

Using VIEWlogic's PROSeries 6.1 with the MPA Design System

Prepared by
Douglas M. Shade
Motorola Programmable Logic Products



APPLICATION NOTE

Using VIEWlogic's PROSeries 6.1 with the MPA Design System

by Douglas M. Shade, Motorola Programmable Logic Products

Introduction

The Motorola family of MPA devices and supporting software provide system designers with a collection of flexible and powerful tools. This application note focuses on the use of VIEWlogic's PROSeries 6.1 schematic capture and simulation programs as front end design tools for the MPA Design System FPGA place and route software. A basic design flow is introduced followed by a more in depth discussion of parameters for place and route and concludes with a discussion on back annotation and simulation procedures.

Basic Design Flow

In this simplest example of using PROSeries, a straight path is taken from design entry through export to the MPA Design System. More detailed discussions on: place and route parameters, I/O parameters, hardware dependent macros, back annotation and simulation are deferred. The reader is assumed to be familiar with PROSeries and have casual knowledge of the MPA Design System.

Libraries

The EDIF net list reader of the MPA Design System is currently constrained to understand only those components passed to it from the MACROLIB, MICROLIB and IOLIB libraries provided. Only a very few other symbols from the BUILTIN library may be used directly in the schematic. These are: IN, OUT and BI; their usage is explained more fully later. Your VIEWDRAW.INI file must contain lines similar to the following in order to steer PROSeries in the correct direction when adding components to your schematic.

```
DIR [rm] C:\mpa\wvlibs\mpalib\macrolib (macrolib)
DIR [rm] C:\mpa\wvlibs\mpalib\microlib (microlib)
DIR [rm] C:\mpa\wvlibs\mpalib\iolib (iolib)
DIR [rm] C:\mpa\wvlibs\mpalib\builtin (builtin)
```

When adding components to your FPGA schematic, be sure to use only components from these first 3 libraries and only the special hierarchical connectors IN, OUT and BI from the BUILTIN.

Capture

There are just a few unique steps to take during schematic capture to ensure a valid MPA Design System EDIF netlist. The netlist importer of the MPA Design System needs to recognize your design's I/O pins. To accomplish this, you may either create a top level symbol for your completed schematic or you may opt to include VIEWlogic's hierarchical connectors.

If you are going to instantiate your completed FPGA schematic into a larger board or system level schematic then generating a top level symbol is the more appropriate method to use. In order to do this, each of the IOLIB components used must have a named net stub attached to their 'external world' pins. Once this task is completed all that is left is to create a VIEWlogic symbol for the entire FPGA schematic. Each of the pin names on the symbol must match the net-stub names exactly. A pin is required for every I/O net-stub.

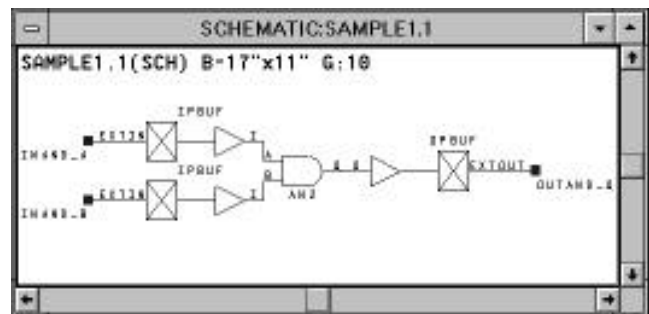


Figure 1. Top Level Schematic, Named Net Stubs

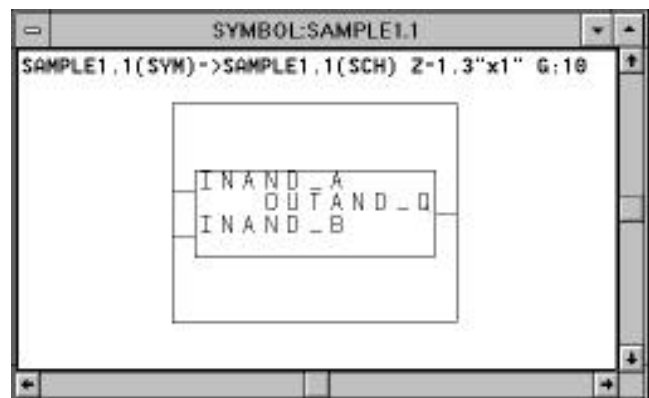


Figure 2. Top Level Schematic's Symbol, Pin Names = Net Stub Names

If on the other hand the schematic you are creating is stand alone for the FPGA, then a short cut method is available to you. As before, place the desired IOLIB components on your schematic. Then from the BUILTIN library select the IN, OUT or BI hierarchical connector as appropriate. Connect this hierarchical connector with to the IOLIB component's 'external world' pin and name the net.

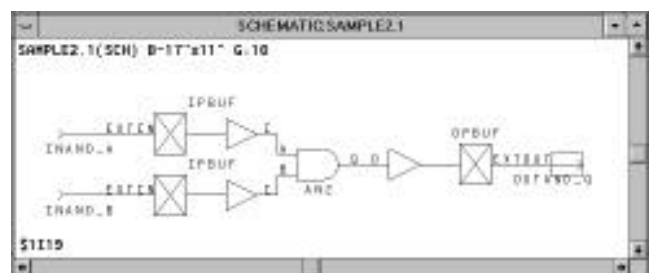


Figure 3. Top Level Schematic Using IN and OUT Symbols from BUILTIN. The OUT Symbol Is Highlighted (Boxed), Instance \$1119.

This name may now be referenced for stimulus/response in the VIEWlogic simulator. Additionally, this net name is passed

in the exported EDIF netlist to the MPA Design System place and route tool.

Net List Export

EDIFNETO is the VIEWlogic tool that translates your completed schematic into an EDIF file importable to the MPA Design System. While the PROSeries netlisting tool is generally available from the pull down menus, some of the requisite features of the tool for MPA compatible netlisting are not. So from a DOS session in your current VIEWlogic design directory, run EDIFNETO. The following log shows the correct answers to EDIFNETO's questions. You need to generate a "flattened" netlist, at the level "micro".

```

MS-DOS Prompt
C:\PROSER61\MYDESIGN>edifneto
EDIF U2.0.0. WIRELISTER - U4.1.2; Workview 4.1.2 032992, 3000 Series
Copyright 1985,1992 by Viewlogic Systems, Inc.
Project Name: sample2
Do you want a flattened netlist? [N] : y
Do you want to evaluate parameterized attributes? [N] :
Level Name: micro
Output File Name [C:\PROSER61\MYDESIGN\SAMPLE2.EDN]:
Author string: Doug Shade
Port/supply type configuration file name (EDIFPTYP.CFG):
Level string: MICRO
41 Comp. 3 Nets) Reading complete.
Using EDIFPTYP.CFG for signal name, port/supply type name pairs.
Updating global interface nets.
Update complete.
Processing complete.
0 errors and 0 warnings found.
2 module(s) written to C:\PROSER61\MYDESIGN\SAMPLE2.EDN
C:\PROSER61\MYDESIGN>_

```

Figure 4. EDIFNETO DOS Window Session

You now have a .EDN EDIF netlist ready for import to the MPA Design System. Give it a try.

Attributes

The MPA Design System's import process can accept a set of attributes to help the designer tune the layout and routing processes. The system also accepts I/O parameters 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 name as the design. The designer may 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.

Place and Route Layout Attributes

The MPA Design System enables the tuning of place and route algorithms in three ways. The first is with the adjustment of the Auto Layout tool options such as annealing temperature, target delays, target zone utilization etc. Additional details on the available options and their use are available in the on-line help facility of the MPA Design System. The second method involves the construction of separate clock files. Here again, additional information is provided in the on-line help system and is not presented in this application note. The third method of influencing place and route results is the inclusion of the following attributes in the schematic, or in an external .PAT file.

Table 1. Valid Attributes

Sch. Component	Attached Place and Route Attribute	Attached I/O Attribute
Net	DPLD_IGNORE_TIMING DPLD_CLUSTER_SEED DPLD_PLACE_PRIORITY DPLD_PAD_PLACE	DPLD_PAD_PLACE
Symbol (instance)	DPLD_IGNORE_TIMING	DPLD_PUP PULLUP or PULLDOWN DPLD_OPDRIVE DPLD_OPLEVEL DPLD_OPSLEW DPLD_IPLEVEL DPLD_PAD_PROPERTIES
Pin	DPLD_IGNORE_TIMING	

DPLD_IGNORE_TIMING

The DPLD_IGNORE_TIMING attribute is used to ignore certain paths for timing purposes. It may be set on an symbol (instance), a net or a pin. If a net has the attribute set, then all segments within that net are ignored. If an instance has the attribute set, then all inputs and output net segments from that instance are ignored. If a driven pin has the attribute set, then the segment driving that pin is ignored. The effect of ignoring a segment for timing purposes is to cause the autolayout to ignore all paths between clocked objects which the segment or instance lies on. 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 will be ignored for timing purposes as well.

Any value given with this attribute is ignored.

DPLD_CLUSTER_SEED

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

DPLD_PLACE_PRIORITY

The DPLD_PLACE_PRIORITY attribute can be applied to a net to force the software to lay out that net in a physically smaller area – in other words, to 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 may be 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. Example: DPLD_PAD_PLACE=C2

I/O Parameter Attributes**DPLD_PUP**

DPLD_PUP – attaches to WPUP primitive cells only, to select either or both of the pull-up resistors available in a WPUP cell. Valid values are 1, 2 or BOTH.

PULLUP or PULLDOWN

PULLUP – set this to 1 if you want to enable the pull-up resistor on the external pin of an I/O pad. Default is 0 (resistor disabled). Example: PULLUP = 0

PULLDOWN – set this to 1 if you want to enable the pull-down resistor on the external pin of an I/O pad. Default is 0 (resistor disabled). Example: PULLDOWN = 0

DPLD_OPDRIVE

DPLD_OPDRIVE – sets the output drive current of an output or bi-directional pad to either 6ma (default) or 12ma.

DPLD_OPLEVEL

DPLD_OPLEVEL – sets the output voltage level of an output or bi-directional pad to either 3v or 5v (default).

DPLD_OPSLEW

DPLD_OPSLEW – sets the output slew rate (transition speed) of an output or bi-directional pad to high (default) or low.

DPLD_IPLEVEL

DPLD_IPLEVEL – sets the input threshold voltage of an input or bi-directional pad to either CMOS or TTL (default).

DPLD_PAD_PROPERTIES

VIEWlogic permits the use of the combined attribute, DPLD_PAD_PROPERTIES which combines the following

attributes into a single comma separated list: PULLUP, PULLDOWN, DPLD_IPLEVEL, DPLD_OPLEVEL, DPLD_OPDRIVE, DPLD_OPSLEW. This is especially useful when defining a block of I/O pins in an external attribute file. When prompted, answer NO to the “Expand Label?” prompt in VIEWlogic. Example: DPLD_PAD_PROPERTIES = 0,0,CMOS,5v,12ma,high

Defaults and Invalid Combinations

The default I/O pad attributes have been selected so that they can connect to either TTL or 5v CMOS without adjustment. The default parameters may not be ideal for every design, and they should be matched to the application in order to achieve the best performance and noise immunity. 3v CMOS users should be especially careful to set DPLD_OPLEVEL to 3v, otherwise damage to peripheral IC's may result.

The following combinations of user attributes are not permitted:

```
DPLD_OPDRIVE = 6ma AND DPLD_OPSLEW = low.
PULLUP = 1 AND PULLDOWN = 1
```

The following combination of user attributes on a single bi-directional pad should be avoided, as it may produce unpredictable results:

```
DPLD_IPLEVEL = CMOS AND DPLD_OPLEVEL = 3v
```

Assigning Attributes in a Schematic

Any of the attributes listed above can be assigned to pins, nets and macro symbols as appropriate. In PROSeries, the method of assigning attributes is straight forward. Select the desired net or symbol (its color will change or a bounding box will appear respectively, identifying it as being 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.

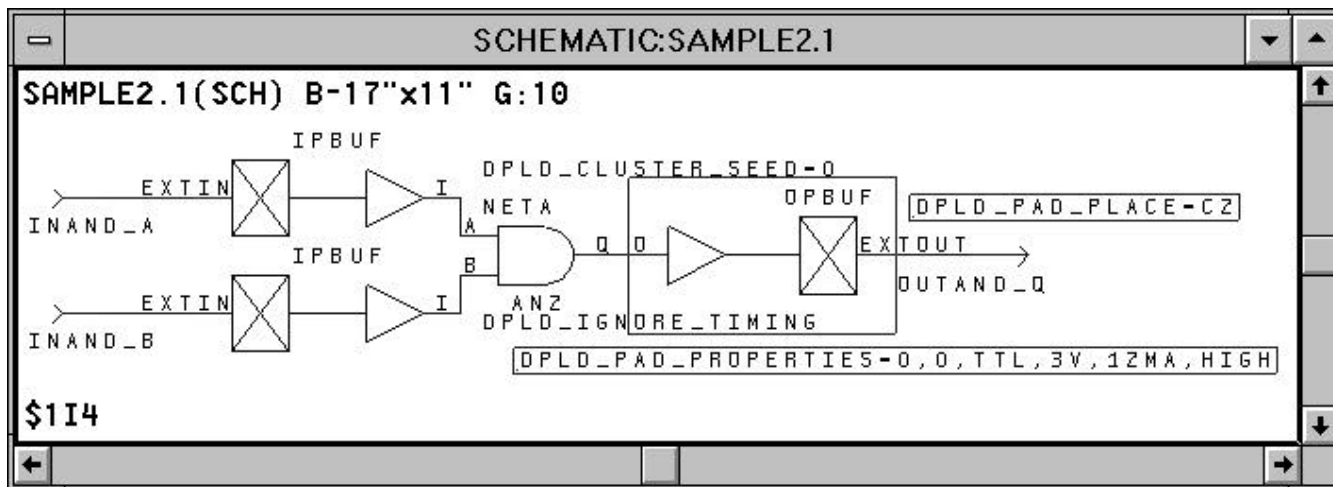


Figure 5. The Net “NETA” Attributed With DPLD_CLUSTER_SEED=0. The “B” Input Pin of AN2 Attributed With DPLD_IGNORE_TIMING. The Selected Component “OPBUF \$1I4” and its Attached Attributes are Boxed.

Selecting a component pin is just a bit trickier. Left click on the desired component (in this example AN2) then right click on the desired input pin. “Add” – “Object Attribute” as described above.

Assigning Attributes & Instances in an External .PAT File

Assigning attributes to a long series of pins, 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 .PAT file must have the same name as the EDIF netlist .EDN file, and must reside in the same directory.

The external attributes file supports three 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.

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:

<code><object-class></code>	is one of port, net, or instance.
<code><object-name></code>	is the netlist name of the object (port, net or instance) being operated on.
<code><operation></code>	is one of attribute or instance.
<code><name></code>	is the name of a definition or an attribute.
<code><value></code>	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.

```
// This is a comment
# This is a comment as well

port sega attribute dpld_pad_place 22
//This results in the IPBUF being placed on the pad associated with package
//pin 22.

port sega attribute dpld_ignore_timing dummy_arg
//For place and route purposes, the design's timing parameters are ignored
```

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

`port <name> attribute <name> <value>`

Attribute <name> with (optional) value <value> is added to the port instance (the instance driven by the formal port). Only works with input and ports.

`port <name> instance <name>`

The port instance (the instance driven by the formal port) is replaced by an instance of the given definition. The only valid definitions are IPCLK, IPRST. This syntax is limited input ports with 1 input pin and 1 output pin (IPBUF for example).

`net <name> attribute <name> <value>`

Attribute <name> with (optional) value <value> is added to the net.

`net <name> instance <name>`

Creates an instance of the given definition and inserts it into the named net. The only valid definitions are ACLK, ARST, APBUF and PABUF.

`instance <name> attribute <name> <value>`

Attribute <name> with (optional) value <value> is added to the instance

Example .PAT File Entries

The following sample .PAT file entries reference the simple schematic shown in Figure 6.

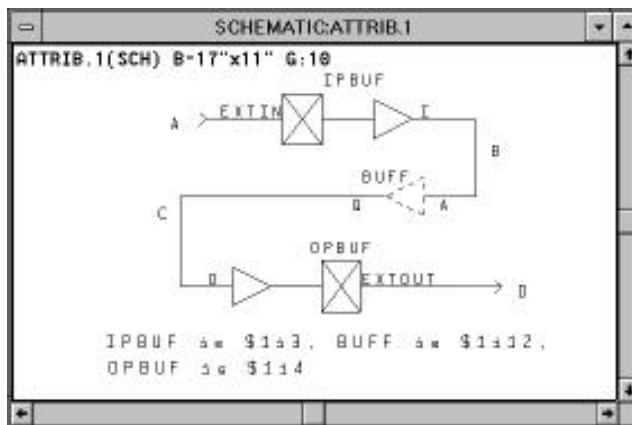


Figure 6. The .PAT File Attributes Are Added to This Simple Schematic. Nets Are Named SEGA, SEGB, SEGC and SEGD.

```

//for the net segment associated with the formal port sega. A dummy
//argument is required for this attribute.

port sega instance ipclk
//This results in the IPBUF being replaced by an IPCLK, forcing the net onto
//a clock network.

net segb attribute dpld_ignore_timing dummy_arg
//For place and route purposes, the design's timing parameters are ignored
//for the entire net "segb". A dummy argument is required for this
//attribute.

net segb attribute dpld_cluster_seed 1
//The dpld_cluster_seed attribute and value shown get assigned to the net B
//for evaluation during place and route.

net segb attribute dpld_place_priority 1
//The dpld_place_priority attribute and value shown get assigned to the net
//B for evaluation during place and route.

net segb instance aclk
//This results in an ACLK buffer being inserted between IPBUF's output and
//BUFF's input. BUFF is now driven off the resulting clock routing
//resource.

instance $1I4 attribute dpld_opdrive 12ma
//This results in the OPBUF getting 12ma drive capability.

instance $1I12 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 $1I12. A dummy
//argument is required for this attribute.

```

All Valid Combinations of Attributes & Instances in an External .PAT File

The following show all the valid combinations of the attributes in the .PAT file.

```

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>
    dpld_pup 1|2|BOTH
    pullup 1|0
    pulldown 1|0
    dpld_opdrive 6ma|12ma
    dpld_oplevel 3v|5v
    dpld_opslew high|low
    dpld_ipevel CMOS|TTL
    dpld_pad_properties (see above
paragraph describing this attribute)
net <name> attribute
    dpld_cluster_seed n (where 0 ≤ n ≤ 1000,
1 is default, 0 means ignore net)
    dpld_place_priority n (where 0 ≤ n ≤ 10, 1
is default)
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>
    dpld_pup 1|2|BOTH
    pullup 1|0
    pulldown 1|0
    dpld_opdrive 6ma|12ma
    dpld_oplevel 3v|5v
    dpld_opslew high|low
    dpld_ipevel CMOS|TTL
    dpld_pad_properties (see above
paragraph describing this attribute)

```

Library Elements Specific to MPA Hardware Features

Most designs can be fit to the desired speed into the MPA family without the designer needing to know much about the details of the device's internal construction. However, to get the most out of the MPA, you should take some time to browse through the on-line help files thoroughly. The help files are rich in detail regarding the hardware specific macros and routing resources macros inherent in the MPA design system. The following topics are presented as simple examples on most of these hardware specific features. For an exhaustive presentation, please refer to the on-line help facility.

Logic One and Logic Zero

It may at times be necessary to modify the functionality of a macro from the supplied libraries by tying one or more of its inputs to logic high or low. There are two special elements provided in MICROLIB for this purpose. ONE produces a logic high to all input pins tied to it; there are no fan-out restrictions for the signal. The ZERO element provides a logic low. An alternate method is to tie the input pins requiring a logic high to a net named VDD, and tie the output pins requiring a logic low to a net named GND.

For all the above methods, the MPA Design System will recognize the logic as static and eliminate superfluous logic elements wherever possible during the place and route process.

Wired-OR

The MPA family allows for connecting many outputs to a single common signal line. The available macros for this function are: WND2, WINV, WOR2, WBUF. When using these macros, 1 or 2 WPUP components are required. The maximum number of connections to a single signal is given by

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

Table 2. Maximum Drivers on a Wired-OR Signal

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

If the number of drivers on the Wired-OR net is below half of the maximum allowed, then only a single WPUP element is recommended. Adding a second WPUP to a very large net, will help speed things up some, but at the cost of increased power consumption. The allowable values for the DPLD_WPUP 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.

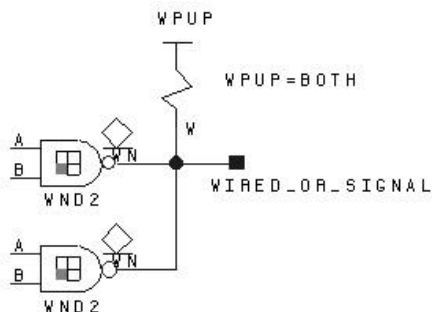


Figure 7. The Diamond Symbol Reminds the User That These Outputs Cannot Source a Logic HIGH, but Are Only Able to Pull the Output to a Logic LOW.

Adding additional wired-or outputs to the net WIRED_OR_SIGNAL will slow it down. Adding additional

inputs to the net has little effect on speed. Low to high transitions are typically slower than high to low transitions on a wired-or signal.

Peripheral Bus

The periphery of the MPA die is bordered with an 8 bit wide Peripheral Bus (P-Bus). The P-Bus can be broken at each corner of the array by switches. (Setting of the switches is handled automatically in the MPA Design System software.) Each of the resulting 4 segments has two programmable pull up circuits, one at each end. The P-Bus is ideal for routing common signals to many I/O macros.

It is easy to build a scenario where many I/O pins have a single or several control signals in common. In this instance you would want to place that signal on the P-Bus using a APBUF. Conversely, pulling a signal off the P-Bus back into the array is accomplished using the PABUF.

The ability to construct Wired-OR nets is not limited to signals internal to the array. 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 rise and especially fall 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. A conservative guide line for number of WPUPS verses number of P-Bus segments is given below.

Table 3. Recommendations for using WPUP on the P-Bus

Number of P-Bus Segments	Recommended Number of WPUPS
4	(4 P-Bus Segments Not Recommended)
3	Use 2 WPUPS
2	1, 2 or 3 WPUPS (Speed/Power Trade-Off)
1	1 or 2 WPUPS (Speed/Power Trade-Off)

Low Skew, Clock and Reset Nets

For a high fan out signal or for a signal where you would like to keep the skew limited to less than 1nS, you should consider moving the signal onto a clock network using the ACLK or ARST macro (they both do the same thing).

MPA I/O Structures

The standard I/O cell of the MPA array is very feature rich. The complexity of this structured 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 (including the P-Bus Wired-OR resources previously mentioned) and it becomes clear that

significant functionality can be achieved before ever using the normal internal resources of the array.

In Figure 8, 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 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.

Of course, all of this functionality could be moved internal to the array, using only the simple I/O macros.

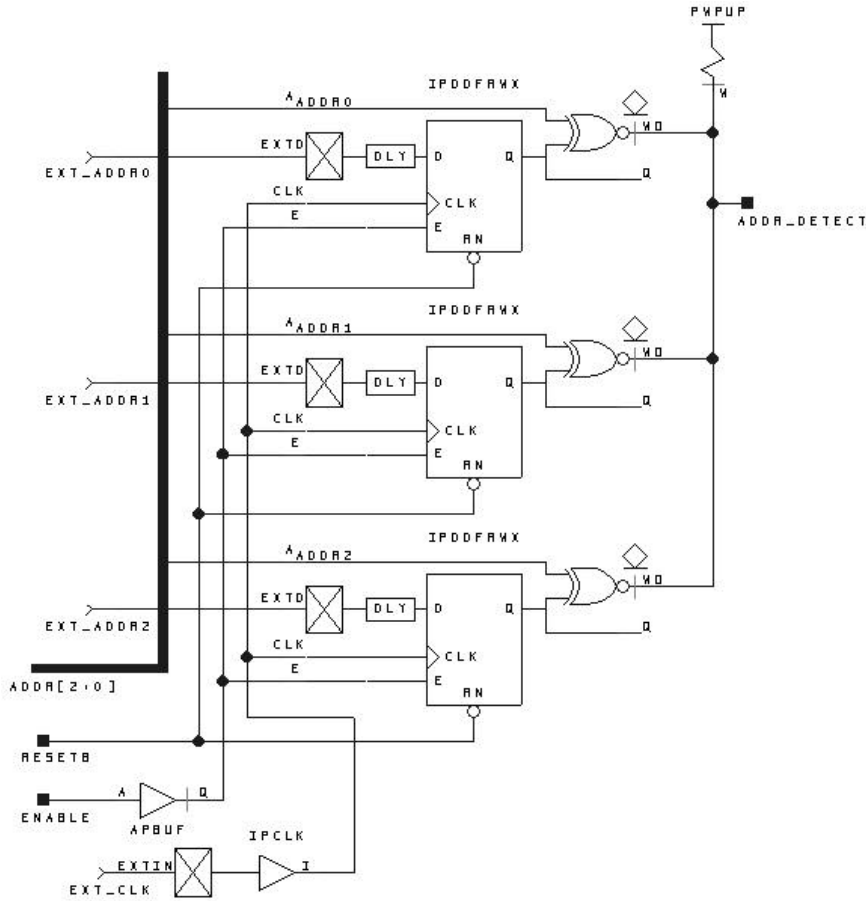


Figure 8. A Three Bit Address Decoder, Only I/O Cell and P-Bus Resources Used

Simple I/O Structures

Your first designs for the MPA will probably only use the above I/O macros, with all the latching, decoding etc. being handled internal to the array. Soon though, you will probably encounter a real world design that constrains you to a specific clock-to-Q specification or some other requirement that will have you going back to re-examine the components available in the IOLIB. There are many variations of latched, registered, and Wired-OR inputs, outputs and bi-directional macros. Again, please invest some time with the on-line help descriptions of the device and the libraries.

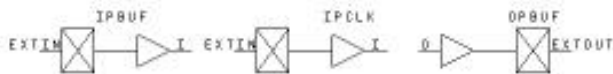


Figure 9. The Three Most Commonly Used I/O Macros

Back Annotation and Simulation

It is assumed that the reader is familiar with the tools and procedures involved in using the VIEWlogic PROSim tools.

The MPA Design System has a facility that provides post place and route back annotation data in a format compatible with the PROSim netlist. Briefly, a VIEWlogic format back annotation data table file (.DTB) can be generated after completion of a place and route. The .DTB file can be read in by VIEWlogic's VSM netlist tool, to provide an accurate simulation netlist of your completed design.

Within the main window of the MPA Design system, select the Tool Options button (or use the pull down menu "Tools – Options"). Select Back Annotation and select "VIEWlogic (.dtb)". Once your design has been imported, placed and routed, the back annotation button becomes selectable. Invoke the back annotation tool by pressing its button. A back annotation file will be placed in the same directory as the rest of the MPA Design System's output files. The files name will be

the same as your .DSN file (visible in the title bar of the MPA Design System main window). Generally, this file needs to be moved up to your VIEWlogic design directory.



Figure 10. VSM Netlister, and its Options sub-window. "BACKANO" is the design name. The back annotation file name is "layout2.dtb".

Invoking the VSM netlister from the PROCapture is done in the normal way. Using the "Tools – Link to PROSim" pull down menu, select "Options" in the PROSim Wirelister window. Enter the name of the desired back annotation file. (You may have several different valid layouts completed at this point, but you can only merge one back annotation file at a time into the simulation netlist.) In the example shown above, the schematic's name is "BACKANO" and the post place and route back annotation file is "layout2.dtb". The MPA Design System generated delays will now be included in the simulation net list.

In Figure 12, ANDB goes high at time 45 and ripples through the AND tree (Q1, Q2, Q3) with the final output ANDOUT going high about 14nS later. When ANDB drops, the third AND gate (Q3) is the first to fall because in this implementation it happened to be placed closer to the ANDB input than AND gates Q1 & Q2. With ANDA held high, gate delay is the limiting factor for a rising ANDOUT and path delay is the limiting factor for a falling ANDOUT.

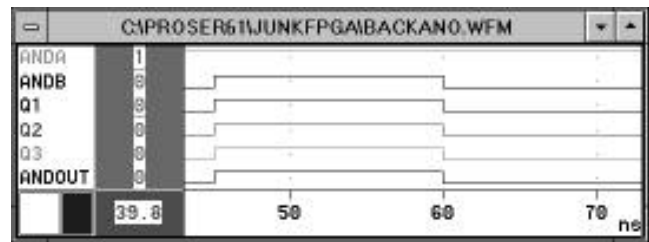


Figure 11. Before back annotation. ANDB going high ripples through all three AND gates and the output instantaneously.

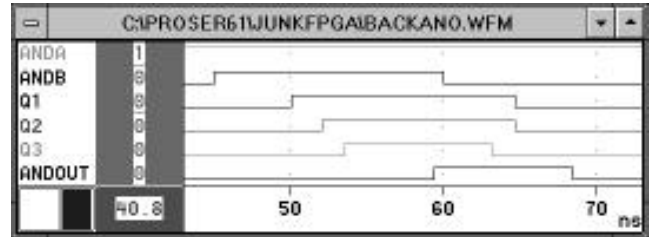


Figure 12. After back annotation of place and route delays. ANDB going high ripples through Q1,2 & 3, but going low happens to take Q3 low first.

Back Annotation and Logic Reduction

The MPA Design System does some "bubble pushing" during the retargeting / fitting of the logical design into the physical implementation. In some cases, redundant gates in the schematic design are eliminated in the physical layout. When this occurs, back annotation proceeds as before with the resulting delays being added only to the input pin of the most downstream logic element not eliminated in the retargeting / fitting process.

In Figure 14, the 4 inverters are eliminated in the retarget / fitting process. The wire delay between the remaining IPBUF and OPBUF gets back annotated to the OPBUF's input pin. The back annotated simulation netlist has 4 zero delay inverters (just as the pre back annotation netlist had) followed by the OPBUF with the real world delay.

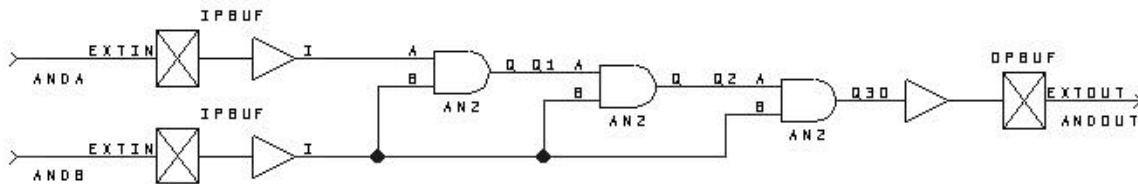


Figure 13. A sample circuit to explore simulation before and after back annotation.

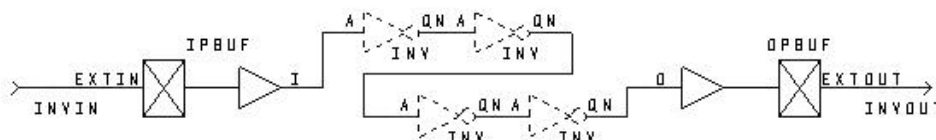



Figure 14. The string of inverters will be removed during place and route.

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

