Pin Preassigning with XC9500 CPLDs

XAPP074 June, 1998 (Version 1.3) Application Note

Summary

This application note describes the planning required for successful pin preassigning and gives a detailed example.

Xilinx Family

XC9500

Introduction

Reducing time to market is critical in today's highly competitive marketplace, and designers often need to prototype their products as swiftly as possible. Because PC board production is often the slowest part of the development process, it is often advantageous to begin PC board layout before the CPLD designs are complete. This requires designers to preassign device pins.

The flexibility of the XC9500 architecture permits pin preassignments, and a high degree of success can be achieved by following a few simple guidelines. The architecture also allows future edits to the design while maintaining the existing pinout. However, some design planning is necessary.

Many designs can be partitioned into datapath and control logic portions. Datapath operations are regular and bit-oriented, maintaining the relative positions of the bits. Datapath operations tend to retain their functionality after they are designed.

Control operations change as timing is adjusted during the design cycle while specifications are altered. These operations evolve as a design progresses. Control operations are much less structured than data operations and therefore the design strategy is different.

Obtaining a reasonable estimate of the design's logic requirements for both control and datapath operations is the key to preassigning pinouts. Once an estimate is obtained, pins are assigned with a high degree of confidence that the function block driving the pins will have the available resources to connect the logic needs of the corresponding macrocells. Because the design needs for datapath and control operations are different, the process of estimating their initial requirements will also differ.

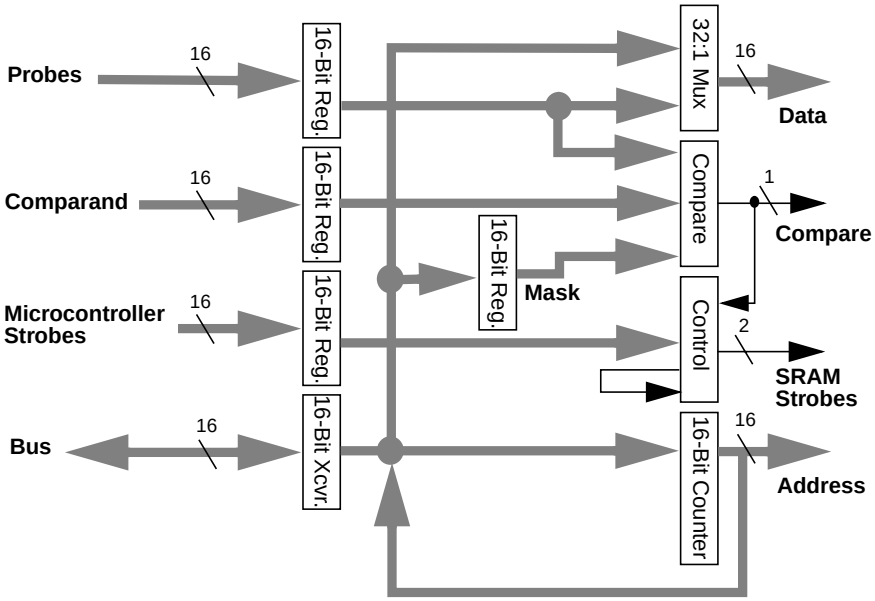

Figure 1: Pin Preassignment Design Example

XC9500 Routing Resources

XC9500 CPLDs combine a locally efficient logic block with a globally flexible interconnect structure to provide ideal connectability for a very large spectrum of designs. The key qualities of the logic block are:

- 36 input signals presented to the logic block.
- Automatic allocation of product terms as needed within the function block. The average is 5 product terms per macrocell, but up to 15 are easily obtained and up to 90 can be used when needed.
- Formation of efficient counters, multiplexers, shifters, and parity circuits with an efficiency of one macrocell or less per bit. The remaining logic is available for use by other functions.

The key properties of the interconnect structure are:

- Any input pin connects to any function block with constant high speed across the entire device.
- Any macrocell output can connect to its own or any other function block with no restriction.
- Macrocells can be internally bused with bit level independent 3-state control, to form internal data buses with global access to all logic blocks. (No other CPLD architecture offers this capability, which saves macrocell logic by using the routing resources to form multiplexers.)

These key features allow significant design changes to be made within CPLDs that are already attached to PC boards.

Datapath Estimation Guidelines

Datapath operations include storage, shifting, multiplexing, arithmetic, comparison, and boolean operations. For most designs, estimating the logic needs for a single bit is sufficient to quickly estimate the entire datapath. Table 1 gives the macrocell needs of various data operations per bit.

Table 1: Macrocell/Product Term Allocation

Data Operation	P-Terms Used
Shift Register	2
Counters	2-4
n:1 Mux	n
Adder	6
EX-OR	2
Storage registers	1

The Xilinx fitter software easily borrows the necessary product terms for denser functions from neighboring macrocells that have spare resources. As shown in Table 1, few functions require more than 4 product terms per macrocell. This suggests that all functions will easily fit into incremental macrocell units, except for the adder, which requires more than one macrocell, and the n:1 MUX which may need more product terms than a single macrocell can supply.

Control Path Estimation Guidelines

Control logic is more complicated to estimate than datapath logic because the logic needs vary and the timing is usually critical. Simple combinatorial control signals are easy to estimate, but state machines are not. However, one-hot encoded state machines, which are flip-flop rich with minimal input logic, may be used for estimation purposes.

One-hot state machine encoding works well for estimating pinouts because:

- No sophisticated state encoding is required.
- The input signals can be managed in a straightforward way.
- Performance degradation can be quantified at the initial estimation time.
- One-hot state machines are easily edited to add or delete states without upsetting the state encoding.
- One-hot solutions give a worst-case estimation, though encoded state machines usually produce logic that is more compact with no speed degradation (in CPLDs).

Editing one-hot machines without significantly altering the overall structure of the machine is easy. States may be inserted or deleted without major impact on the next state or output logic encoding. One-hot designs can also be implemented in either the Mealy or Moore versions as required. Figure 2 shows the basic one-hot encoding structure. The darker region of the figure shows additional logic added to alter the four state machine (A,B,C,D) to a five-state machine (A,B,C,D,E) without completely re-encoding.

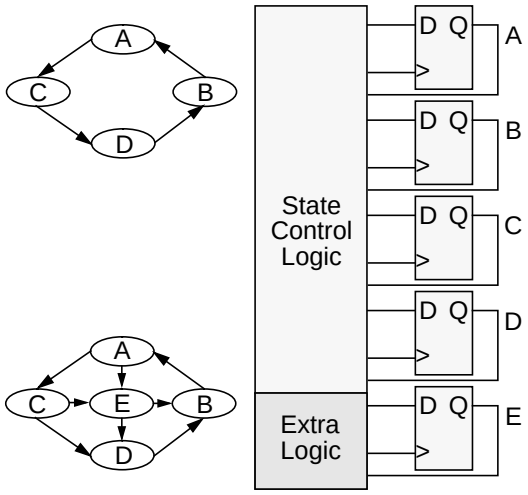

Figure 2: One-Hot Encoded State Machine Editing

Pin Preassigning Guidelines

The easiest and most successful method for preassigning pins is to do an estimated design and let the fitter software assign the pins. However, the following guidelines will help if a manual estimate is required:

- Partition the data and control sections, and estimate the needs of each section.
- Consider One-hot control unit designs to permit future editing and stable pin assignments.
- Use the following cell arrangement techniques:
  - Spread the outputs among all Function Blocks.
  - Within Function Blocks, spread macrocells evenly, with unused macrocells interleaved.
  - Assign outputs using common inputs to the same function blocks.
  - Don't use global pins (clocks, resets, 3-state, enables) for logic inputs or outputs unless the global pins are not needed.
- XC9500 CPLDs have common footprint packages for various density parts. By designing for the smallest capacity in a specific package footprint, there is always the option of a pin compatible density upgrade that matches the existing pinout. See Table 2 for package availability.
- Interleave output functions that need many product terms with functions that need few.

Manual Estimation Example

Figure 1 shows a block diagram of a proposed system that needs pinout assignments for early PCB delivery. The first step is to make a rough estimate of the required device capacity in order to target a particular XC9500 CPLD.

In this design, there are:

- 4 — 16-bit registers using 64 macrocells.
- 1 — 32:16 bit mux using 16 macrocells.
- 1 — 16 bit loadable counter using 16 macrocells.
- 1 — 16-bit transceiver using 16 pins (no macrocells).
- 1 — 16-bit maskable comparator using X macrocells.
- 1 — Control unit using Y macrocells.

To complete this estimate, more detail is required. First, the 16-bit transceiver will need 16 bi-directional pins and a global 3-state direction control signal. This partition will combine with the 32:16 mux and the 16-bit counter. One bit of this is shown in Figure 3.

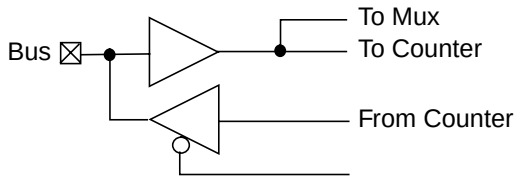

Figure 3: Transceiver Logic

The comparator will be created as an expansion of the diagram shown in Figure 4.

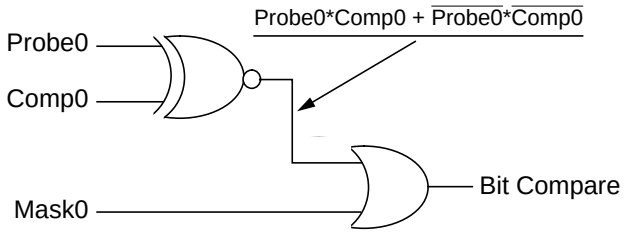

Figure 4: Bit Compare Logic

16-bit compares must be ANDed to create the overall signal "compare". Notice that each bit compare is easily computed with one bit per macrocell, but a total of 48 inputs are required. This exceeds the number of available inputs to an XC9500 function block. Also, 16 intermediate bit compares must be ANDed to permit ANDing a large number of such signals. Depending on the device, 12 bits may be assigned to a single FB using all FB inputs and still require making 4 more bit compares. This approach leaves 6 unusable macrocells remaining in an FB. A better approach is to assign 8 bit compares to an FB using 24 inputs with 12 inputs to support the remaining 10 macrocells. This is a natural way to interleave sparse resource macrocells (registers) with heavier resource macrocells (comparators). See Figure 5.

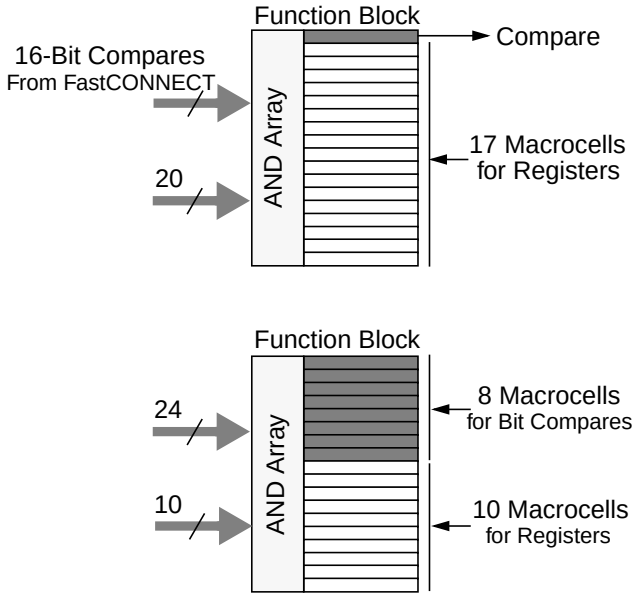

Figure 5: Possible Arrangement for Register and Compare Bits

Another version that may be more flexible is to interleave the registers with the compares in case additional variety (creeping elegance) may occur with the comparator. This approach makes single cascade groups of four product terms additionally available at each bit compare site. Com-

parator tally equals 17 macrocells. The 16-bit counter is loadable and synchronous. It needs 16 data inputs, 16 state feedbacks, and a load/count signal. This leaves only 3 available FB inputs, but clusters all macrocells within a single FB for faster speed.

The control unit will have a loadable counter used as an internal event timer. It will also require a simple state machine with about 4 states. Figure 6 estimates the Control Unit.

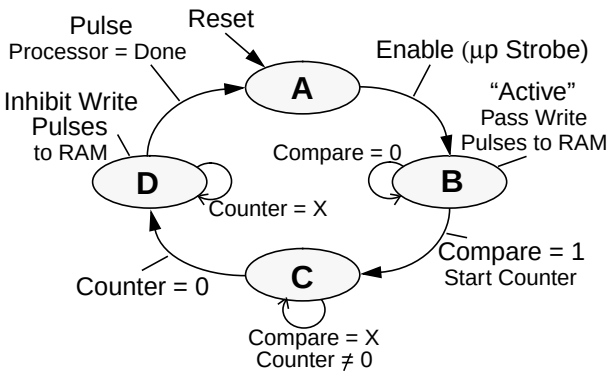

Figure 6: Control Unit State Machine

The loadable counter requires 33 inputs and 16 macrocells, so it makes sense to interleave this module among the registers which require few inputs and p-terms. As a one-hot state machine, the control unit state machine, with no simplification, requires 4 flip-flops and 4 inputs. Figure 7 gives an approximation of the design.

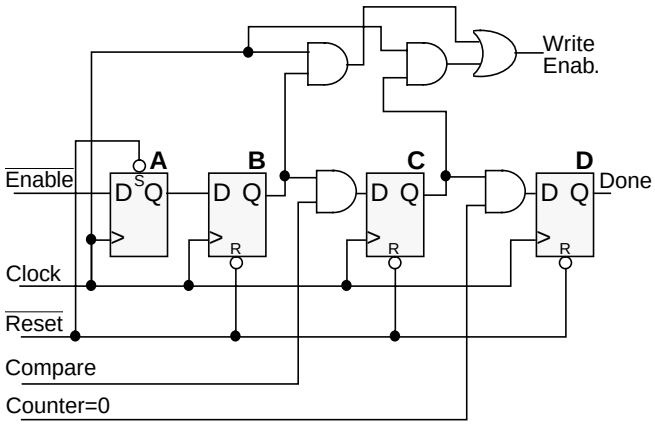

Figure 7: One-Hot State Machine Schematic

The tally can now be completed by adding in the maskable comparator which uses 17 macrocells and the control unit which uses 20 macrocells. The total is 96 + 37 = 133 macrocells. The smallest XC9500 device that can contain this is the XC95144. The I/O pins required are 108. Figure 8 suggests a reasonable pin assignment by associating macrocells to explicit pins in the PQ160 package.

Table 2: XC9500 Device Package Availability

Pins	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288
44	X	X				
48	X					
84		X	X			
100		X	X	X		
160			X	X	X	
208					X	X
352					X	X

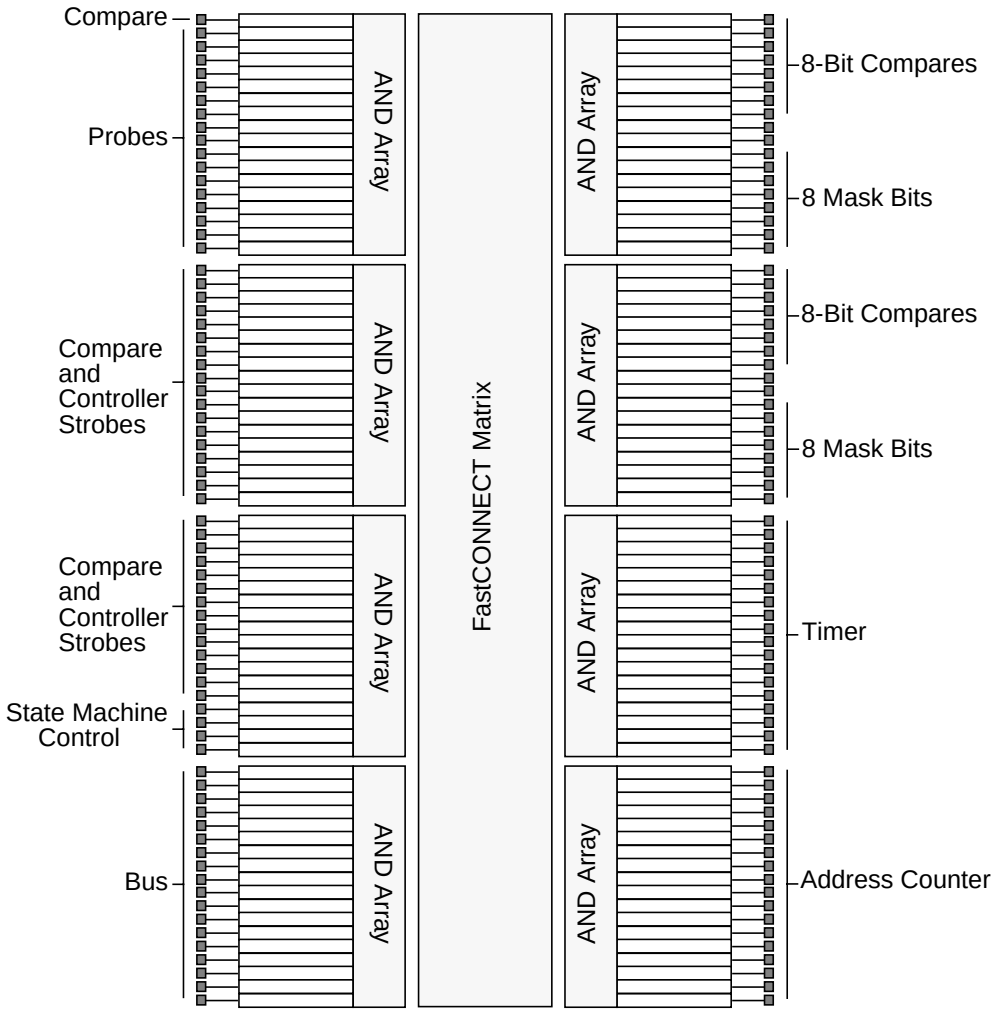

Figure 8: Pin Preassign Strategy for the XC95144

Conclusion

Pin preassigning is successfully achieved with XC9500 CPLDs by following simple guidelines. It is always recommended to do an estimated design and let the design software assign pins before committing to a printed circuit board.

The benefits of preassignment include faster time to prototype and ultimately faster time to market. Xilinx XC9500 CPLDs provide the speed and flexibility to make this goal a reality.