

Recommended Design Methods

Introduction

Described here are a series of guidelines for designing with AT6000 Series field programmable gate arrays (FPGAs). Among the topics covered are basic cell functionality, building simple functions, general manual placement-and-routing rules, and schematic-entry tips that can make time spent in the Interactive Editor more productive. Keeping these guidelines in mind can reduce design time and produce more efficient circuits.

The Basics

Before beginning a design, it is important to understand the building blocks of the Atmel architecture. AT6000 Series devices consist of a symmetrical array of cells that perform logic functions and are connected to a comprehensive busing structure. Symmetrical interconnects on the four sides of each cell provide cell-to-cell and cell-to-bus connections. Figure 1 shows the logic contained in

each cell. A more detailed explanation of cell functionality can be found in the AT6000 Series data sheet.

Logical Functions and Cell Configurations

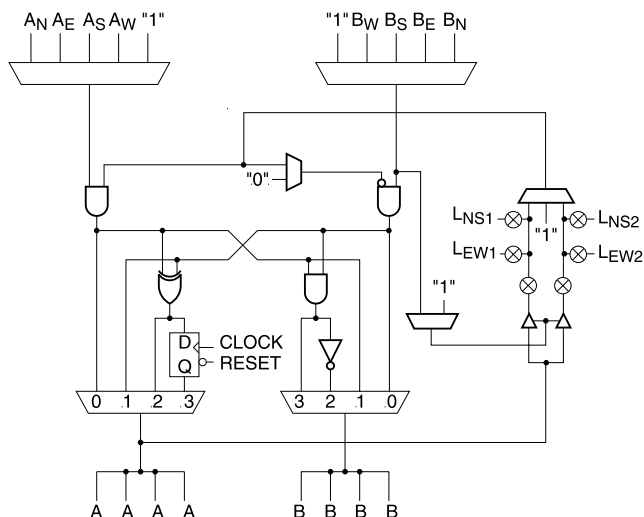
The logical function is the function a cell is performing; the cell configuration is how the cell is configured to perform that function. Cells can perform 44 logical functions, each corresponding to at least one cell configuration.

For example, a cell can be configured six different ways to perform the same inverter function. Figures 2 and 3 show two of these configurations. The inverter in 2 goes from a local bus input (L) to a B output. The inverter in 3 takes an A input signal, inverts it, and drives it to the A, B, and L outputs. Depending on routing conditions and the use of neighboring cells, one configuration may be more appropriate than another.

Field Programmable Gate Array

Application Note

Figure 1. Cell Structure



The following table shows the number of cell configurations associated with each logical function:

Logical Function		Cell Configurations
Combinatorial (1 output)		
INV	Inverter	6
AN2	2-input AND	4
ND2	2-input NAND	4
XO2	2-input XOR	2
ORT	2-input OR	1
MUX	$A = (A \bullet L) \text{ XOR } (B \bullet L')$	1
AN2L	$(B \bullet L')$ 2-input AND with 1 Inverted Input	1
ORL	$(A + L')$ 2-input OR with 1 Inverted Input	1
AN3	3-input AND	1
ND3	3-input NAND	1
ANXO	2-input AND Feeding an XOR	1
BUF	Buffer	1
Combinatorial (2 outputs)		
INVW	Local Bus Input Inverter with Thru Wire	2
INVINV	Two Inverters	3
AN2S	2-input AND and B Wire	1
AN2X	2-input AND and B Cross Wire	1
AN2INV	$(A \bullet L, L')$ 2-input AND and Inverter	1
INVAN2	$(L', A \bullet L)$ Inverter and 2-input AND	1
NDND	Two 2-input NANDS	1
XOND	2-input XOR & NAND	2
SELBUFS	$A = (A \bullet L); B = (B \bullet L')$	1
SELBUFEX	$A = (B \bullet L'); B = (A \bullet L)$	2
XOND3	$A = (A \bullet L) \text{ XOR } B; B = (A \bullet L) \text{ NAND } B$	1
WAN2L	Wire & $(B \bullet L')$	1
AN2LW	$(B \bullet L')$ & Wire	1
Flip-Flop		
FD	D Flip-flop Q Out	1
FDN	D Flip-flop QN Out	3
FDHA	D Flip-flop = Half-Adder Sum	2
FDMUX	D Flip-flop = $(A \bullet L) \text{ XOR } (B \bullet L')$	1
FDND	D Flip-flop = 2-input NAND	1
FDOR	D Flip-flop = 2-input OR	1
FDORL	D Flip-flop = 2-input OR with 1 Inverted Input	1
FDXOAN3	D Flip-flop = $(A \bullet L) \text{ XOR } B; B = (A \bullet L) \text{ AND } B$	1
CLKEDGE	Clock Edge Detect	1
Tri-state		
BUFZ	Tri-state Buffer	1
FDZ	Tri-state D Flip-flop	1
LZ	"0" or "Z" (high impedance)	1
HZ	"1" or "Z" (high impedance)	1
CLKEDGEZ	Clock Edge Detect or "Z"	1
Constant		
ONE	Logic One	2
ZERO	Logic Zero	2
ONEONE	Two Logic Ones	1
ZEROZERO	Two Logic Zeros	1
ZEROONE	Logic One and Logic Zero	2

Figure 2. First Inverter Configuration

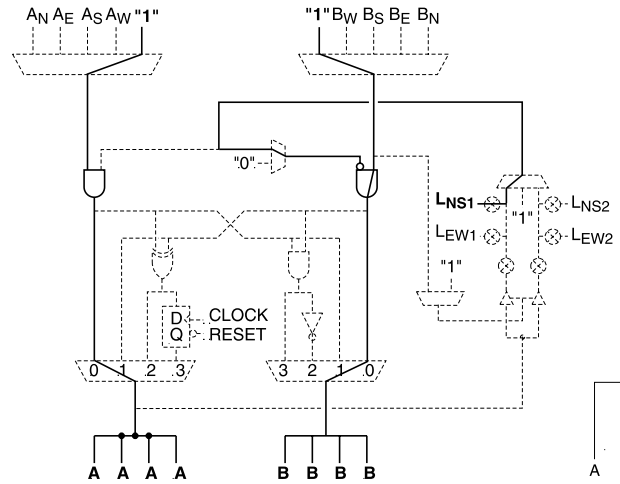
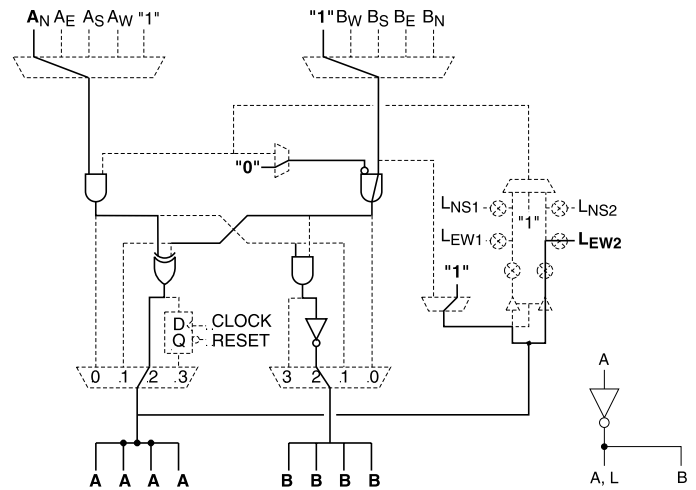


Figure 3. Second Inverter Configuration



Combining Cell Functions

When cells are used in combination, more functions are available. The combination of two cells, for example, can produce a four-input AND gate as shown in Figure 4. Other two-cell functions include a three-input XOR and a set/reset NAND latch.

The combination of three cells can produce more complex operations such as the two-input AND/OR function shown in Figure 5.

The three cells in combination produce a specific output, but the outputs of each cell can be used as part of another function. The following table lists the intermediate outputs available from this function:

Inputs		Outputs					
A	B	NAND	XOR	XNOR	NOR	OR	AND
0	0	1	0	1	1	0	0
0	1	1	1	0	0	1	0
1	0	1	1	0	0	1	0
1	1	0	0	1	0	1	1

This same two-input AND/OR function could be implemented using a single cell, but that would preclude use of the intermediate outputs.

Macros

When cells are grouped together to perform a specific logical function, they form a macro. A single-macro function, like a single logical function, can be configured in more than one way. Each of these physical variations is called a shape. Shapes vary in their use of routing resources, the relative placement of logic cells, and the number and type of physical primitives used. Routing varies from shape to shape, making some shapes faster than others. Designs can therefore be tuned for speed or size.

Macros are either hard or soft. Hard macros maintain the relative placement of each logic cell. The timing of a hard macro can be fully characterized and remains constant regardless of the macro's orientation in the layout because

the relative placement of its components remains unchanged. Because the AT6000 Series architecture is symmetrical, hard macros can be flipped or rotated without affecting internal macro timing.

Soft macros are made by dismantling, or softening, a hard macro, or by creating a schematic symbol that represents a series of hard macros. (In essence, every unplaced design is really a large, soft macro.) Each cell in a soft macro can be placed individually, so the timing of a soft macro depends on cell placement and interconnect.

A list of the more than 200 macros included in the Atmel Macro Library appears in the Integrated Development System data sheet. Most macros have more than one shape, yielding a total of over 350 configurations.

Macros can be combined to make more complex functions. For example, six half-adders (HA1, shown in Figure 6) can combine to create the 4-bit Summer shown in Figure 7. The Summer uses four binary inputs and defines its outputs as follows:

If 0 bits = 1, then output = 0 (Binary 000)

If 1 bit = 1, then output = 1 (Binary 001)

If 2 bits = 1, then output = 2 (Binary 010)

If 3 bits = 1, then output = 3 (Binary 011)

If 4 bits = 1, then output = 4 (Binary 100)

The Summer counts how many of the four-input bits are true and outputs the sum. Figure 8 shows the layout of the Summer in the Atmel architecture.

Placement and Routing

Once the macros to be used in a design have been selected and the schematic is complete, the design is ready to be placed and routed. Placement and routing are interdependent because cells can be used for both logic and routing. The relationship between placement and routing creates a number of trade-offs involving circuit speed, cell utilization and location of routed nets. For example, cell placement can interfere with routing by blocking a cell and making the net unroutable. Understanding these trade-offs makes place and route easier and more efficient.

Figure 4. Two Cells Combine to Make a Four-input AND

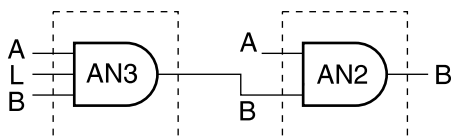


Figure 5. Two-input AND/OR Functions uses Three Cells

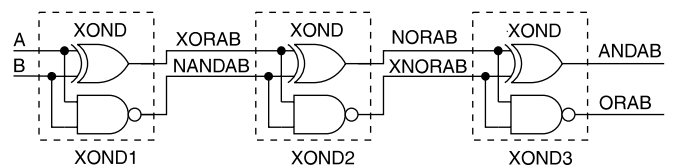


Figure 6. Schematic of HA1

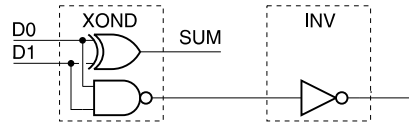


Figure 7. Six HA1 Macros used to Build a 4-bit Summer Function

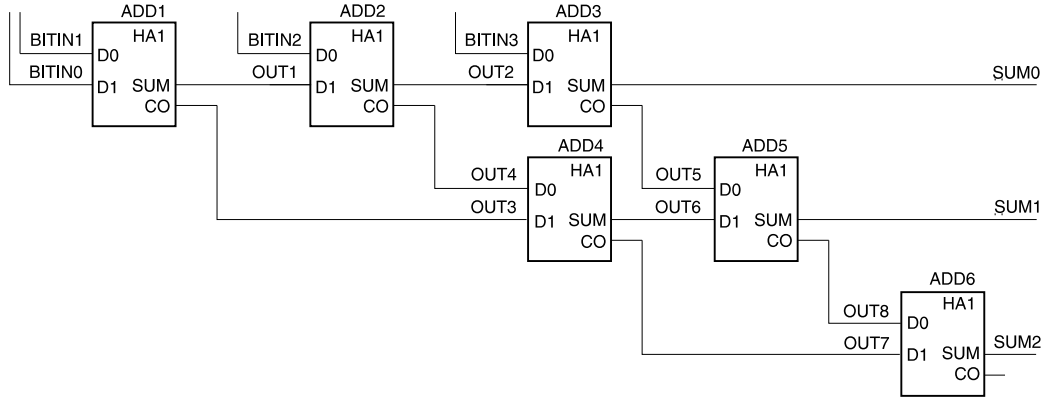
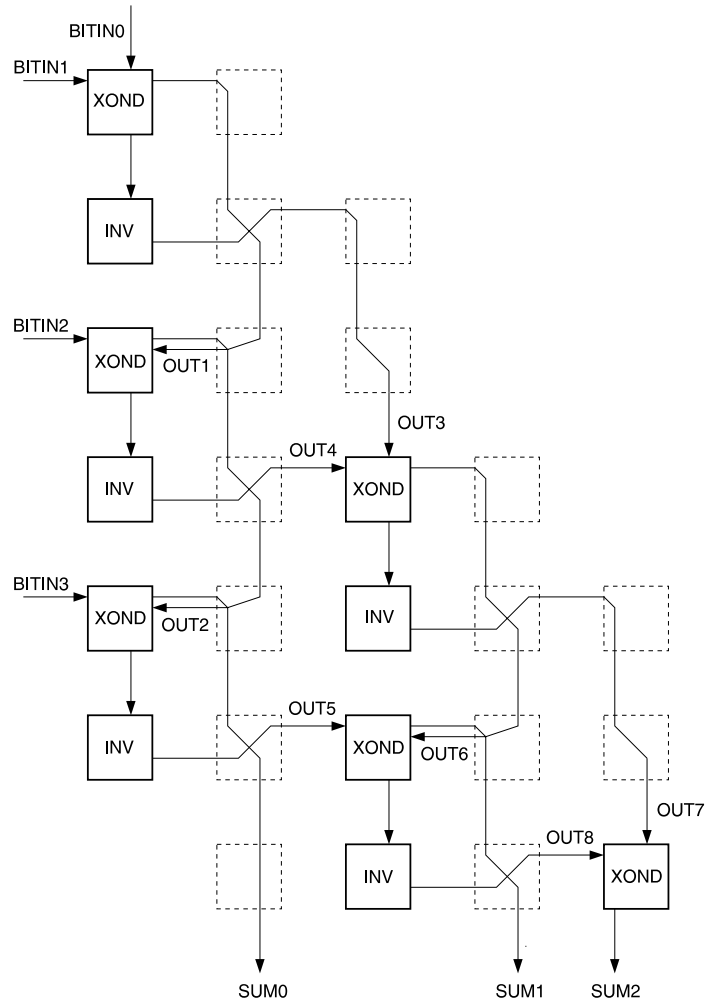


Figure 8. Layout of the 4-bit Summer



Routing Rule 1: Conserve busing resources.

In general, because cells are more plentiful than buses, cells should be used in place of buses for routing signals over short distances. Placing interconnected cells in adjacent locations in the array, or abutting them, avoids routing delays and makes it easy to route the signals together.

To conserve buses, it is sometimes better to use more than the minimum number of cells to implement a function. For example, a two-input OR gate can be performed in one cell, but requires the use of a local bus, as shown in Figure 9. If the local bus is already in use, then it is better to implement the function using three cells and no local bus, as in Figure 10. Using a bus-free macro implementation makes layout more flexible because the macro can be placed and routed with less restriction.

Busing resources are best saved for routing signals across longer distances (more than five cells) in the array, for tri-state capabilities, for signals with high fanout, and for hard-to-route nets.

Routing Rule 2: Align common signals used as inputs.

When a number of signals provide the inputs for many functions, group the signals together and route them in parallel.

Consider implementing the decoder function shown in Figure 11. The decoder implements full or partial product terms:

$$Q1 = S3' \& S2 \& S1 \& S0'$$

$$Q2 = S3 \& S2' \& S1' \& S0'$$

$$Q3 = S3 \& S0'$$

Figure 9. Two-input OR using One Cell and a Local Bus

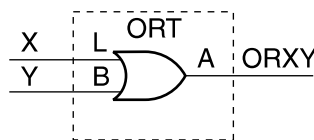
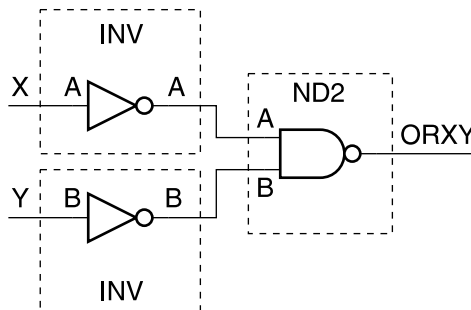


Figure 10. Two-input OR using Three Cells and No Bus



The layout of the decoder is shown in Figure 12. Each cell receives the same input via a local bus, and directs the input to an inverter (INV), a two-input AND gate (AN2), or a two-input gate with one inverted input (AN2L). Because Q1, Q2, and Q3 go through the same number of cells, timing for each signal is regular.

Consider also the two-to-four decoder shown in Figure 13. The logic schematic and physical layout (Figure 14) show this function implemented using four cells and two local buses. As in the product term example above, the two inputs S1 and S0 enter the cells from the local bus, but signals could enter from adjacent cells if the local bus was already being used to route another signal. The outputs exit from the A and B cell outputs.

Routing Rule 3: Use express buses whenever possible.

Because they are not connected directly to cells and thus have lower capacitive loads, express buses are faster than local buses and should be used whenever possible to increase design performance. Also, using an express bus in place of a local bus frees up the local bus for other routing purposes. In some cases, however, substituting an express for a local bus will not be possible:

- when connecting directly to a cell
- when using a bi-directional signal
- when making 90° turns

For increased performance it is best to limit the number of local bus segments carrying a signal and to cross repeater boundaries only if necessary.

Figure 11. Schematic of Decoder Function

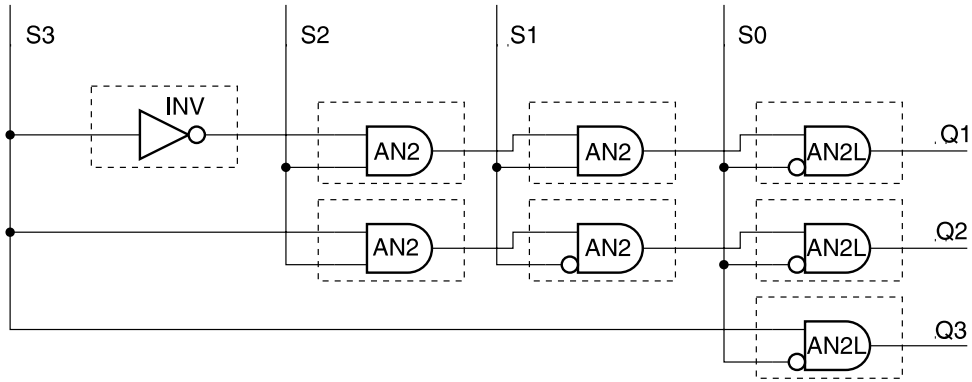


Figure 12. Layout of the Decoder

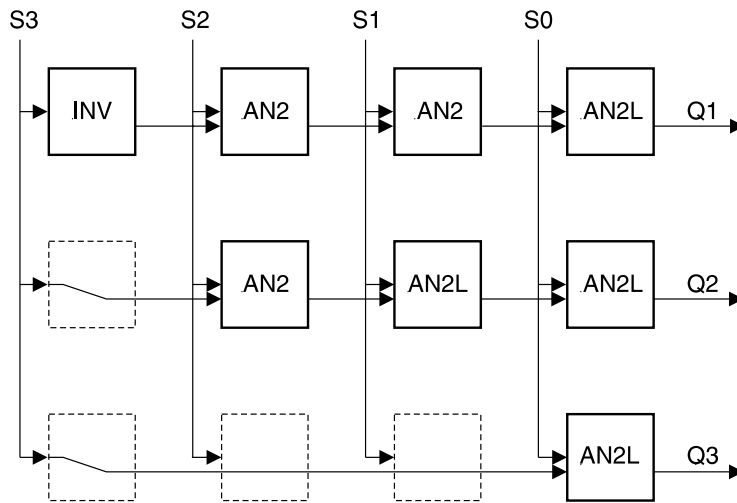


Figure 13. Schematic of Two-to-Four Decoder Function

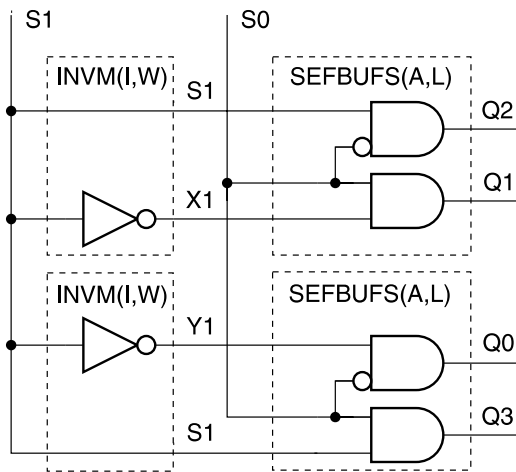
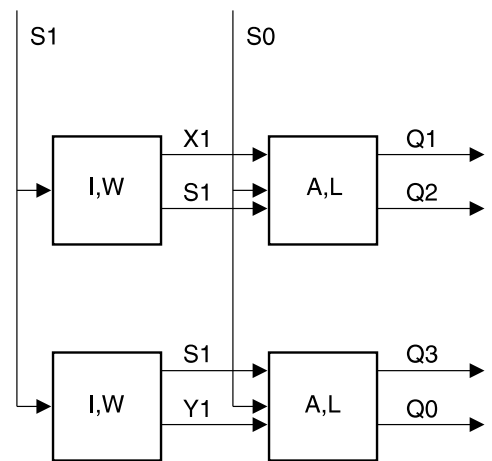


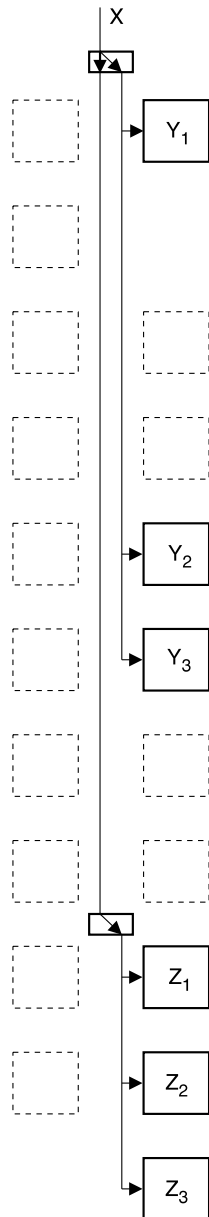
Figure 14. Layout of Two-to-Four Decoder Function



Branching the express bus signal to the local bus at each repeater can be beneficial when the fanout of a signal is greater than eight or the signal goes through more than one repeater. This helps balance the load of each local bus segment, making timing consistent across the length of the array.

Figure 15 illustrates an example of branching. Signal X is routed through the express bus and branches at the repeaters to the local bus segments driving the Y and Z cells. If signal X had been routed via the local bus to cells Y_1 , Y_2 , and Y_3 , and through a repeater (local bus to local bus) to cells Z_1 , Z_2 , and Z_3 , the load of the Y cells would impact the speed of the signal reaching the Z cells.

Figure 15. Branching Signals Increases Performance



Routing Rule 4: Forced redundancy can save routing resources and minimize skew.

If the same function is being used at different locations in the array, two methods can be used to distribute the functions. Figure 16 illustrates the first method. The function can be performed at a single location and the output routed to other locations in the array. With the second method, shown in Figure 17, the inputs of the function can be routed to each location of the array and the function performed at each location. Unless the signals are already accessible throughout the array, this option can require extra routing resources. If routing resources are available, however, the second method may be preferable because it lets you align signal paths in parallel and makes timing more consistent.

Figure 16. Function Outputs are Routed to Other Locations in the Array

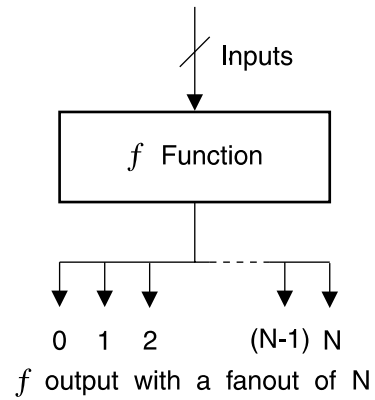
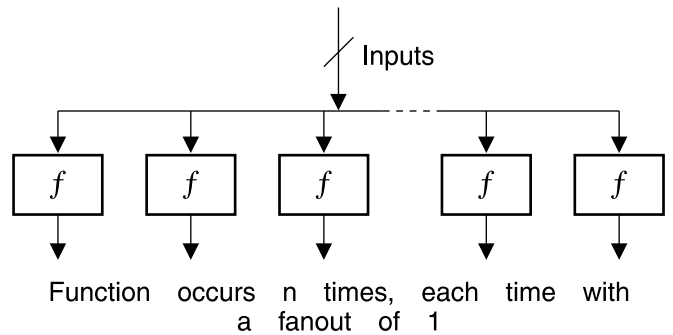


Figure 17. Function is Performed at Different Locations in the Array



Routing Rule 5: Organize registers in columns.

Clocks are distributed along the top edge of the Atmel array and allow each column of logic cells to be clocked individually (see the AT6000 Series data sheet for a detailed description). Grouping cells that use the same clock in a limited number of columns frees up the clocks in the remaining columns. Similarly, registered I/O's should be placed at the sides of the array rather than the top or bottom so the corresponding entrance and exit cells can be controlled by the same column clock.

Although independent clocking increases the flexibility of the AT6000 Series architecture, synchronous designs based on a single global clock yield better results. Asynchronously clocked registers can restrict placement and routing because they make grouping cells together more difficult and, depending on the number of different clocks in a design, can use up limited clock resources too quickly. The global clock is the most efficient clock because it is available to all cells and has low skew.

Use of the asynchronous reset logic along the bottom of the array is similar to that of the column clocks. Place logic requiring the same reset in the same column, and limit the variety of resets in a design.

Schematic Entry Tips

Although it is possible to implement designs directly in the layout using the Interactive Editor, many engineers describe their designs in a schematic and use the resulting netlist for placement and routing. The tips included here describe how to label macros and nets in Viewdraw for use with the Interactive Editor in netlist-driven mode.

Labels applied in the schematic carry over to the “to-be-placed” list of macros in the Interactive Editor. The label from each hierarchical level is included in the list, beginning with the top-level symbol and continuing down the hierarchy to the macro label. For easy viewing in the placement menu, use a short label to identify each instance of a component on the schematic.

Component labels are used in the Interactive Editor for placement. Labels can be entered at the keyboard or selected from a placement menu. The placement menu lists labels in alphabetical order, with the first item preselected. Careful use of labels can organize macros such that functions appear together, making placement go more quickly.

Nets that connect components can also be labeled for easy identification, making it simple to correlate placement and routing with the design schematic. Workview automatically assigns names to unlabeled nets (see Workview manuals for details), so it is best to create meaningful labels as you create the schematic.

Making Your Own Components

ViewLogic lets you create components by defining symbols to represent underlying schematics. The elements of the ViewLogic component appear in the Interactive Editor's placement list under a common root name. The underlying elements can be selected from the list and placed like any other Atmel macro.

The Design Manager automatically creates a symbol for the Atmel FPGA using the 132-pin PQFP package by default. This symbol can be modified to specify I/O connections, or replaced with any of the other package symbols.



Atmel Headquarters

Corporate Headquarters

2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe

Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia

Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan

Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs

1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset

Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand

North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

BBS

1-(408) 436-4309

© Atmel Corporation 1999.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Marks bearing ® and/or ™ are registered trademarks and trademarks of Atmel Corporation.

Terms and product names in this document may be trademarks of others.



Printed on recycled paper.

0460C-09/99/xM