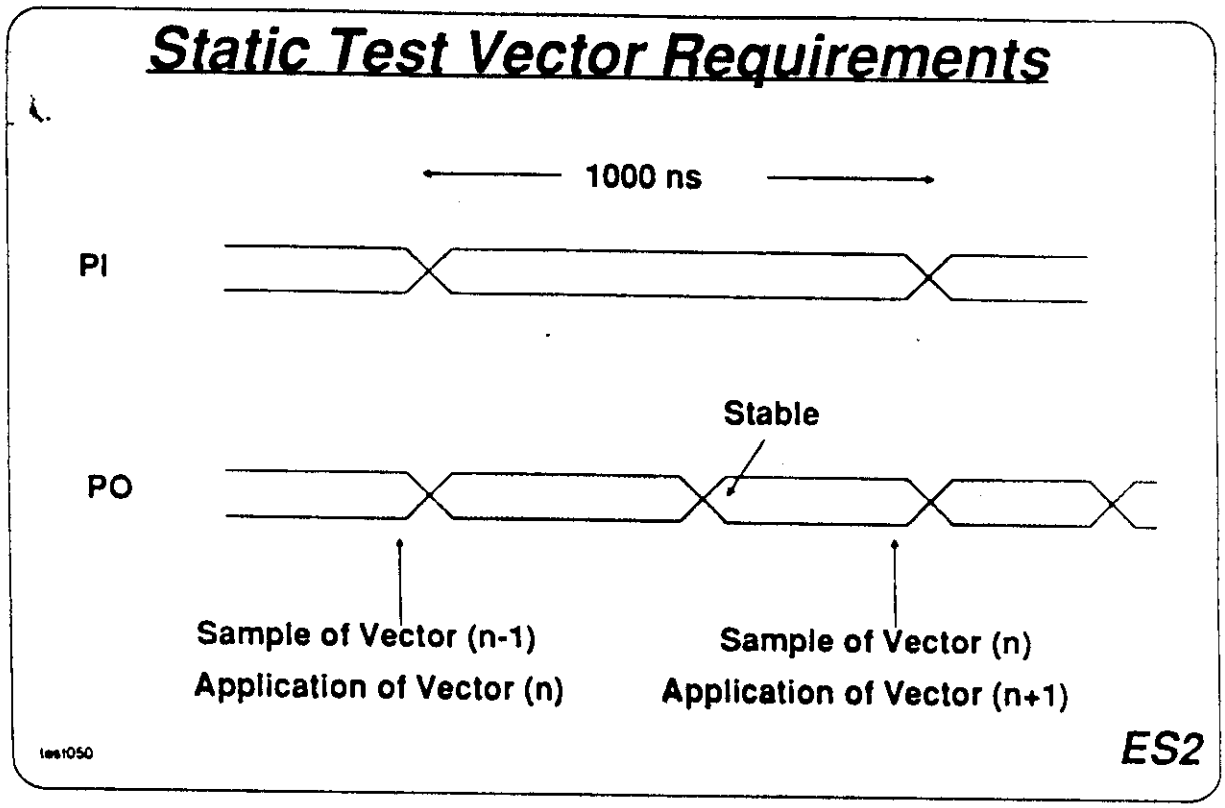
_____



Edge sensitive parts

Since the clock is applied to the ASIC synchroneously with any change of applied data, you must separate active clock edges away from changes in data. It can be seen that data will then change at 500KHz rate. If the system is sensitive to both edges at the primary input, clock must only change during data high or data low reducing data rates to 250KHz. This is only for testing. We will skew all inputs by 100ns both forward and backward to identify test vectors that do not meet this requirement.

Notes:

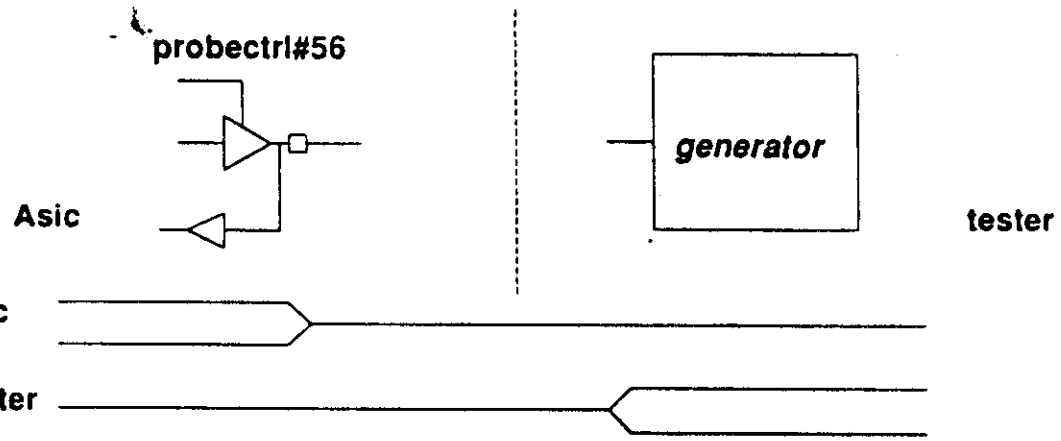


Static Test Patterns

We do not know the settling time of a given Asic design. To streamline the design acceptance procedure, we specify that new test vectors must not be applied at a rate exceeding 1MHz. This provides us with a margin of safety that should cover most Asics that we accept. For analogue designs, the 1000nS required period is increased to 10000nS. Additional screening for speed may be applied to select faster devices.

Notes:

Bidirectional IO ports



Allow bidirectional nets to float for a single tester period (1000ns) on each change of direction. Also: Mark for tracing any bidirectional control line using"probe" facility in draft and template.

tes1080

ES2

Bidirectional Data Ports.

The tester has extremely low source impedance and will change from listening to talking exactly on the 1000nS tick. This can pull the Asic's power supply and so cause corruption of the internal state. Always allow the bus to float for a single tester cycle when the Asic stops talking and the tester starts. Control lines for bidirectional, tristate and analogue buffers are probed automatically by the Solo software.

Notes:

UNIX

UNIX™

Contents: Login and User Environment
Windows
UNIX Commands
The ViEditor
Wild Cards
File Manipulation and Archiving

UNIX is a registered Trade Mark of Bell Laboratories

UNIX010 ES2

UNIX

Notes:

UNIX QUICK REFERENCE

cd	change directory
ls	list directory
more	list text file by page
rm	remove file
mv	move or rename file
cp	copy file
man	manual pages
vi	screen editor

UNIX000 ES2

UNIX Quick Reference

Notes:

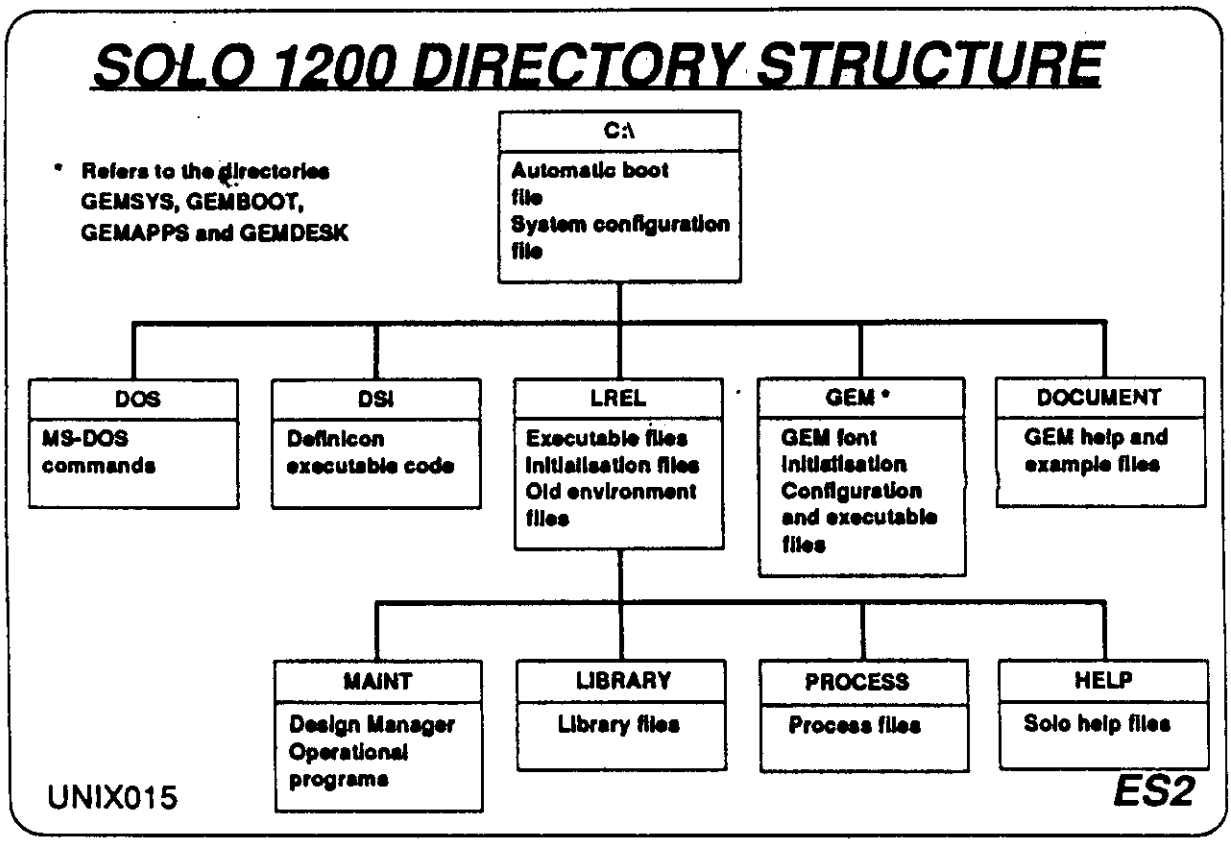
VI QUICK REFERENCE

↑ ↓ ← →	cursor control
x	delete
a	append
i	insert
esc	exit from append or insert
dd	delete line
J	join line
:wq	write to file and quit

UNIX001 ES2

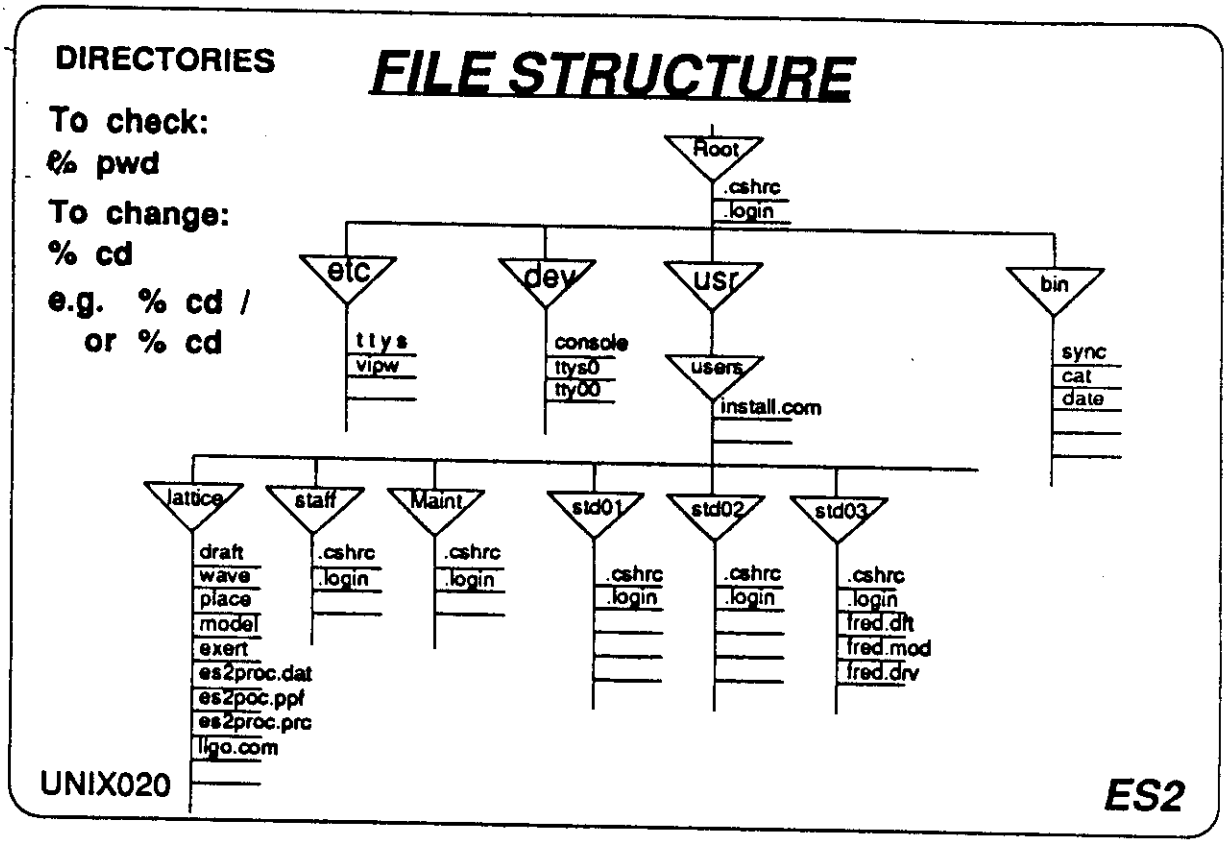
vi Quick Reference

Notes:



Directory Structure

Notes:



File Structure

Notes:

UNIX WINDOWS

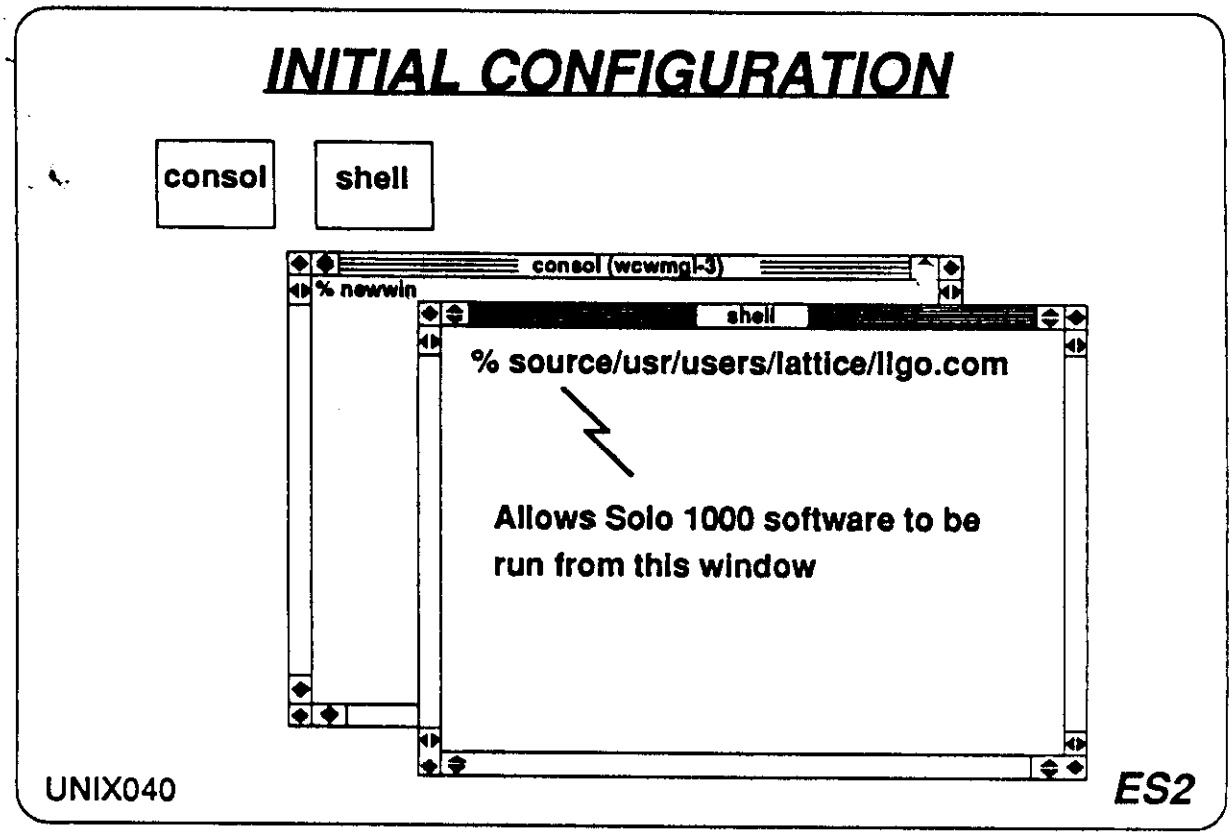
The UNIX windows give:

- **Multiple tasking of jobs.**
- **Better observability of files.**
- **Ease of operation in multiple directories.**
- **Better management of both the computer's and user's time.**

UNIX030 *ES2*

UNIX Windows

Notes:

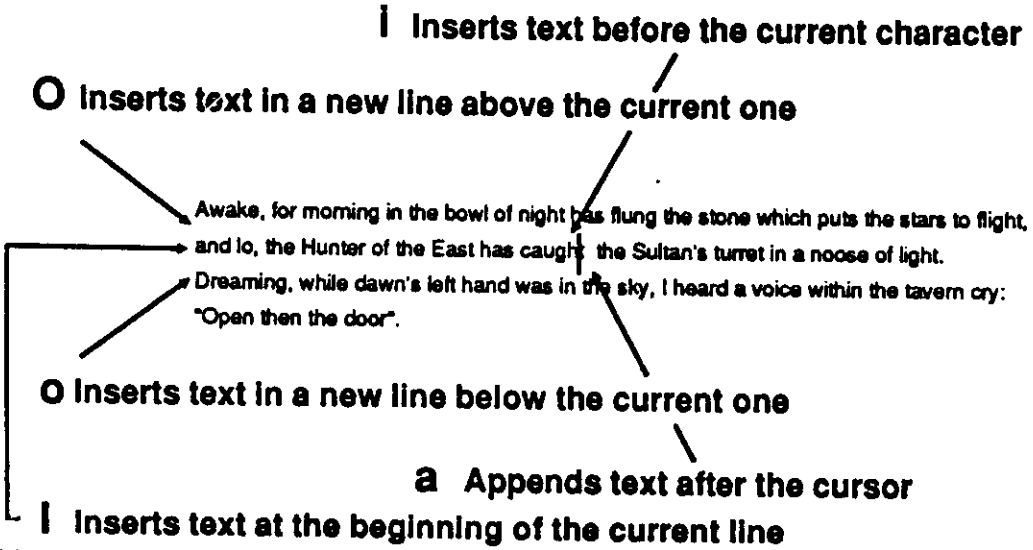


Initial Configuration

Notes:

VI IN TEXT MODE

With the cursor in the position shown, the effects of the following commands will be:



UNIX050 ES2

vi in Text Mode

Notes:

VI QUICK REFERENCE

Either: COMMAND MODE or TEXT MODE

a) STARTING vi

Enter: vi <filename>

b) Use one of the following to switch to text mode:

a to append the text after the cursor
A to append text to the end of the line
i to insert text before the cursor
I to insert text at the beginning of the line
o to open a blank line below the cursor for text entry
O to open a blank line above the cursor for text entry

UNIX060 *ES2*

vi Quick Reference

Notes:

VI QUICK REFERENCE

c) CURSOR COMMANDS

h	left one character	^u	up half a screen
j	down one line	^d	down half a screen
k	up one line	M	middle of screen
l	right one character	^f	forward a screen
		^b	back up a screen

w	to beginning of next word
e	to end of next word
b	back one word

[Note: ^ = Ctrl]

UNIX061 *ES2*

vi Quick Reference

Notes:

VI QUICK REFERENCE

d) FIND COMMANDS

/text	locates text in a file when you type a slash (/) followed by the text to be located, and press RETURN.
/	repeates the previous FIND command when you type the slash (/) and press RETURN.
n	Repeats the previous FIND command in the direction of the initial search.

UNIX062 *ES2*

Notes:

VI QUICK REFERENCE

e) EDITING COMMANDS

x	delete character at cursor
r	replace character at cursor
dd	delete line at cursor
dw	delete word at cursor
u	undo last change
U	undo all changes to last line

UNIX063 *ES2*

vi Quick Reference

Notes:

vi QUICK REFERENCE

f) FILE COMMANDS

:w	save and resume editing
:wq	save and quit
:q	quit if no changes
:q!	quit and discard changes since last save
:e!	discard any changes since last save and continue editing

UNIX064 **ES2**

vi Quick Reference

Notes:

UNIX COMMANDS

ls	list files in a directory
cat	list text file
pr	list text file (formatted)
more	list text file (page-at-a-time)
rm	delete file
cp	copy file
mv	move or rename file
vi	invoke the full screen editor
flar	floppy disk archiving

UNIX070 *ES2*

UNIX Commands

Notes:

UNIX WILD CARDS

These are characters which can be used to find a match for others.

Thus:	**?**	**can match any character**
	*****	**can match any sequence of characters**

For example:

ab?	**includes the set**	**aba, abb, abc, ab1, ab2, ab3, abz, abx**
a??	**"** **"**	**aaa, aab, aba, abc, a01, a02, a99, a9z**
a*	**"** **"**	**aa, a1, abc, a02, abort, afile, adirectory**
?n*	**"** **"**	**0n0, one unix, snooz, 8nty, inthefile**

UNIX080 *ES2*

UNIX Wild Cards

Notes:

UNIX SHORTCUTS

!	repeat the last command.
!mo	repeat the last command beginning with "mo" (e.g. model fred -o).
!32	repeat command 32 as listed in the "history".
!c!d	repeat the last command, but substitute a "d" for a "c" (e.g. mocel fred -o).
!$	can be used to repeat the last parameter of the last command (e.g. "route fred" then "draw!$" instead of "draw fred").

UNIX090 *ES2*

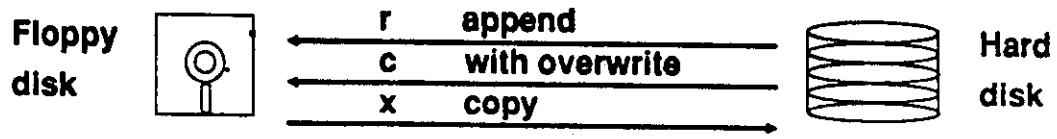
UNIX Shortcuts

Notes:

FLOPPY DISK ARCHIVING WITH FLAR

Command: flar <key> <filename>

(The keys are: x, c, r, t, v)



flar r fred.*	**Add fred.MOD, fred.DFT, fred.IDL to the floppy disk**
flar c fred.*	**Put fred.MOD, fred.DFT, fred.IDL, onto floppy. (Previous contents of the floppy are lost!).**
flar x	**Copy all files from the floppy to hard disk.**
flar xv	**Copy, and list, all of the floppy's files to the hard disk.**
flar t	**List the files on the floppy disk.**

UNIX110 *ES2*

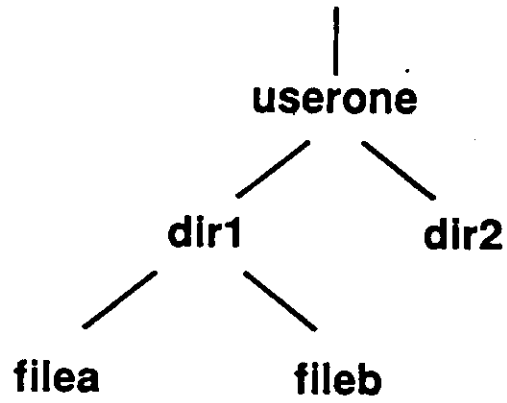
Disk Archiving

Notes:

Notes:

COPYING FILES

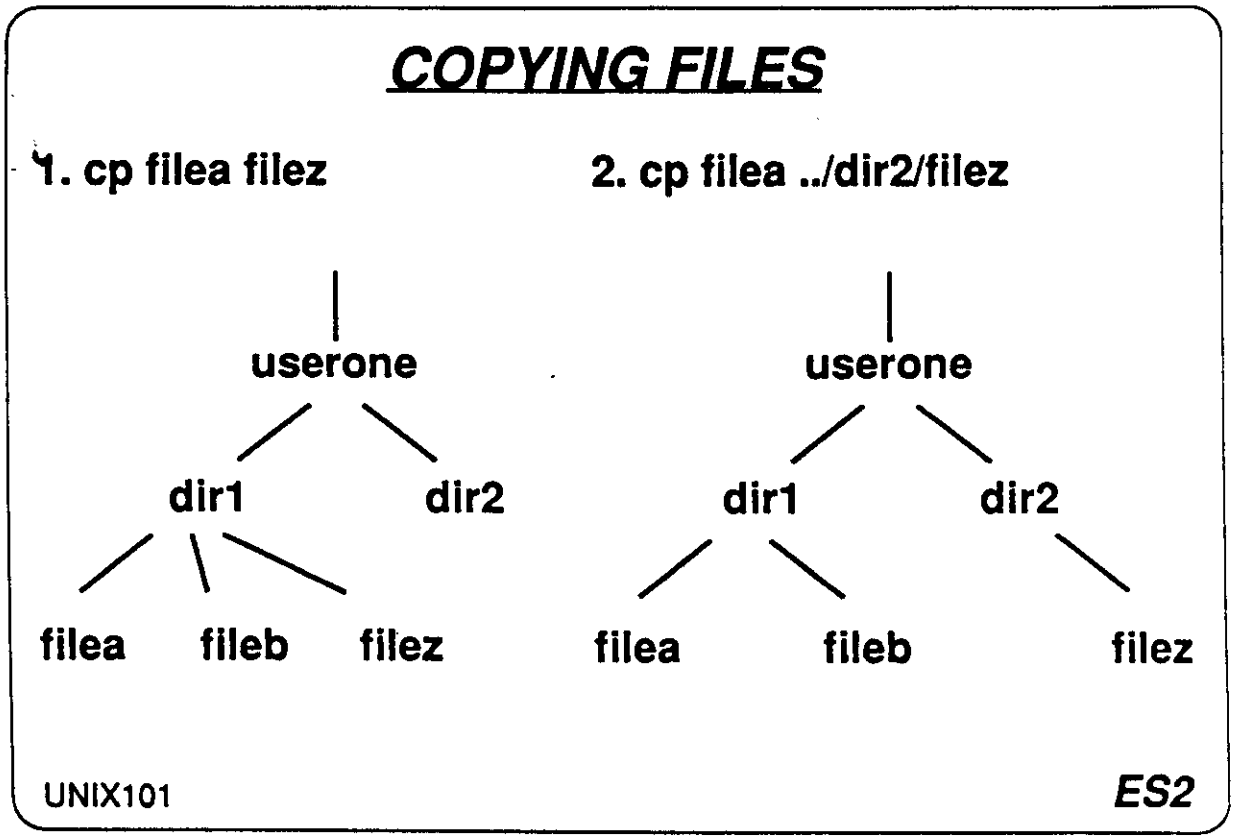
Each use of the COPY command changes the directory structure to that shown under the respective command.



UNIX100 *ES2*

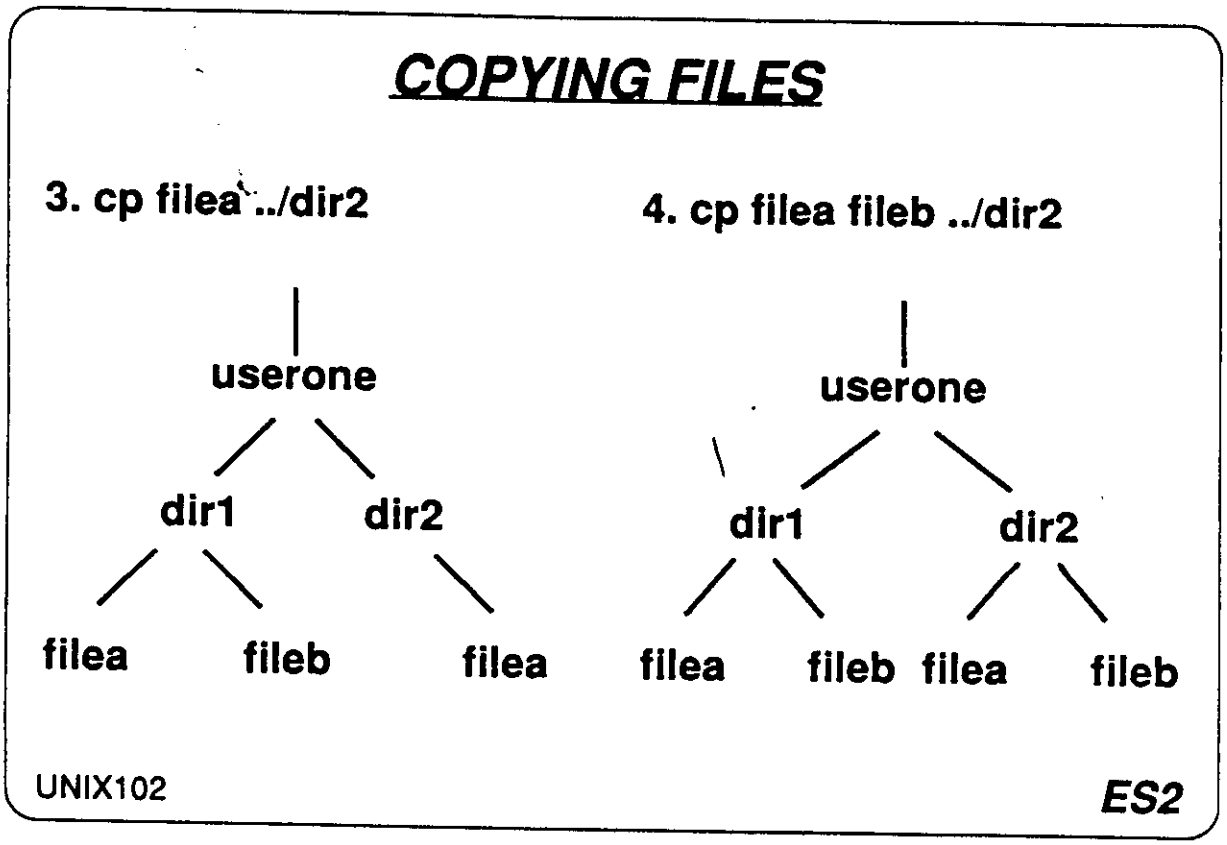
Copying Files

Notes:



Copying Files

Notes:



Copying Files

Notes: