

# Tuning the MPA Design System for Speed

*Prepared by*  
**Douglas M. Shade**  
Motorola Programmable Logic Products



# APPLICATION NOTE

## Tuning the MPA Design System for Speed

by Douglas M. Shade, Motorola Programmable Logic Products

### Introduction

This application note covers methods for maximizing the clock frequency of a given design using the MPA1000 generation of Motorola MPAs. The discussion is limited to those areas specific to the MPA Design System: MPA library components, pin, net and instance attributes, MPA Design System Tool Options and clock files. Generic fast logic design techniques such as look ahead carry and pipelined logic are not covered in this note. A section which covers the progress monitors of the tool is also included.

Front end techniques discussion is limited to schematic entry however the basic concepts of each of the sections of this application note apply to all design entry methods.

### Taking Advantage of the Architecture

A very good investment of your design time is to spend a while reviewing the data book section "MPA1000 Architectural Overview" and the extensive on line documentation included with the MPA Design System. In particular use the pull down menus: "Help > Help on Device", and "Help > Help on Libraries". Read these sections thoroughly.

### Clock Resources

Each member of the MPA1000 family has an identical set of clock distribution resources. Two pads on each edge of the die may be dedicated to driving the 8 clock distribution lines. These primary clock distribution lines roughly bisect the die along the horizontal and vertical. From there, the lines branch to form a load balanced distribution comb covering the entire die. Skew is held to within 1nS for the complete clock network.

Using these clock resources to distribute register clock and latch enables is a conventional requirement of most designs. These resources are mentioned in this application note because they may also be used to distribute internal logic (non-clock or reset) signals with high fanout or otherwise tight skew requirements. Speeding up high fanout signals by putting them on one of the 8 available clock networks is accomplished by appending the output of the source driver with an ACLK or ARST buffer.

Similarly, external non-clock and non-reset signals can be distributed throughout the array on the clock network by using the IPCLK or IPRST input buffers at one of the 8 valid pad locations. Using one of these buffers without specifying the pad location will result in the MPA Design System automatically assigning the buffer to one of these 8 valid pad locations.

The MPA Design System treats ACLK and ARST identically. The same holds IPCLK and IPRST. However, each Zone is limited to two unique primary clock and two unique primary reset signals. Since most designs have more clocks than resets, it may be more prudent to use ARST and IPRST macros for routing speed critical high fan out nets.

### Taking Advantage of the I/O Cell

The standard I/O cell of the MPA array is very feature rich. The complexity of this structure is apparent in the large number of choices available in the I/O macro library IOLIB. (Here again, the reader is encouraged to invest some time in the on-line help facility, in particular "Help on Libraries > Input and Output Pads" and "Help on Device > Functional Description > I/O Cell". Defining a bookmark for these two particularly useful help sections will provide a short cut for referencing them in the future.)

Each of the macros in the IOLIB fit in a single I/O cell. Couple these relatively complex functions with the availability of P-Bus routing resources and it becomes clear that significant functionality can be achieved before ever having to slow down to use the normal internal resources of the array. Another advantage to using the registers in the I/O cells is guaranteed clock to out timing.

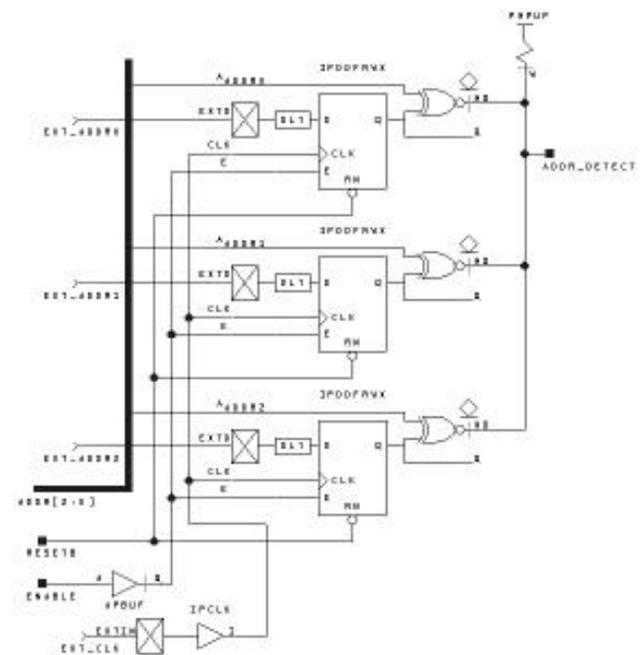


Figure 1. A Three Bit Address Decoder, Only I/O Cell and P-Bus Resources Used

In the example above, a three bit address decoder was implemented using only the resources available in the complex I/O Cells and the associated P-Bus. No internal logic or routing resources were consumed. The features used unique to the I/O Cell and P-Bus include: input delay to synchronize external data with the buffered clock signal, APBUF used to bus a common internal enable to signal to several I/O sites, XNOR used to compare external and internal address values, and finally the Wired-OR P-Bus line. The P-Bus Wired-OR of Figure 1 is shown only for reference.

Wired-OR should be avoided on speed critical nets; explained more fully in the next section.

Of course, all of this functionality could be moved internal to the array, using only the simple I/O macros, but the design will generally incur a slight speed penalty when doing so.

### Wired-OR, Not the best choice

Several elements of the MPA library give you access to the chip's Wired-OR (open drain) structures. While the high to low transition time of such nets is generally acceptable, the low to high 'drive' of such nets is provided only by pull-up resistors. As such, the low to high transition time suffers and so such structures should be avoided on speed critical nets.

The pull-up resistor structure is provided in the WPUP component. The number of parallel pull-up resistors used in the WPUP is under control of the DPLD\_PUP attribute. If it becomes necessary to use such a net on a speed critical path, be sure to use the DPLD\_PUP attribute set to BOTH.

The available macros for the internal Wired-OR functions are: WND2, WINV, WOR2, WBUF. When using these macros, exactly one WPUP is required per Wired-OR net. The maximum number of drivers of a single Wired-OR signal is given by

$$\frac{1}{2} \sqrt{\# \text{ of core cells.}}$$

MPA Family Member	Max Drivers
MPA1016	20
MPA1036	30
MPA1064	40
MPA1100	50

Figure 2. Maximum Drivers on a Wired-OR Signal

If the number of drivers on the Wired-OR net is below half of the maximum allowed, then only a single pull-up resistor is recommended for power conservation. The allowable values for the DPLD\_PUP attribute are 1, 2 or BOTH. Resistors 1 & 2 are of equal value, so it makes no difference which one you select. BOTH ties resistors 1 & 2 in parallel and decreases the low to high transition time, but at the expense of extra power consumption.

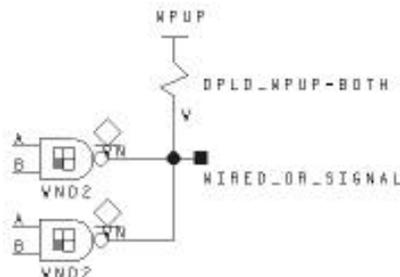


Figure 3. The diamond symbol reminds the user that these outputs can not source a logic high, but are only able to pull the output to logic low.

Adding Wired-OR drivers to the net WIRED\_OR\_SIGNAL will slow it down. Adding inputs to the net has little effect on speed.

The ability to construct Wired-OR nets is not limited to signals internal to the array. Peripheral Bus (P-Bus) Wired-OR nets can be constructed using APWBUF (or APWINV) and their associated pull-up structure PWPUP. However, using these features requires the user to be aware of the natural consequences of adding capacitance and pull-ups to a resistive bus. Adding additional segments of P-Bus (by assigning I/Os to different edges of the die) increases capacitance that effects fall and especially rise times. Also, the somewhat resistive nature of the P-Bus can cause Vol noise margin problems if the active low P-Bus driver is far from the pull-up element and driven element.

The assignment of P-Bus pull-up resistors is automatic. On import, all PWPUP instances defined by the designer are removed from the netlist. The tool automatically balances the number of pull-up resistors against the P-Bus loading (incurred by the use of multiple die edges) by automatically inserting additional pull-up capacity for each P-Bus segment used. During autolayout, the MPA Design System re-attaches a single peripheral bus pull-up resistor per occupied die edge, for each unique P-Bus signal. For example, if several I/Os that use a P-Bus Wired-OR signal get split between the 'top' and 'right' edges of the die during autoplacement, the tool would assign two pull-up resistors to the net.

The Wired-OR resources are provided to help simplify some logic designs, however, their use should be avoided on speed critical paths.

### Guiding Layout with Attributes

The MPA Design System's import process can accept a set of attributes to help the front end designer tune the layout and routing processes. The system also accepts I/O attributes to specify CMOS/TTL compatibility, I/O drive, package pin assignment and slew rate control. Declaring attributes in the schematic will result in their being passed into the EDIF netlist and then imported into the MPA Design System. Optionally the designer may prefer to include attributes in an external .PAT file of the same root file name as the EDIF netlist. The designer may otherwise choose to use the combination of the two methods, but it should be noted that attributes passed into the MPA Design System in .PAT files will always take precedence if declared in both places.

Sch. Comp'nt	Attached Place and Route Attribute	Attached I/O Attribute
Net	DPLD_IGNORE_TIMING DPLD_CLUSTER_SEED DPLD_PLACE_PRIORITY	
Symbol (instance)	DPLD_IGNORE_TIMING DPLD_PAD_PLACE (I/Os only)	DPLD_PUP PULLUP or PULLDOWN DPLD_OPDRIVE DPLD_OPLEVEL DPLD_OPSLEW DPLD_IPLEVEL DPLD_PAD_PROPERTIES
Formal Port	DPLD_IGNORE_TIMING	

Figure 4. All Valid Attributes. Place and Route Attributes can be used to affect a design speed up.

## Place and Route Attributes

Place and Route Attributes can be used to affect a design speed up by providing guidance to the autolayout tool about which are the unimportant nets, and which nets should be clustered and placed tightly together. Absolute placement of I/Os and relative placement of instances are also used as autolayout guides.

### DPLD\_IGNORE\_TIMING

The DPLD\_IGNORE\_TIMING attribute is used to inform the tool which nets to ignore timing on. It may be set on an symbol (instance), a net or an external pin (formal port). If a net has the attribute set, then all delay paths associated with that net are ignored. If an instance has the attribute set, then all input delay paths driving and all delay paths being driven from that instance are ignored. Assigning the attribute to a formal port has exactly the same effects as assigning to the I/O instance itself. Once all the objects to be ignored have been identified, their paths are propagated forwards and backwards through combinatorial gates until clocked objects (or top level circuit I/O) are reached. The result is that additional segments other than those explicitly specified may be ignored for timing purposes as well.

You are required to use a dummy value with this attribute, but the value stated is otherwise ignored.

Assigning this attribute to a symbol, net or formal port frees the timing driven autolayout algorithms to more optimally cluster, place and route the speed critical nets.

### DPLD\_CLUSTER\_SEED

Once a netlist import is complete, the first step of autolayout is clustering. During clustering the tool attempts to group related chunks of logic together. This helps simplify the place and route problem by reducing the total number of 'things' the place and route algorithm has to deal with.

The DPLD\_CLUSTER\_SEED attribute is used to assign a cluster seed to a net. This will cause the clustering to treat all instances that connect to that net specially. The action taken depends on the value of the attribute, as follows:

- 0 ignore this net during clustering. Setting this attribute on a net is likely to cause the net to be implemented in global interconnect.
- 1 default operation
- <1000 weight this net by the given factor in the clustering

Assigning a high value cluster seeds on your most speed critical nets results in a tighter clustering and consequently shorter delays for these nets.

### DPLD\_PLACE\_PRIORITY

The DPLD\_PLACE\_PRIORITY attribute can be applied to a net to guide the software to lay out that net in a physically smaller area – in other words, to physically place the instances connected to that net closer together. The value of DPLD\_PLACE\_PRIORITY should be an integer in the range 1 to 10 (1 is the default). Higher values of place priority let you prioritize nets relative to each other.

### DPLD\_PAD\_PLACE

DPLD\_PAD\_PLACE – instructs the I/O pad to be allocated to the package pin number specified. Only one pad is allocated to any pin. Automatic placement of I/O pads usually results in a better layout, so this attribute should only be added when it is necessary to back fit an existing PCB layout. Example: DPLD\_PAD\_PLACE=C2

## Assigning Attributes in a Schematic

In PROSeries, the method of assigning attributes is straight forward. With a left mouse click, select the desired net or symbol (instance). A net's color will change or a bounding box will appear around the instance, identifying it as the currently selected object. Then from the "Add" pull down menu, select "Object Attribute", the bottom of the screen (the text input area) will then prompt you with "Attribute Text String". Type in the attribute and the value (if appropriate) and hit enter. The attributes will then be included in the EDIF netlist once EDIFNETO is run.

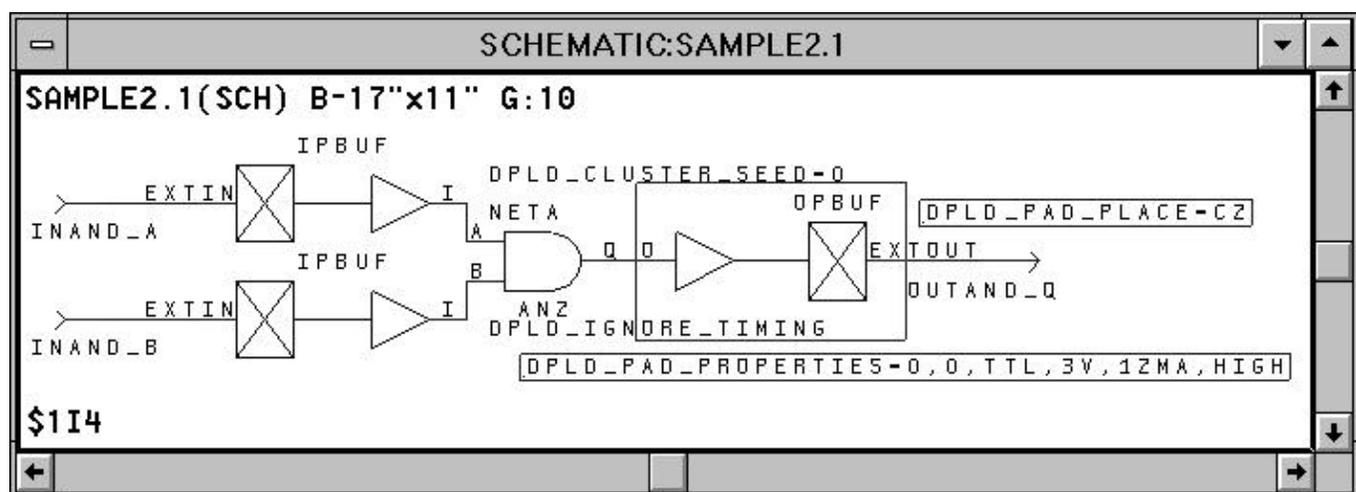


Figure 5. The net "NETA" attributed with DPLD\_CLUSTER\_SEED=0, The AN2 is attributed with DPLD\_IGNORE\_TIMING. The selected component "OPBUF \$1I4" and its attached attributes are boxed.

## Assigning Attributes & Instances in an External .PAT File

Assigning attributes to a long series of instances, or a variety of nets in the above manner can be time consuming and may be error prone. The MPA Design System gives the designer the option to enter all the valid attributes in an external .PAT file. Entries in the .PAT file take precedence over any attributes that may have also been instantiated in the EDIF netlist via schematic entry.

The external attributes file supports four main operations:

- 1) Insertion of attributes to specify pin placements and pad characteristics.
- 2) Insertion of special pad cells, IPCLK/IPRST, to drive the primary clock/reset network.
- 3) Insertion of special buffer primitives into a named net.

This has two uses:

- a) Force a named net onto/off the peripheral bus, by inserting the primitives APBUF/PABUF respectively.
  - b) Force a named net onto the primary clock/reset network, by inserting the primitives ACLK/ARST respectively.
- 4) Attaching place and route attributes to existing nets, instances and formal ports.

The external attributes file must exist in the same directory and with the same name as the EDIF netlist, with the file extension .PAT During import, the MPA Design System automatically checks for the existence of a .PAT file and uses it when one is found.

### Syntax of the External Attributes (.PAT) File

The external attributes file contains a list of commands, one per line. Each command contains up to five fields, as follows:

**<object-class> <object-name> <operation> <name> [<value>]**

where:

<b>&lt;object-class&gt;</b>	is one of port, net, or instance.
<b>&lt;object-name&gt;</b>	is the netlist name of the object (port, net or instance) being operated on.
<b>&lt;operation&gt;</b>	is one of attribute or instance.
<b>&lt;name&gt;</b>	is the name of a definition or an attribute.
<b>&lt;value&gt;</b>	is only used in attribute operations, and is the value to be given to the attribute. This field is only required when an attribute requires a value.

The following are specific syntax forms of all valid attribute or instance assignments.

**port <name> instance**

(name here can only refer to input instances with one input pin and one output pin)

**ipclk**

**iprst**

**port <name> attribute**

**dpld\_ignore\_timing dummy\_arg**

**dpld\_pad\_place <value, see data book for package being used>**

**pullup 1|0**

**pulldown 1|0**

**dpld\_opdrive 6ma|12ma**

**dpld\_oplevel 3v|5v**

**dpld\_opslew high|low**

**dpld\_iplevel CMOS|TTL**

**dpld\_pad\_properties 0|1, 0|1, CMOS|TTL, 3v|5v, 6ma|12ma, high|low**

**net <name> attribute**

**dpld\_cluster\_seed n** (where, 1 is default, 0 means ignore net)

**dpld\_place\_priority n** (where, 1 is default, 10 is highest priority)

**net <name> instance**

**aclk**

**apbuf**

**arst**

**pabuf**

**instance <name> attribute**

**dpld\_ignore\_timing dummy\_arg**

**dpld\_pad\_place <value, see data book for package being used>**

**pullup 1|0**

**pulldown 1|0**

**dpld\_opdrive 6ma|12ma**

**dpld\_oplevel 3v|5v**

**dpld\_opslew high|low**

**dpld\_iplevel CMOS|TTL**

**dpld\_pad\_properties 0|1, 0|1, CMOS|TTL, 3v|5v, 6ma|12ma, high|low**

The nomenclature of port, net and instance and their use can be a little confusing, I'll attempt to clarify a bit. A 'net' is simply the name of the net of interest. In Figure 6 the valid net names are SEG[0:8]. An 'instance' is the unique designator for the instantiated library macro. In Figure 6, \$1146 is the instance of the OR gate. A 'port' refers to only formal ports (external I/O pins).

Another point of confusion is that the valid attribute sets for "port <name> attribute" and "instance <name> attribute" are identical and each guides the MPA Design System to respond in an identical fashion. You may use either syntax, but for the sake of simplicity, stay consistent in your methodology.

### Example .PAT File Entries

The following sample .PAT file entries reference the simple schematic shown in Figure 6. The entries of Figure 7 should be considered one at a time; considering them all to be part of the same .PAT file, all operating on this simple circuit simultaneously is not the intended interpretation.

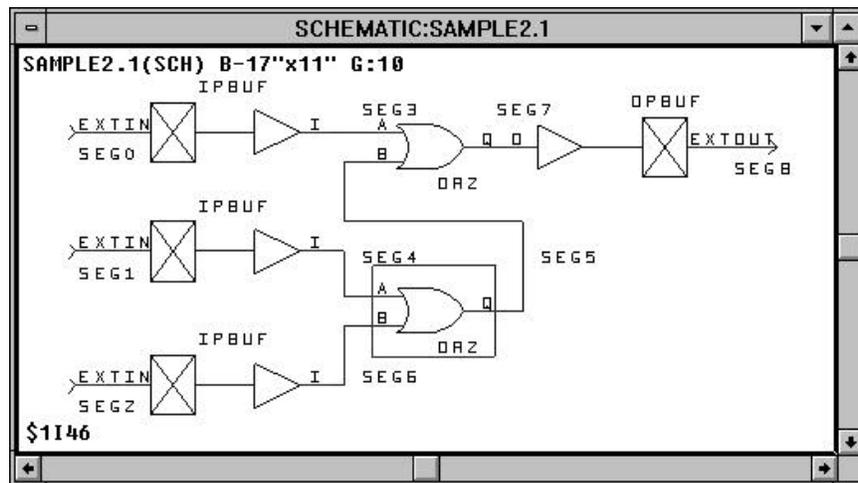


Figure 6. A sample schematic to add .PAT file attributes to. Nets are named SEG[0:8].

```
// This is a comment
# This is a comment as well
port seg0 instance ipclk
//This results in the top IPBUF being replaced by an IPCLK, forcing the
//net SEG3 onto a clock network.
port seg0 attribute dpld_pad_place 22
//This results in the top IPBUF being placed on the pad associated with
//package pin 22.
port seg2 attribute dpld_ignore_timing dummy_arg
//For place and route purposes, the design's timing parameters are ignored
//for the entire delay path driven from the formal port SEG2. A dummy
//argument is required for this attribute.
net seg5 attribute dpld_ignore_timing dummy_arg
//For place and route purposes, the design's timing parameters are ignored
//for the entire delay path associated with the net SEG5. For the design
//of Figure 6, this statement has the same effect as the previous one.
//A dummy argument is required for this attribute.
net seg5 attribute dpld_cluster_seed 500
//During clustering, the MPA Design System will strongly associate
//the top and bottom OR2s. The resulting layout will likely have
//these two instances in the same cluster.
net seg5 attribute dpld_place_priority 10
//With the net SEG5 attributed with a high place priority, the autolayout
//tool will likely place the top and bottom OR2s physically adjacent to one
//another.
net seg5 instance aclk
//This results in an ACLK buffer being inserted after the bottom OR2's output
//and the top OR2's B input. The B input of the top OR2 will now be driven
//off of a primary clock routing resource.
instance $1I23 attribute dpld_opdrive 12ma
//This results in the OPBUF getting 12ma drive capability.
instance $1I46 attribute dpld_ignore_timing dummy_arg
//For place and route purposes, the design's timing parameters are ignored
//for the all nets associated with the instance $1I46. A dummy
//argument is required for this attribute.
```

Figure 7. Sample .PAT file entries showing the major syntax variations allowed. Consider each entry individually, this entire figure is not applicable as a .PAT file for the referenced design.

## Tool Options – Autolayout

Referring to Figure 8, clicking on the Tool Options button brings you to the Tool Options window of Figure 9.

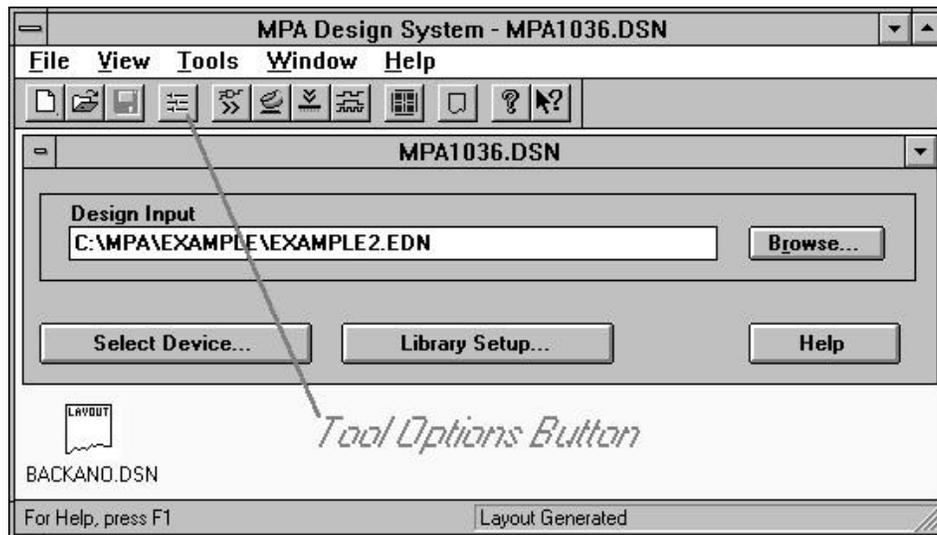


Figure 8. A sample tool context window. In this example, two design (.DSN) files are available.

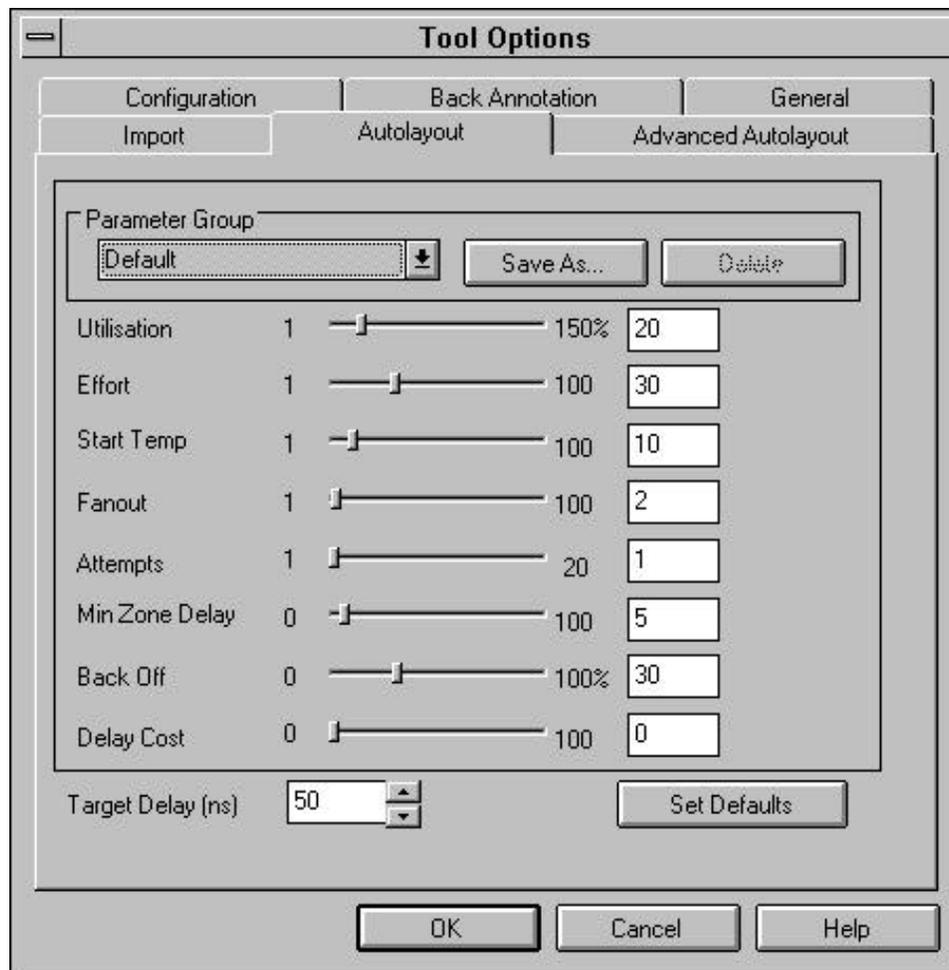


Figure 9. Almost all autolayout parameters are adjusted from this window. 'Seed' is available under the Advanced Autolayout tab.

The 'Parameter Group' roll down menu has a scrollable list of pre-defined tool settings to choose from: Default, High Utilisation, Minimum Delay, Try Harder, and Ignore Timing. Each of these parameter groups was established as a result of studying many sample designs, each with varying design styles and densities. These provided Parameter Groups are generally very good starting points for guiding the MPA Design System through optimization of your designs. If after some experience, you find a new unique set of parameter settings that works better for you, you may save your own custom parameter groups. (The changes are written to a file called PMEL.INI in the Windows installation directory.)

### Target Delay

Minimum 1                      Maximum 9999                      Default 50 (5nS)

Target delay is the most significant guiding parameter of the autolayout process. The 'delay' of a combinatorial network is calculated as the longest path from input to output. For synchronous circuits, path delays are calculated as I/O to clocked element (register), register to I/O and from register to register.

Autolayout attempts to keep the delay of each of these paths types below the target delay value. For a single clock design, you set the target delay to the reciprocal of the desired operating frequency.

The units are 10<sup>-10</sup>s. For example, 80 yields an 8ns delay target. (A screen shot of Version 2.2.3 of the MPA Design System is shown in Figure 9. The panel shows the units as "ns", this is incorrect and will be corrected on version 2.2.4 and later.)

### Utilisation

Minimum 1                      Maximum 150                      Default 80

The percentage of the recommended maximum number of core cells partitioning will attempt to use in a zone. Partitioning will exceed this number if necessary to complete.

For the MPA1000 family, the recommended number of cells per zone to use is around 50. Setting utilisation to 100% tells the partitioning tool to consume approximately 50 of the 100 possible cell sites per zone. If the utilisation parameter is set too high, then the autolayout tool may spend a lot of effort on a few highly utilized zones, while other zones are left empty or underutilized. If the utilisation parameter is too low, then the autolayout tool will adjust the parameter upward until it is compatible with the design. Increasing the value of the utilisation parameter may increase the operating speed of the circuit, only at the cost of increased tool run time. Values greater than 100% are not recommended.

### Effort

Minimum 1                      Maximum 100                      Default 30

The relative amount of work applied to the partitioning phase. When setting utilisation high, effort should be set high as well.

The two major phases of autolayout that typically consume the most time are partitioning and zone routing. The time taken in partitioning is directly related to effort.

### Start Temp

Minimum 1                      Maximum 100                      Default 10

Partitioning and zone routing are performed using a simulated annealing algorithm. Setting the start temperature higher gives the partitioning and zone routing tools the freedom to make more aggressive moves in the search for the optimal solution. With the start temp set low, the respective algorithms may only be free to find local minimum solutions whereas the best overall solution lies over some other cost hump that the tool was otherwise constrained from traversing. It is usually best to increase start temp in a highly utilized device. If increasing start temp, be sure to increase effort as well.

### Fanout

Minimum 1                      Maximum 100                      Default 2

This is the maximum fanout of a net which is still included in the clustering. All nets which have a fanout greater than the number specified are ignored in the clustering phase. Any nets ignored during clustering are more likely to be routed on global resources (typically a bit slower than medium or zonal routing resources).

### Attempts

Minimum 1                      Maximum 20                      Default 1

The number of runs through partitioning phase. More runs typically yield a better solution, only at the expense of extending tool run time.

### Min Zone Delay

Minimum 1                      Maximum 9999                      Default 5 (0.5nS)

The units are 10<sup>-10</sup>s. The zone router will attempt to keep the delay of all net segments within the zone being routed to less than the value specified.

### Back Off

Minimum 0%                      Maximum 100%                      Default 30%

Percentage of relaxation to apply if the target delay was unobtainable. Expressed as a percentage of the previously described Target Delay.

### Delay Cost

Minimum 1                      Maximum 100                      Default 5

This is the weighting given to timing during partitioning. If a particular partition includes a net that is failing to meet its timing target, then the cost of that partition will be artificially raised by an amount proportional to the delay cost. Increasing the delay cost is likely to trade off against achievable utilization. Setting delay cost to zero will result in much reduced tool run times and increased achievable utilization, however this is accomplished at the expense of lowering the design's maximum frequency.

### Seed

A 'seed' value may be set in the Advanced Autolayout panel of the Tool Options window. The seed is as a starting point for the autolayout's pseudo-random number generator. Random numbers are used in several of the autolayout algorithms. Because it is not a true random number generator, two layout runs with identical settings will yield identical results. Changing nothing but the seed value may yield significantly

different solutions. The PC and Workstation versions of the MPA Design System have different pseudo-random generators and so solutions from each will always differ in spite of identical initial seed values. It is mentioned here only for completeness. Changing the seed value does not guarantee a faster place and route solution, only a different one.

## Clock Files

In a simple single clock design, it is sufficient to declare a Target Delay value in the autolayout panel of the tool options window. The autolayout attempts to achieve a place and route solution in which all I/O to clocked element path delays and all clocked element to clocked element path delays are shorter than the target delay.

However, more complex designs may have multiple clocks, each running at unrelated frequencies. In such instances it may be unwise to ask the autolayout tool to constrain every path of the design to the delay target of the fastest path required.

Clock files provide a method of grouping related components of your design into unique timing groups. The timing groups may then each be assigned unique clock specifications, only as restrictive as required. This assists the autolayout tool by guiding it to complete the more speed

critical nets using the more valuable placement, routing and switching resources as required.

Clock files also provide a method of specifying target delays between such timing groups. The syntax presented below can be a bit daunting at first. Please read through to the examples, and it should become a bit clearer on just how to use a clock file.

### Clock File Syntax

The following meta-syntax conventions are used in this definition:

<code>::=</code>	introduces a rule
<code>;</code>	terminates a rule
<code>{...}+</code>	Indicate one or more occurrences of the phrase inside the braces
<code>{...}*</code>	Indicates zero or more occurrences of the phrase inside the braces
<code>[...]</code>	Indicates an optional single occurrence of the phrase inside the brackets
<code> </code>	Separates alternative choices

Timing groups identify those portions of the design driven by a particular clock with a particular specification. The syntax for Timing Group Definitions is given as:

```
timingGroupDefinitions ::= '(' 'timingGroupDefinitions' {timingGroup}+ ')';
where
timingGroup ::= '(' 'timingGroup' timingGroupDef [clock]
               timingGroupInstanceList ')';
where
timingGroupDef ::= timingGroupName; any string, ie my_timing_group
clock ::= '(' 'clock' period [phase] ')';
where
period ::= time_in_units_10-10_seconds (50 = 5nS), ie '50'
phase ::= time_in_units_10-10_seconds
timingGroupInstanceList ::= {allIO|allFlipFlops|netRef|instanceRef}+;
where
allIO ::= '(' 'allIO' ')';
allFlipFlops ::= '(' 'allFlipFlops' [clockInputSense] ')';
where
clockInputSense ::= 'INVERTED'|'NONINVERTED'
netRef ::= '(' 'netRef' netName [clockInputSense] ')';
where
netName ::= any string, ie my_net_name
clockInputSense ::= 'INVERTED'|'NONINVERTED'
instanceRef ::= '(' 'instanceRef' instanceName ')';
where
instanceName ::= '"any_instance"', ie "$I19"
```

Intended Target Delay is the method used to limit the delay between the previously defined timing groups. The syntax for Intended Target Delay is given as:

```
intendedTargetDelay ::= '(' 'intendedTargetDelay' targetDelay driverGroupRef
                        drivenGroupRef ')';
where
targetDelay ::= time_in_units_10-10_seconds (50 = 5nS), ie '50'
driverGroupRef ::= timingGroupName; any string, ie my_timing_group
drivenGroupRef ::= timingGroupName; any string, ie my_timing_group
```

**Clock File Examples**

The syntax specification for clock files is harder to read than it is to type, and it wasn't easy to type. So in an effort to clarify the topic some, let's look at some clock file examples.

```
( timingGroupDefinitions
  ( timingGroup My_Flip_Flops
    ( clock 50 )
    ( allFlipFlops )
  )
  ( timingGroup My_IO
    ( allIO )
  )
)
( intendedTargetDelay 10000000 My_Flip_Flops My_IO )
( intendedTargetDelay 10000000 My_IO My_Flip_Flops )
```

In the above example all of the flip-flops are intended to be clocked with a 5nS period clock. I'm further specifying that I don't care how long it takes to get signals in and out of the design by defining a very large intendedTargetDelay between groups and not specifying a target clock period set for "allIO",

```
( timingGroupDefinitions
  ( timingGroup My_CLK1_Group
    ( clock 200 )
    ( netRef CLK1 )
  )
  ( timingGroup My_CLK2_Group
    ( clock 100 )
    ( netRef CLK2 )
  )
)
```

In the above example, two clock groups were specified: 'My\_CLK1\_Group' and 'My\_CLK2\_Group'. All instances driven with the clock named CLK1 will be placed and routed to meet a 20nS delay criteria, and all instances driven with the clock named CLK2 will accommodate a 10nS clock.

```
( timingGroupDefinitions
  ( timingGroup My_Group
    ( clock 200 )
    ( instanceRef "$1I*" )
  )
)
```

In the above example, the wild card \* is used to define the instances that are members of the timing group 'My\_Group'.

```
( timingGroupDefinitions
  ( timingGroup My_CLK1_Group
    ( clock 200 )
    ( netRef CLK1 )
  )
  ( timingGroup My_CLK2_Group
    ( clock 100 )
    ( netRef CLK2 )
  )
)
( intendedTargetDelay 100 My_CLK1_Group My_CLK2_Group )
( intendedTargetDelay 100 My_CLK2_Group My_CLK1_Group )
```

In the above example, two clock groups were specified: 'My\_CLK1\_Group' and 'My\_CLK2\_Group'. All data paths between these two groups are constrained to meet the more restrictive of the two timing requirements.

## Tool Progress Monitoring

As the you gain experience and the density of your designs starts to increase, and our your speed requirements become more restrictive, you may find the MPA Design System working longer on your designs. As such, this section is included to give you a clear explanation of all the status windows and displays the system gives you during the course of design importation and auto layout.

### Import

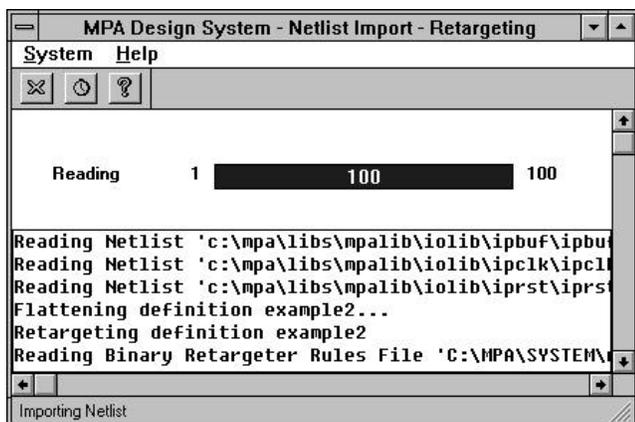


Figure 10. Importing and Retargeting

During the import and retarget phase, the MPA Design System takes the EDIF netlist in, checks it for compatibility and errors. The macro elements of the EDIF netlist are mapped to the cell definitions available to the MPA family. If there are no problems found, the import phase concludes with the output of a .NET file, the native netlist format of the tool.

### Autolayout

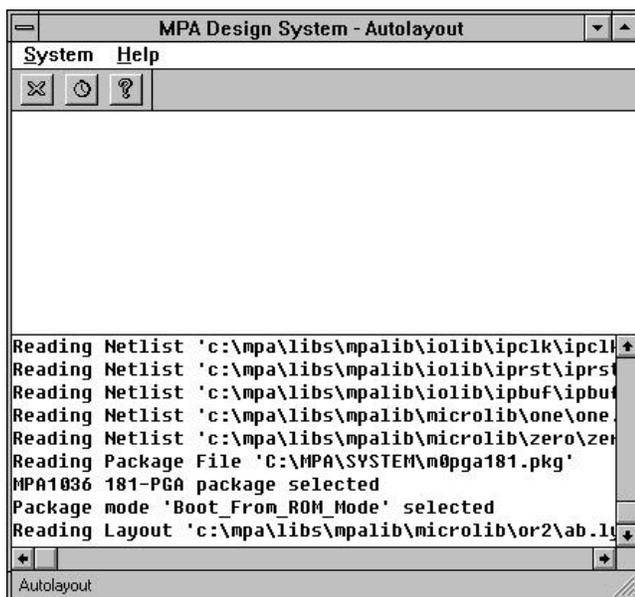


Figure 11. The beginning of Autolayout

During the brief initialization period of the autolayout process, the previously mapped component nets are read in, the package information is read in, I/O pads are counted along with core cells. A figure of utilization is echoed back and the tool proceeds to the clustering phase. No status bars are presented in this step.

### Clustering

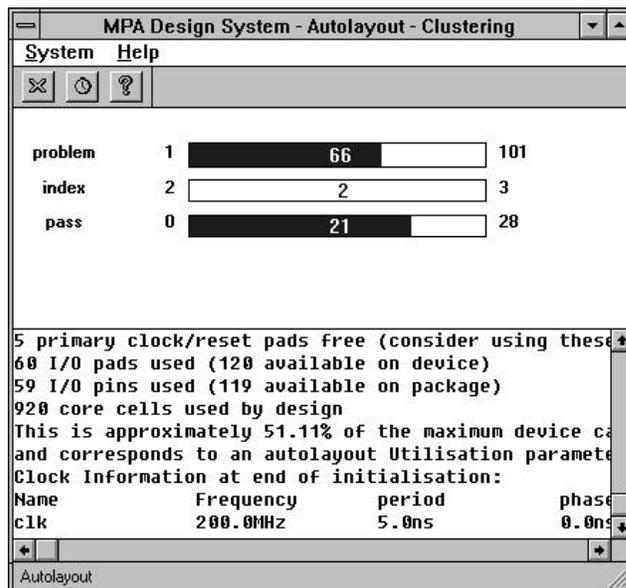


Figure 12. The Clustering Status

Clustering combines groups of related logic together to help break up the place and route problem into more manageable chunks. In Figure 12: **Problem** refers to the number of root level clustering problems to be solved. A root level clustering problem is a cluster containing sub-clusters or and I/O. **Index** refers to the 3 phases of clustering for that problem. Phase 1 is the stage where the clustering algorithm decides which clusters are root level clusters, and is already complete. Therefore the progress bar starts on 2 and finishes on 3. **Pass** refers to the estimated number of items per root level problem.

### Partitioning and Pre-Placement

About half of the tool's time is spent in the partitioning phase. During this phase, the MPA Design System places the clusters formed in the previous process into zones. Pre-placement refers to the placement of I/O sites that were optionally fixed via the DPLD\_PAD\_PLACE attribute.

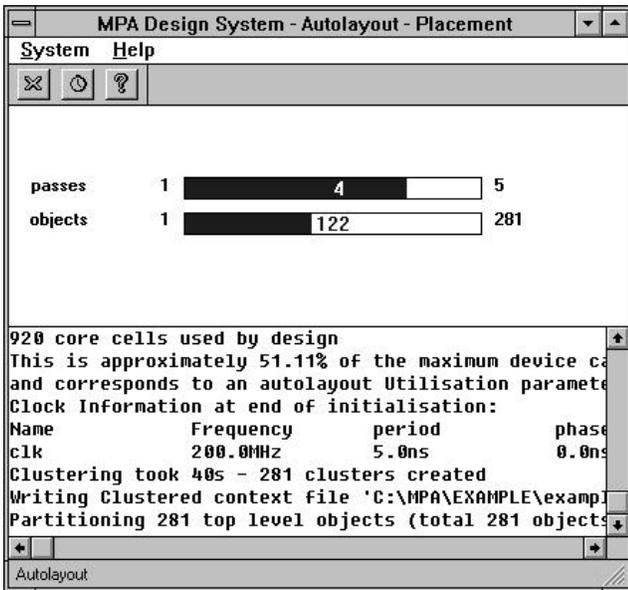


Figure 13. The Initial Partitioning Phase

The first part of the partitioning algorithm completes the initial placement tasks. **Passes** refers the fixed number of passes at the placement problem. The first pass yield an initial placement of clusters, the remaining 4 passes are refinements of this initial placement. **Objects** refers to the number of clusters created in the previous clustering phase.

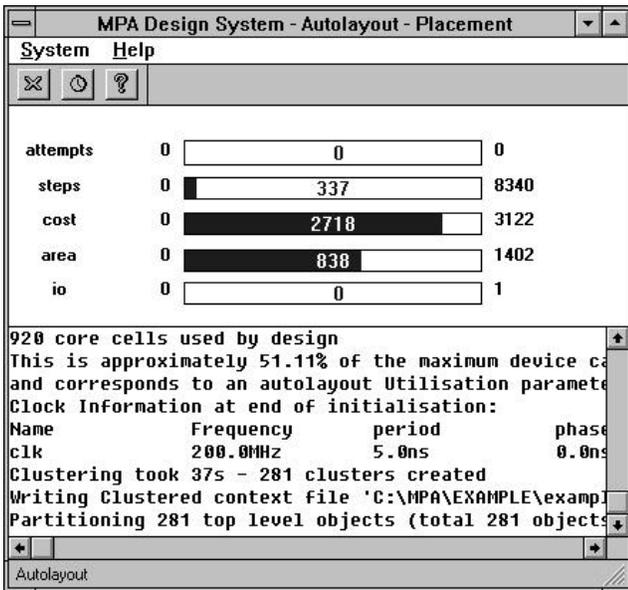


Figure 14. The Second Partitioning Phase

This phase takes the initial placement results and refines it by assigning specific zones and routing ports to the clusters. **Attempts** refers to the number of times this entire phase

will/has run. Attempts is set as a autolayout tool option 'Attempts'. Partitioning is the most important phase of the autolayout process. If you are looking for more speed from your design this is a good place to experiment. Increasing 'Attempts' in the autolayout tool options will generally yield improved results, but at the expense of extending the tool's run time of course. **Steps** refers to the number of steps the tool must take to complete the partitioning. The step rate increases as the tool proceeds. This is the indicator to watch to get the best feel for how long partitioning will take. **Cost** refers to the total cost of the design (the sum of the cost of the nets plus the cost of the cost of the parts). Cost generally starts off high and proceeds in a downward trend as a solution is approached. It gives you an indication of the quality of the partition in the current stage, whether or not the tool is converging to a solution, how good that solution is, and how sensitive the design is to changes in the way the clusters are placed. **Area** is proportional to the sum of the number of instances in the zones in excess of the number specified in the utilisation parameter. **IO** refers to number of over congested areas in the IO zones.

### Global Routing

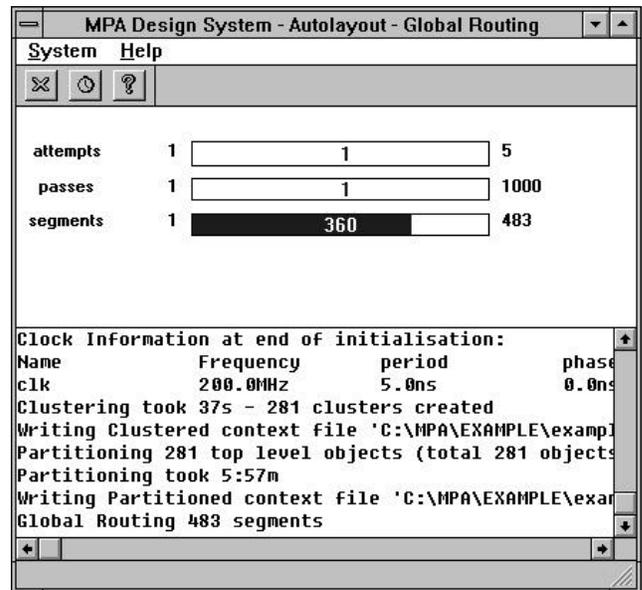


Figure 15. Global Routing Phase

After partitioning has assigned all the clusters to zones, global routing assigns signals to zone port cells and routes these ports to one another as required. **Attempts** refers to the number of attempts to route all segments. If global router fails to complete the routes on the fifth iteration it will give up. **Passes** refers to the number of attempts per segment to route globally (1000 per segment). **Segments** refers to the total number of segments to be routed.

## Zone Routing

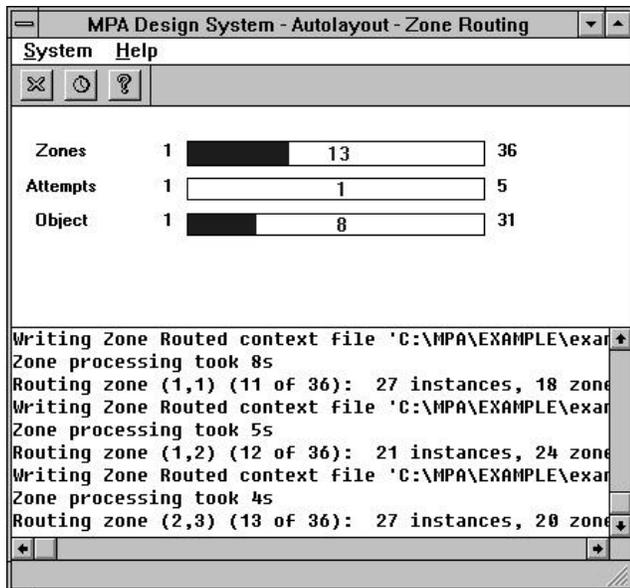


Figure 16. Simple Zone Routing Example

Zone routing is the final major step in completing the autolayout problem. The tool can often spend more than half of its time completing this task. Partitioning has placed clusters into zones, and global routing has assigned and routed signals from zone to zone and zone to I/O. Now the zone routing tool must complete the connections at the lowest level of hierarchy. **Zones** refers to which zone is currently being processed. **Attempts** is the number of attempts taken at completing the zone routing. If the zone router fails to route the zone on the fifth iteration, it will give up on the current placement solution and attempt to replace instances within the zone as described in Figure 17. **Object** refers to the total number of objects to be placed and routed in the zone.

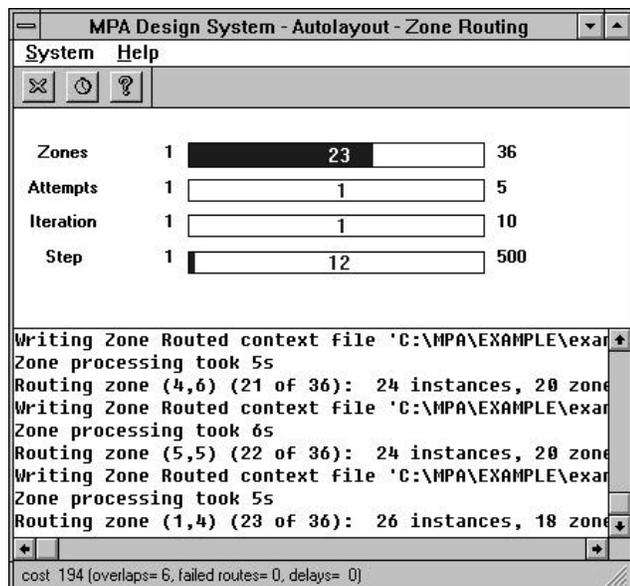


Figure 17. Complex Zone Routing Problem

If the instance placement solution provided by partitioning can not be zone routed, the tool (zone router) gives up on the process described in Figure 16 and moves to a second phase of zone routing. In this phase the tool discards the placer's solution and provides new possible solutions to try. In this case, **Zones** refers to the zone currently being processed. **Attempts** is the number of attempts taken at completing the zone routing. **Iteration** is the number of attempts to take to achieve the targets for zone routing. **Step** refers to the maximum number of steps (per iteration) to obtain the zone routing targets.

## Back Annotation

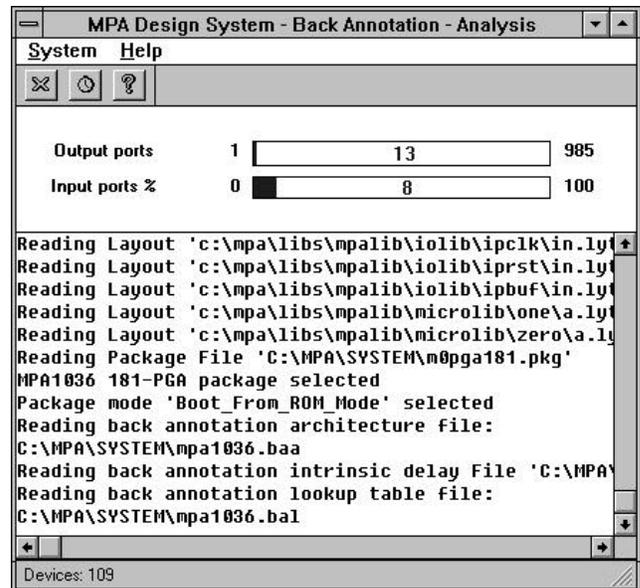


Figure 18. Back Annotation Status

Once you've successfully completed a layout, you may want to generate a back annotation file so timing data can be passed back to your simulator. In this process the estimated RC and routing switch delays are compiled and passed back to a netlist format of your choosing. **Output ports** refers to the every driving pin of the design. **Input ports** (expressed as a percentage) refers to all the inputs that could possibly affect the output port being considered.

## Summary

Figure 19 is a matrix that shows how each of the methods discussed in this application note interacts with the various phases of the autolayout process. You can see from this matrix that the most critical phase of the autolayout process is partitioning. If you are having trouble meeting a restrictive timing requirement, this is where you should concentrate your efforts.

The number of different combination of design styles, tool settings and use of attributes is very large and so it is impossible to give exact guidance on how to use the MPA Design System to get the absolute best possible speed solution for your particular design. However, the MPA Design System and the MPA1000 architecture were co-developed and as such should provide you with a total system ready to implement your requirements quickly and easily with no need to adjust the parameters and attributes as described in this note. If however your timing requirements are not met on your first pass through the tool, hopefully you have been provided with enough information in this note to start experimentation in an informed manner.

	Map (import)	Pre-Place	Cluster	Partition	Global	Zonal
<b>Front End Design</b>						
Macro Selection	X					
<b>PIN Attribute</b>						
DPLD_IGNORE_TIMING			X	X	X	X
<b>INSTANCE Attribute</b>						
DPLD_IGNORE_TIMING			X	X	X	X
<b>NET Attributes</b>						
DPLD_IGNORE_TIMING			X	X	X	X
DPLD_CLUSTER_SEED			X			
DPLD_PLACE_PRIORITY					X	
DPLD_PAD_PLACE		X				
<b>Tool Options</b>						
Target Delay			X	X	X	X
Utilization				X		
Effort				X		
Start Temp				X		
Fan Out			X			
Attempts				X		
Min Zone Delay						X
Back Off				X	X	X
Delay Cost				X	X	X
Seed			X	X	X	X

Figure 19. Effected Modules for Macros, Attributes, and Tool Option Settings

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

### How to reach us:

**USA/EUROPE/Locations Not Listed:** Motorola Literature Distribution;  
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447 or 602-303-5454

**MFAX:** RMFAX0@email.sps.mot.com – TOUCHTONE 602-244-6609  
**INTERNET:** <http://Design-NET.com>

**JAPAN:** Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center,  
3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-81-3521-8315

**ASIA/PACIFIC:** Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,  
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

