



## XC6200 Field Programmable Gate Arrays

---

### *Table Of Contents*

Features	2
Description	3
Architecture	3
Logical and Physical Organization	3
Additional Routing Resources	6
Magic Wires	6
Global Wires	6
Function Unit	6
Cell Logic Functions	7
Routing Switches	8
Clock Distribution	9
Clear Distribution	10
I/O Architecture	10
Pull-Up, Pull-Down And Slew	13
Border Routing	13
GCLK, OE And Reset Routing	13
Designing with XC6200	14
Register Access	16
Map Register	17
Mask Register	19
Programming	19
FastMAP™ Parallel CPU Interface	19
Understanding The Configuration Bits	20
Wildcard Registers	24
Device Configuration Register	25
Device Identification Register	26
Control Register Memory Map	26
Serial Programming Interface	30
Reset And Initialization	34
Packaging	35
Pin Descriptions	35
Electrical Parameters	36
Timing Diagrams	44
XC6216 Pinouts	45
Ordering Information	53



# XC6200 Field Programmable Gate Arrays

January 9, 1997 (Version 1.8)

Advance Product Information

## Features

- High-Performance Sea-Of-Gates FPGA
  - Thousands of configurable cells
  - Fine-grain architecture, abundant registers, gates and routing resources
  - Extremely high gate count for structured logic or datapath designs
  - High-speed SRAM control store
  - 220 MHz flip-flop toggle rates
- Extremely Flexible Cell Architecture
  - Over 50 distinct logic functions per cell
  - One register and gate/multiplexer possible for every cell
- Advanced Processor Compatible Architecture
  - Xilinx FASTmap™ processor interface
  - Direct processor read/write access to *all* internal registers in user design with no logic overhead
  - All user registers and SRAM control store memory mapped onto processor address space
  - Programmable data bus width (8, 16 or 32-bits)
  - Easily interfaced to most microcontrollers and microprocessors
- Advanced Dynamic Reconfiguration Capability
  - High-speed reconfiguration via parallel CPU interface
  - Full or partial reconfiguration/context switching possible
  - Unlimited reprogrammability
  - Ideal for custom computing applications
- Flexible Pin Configuration
  - All User I/Os programmable as in, out, bidirect, three-state or open drain.
  - Configurable pull-up/down resistors
  - CMOS or TTL logic levels
- Flexible Interconnect Architecture
  - Low-delay FASTlane™ hierarchical routing scheme gives large number of fast 'Longlines'
  - Any cell can be connected to any other
  - Suited to both structured synchronous data path type designs or irregular random logic
  - Completely flexible clocks and asynchronous clears for registers
  - 4 Global low-skew signals
- Testability
  - Pre-tested, high-volume, standard part
  - JTAG capability with library macrocells
- Sophisticated CAD Tools
  - Implement designs using familiar tools like Viewlogic and Synopsys
  - Dedicated XACTstep Series 6000 back-end tools
  - Use PC or Unix workstation platforms
  - Fully automatic mapping, placement and routing
  - Interactive Physical Editor for design optimization
  - Large Xilinx parts library for schematic capture
  - VHDL synthesis

**Table 1: The XC6200 Family of Field Programmable Gate Arrays**

Device	XC6209 <sup>†</sup>	XC6216	XC6236 <sup>†</sup>	XC6264 <sup>†</sup>
Typical Gate Count Range	9000-13000	16000-24000	36000-55000	64000-100000
Number of Cells	2304	4096	9216	16384
Number of Registers	2304	4096	9216	16384
Number of IOBs	192	256	384	512
Cell Rows x Columns	48x48	64x64	96x96	128x128

<sup>†</sup> = Planned Product

## Description

The XC6200 family is a new type of high performance FPGA from Xilinx.

XC6200 is a family of fine-grain, sea-of-gates FPGAs. These devices are designed to operate in close co-operation with a microprocessor or microcontroller to provide an implementation of functions normally placed on an ASIC. These include interfaces to external hardware and peripherals, glue logic and custom coprocessors, including bit-level and systolic operations unsuited to standard processors.

The XC6200 can provide extremely high gate counts for data path or regular array type designs. In these cases the actual gate count may turn out to be a factor of two or more greater than those given in [Table 1](#).

An XC6200 part is composed of a large array of simple, configurable cells. Each basic cell contains a computation unit capable of simultaneously implementing one of a set of logic level functions *and* a routing area through which inter-cell communication can take place. The structure is simple, symmetrical, hierarchical and regular, allowing novice users to quickly make efficient use of the resources available.

The nearest-neighbor interconnect of the underlying cells is supplemented with wires of length 4 cells, 16 cells and Chip-Length, which provide low delay paths for longer connections. In addition there are four global input signals which provide a low skew distribution path for critical high fan-out nets such as clocks and initialization signals.

XC6200 parts are configured by an integral, highly stable six-transistor SRAM control store. This allows XC6200 parts to be quickly reconfigured an unlimited number of times. The SRAM control store can be mapped into the address space of a host processor and additional support logic is provided to allow rapid reconfiguration of all or part of the device. In addition, the outputs of function units within the device can be read by a processor through the FASTmap™. Processors can read or write registers within logic implemented on the device. Data transfers can be 8, 16 or 32 bits wide, even when register bits are distributed over a column of cells. These capabilities allow XC6200 FPGAs to support virtual hardware in which circuits running on the FPGA can be saved ('swapped out') to allow the FPGA resources to be assigned to a different task, then restored ('swapped in') at a later time with the same internal state in their registers. Sections of the device can be reconfigured without disturbing circuits running in other sections. Thus an XC6200 in a coprocessor application can be shared by several processes running on the host computer.

Design entry and proving may be carried out with Xilinx software products using industry standard schematic

capture, synthesis and simulation packages such as Viewlogic, Mentor Graphics and Synopsys. A comprehensive library of parts, ranging from simple gate primitives to complex macro-functions, exists to make this an easy task.

Below the top level design tools, the XC6200 product family is supported by XACTstep Series 6000. This contains tools ranging from simple symbolic editors for high-efficiency user designs to sophisticated cell-compilation tools. These tools help to ensure the design captured is laid out efficiently with no user intervention. Node delays can then be back-annotated to the front-end logic simulator for design proving. The tools allow for manual intervention in the layout process if desired. Incremental design is also supported: if a design is laid out and subsequently changed, only the modified block has to be re-laid out.

The functions available within each cell provide a good target for logic synthesis programs. The simple cell architecture allows arbitrary user logic designs to be mapped onto a number of cells, rather than having to split the design up into medium-complexity mini-functions for mapping to a larger configurable logic block. Because each cell can be configured as a register, designs containing far more registers than would be possible with a larger configurable block are achievable.

## Architecture

### Logical and Physical Organization

The XC6200 architecture may be viewed as a hierarchy. At the lowest level of the hierarchy lies a large array of simple cells ([Figure 1](#)). This is the 'sea of gates'. Each cell is individually programmable to implement a D-type register and a logic function such as a multiplexer or gate. Any cell may also be configured to implement a purely combinatorial function, with no register. This is illustrated in [Figure 7](#).

First generation fine-grain architectures implemented only nearest-neighbor interconnection and had no hierarchical routing ([Figure 1](#)). XC6200 is a second generation fine-grain architecture, employing a hierarchical cellular array structure. Neighbor connected cells are grouped into blocks of 4x4 cells ([Figure 2](#)) which themselves form a cellular array, communicating with neighboring 4x4 cell blocks. A 4x4 array of these 4x4 blocks forms a 16x16 block ([Figure 3](#)). In the XC6216 part, a 4x4 array of these 16x16 blocks forms the central 64x64 cell array which is then surrounded by I/O pads ([Figure 4](#)).

Each level of hierarchy (unit cells, 4x4 cell blocks, 16x16 cell blocks, 64x64, etc.) has its own associated routing resources. Basic cells can route across themselves to connect to their nearest neighbors and thus provide wires

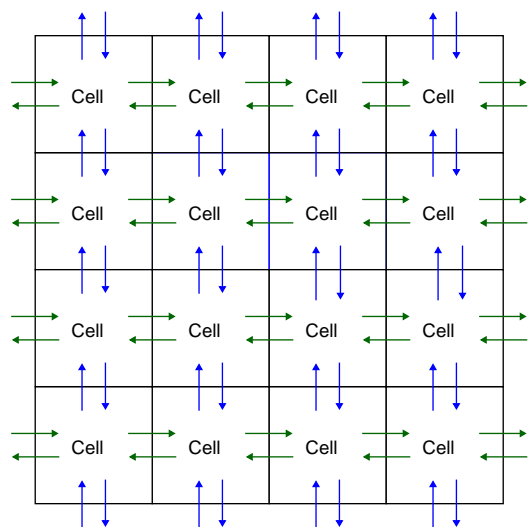


Figure 1. Nearest-Neighbor Interconnect Array Structure

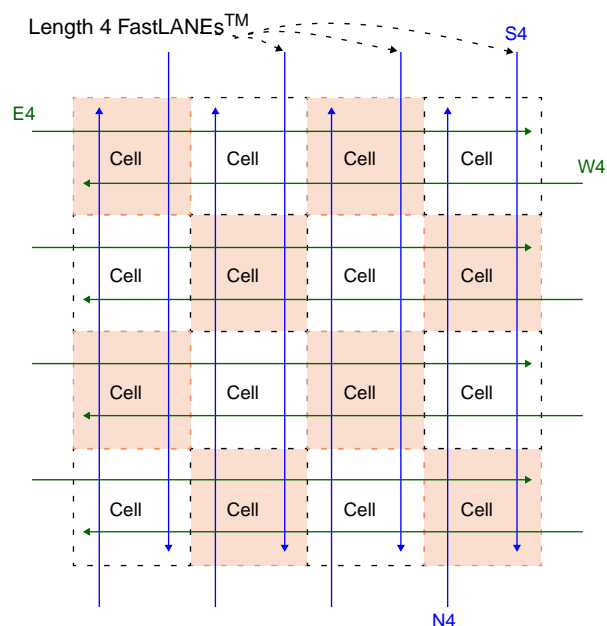


Figure 2. XC6200 4x4 Cell Block

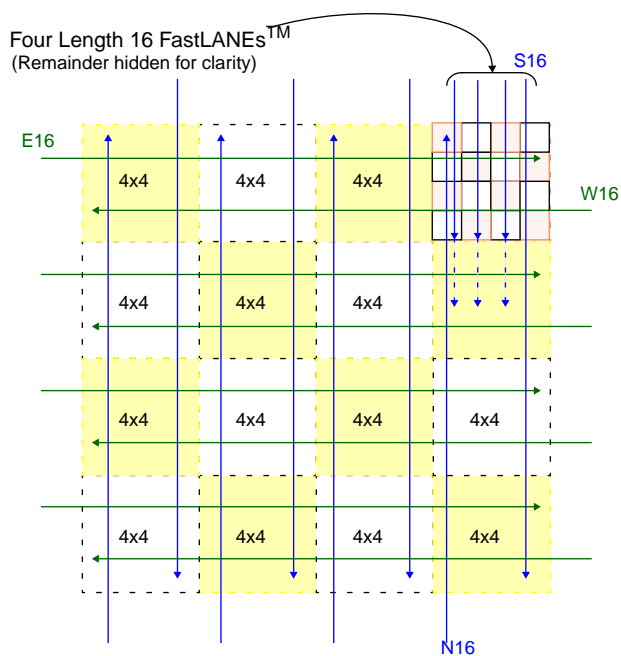


Figure 3. XC6200 16x16 Cell Block

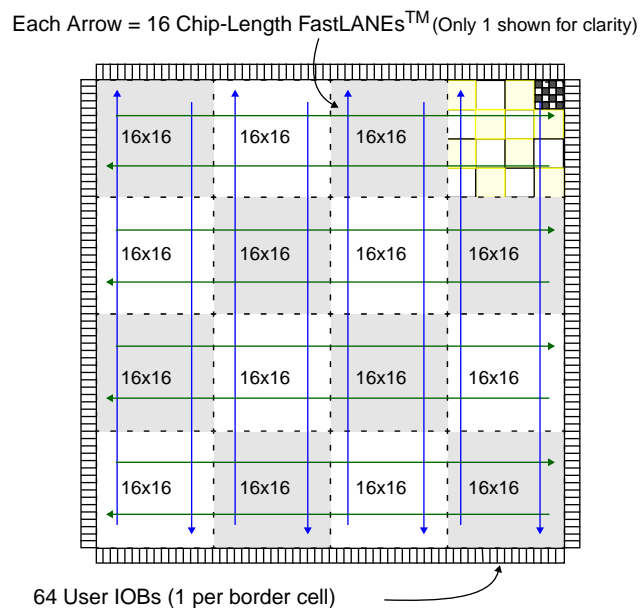


Figure 4. XC6216 Device

of length 1 cell. Note that cells used for interconnect in this manner can still be used to provide a logic function. Wires of length four cells are provided to allow 4x4 cell blocks to route across themselves without using unit cell resources. Similarly 16x16 cell blocks provide additional wires of length 16 cells and the 64x64 array provides Chip-Length wires. Larger XC6200 products extend this process to 256x256 cell blocks and so on, scaling by a factor of 4 at each hierarchical level as required. Intermediate array sizes (e.g. 96x96) are created by adding more 16x16 blocks. Switches at the edge of the blocks provide for connections between the various levels of interconnect at the same position in the array (e.g. connecting length 4 wires to neighbor wires).

The longer wires provided at each hierarchical level are termed 'FastLANEs™'. It is convenient to visualize the structure in three dimensions with routing at each hierarchical level being conceptually above that in lower hierarchical levels, with the cellular array as the base layer. The length-4 FastLANEs™ are driven by special routing multiplexers within the cells at 4x4 block boundaries. All routing wires are directional. They are always labeled according to the signal travel direction. For example, S4 is a length-4

FastLANE™ heading from North to South. In **Figures 2, 3 and 4** each individual cell has a length 4, 16 and Chip-Length FastLANE™ above it. However only a small number are shown for clarity.

The benefit of the additional wiring resources provided at each level of the hierarchy is that wiring delays in the XC6200 architecture scale logarithmically with distance in cell units rather than linearly as is the case with the first generation neighbor interconnect architectures. Since 4x4 cell block boundaries lie on unit cell boundaries, the switching function provided at 4x4 cell boundaries is a superset of that provided at unit cell boundaries; i.e. it provides for neighbor interconnect between the adjacent cells as well as additional switching options using the length 4 wires. Similarly, the switching unit on 16x16 cell block boundaries provides a superset of the permutations available from that on the 4x4 cell block boundaries. Further switching units are also provided on the 64x64 cell boundaries to provide the Chip-Length FastLANEs™.

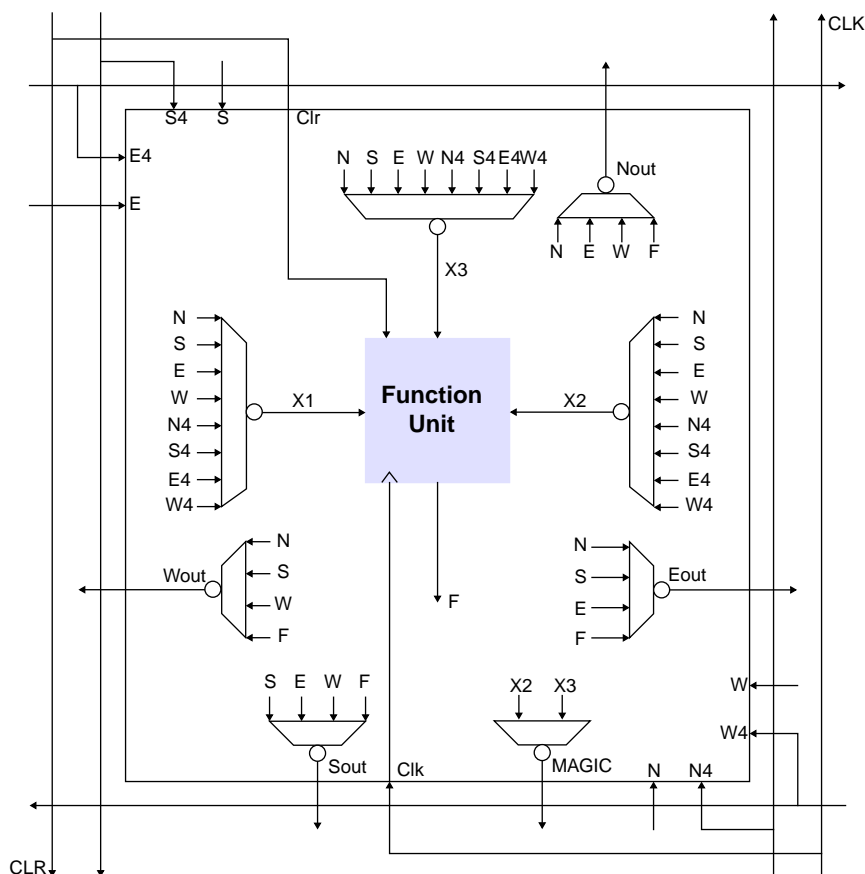


Figure 5. XC6200 Basic Cell

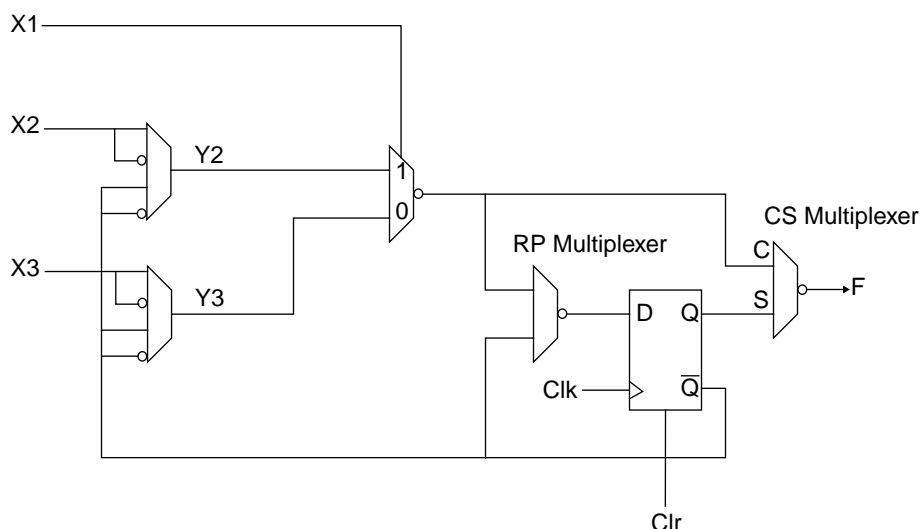


Figure 6. XC6200 Function Unit

## Additional Routing Resources

### Magic Wires

The majority of interconnections are routed using the nearest-neighbor and FastLANEs™ described above. Each cell has a further output (labeled 'Magic') which provides an additional routing resource. A cell's Magic output is not always available for routing. Its availability is dependent on the logic function implemented inside the cell. More information on the physical nature of the Magic wires is given in the section "Function Unit" on page 6.

Each cell's Magic output is routed to two distinct 4x4 block boundary switches. The Magic wire can be driven by N, S, E or W from adjacent cells or from the N4, S4, E4 or W4 FastLANEs™ passing over the cell. This makes it particularly useful for corner-turning (all other routing resources are straight).

The Magic wires are illustrated in Figure 8.

### Global Wires

The XC6200 architecture permits registers within a user design to be clocked by different clocks and cleared by different asynchronous clears. Clocks and Clears may be provided by any user I/O pin or generated from user logic internally. In line with good synchronous digital design practices, it is recommended that a single global Clock and Clear are used. This minimizes the likelihood of timing problems and gives more reliable simulations.

Four Global wires (G1, G2, GClock and GClear) are provided for low skew, low delay signals. These wires are intended for global Clock and Clear or other high fan-out signals

and are distributed throughout the array in a low skew pattern. A global signal can reach the clock and clear inputs of any cell on the array passing through very few routing switches. The four Globals are very similar. It would be possible to use GClock as a global Clear signal, however for minimum delay, it is recommended that GClock be used for global clocks and GClear for global clears. GClock and GClear can reach the inputs of any register in the array, passing through only a single routing switch. G1 and G2 may be used for secondary global clocks or clears. G1 and G2 have a slightly larger delay than GClock and GClear.

### Function Unit

Figure 5 shows the basic XC6200 cell in detail. The inputs from neighboring cells are labeled N, S, E, W and those from length 4 wires N4, S4, E4, W4 according to their signal direction. Additional inputs include Clock and Asynchronous Clear for the Function Unit D-type register. The output from the cell function unit, which implements the gates and registers required by the user's design, is labeled F. The Magic output is used for routing as described earlier. The multiplexers within the cell are controlled by bits within the configuration memory. As can be seen from Figure 5, the basic cells in the array have inputs from the length 4 wires associated with 4x4 cell blocks as well as their nearest neighbor cells. The function unit design allows the cells to efficiently support D-type registers with Asynchronous Clear and 2:1 multiplexers, as well as all Boolean functions of two variables (A and B) chosen from the inputs to the cell (N,S,E,W,N4,S4,E4,W4) (Table 2). Figure 7 shows the schematic representations of the basic cell functions pos-

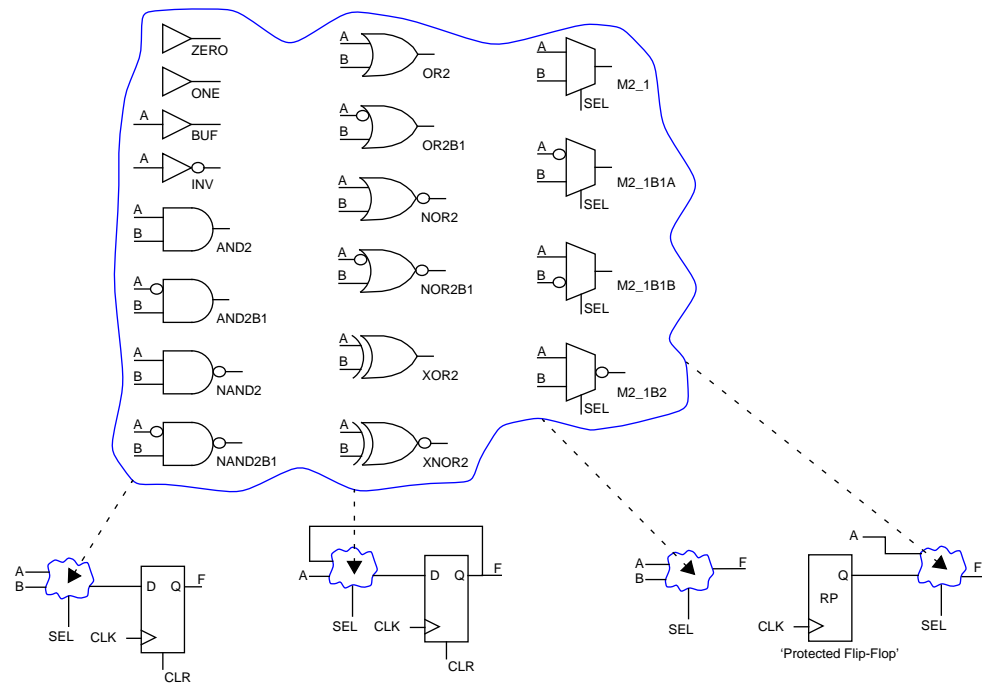


Figure 7. Cell Logic Functions

sible. The Magic routing output can only be used if the signal to be routed can be placed on X2 or X3.

Figure 6 shows the implementation of the XC6200 function unit. The design uses the fact that any function of two Boolean variables can be computed by a 2:1 multiplexer if suitable values chosen from the input variables and their complements are placed on its inputs. The Y2 and Y3 multiplexers provide for this conditional inversion of the inputs. The CS multiplexer selects a combinational or sequential output. The RP multiplexer allows the contents of the register to be 'protected'. If register protection is enabled then only the programming interface can write to the register. It does not change when the X inputs to the function unit change, **even if it is clocked or cleared**. This feature is useful in designs containing control registers which are only to be written by an external microprocessor. The control inputs of all the multiplexers, except the one switched by X1, come from configuration memory bits.

## Cell Logic Functions

Each cell can be configured as any two-input gate function, any flavour of 2:1 multiplexer, constant 0 or 1, single input functions (buffer or inverter) or any of these in addition to a D-type register. This is illustrated in Figure 7. The gate names given correspond to standard Xilinx library part names for these primitives. Although three inputs are shown entering the combinational 'cloud', dual and single input functions are also possible. e.g. inverter

+ register or register alone. The buffer symbol is available in the CAD libraries. There is no requirement for the designer to buffer signals with this architecture. This is because signals are regularly buffered by routing multiplexers. Symmetrical functions are also possible but not shown in Figure 7. e.g.  $\bar{A}.B$  (AND2B1) is shown but  $A.\bar{B}$  (AND2B2) is not. This is because A and B are assigned to user signals by the logic mapping software to provide the required function. Thus a multiplexer with inversion on the SEL input is unnecessary because the mapping software can simply swap the signal assignments for A and B.

The sources of the X1, X2 and X3 input multiplexers are set automatically by CAD software during the logic mapping phase. Table 2 shows the assignments for all the cell multiplexers to compute the various logic gate functions. A NAND2B1 is equivalent to an OR2B1 with the inputs swapped and a NOR2B1 is equivalent to an AND2B1 with the inputs swapped therefore these gates are not listed in Table 2. The C and S signals are taken as the 'true points' for the gate mappings in this table.

If the register within a cell is not used in the design then a special 'fast' version of most gates can be configured, using the register to provide a constant 1 or 0. For example a fast AND gate ( $A.B$ ) can be configured by setting the register to 0 during configuration and assigning  $\bar{Q}$  to Y3. A is routed to X1 and B to X2.  $\bar{X}2$  is assigned to Y2. When A changes to 0, Y3 is selected and F is forced Low as soon as the X1-controlled multiplexer



switches. In the normal AND gate, there would be an additional delay as  $A$  propagated through the Y3 multiplexer. Fast or normal gates may be specified by the designer but for optimal layout density this is best left to the logic mapping software.

The multiplexer functions have a straightforward mapping with fixed assignments to X1,X2 and X3, with Y2 and Y3 providing input inversions as required.

## Routing Switches

As described earlier, each cell within a 4x4 block is able to drive its output to its nearest neighbors to the N,S,E and W. In addition to this, cells at 4x4 block boundaries are also able to drive their outputs onto length-4 Fast-LANEs<sup>TM</sup>. Special switch units are provided around each 4x4 block boundary to facilitate these connections. This is

illustrated in [Figure 8](#). These switches also allow higher levels of hierarchical routing (e.g. length 16 and Chip-Length FastLANEs<sup>TM</sup>) to be connected to length-4 Fast-LANEs<sup>TM</sup>.

[Figure 8](#) also shows the connections for each cell's Magic output. Each Magic output is routed to two destinations for increased routing flexibility. The two connections are labeled M and MA. The Magic wires allow cell outputs to jump to the edge of the 4x4 block and hence onto Fast-LANEs<sup>TM</sup> or into the next 4x4 block. They are also a particularly efficient way of making large busses turn corners.

N,S,E and W switches are similar, however the N switches contain additional multiplexers to drive the register Clock lines. The contents of the boundary switches are shown in [Figures 9 to 12](#). These multiplexers

**Table 2: Function Derivation**

Function	X1	X2	X3	Y2	Y3	RP	CS	Q
0	A	A	A	X2	$\overline{X3}$	X	C	X
1	A	A	A	$\overline{X2}$	X3	X	C	X
BUF (Fast)	A	X	X	Q	$\overline{Q}$	Q	C	0
BUF	X	A	A	$\overline{X2}$	$\overline{X3}$	X	C	X
INV (Fast)	A	X	X	$\overline{Q}$	Q	Q	C	0
INV	X	A	A	X2	X3	X	C	X
A.B (Fast)	A	B	X	$\overline{X2}$	$\overline{Q}$	Q	C	0
A.B	A	B	A	$\overline{X2}$	$\overline{X3}$	X	C	X
$\overline{A}.B$ (Fast)	A	X	B	$\overline{Q}$	$\overline{X3}$	Q	C	0
$\overline{A}.B$	A	A	B	X2	$\overline{X3}$	X	C	X
$\overline{A}.\overline{B}$ (Fast)	A	B	X	X2	Q	Q	C	0
$\overline{A}.\overline{B}$	A	B	A	X2	X3	X	C	X
A+B (Fast)	A	X	B	Q	$\overline{X3}$	Q	C	0
A+B	A	A	B	$\overline{X2}$	$\overline{X3}$	X	C	X
$\overline{A}+B$ (Fast)	A	B	X	$\overline{X2}$	Q	Q	C	0
$\overline{A}+B$	A	B	A	$\overline{X2}$	X3	X	C	X
$\overline{A}+\overline{B}$ (Fast)	A	X	B	$\overline{Q}$	X3	Q	C	0
$\overline{A}+\overline{B}$	A	A	B	X2	X3	X	C	X
$A\oplus B$	A	B	B	X2	$\overline{X3}$	X	C	X
$\overline{A}\oplus\overline{B}$	A	B	B	$\overline{X2}$	X3	X	C	X
M2_1	SEL	A	B	$\overline{X2}$	$\overline{X3}$	X	C	X
M2_1B1A	SEL	A	B	X2	$\overline{X3}$	X	C	X
M2_1B1B	SEL	A	B	$\overline{X2}$	X3	X	C	X
M2_1B2	SEL	A	B	X2	X3	X	C	X



invert some inputs. This is not shown in the figures. See the **"Programming"** section for details.

The multiplexers driving the NOut, SOut, EOut and WOut lines are actually implemented within the cell adjacent to the switch. These multiplexers take the place of the neighbor multiplexers found in the basic cell (see **Figure 5**). Boundary cells contain additional RAM bits to control the larger multiplexers. An additional output is available from these multiplexers. This output reflects the output that would have come from the cell's neighbor multiplexer had it been a basic non-boundary cell. To distinguish this from the output of the boundary switch (NOut, SOut, EOut or WOut), it is suffixed with a 'C' (Cell). e.g. NC for an Nswitch. NC is one of NIn, E, W, or F depending on the least-significant two bits of the NOut multiplexer select lines. Hence NC is identical to NOut if NOut is one of F, NIn, E or W. If NOut is one of N4In, N16, PS4 or MN then NC is one of F, NIn, E or W depending on which signal is routed to NOut. The 'C' signal is one of the upper four inputs to the 8:1 multiplexers shown in **Figures 9 to 12**, the actual value being selected by the two least-significant multiplexer select lines. Similar 'C' signals are generated in the Sswitch, Eswitch, and WSwitch.

The S4 input to the NOut multiplexer in the Nswitch is actually the S4 *input* to the adjacent Sswitch in the 4x4 block immediately to the North of this block. This should not be confused with the S4Out signal from that block's Sswitch. This is also true of some of the other inputs to the multiplexers in other boundary switches. To avoid confusion, these inputs are prefixed with the letter 'P' (for

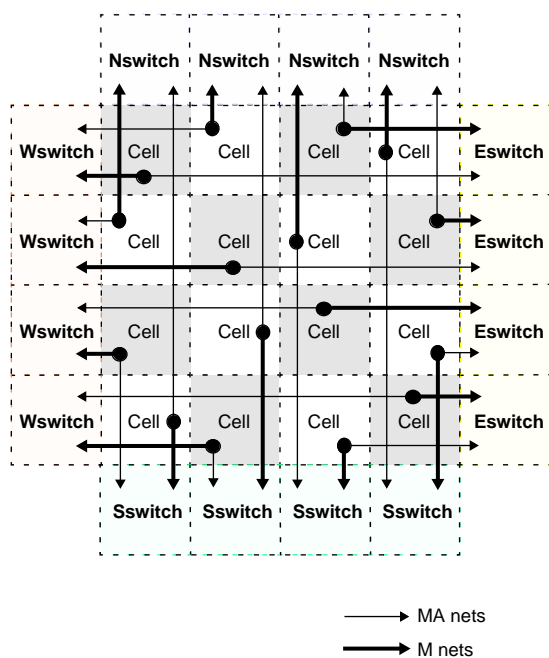


Figure 8. Routing Switches at 4x4 Block Boundary

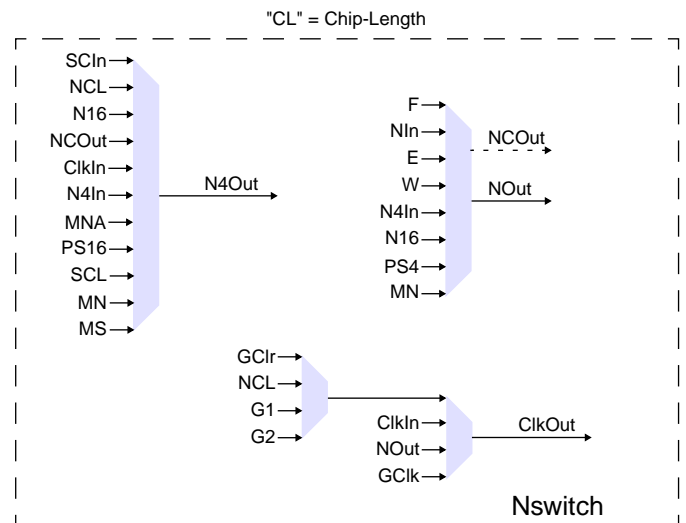


Figure 9. Contents of Nswitch

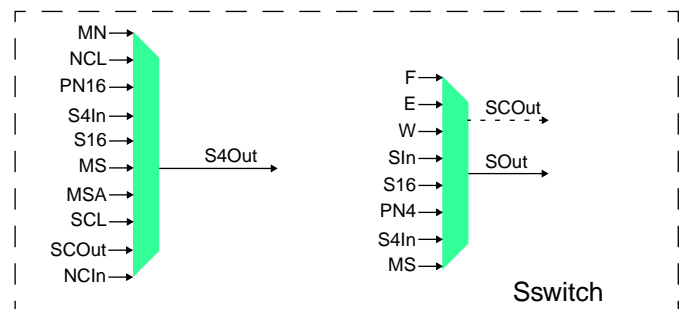


Figure 10. Contents of Sswitch

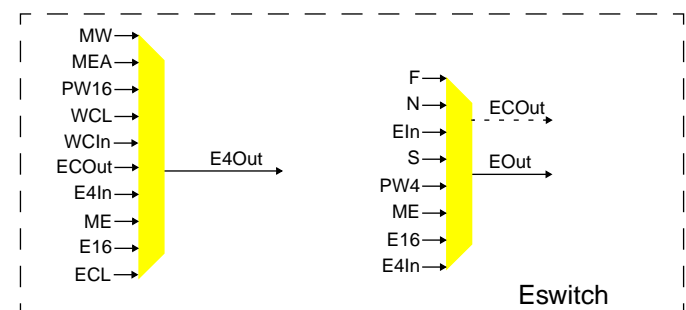


Figure 11. Contents of Eswitch

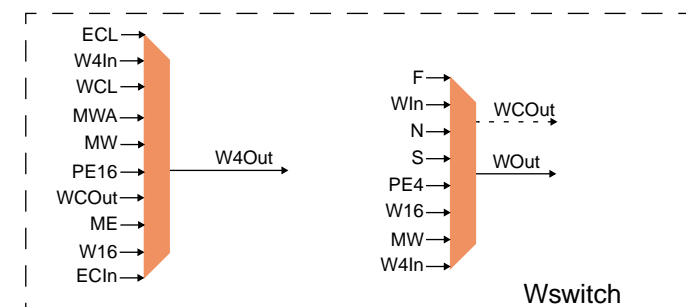


Figure 12. Contents of Wswitch

Previous). e.g. PS4. This feature allows FastLANEs™ to perform U-turns.

## Clock Distribution

As described previously, register clock inputs may be driven from any source but it is recommended that the GClk signal is used. GClk also has the advantage that it can be stopped by writing to the Device Configuration Register (see Table 22). The Global wires enter the part through dedicated input pins and are distributed in a special low-skew 'H' pattern (Figure 13). Each vertically aligned (South to North) group of four cells within a 4x4 block is clocked by its own clock source. This is driven from a multiplexer in the Nswitch immediately to the South of the group of cells. The connections for this multiplexer are shown in Figure 9. ClkOut drives the Clk inputs to each of the four cells in the group. As can be seen from Figure 9, the register clock for each group of four cells can be driven by ClkIn, NOut, GClk, GClr, G1, G2 or NCL (N Chip-Length). ClkIn is the ClkOut from the 4x4 block to the South, allowing vertical daisy-chaining of clock signals. NOut is the N output from the cell associated with the Nswitch. This can be used to provide local user-generated or gated clock signals if required. GClk is the Global Clock signal direct from the device GClk input. Clearly this signal only has to pass through one 4:1 multiplexer whereas GClr, G1 and G2 have to pass through two. This is one reason why there is less delay on GClk.

It is also possible to route North Chip-Length wires onto the Clock lines. This allows up to 64 (for a XC6216) locally used clocks to be provided which can still run the entire length of the chip with minimal skew. These local clock signals may be generated internally (e.g. by dividing a faster clock) or sourced directly from the device programmable I/O pins.

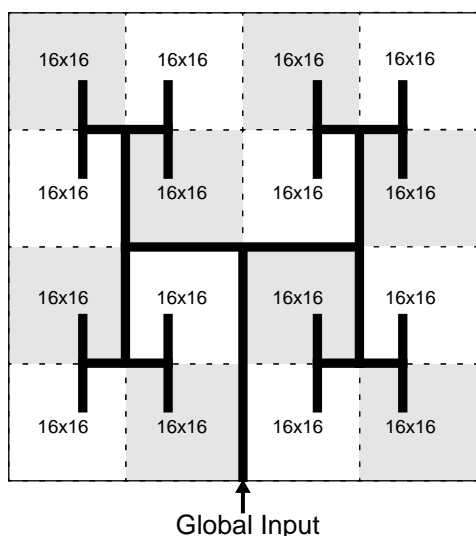


Figure 13. Low Skew 'H' Distribution Of Global Signals (XC6216)

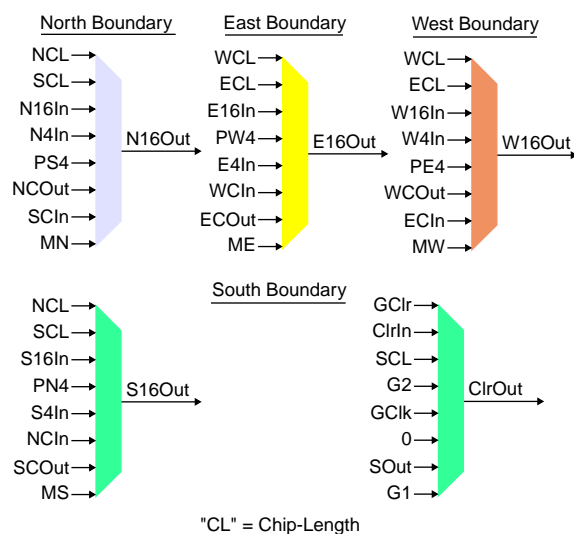


Figure 14. Additional Switches at 16x16 Boundaries

Where a fast clock is required by only a small fraction of the logic on the device it may be preferable to employ user interconnect resources rather than a Global or Chip-Length signal. This is because limiting fast clock distribution to the area of the device where it is required reduces power consumption.

## Clear Distribution

Register Clear inputs are routed in a similar manner to Clock inputs. In this case vertical groups of 16 cells, within a 16x16 block, share a common Clear. Clear lines run in a Southerly direction and are sourced from the Sswitch unit of 4x4 blocks which also lie on a 16x16 boundary. All of the boundary switches at 16x16 boundaries contain additional switching multiplexers. These are illustrated in Figure 14. These multiplexers invert some inputs. This is not shown in the figures. See the "Programming" section for details.

ClrOut drives the Clr inputs to each of the sixteen cells in the group. The S and SCL connections allow the output of a cell to provide a user-generated local Clear signal.

## I/O Architecture

User-configurable Input/Output Blocks (IOBs) provide the interface between external package pins and the internal logic.

One IOB is provided for every cell position around the array border. IOBs are connected to fixed pad locations. There are more IOBs than available pads, hence some IOBs are 'padless'. However it is still possible to route signals from padless IOBs to device pins.

Figure 15 is a simplified diagram of an IOB and its associated IO pad. The IOB is located at the array border and the pad is located close to its device pin. The pad may be located some distance from its associated IOB. The

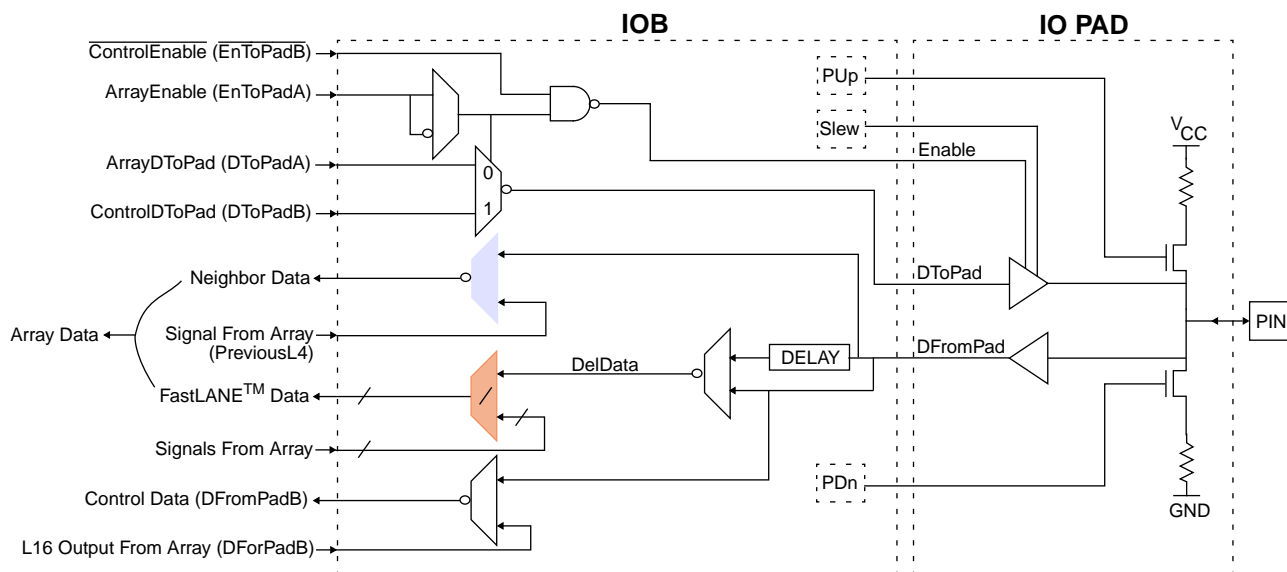


Figure 15. Input/Output Architecture

mapping of IOBs to device pins is given in the pinout tables starting on [page 45](#).

The XC6200 IOB architecture incorporates a novel and very powerful feature: every IOB has the capability of routing either an array signal or a control logic signal to/from the device pin. Every signal, including all the control signals (e.g.  $\overline{CS}$ ,  $Rd\overline{Wr}$ , Address Bus, Data Bus, etc.), passes through an IOB. This means that all the control signals can be routed into the logic array for use in user designs. Similarly, user logic can control the XC6200 internal control circuitry. For example a user signal could

be used to drive the internal  $\overline{CS}$  signal rather than the  $\overline{CS}$  pin.

As an example of the power of this feature, an XC6200 design could include an address decoder which decoded microprocessor read/write cycles and produced appropriately retimed signals for all the parts on a board *including itself*, thereby removing the need for address decoding PALs or discrete logic.

Each IOB has an array data input and a control data input, labeled *ArrayDToPad* and *ControlDToPad* in [Figure 15](#). Associated with these inputs are two enable signals - *ArrayEnable* and *ControlEnable*. These signals control whether the pad associated with this IOB is in the input or output mode. Each IOB also supplies *ArrayData* and *ControlData* when acting as an input.

The 'Control' signals are routed to the internal XC6200 control circuitry. If control signals are not required all the time then these IOBs can be used to route other user signals into the array. For example if only eight data bus bits were continuously required, the remaining twenty-four IOBs associated with the data bus could be used to route user signals to/from the array. *ControlEnable* comes either from the internal XC6200 control circuitry if there is a bidirectional control signal or output signal on that IOB, or it is tied inactive.

The 'Control' signals are also referred to as 'B' signals in this data sheet.  $ControlDToPad = DToPadB$ ,  $ControlEnable = EnToPadB$  and  $ControlData = DFromPadB$ . The L16 output from the array, which can be routed onto *ControlData*, is also referred to as *DForPadB*.

The *DFromPadB* output is unconnected in IOBs which have an output-only 'B' signal, such as  $\overline{SECE}$ . IOBs which have an input-only 'B' signal, such as  $\overline{CS}$ , have the *DToPadB* input tied Low and the *EnToPadB* input tied High. IOBs which have no associated 'B' signal also have

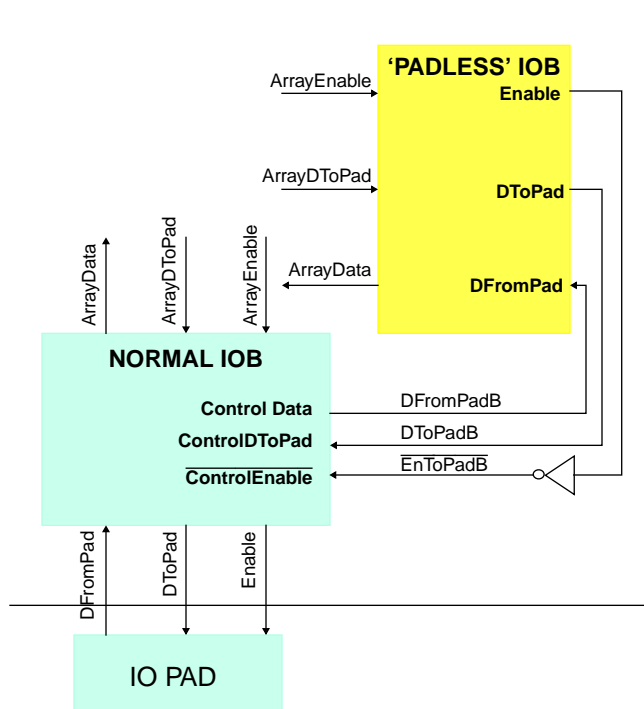


Figure 16. 'Padless' IOB Configuration

$DToPadB$  tied Low,  $\overline{EnToPadB}$  tied High and  $DFromPadB$  unconnected.

There are less real control signals than IOBs, hence the three signals,  $DFromPadB$ ,  $DToPadB$  and  $\overline{EnToPadB}$ , on some IOBs are not connected to the XC6200 control logic. Some of these spare 'B' signals are used to route data to and from the padless IOBs mentioned above. The 'B' signals on the padless IOB are not used. This is illustrated in [Figure 16](#). This arrangement allows data to be routed in or out of the chip via an IOB which has no associated IOPAD. The padless IOBs and their padded partner IOBs are detailed in the pinout tables starting on [page 45](#). For example, in a XC6216 IOB W0 is padless and is partnered with IOB S12, which has a pad.

The *ArrayEnable*, *Array Data* and *Control Data* multiplexers are controlled by configuration RAM bits. A fixed delay may be optionally applied to *Array Data* inputs. This allows the input data hold time specification to be removed.

The *ArrayEnable* and *ArrayDToPad* signals can be configured to constant 0 or 1 values within the logic array. The constant values are particularly useful for the enable signal when the pin is to function as an input or output rather than a bidirectional pin. Constant values on the data signal and a computed value on the enable signal produce open drain pull-up ( $DToPad=1$ ) or pull-down ( $DToPad=0$ ) pins.

**Table 3: Connections Between IOBs And Built-In XC6200 Control Logic**

B Signal Type	Example	$\overline{EnToPadB}$	$DToPadB$	$DForPadB$	$DFromPadB$
Input Only	$\overline{CS}$	1	0	L16 Output From Array	Drives XC6200 Control Logic $\overline{CS}$ Input
Output Only	$\overline{SECE}$	Driven By XC6200 Control Logic	$\overline{SECE}$ Out From XC6200 Control Logic	L16 Output From Array	Not Connected
Bidirectional	Data Bus	Driven By XC6200 Control Logic	$\overline{Data-Bus}$ Out From XC6200 Control Logic	L16 Output From Array	Drives XC6200 Internal FASTmap™ Data Bus Inputs
From Padless IOB	South IOB12	$\overline{Enable}$ Output From W0 IOB	$DToPad$ Output From W0 IOB	L16 Output From Array	Drives W0 IOB $DFromPad$ Input
None	North IOB30	1	0	L16 Output From Array	Not Connected

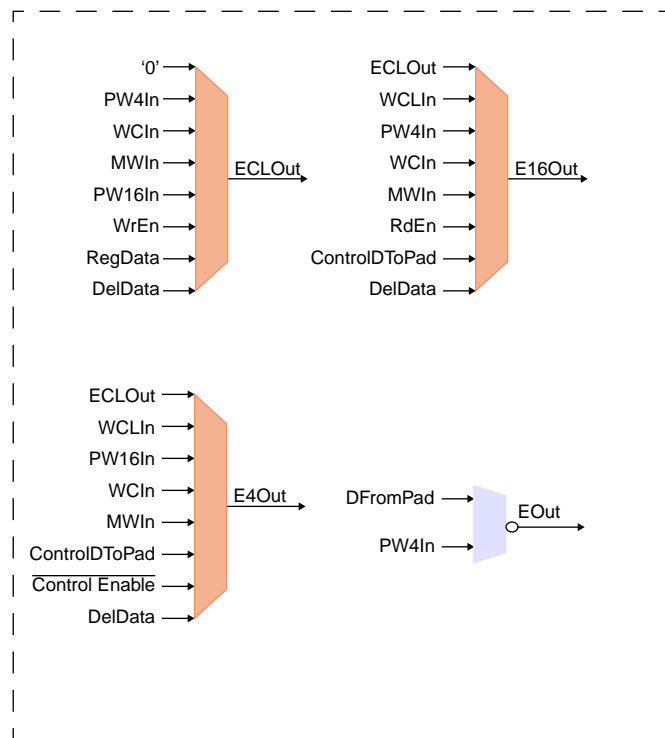


Figure 17. Array Data Sources In West IOBs

### Pull-Up, Pull-Down And Slew

Three configuration RAM bits within each IOB control the programmable aspects of its IO pad. These RAM bits have no effect for padless IOBs. 'PU' and 'PDn' enable the pull-up and pull-down resistors. The resistors may be used to tie floating logic inputs to a known value. 'Slew' slows the output transition time to reduce supply noise and ground-bounce. The default condition is pull-up off, pull-down off and slew on.

During reset, all the output drivers are disabled and the pull-up resistors are enabled. The pull-up and pull-down RAM control bits have no effect. After a reset the output drivers remain in this state. For the output drivers to be enabled, the global  $\overline{OE}$  signal must be asserted (Low) and a valid configuration must be present in the device ID register. The ID register is usually the last thing to be written during configuration and acts as a check that the programming interface is operating correctly. More details of this are given in the 'Programming' section. The  $\overline{OE}$  signal provides a quick way of disabling all the output drivers and may be activated at any time. Only when  $\overline{OE}$  is active and there is a valid ID pattern in the ID register, do the pull-up and pull-down RAM control bits determine the IO-pad resistor configuration. When  $\overline{OE}=1$  or the ID pattern is not valid, the pull-ups default to on and the pull-downs to off. The only exception to this default occurs in the Leakage Test Mode (see "Serial Interface State Machine", State 1) where all the pull-up/down resistors are disabled.

### Border Routing

The array signals to and from the IOBs are generally just the signals which would have passed between two cells in the array. The *ArrayDToPad* signal in Figure 15 is actually the neighbor output from the border cell associated with the IOB. The *Array Enable* signal is the length-4 FastLAN-E™ output from the same cell.

The *Array Data* multiplexer in Figure 15 is actually a collection of multiplexers which source the neighbor, length-4, length-16 and Chip-Length wires into the array. South IOBs (IOBs at the South edge of the array) also source the local clock signals into the array. North IOBs source the local clear signals. The actual signals which can be routed onto the IOB *Array Data* outputs are detailed in Tables 14 and 19. This is illustrated for a West IOB in Figure 17. Inversions are not shown. See "Programming" section for details. These multiplexers also allow a number of other internal control signals to be routed into the array: WrEn and RdEn are signals which are active during state register accesses. 'RegData' is the state register output value for this row during a state access. Details of the timing of these signals are given in the "Parallel CPU Interface" section. Note that in order to provide a minimal delay signal path into the core array, the neighbor data output from the IOB cannot select the delayed version of *DFromPad*. Only the un-delayed *DFromPad* and the Previous Length-4 Input can be routed onto the neighbor data output. Therefore the neighbor data output is unaffected by the value of the configuration memory which controls the DelData multiplexer in Figure 15.

The length-4 and length-16 routing multiplexers at the array border also expect some inputs which are not available. For example at the West edge, MEIn, ECIn, PE4In and PE16In are non-existent. These inputs are tied to ground or  $V_{CC}$ , thereby providing an abundant source of constant zeros and ones at the array border. These can be used to provide constant values to drive the *ArrayEnable* inputs to IOBs.

### GClk, $\overline{OE}$ And Reset Routing

The connections from the GClk I/O pad differ from all the other I/O signals. This is necessary because GClk is used to clock all the built-in FPGA control logic.

In common with the  $\overline{Reset}$  and  $\overline{OE}$  pads, the GClk pad cannot be enabled as an output pad. However, the main difference between these and all the other control signals is that they are routed to the FPGA control logic directly from the pad and **not** from the IOB. This means that a user circuit cannot modify them before they reach the control logic. This is illustrated for GClk in Figure 18.

GClk to the logic array is supplied from the IOB. It is this signal which passes through the enable circuit controlled by a single bit in the device control register. Thus it is still possible to route any signal onto the array GClk net using



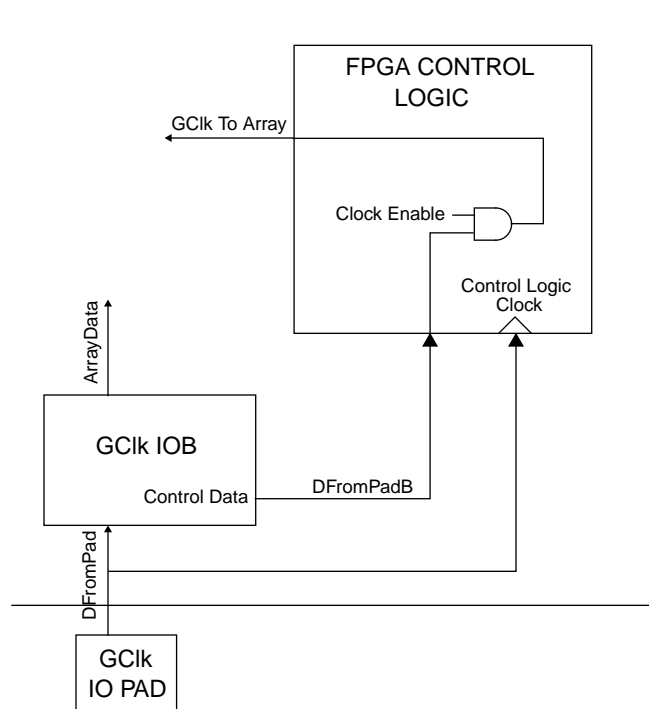


Figure 18. GClk Routing

the L16 output from the array (see Figure 15). However this GClk can only be safely stopped and started (without glitches) if the DFromPadB signal is the DFromPad signal from the GClk I/O pad.

## Designing with XC6200

Designing of XC6200 into systems may be partitioned into three distinct activities.

### 1. Board design with XC6200

An XC6200 part may be used on a board design as a microprocessor peripheral part, as an ASIC-type device or as both. In the first instance the XC6200 part has conventional SRAM data, address and control signals as illustrated in Figure 19. In other cases it may only require the user defined I/O signals of an ASIC. Packaging information for the part is given in section "Packaging" on page 35. The number of user I/O signals depends on the exact package used.

Several XC6200 devices may be tiled together on a board to form a larger array. The regular array structure of XC6200 makes this particularly easy. East I/Os on one chip would connect to West I/Os on the adjacent chip. North I/Os would connect to South I/Os on the adjacent chip and so on until the required array size was reached. It may be required to use the control signal pins, such as the Data and Address busses, on every part in the array. The control signals use every second IOB, leaving evenly distributed IOBs in between which can be used to inter-

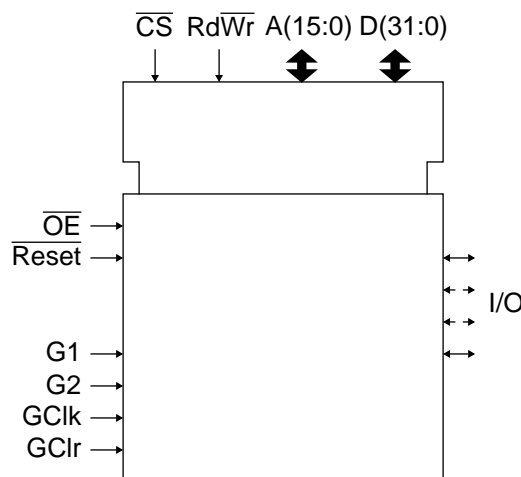


Figure 19. XC6216 Logic Symbol

connect the XC6200s. See the device pin-out tables, starting on page 45 for details.

The configuration RAM bits in the IOBs allow for a number of different programmable options to make interfacing to other ICs easier. This is more fully defined in the "I/O Architecture" section.

### 2. Logic design with XC6200

This can be approached as an ASIC type design using the function and routing architecture defined in the previous sections. An example design flow is illustrated in Figure 20. The design may be carried out in a variety of different ways. Hardware description languages such as VHDL may be used with the synthesized design targeted to the XC6200 architecture. Alternatively, schematic capture, using the extensive Xilinx Unified Library, with commonly used front end design tools (e.g. ViewLogic PROcapture/ViewDraw) may be used. These tools produce an EDIF netlist which is subsequently passed to the underlying XC6200 place and route software. This automatically maps the user's design to the XC6200 architecture in an efficient way and provides individual node delays which can be passed back to the high level simulation tools such as Viewlogic PROsim/ViewSim for accurate simulation. Simulation may be carried out prior to placement to check the logical correctness of the design using nominal delays. The place and route software also has optimization capability to carry out tasks such as redundant gate removal. A binary configuration file which can be written to the XC6200 device via the programming

interface is also produced automatically. The underlying CAD software is highly integrated with the high level CAD tools, providing user-friendly pull-down menus and dialog boxes to carry out all tasks.

These methods allow designers with little or no knowledge of the XC6200 architecture to quickly produce large and complex designs. Some designers may wish to carry out detailed hand placement and routing to produce ultra-optimized very high-speed/small area sections in their designs. Others may wish to generate large regular structures such as systolic arrays or perform floor-planning for extra efficiency. For these cases, a sophisticated physical editor is available which allows designers to graphically modify the automatic placement of gates/registers into cells and modify the routing as much as required. Alternatively this software may simply be used to see how the automatic placement and routing software has optimized a design. If a modification is subsequently made then only the modified part of the design needs to be re-laid out. This incremental design process gives a very rapid change cycle during debugging.

All the design tasks may be carried out on PC or Unix workstation platforms.

As an example, the simple accumulator circuit of [Figure 21](#) is mapped onto the XC6200 architecture. [Figure 22](#) shows the resulting layout as displayed by the Physical Editor, running under Microsoft Windows in this case. The Physical Editor tools are also available for Unix worksta-

tions. The boundaries of basic cells within the array are denoted by the squares, with larger rectangles representing the switch units on 4 cell boundaries. The wiring resources used by the design mapped onto the array are indicated by the darker black lines. When a cell function unit is used by the design it is annotated with the instance names of the mapped primitives. The primary inputs and outputs are not shown in this example. The rectangles around the edge represent IOBs and their pads.

The inputs and outputs to the function unit are connected to the edges of the cell box. The small squares within each cell represent the input ports (X3, X2, X1) and the output port (F).

The Physical Editor allows cells to be selected and moved. The inter-cell routing rubber-bands and adapts automatically to the new placement. The routing may also be manually modified if desired.

Full details of using the software are contained in the on-line help. An interactive software demonstration is also available.

### 3. Software design with XC6200

This is the design of a program for the host processor which interacts with a design running on the XC6200. Here various registers within the XC6200 design appear as locations within the processor's memory map. In addition the configuration memory of the device appears within the memory map and portions of the device can be

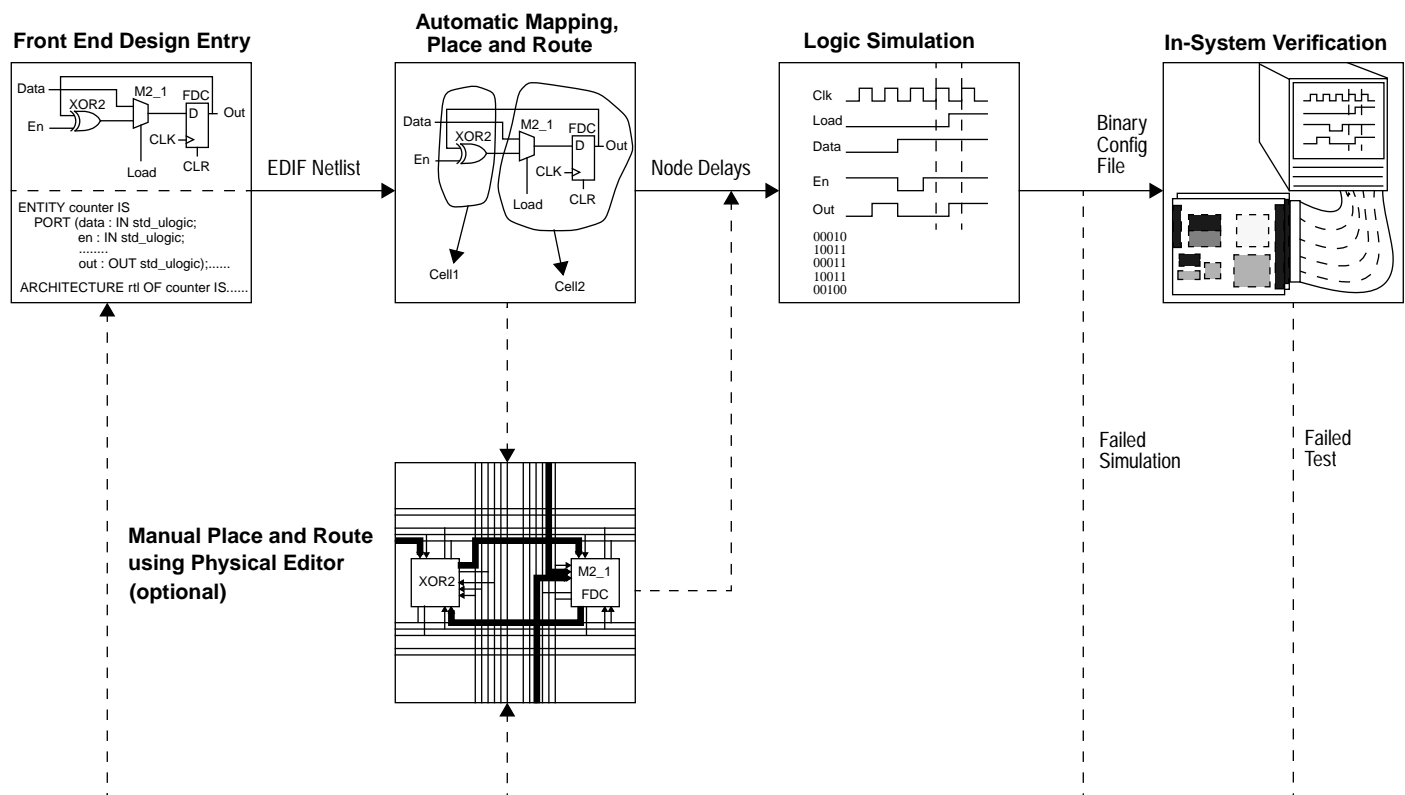


Figure 20. XC6200 Logic Design Flow



reconfigured as required. Predefined device drivers and an efficient run-time library are available to make optimal use of XC6200's high speed reconfiguration capabilities with minimal development time.

## Register Access

XC6200 supports direct accesses from the processor to nodes within the user's circuit: the output of any cell's function unit can be read and the flip-flop within any cell can be written. During state reads a number of cell outputs are routed onto the CPU data bus. The signal which is actually read is either C or S in [Figure 6](#), depending on whether the combinatorial or sequential output is selected. See [Table 11](#) on [page 21](#) for details of signal inversions. These inversions are cancelled out by the readback circuit so that the true value of C or S is read.

These accesses are carried out through the control store interface and involve no additional wiring within the user's design. The CPU interface signals involved in addressing the cell state can be routed into the configurable array so that user circuits can detect that an access has been made and take appropriate action: for example, calculate a new value for an output register or process a value placed in an input register.

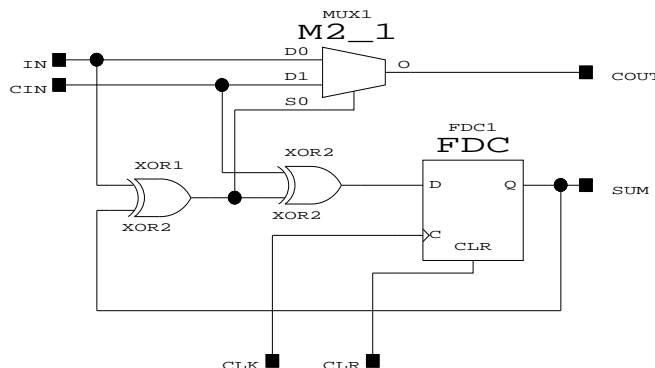


Figure 21. Accumulator Schematic

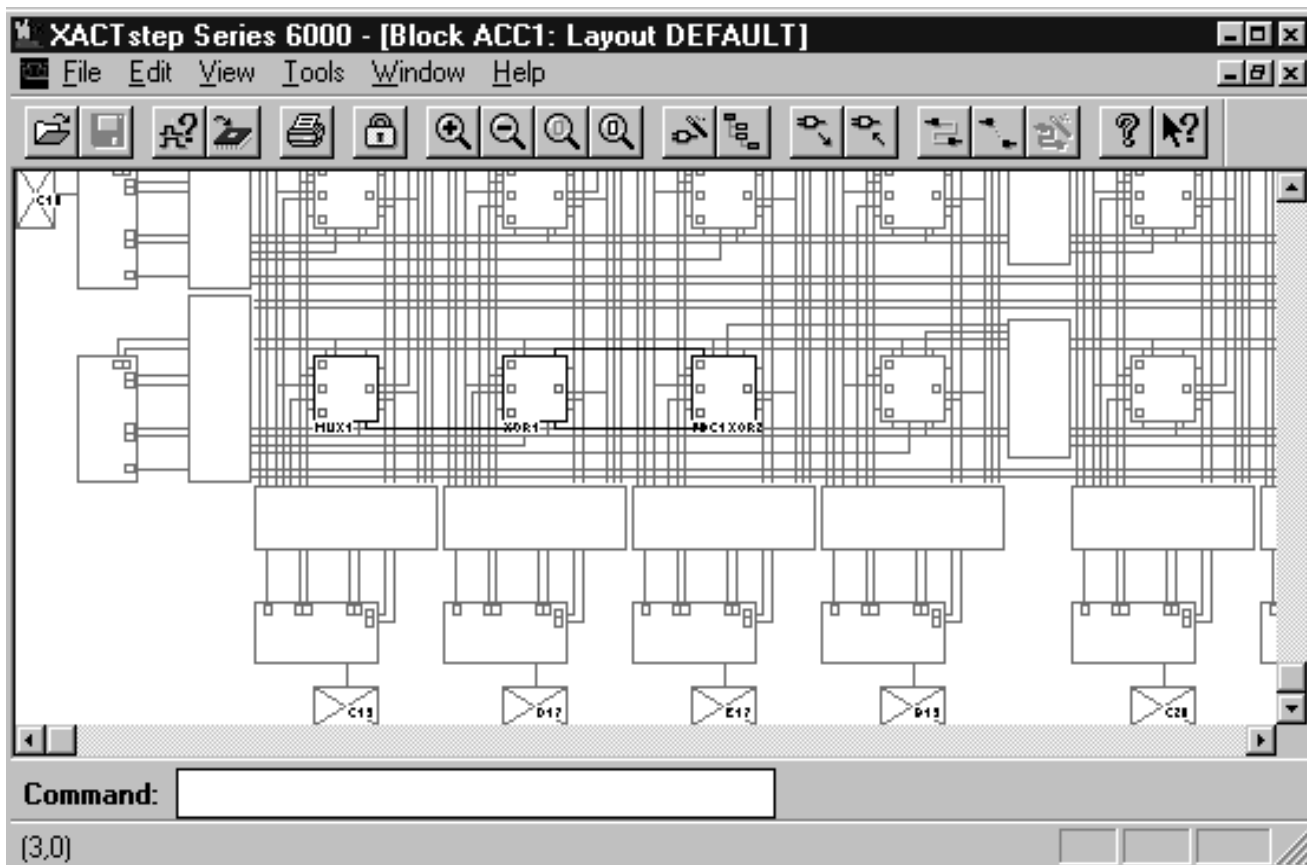


Figure 22. Accumulator Physical Editor View

In many applications this access to internal nodes is the main path through which data is transferred to the processor and in some coprocessor type applications it may be the only external I/O method: user programmable I/O pads may not be required at all.

To allow high bandwidth transfers between the processor and internal nodes it is necessary to be able to transfer a complete processor data word of up to 32 bits in one memory cycle. For this reason access bits within XC6200 are mapped into a separate region of the device address space from configuration bits so that all the bits in a word contain access bits. Figure 23 is a block diagram of the XC6216 part, showing the row and column address decoders. Figure 24 shows the mapping of this area of the address space: there are 64 I/O signals from each column of cells and a 6-bit column address selects a particular column of cells to access. This row and column addressing scheme puts a constraint on the placement of registers within the user's design which are to be accessed word-wide: they must be on the same column of cells within the array.

Map Register

XC6200 also provides a mechanism for mapping all the possible cell outputs from a column onto the 8,16 or 32-bit external data bus, selecting only those cells which implement bits of the register to be accessed. Without this unit the processor would have to implement a complex sequence of shift and mask operations to discard those bits corresponding to cells not within the register, or the user would have to constrain the layout so that the register bits were in adjacent cells. The mecha-

nism provided by XC6200 takes the form of a **Map Register**, one bit for each row I/O signal from the array. This Map Register can be read and written through the control store interface and is set up prior to state accesses. A logic 0 in the Map Register indicates that the cell in the corresponding row is part of the register to be accessed. The unit maps rows from the cell array onto external data lines starting with the least significant bit: thus the first row with a 0 in the Map Register connects to external data bus bit 0, the second row with a 0 in the Map Register to data bus bit 1 and so on.

This technique puts a further constraint on the user's layout: the cells implementing the bits of the register must be ordered so that less significant bits occur below more significant bits. However, there are no constraints about the relative separations of the cells. In practice these two placement constraints: cells occurring in the same column and in order vertically are easy to meet in datapath type designs.

Normally, the Map Register is set once to indicate the placement of the user I/O register which is then accessed many times. Therefore the two write operations required with a 32-bit bus to set up the Map Register represent a small overhead. In data path type designs where several registers are required, for example two input operand registers and a result register, it is easy to ensure that the corresponding bits of the registers occur on the same row but different columns of the array so that the same Map Register value can be used with different column addresses to access the various registers.

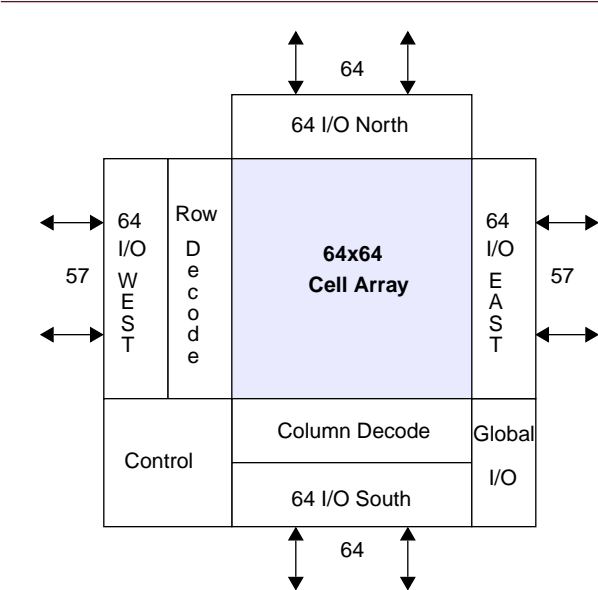


Figure 23. XC6216 Block Diagram

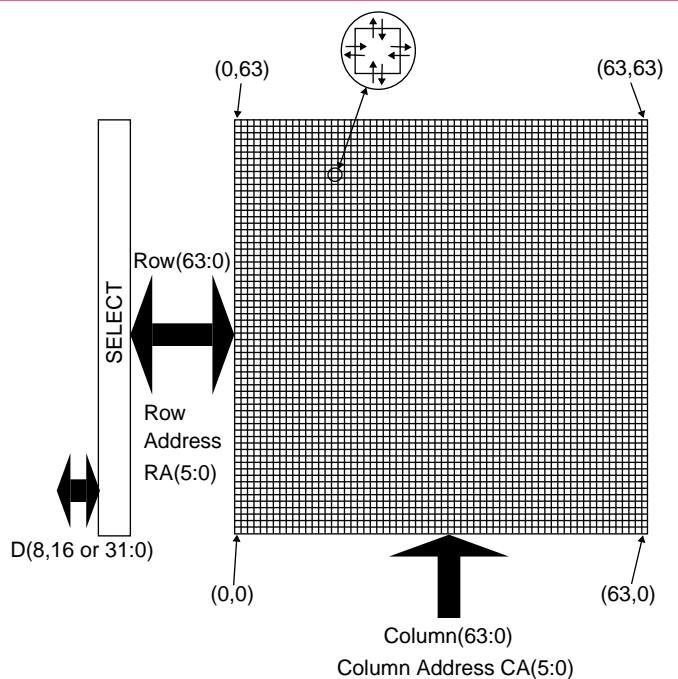


Figure 24. Memory Mapped I/O

## 8-Bit Data Bus Example

XC6200 Boundary

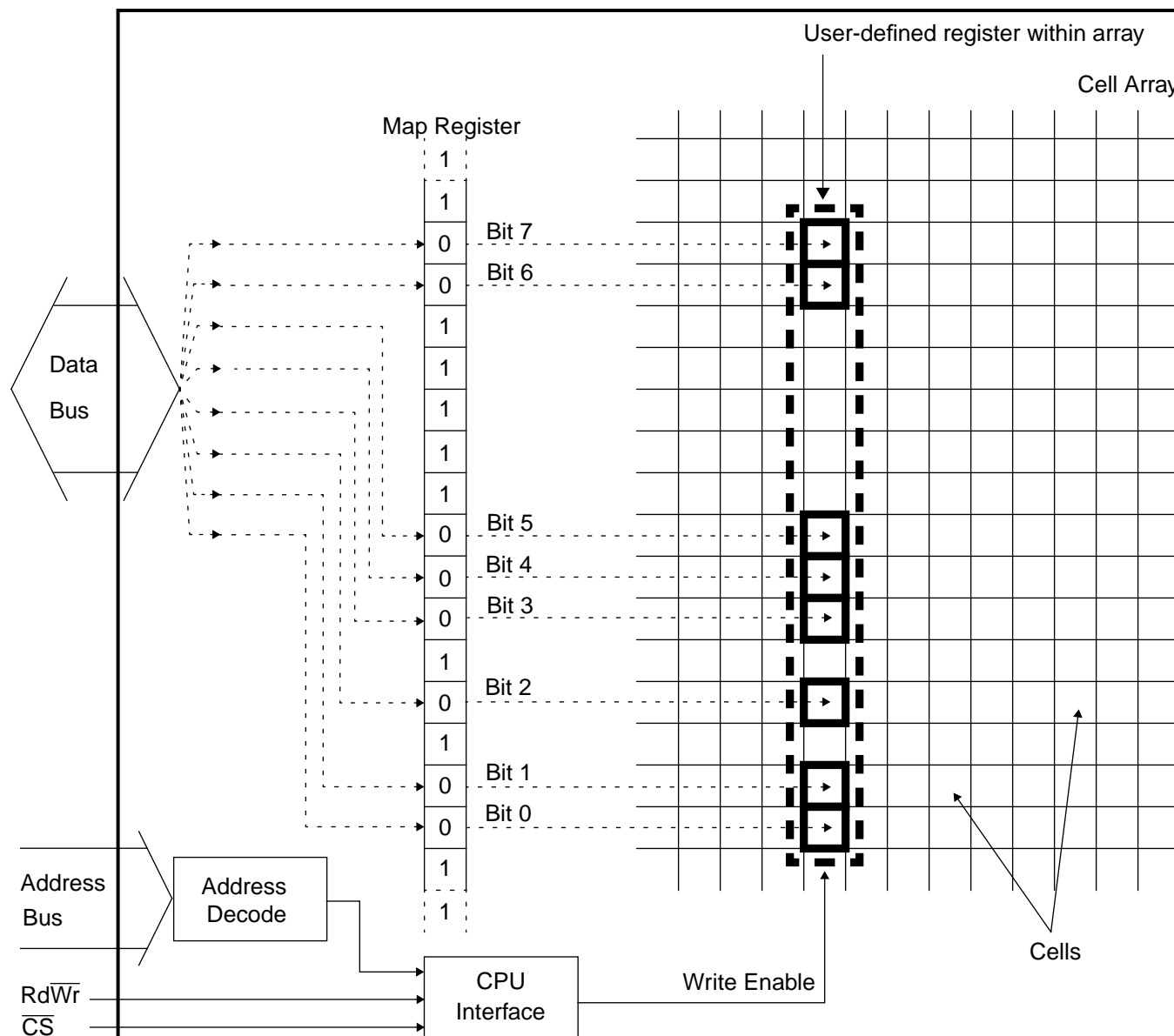


Figure 25. Internal Register Access

If more 0s exist in the Map Register than there are valid data bus bits then a form of wildcarding occurs during writes. The data bus bits are allocated to the rows of the array with a 0 in their Map Register bit. Once all of the data bus bits have been allocated, Bit 0 of the data bus is allocated to the next row whose Map Register bit is a 0, Bit 1 of the data bus to the next row and so on. This feature means that an entire column of state registers can be written with a single 8-bit write. For example, if the Map Register contains all 0s and the CPU writes FFh to a particular column. All the state registers in that column are written with a 1. The default state of the Map Register is all 0s.

During reads, if there are more 0 bits in the Map Register than data bus bits, the first rows with 0 bits are mapped onto the bus.

If there are fewer 0s in the Map Register than data bus bits, the upper data bus bits, which are not mapped, are undefined during CPU reads and ignored during CPU writes.

An example of Map Register operation is shown in Figure 25. The position of the user-defined register within the cell array is defined by the 0s in the Map Register. Similar registers could be defined for every column in the array if desired.

There is a delay of  $T_{MPST}$  (see page 42) after a write to the Map Register before the change takes effect. No state

accesses should be carried out during this time. There are important notes on the register clocking requirements for reliable register reading and writing in the “[Parallel CPU Interface](#)” section.

## Mask Register

A mask unit controlled by a 32-bit register is placed between the external data bus and the internal data connections. When the external data bus is 8 or 16 bits wide only the bottom 8 or 16 bits of this register are significant. A logic ‘1’ in a bit of this register indicates that the corresponding bit of the internal data bus is *not* relevant. Bit locations corresponding to 1s in the Mask Register retain their values when written. On a write operation the corresponding bit line is not enabled and the state information for that bit is not changed. When the device is reset the Mask Register contains all logic 0s corresponding to all data bus bits valid.

During CPU reads, valid register bits which are disabled are read as ‘0’. Invalid bits (bits which do not physically exist for the register being read) may be read as ‘0’ or ‘1’.

The Mask Register does not affect state register accesses. In this case the Map Register can be used to prevent certain bits being modified.

The Mask Register is also ignored during reads and writes to XC6200 Control Registers. These are memory locations which control various XC6200 functions and are defined in “[Address Mode 11 - Programming Control Registers](#)” on page 24

## Programming

The binary data for configuring XC6200, generated by CAD software from the textual description of a user design, must be downloaded into the part itself. This may be performed in several ways. Generally the fastest and most efficient way is by writing directly to the control store, mapped into the address space of a host processor. If a microprocessor or other parallel data source is not available then the serial programming interface may be used.

## Parallel CPU Interface

XC6200 has a full parallel CPU interface, referred to as ‘FastMap™’. This makes all the configuration SRAM and logic cells appear as conventional memory mapped SRAM. The FastMap™ interface is based on Chip Select ( $\overline{CS}$ ) and Read/Write ( $Rd\overline{Wr}$ ) control signals. The  $\overline{CS}$  signal can be used to address a single part within an array of devices and allows data to be read or written. Timing for these signals is illustrated in [Figures 30 and 31](#). These figures show that the programming interface is synchronous. The GClk input is used to sample all the

interface signals. GClk is also used when accessing user registers as illustrated in [Figure 25](#). This is an important point, as only registers clocked directly by GClk can be reliably read or written using this method. This is because the value written by the CPU is presented to the inputs of the cell registers just before  $t_2$  in [Figure 30](#) and held there until GClk goes Low again. Thus it is essential that the register receives a rising clock edge at  $t_2$ . This can be guaranteed if GClk is used to clock the registers. If another signal is used then it must have a rising edge at  $t_2$  for FastMap™ register writes to work. For reliable register reads, the register contents must be stable between  $t_1$  and  $t_2$  in [Figure 31](#).

[Figure 31](#) shows two separate read cycles - a normal cycle immediately followed by an extended cycle. In the normal read cycle  $\overline{CS}$  is sampled Low on the first rising GClk edge ( $t_1$ ) and High on the next ( $t_2$ ). The data bus is then driven until the next rising GClk edge ( $t_3$ ). In cases where this is not long enough, the read cycle can be extended by keeping  $\overline{CS}$  asserted beyond  $t_2$ . This is equivalent to adding wait states. In this case the data bus is driven until  $\overline{CS}$  is deasserted.  $\overline{CS}$  should not be allowed to go High and Low again. This would cause another cycle to begin.  $\overline{CS}$  is sampled on every rising GClk edge. Other CPU interface signals such as  $Rd\overline{Wr}$  and the Address Bus are only sampled on the first GClk edge of the cycle ( $t_1$  for the first cycle and  $t_3$  for the second in the figure examples).

Extended write cycles are also possible, however these are functionally no different to normal write cycles, the data and address busses still being sampled on the first rising GClk edge of the cycle ( $t_3$  in [Figure 30](#)).

$\overline{CS}$  must always be sampled as a ‘1’ before the next cycle can begin. In [Figure 31](#) the extended read cycle starts immediately after the normal read cycle at time  $t_3$ . A write cycle could not start until the *next* rising GClk edge as the data from the read cycle is still on the data bus.

The SRAM programming interface is supplemented by additional hardware resources designed to minimize the number of processor cycles required for reconfiguration. These resources are initially inactive after a reset so the device looks like an SRAM.

The control store layout is designed to minimize the overhead of computations required for dynamic access while maintaining adequate density to minimize the external storage required for device configurations. When an external processor is used to configure the device it may be convenient to use a compressed format of the configuration information.

A feature of the XC6200 architecture is that a rectangular area of cells specified as a hierarchical block within a user’s design corresponds directly with a rectangular area within the configuration memory of the XC6200 device.

This means that a block within the user's design can be dynamically replaced with another block by the host processor, reconfiguring only the corresponding area of the control store. The binary data for both blocks can be pre-calculated from the cellular design and the actual replacement can be carried out very rapidly using block transfer operations.

The format of the address bus to the XC6216 device is shown in Table 4. Larger XC6200 devices have proportionally more bits allocated to row and column addresses.

Mode(1:0)	Column(5:0)	Column Offset<1:0>	Row(5:0)
15:14	13:8	7:6	5:0

**Table 4: Address Bus Format (XC6209 and XC6216)**

Mode(1:0)	Column(6:0)	Column Offset<1:0>	Row(6:0)
17:16	15:9	8:7	6:0

**Table 5: Address Bus Format (XC6236 and XC6264)**

All the configuration memory can be accessed as 8-bit bytes. When a 16-bit transfer occurs Address<0> is irrelevant. When a 32-bit transfer occurs Address<1:0> is irrelevant. Data Bus bits <7:0> are written to the address with Address<1:0>=00, bits <15:8> are written to the address with Address<1:0> = 01, etc. The Address Mode bits are used to determine which area of the control store is to be accessed according to Table 6.

Mode1	Mode0	Area Selected
0	0	Cell Configuration and State
0	1	East/West Switch or IOB
1	0	North/South Switch or IOB
1	1	Device Control Registers

**Table 6: Address Mode Selection**

## Understanding The Configuration Bits

The full memory map is given in Tables 25 and 26. From these tables it is possible to work out the address for any bytes of configuration or cell state register in the FPGA. The address/data pairs are normally calculated automatically by XACTstep Series 6000 and written to a .cal file.

From Tables 7 to 21 it is possible to work out what data needs to be written to the above addresses to change the

configuration of any routing multiplexer or cell in the FPGA. These tables are split into subsections - Cells, East/West Switches and North/South Switches. Within each section, the bytes which control the switches are defined first. This table is then followed by a group of tables which define the coding of the bits within these bytes.

This sequence of tables details the coding for the various multiplexer select lines. These are always referred to as 'Sel'. The columns refer to the output of the appropriate multiplexer when Sel is at a particular value. As an example, in Table 7, Cell Routing Register Byte 00, Bits [7:6] correspond to the North Neighbor Multiplexer Sel[1:0] in Table 8. When Sel[1:0] = 10, the North output of the cell is routed from the East Neighbor input.

As another example, in Table 13, the EOut multiplexer in the West IOB is controlled by Bit4 of Byte 1. This corresponds to the single Sel bit in the West IOB block of Table 14. Thus when Bit4=0, EOut=PW4. When Bit4=1, EOut=DFromPad.

Note that most of the routing multiplexers invert their outputs to reduce propagation delays. This has not been shown in earlier figures.

## Address Mode 00 - Cell Mode

In Mode 00 the 6-bit row and column addresses are effectively a Cartesian coordinate pointer to a particular cell. (0,0) is the cell in the South-West corner of the array. Once a particular cell has been pin-pointed, the 2-bit Column Offset determines which cell configuration RAM byte is accessed. Each cell has 3 separate 8-bit configuration bytes and 1 single-bit state register. The state register is the cell register shown in Figure 6. The three cell configuration bytes are described below.

In Mode 00 bytes read from and written to the control store have the format shown in Table 7. Bit 7 is the msb. Column Offset = 00 addresses the neighbor routing multiplexers select lines. Thus a single byte controls all the neighbor routing multiplexers within a basic cell. Bytes 01 and 10 control the remaining cell routing. See Figures 5 and 6 for the connections to these multiplexers.

Column Offset<1:0>	DATA BIT							
	7	6	5	4	3	2	1	0
00	North		East		West		South	
01	CS		X1[2:0]		X2[1:0]		X3[1:0]	
10	M	RP	Y2[1:0]		Y3[1:0]		X3[2]	X2[2]

**Table 7: Cell Routing**



CS is the Combinatorial/Sequential multiplexer select line. M is the Magic multiplexer select line and RP is the Register Protect bit for the cell.

Column Offset = 11 is used for state accesses. In this case the Data Bus bit values are determined by the Map Register. The Row bits of the address bus are ignored.

The multiplexer which selects between Y2 and Y3 in **Figure 6** and the RP multiplexer also invert. Thus the inputs to the RP multiplexer are actually C and  $\bar{Q}$ .

Sel[1:0]	North	South	East	West
00	$\bar{F}$	$\bar{F}$	$\bar{F}$	$\bar{F}$
01	$\bar{N}$	$\bar{E}$	$\bar{N}$	$\bar{W}$
10	$\bar{E}$	$\bar{W}$	$\bar{E}$	$\bar{N}$
11	$\bar{W}$	$\bar{S}$	$\bar{S}$	$\bar{S}$

Table 8: Neighbor Multiplexer Selection

Sel[2:0]	X1	X2	X3
000	$\bar{S}$	$\bar{S}$	$\bar{S}$
001	$\bar{E}$	$\bar{W}$	$\bar{E}$
010	$\bar{W}$	$\bar{E}$	$\bar{W}$
011	$\bar{N}$	$\bar{N}$	$\bar{N}$
100	$\bar{W4}$	$\bar{W4}$	$\bar{W4}$
101	$\bar{S4}$	$\bar{E4}$	$\bar{S4}$
110	$\bar{E4}$	$\bar{S4}$	$\bar{E4}$
111	$\bar{N4}$	$\bar{N4}$	$\bar{N4}$

Table 9: X Multiplexer Selection

Sel[1:0]	Y2	Y3
00	X2	X3
01	$\bar{Q}$	$\bar{X3}$
10	$\bar{X2}$	Q
11	Q	$\bar{Q}$

Table 10: Y Multiplexer Selection

Sel	CS	M	RP
0	$\bar{Q}$	$\bar{X3}$	NOT PROTECTED (D)
1	$\bar{C}$	$\bar{X2}$	PROTECTED (Q)

Table 11: Remaining Configurable Cell Multiplexers

### Address Mode 01 - East/West Switches And IOBs

Mode 01 addresses the switches which control the East and West FastLANEs<sup>TM</sup> and the IOBs along the East and West edges. The upper Column address bits pin-point a particular 4x4 block. Column[1:0] then selects which edge of the 4x4 block.

Column[1:0]	Switches
00	West edge of 4x4 block
01	West IOB (Col[5:2] = 0000)
10	East IOB (Col[5:2] = 1111)
11	East edge of 4x4 block

Table 12: East/West Column Decoding

The IOBs are only addressed if Column[5:2] = 0000 or 1111, in a XC6216.

The Row address bits select an individual row within the array. For example 000000 selects the row at the South edge of the array and 111111 selects the row at the North edge of the array, in a XC6216.

Having pin-pointed a particular switch group, the Column Offset selects the exact switch required. For length-4 and 16 FastLANE<sup>TM</sup> switches within the array, Column Offset = 00. If an IOB is selected then the Column Offset selects a particular register within the IOB.

### East/West IOB Configuration

Column Offset <1:0>	DATA BIT							
	7	6	5	4	3	2	1	0
East IOB	00	W4[2:0]			Del	W	W16[2:0]	
	01	WCL[2:0]			En	DfPB	$\bar{Slew}$	PDn PUp
West IOB	00	PUp	PDn	$\bar{Slew}$	DfPB	En	ECL[2:0]	
	01	E16[2:0]			E	Del	E4[2:0]	

Table 13: East/West IOB Configuration Registers

See **Figure 15** for details of the IOB architecture. 'Del' allows a fixed delay to be added to the DFromPad signal. 'E' and 'W' are the neighbor outputs into the array. 'E4', 'W4', etc are the FastLANE<sup>TM</sup> outputs into the array. 'DfPB' is the select line for the Control Data (DFromPadB) output from the IOB. 'En' allows the Array Enable signal

to be inverted. PUp = 1 enables the pad pull-up resistor. PDn = 1 enables the pad pull-down resistor. Slew = 0 causes the output driver to slew its output. See [“Pull-Up, Pull-Down And Slew” on page 13](#).

The coding for the individual multiplexers is as follows:

West IOB				
Sel	Del	En	E	DfPB
0	Delay	W4In	PW4	DFromPad
1	No Delay	W4In	DFromPad	W16

West IOB			
Sel[2:0]	E4 Mux	E16 Mux	ECL Mux
000	ECLOut		0
001	WCLIn		PW4
010	PW16	PW4	WCLIn
011	WCLIn		MW
100	MW		PW16
101	DToPadB	RdEn	WrEn
110	EnToPadB	DToPadB	RegData
111	DelData		

East IOB				
Sel	Del	En	W	DfPB
0	Delay	E4In	PE4	DFromPad
1	No Delay	E4In	DFromPad	E16

East IOB			
Sel[2:0]	W4 Mux	W16 Mux	WCL Mux
000	WCLOut		0
001	ECLIn		PE4
010	PE16	PE4	ECLIn
011	ECLIn		ME
100	ME		PE16
101	DToPadB	RdEn	WrEn
110	EnToPadB	DToPadB	RegData
111	DelData		

**Table 14: East/West IOB Configuration Coding**

RdEn and WrEn are signals which are active during reads and writes to state registers in the row corresponding to the IOB. They can be routed back into user designs to detect CPU reads and writes of state registers. These signals go active after the first rising clock edge of a CPU cycle and go inactive after the second falling clock edge. RegData is the value being written to or read from a state register during a state access. It is valid during the second High phase of GClk during the cycle.

## East/West Switch Configuration

Column Offsets of 00 select array routing switches at 4x4 or 16x16 boundaries:

	DATA BIT							
	7	6	5	4	3	2	1	0
East	E16[2:0]			E[2]	E4[3:0]			
West	W4[3:0]			W[2]	W16[2:0]			

**Table 15: East/West Routing Configuration Registers**

The length-16 FastLANE™ bits are only of relevance at 16x16 boundaries. The E[2] and W[2] bits are the MSBs controlling the cell neighbor output. The LSBs are the normal neighbor selection bits in the cell routing register. See [“Routing Switches” on page 8](#).

Sel[2:0]	E Mux	W Mux	E16 Mux	W16 Mux
000	F	F	ECL	WCL
001	N	W	WCLIn	PE4
010	E	N	E4In	W4In
011	S	S	PW4	WCOOut
100	PW4	PE4	ECOOut	ECLIn
101	ME	W16	WCL	ECL
110	E16	MW	E16	W16
111	E4In	W4In	ME	MW

Sel[3:0]	E4 Mux	W4 Mux	Sel[3:0]	E4 Mux	W4 Mux
0000	MW	ECL	1000	ECOOut	-
0001	MEA	W4In	1001	E4In	WCOOut
0010	PW16	WCL	1010	ME	-
0011	WCL	MWA	1011	E16	ME
0100	WCLIn	-	1100	ECL	-
0101	-	MW	1101	-	W16
0110	-	-	1110	-	-



Sel[3:0]	E4 Mux	W4 Mux	Sel[3:0]	E4 Mux	W4 Mux
0111	-	PE16	1111	-	ECIn

**Table 16: East/West Routing Configuration Coding**

At the edge of the array some inputs clearly do not make sense. For example there can be no PE4 input to a W Multiplexer at the West edge of the array as there are no cells to the West of this multiplexer. In these cases the signals are tied to ground. The only exceptions are the Previous16 inputs to the length-4 multiplexers, which are tied to  $V_{CC}$ . Thus for an E4 multiplexer at the extreme Eastern edge of the array, MW=0, PW16=1 and WCIn=0.

### Address Mode 10 - North/South Switches And IOBs

Mode 10 addresses the switches which control the North and South FastLANEs™ and the IOBs along the North and South edges. Row[5:2] pin-points a particular 4x4 block. Row[1:0] then selects which edge of the 4x4 block.

Row[1:0]	Switches
00	South edge of 4x4 block
01	South IOB (Row[5:2] = 0000)
10	North IOB (Row[5:2] = 1111)
11	North edge of 4x4 block

**Table 17: North/South Row Decoding**

The IOBs are only addressed if Row[5:2] = 0000 or 1111.

Column[5:0] selects an individual column within the array. For example 000000 selects the column at the West edge of the array and 111111 selects the column at the East edge of the array.

Having pin-pointed a particular switch group, the Column Offset selects the exact switch required. If an IOB is selected then the Column Offset selects a particular register within the IOB.

### North/South IOB Configuration

	Column Offset <1:0>	DATA BIT							
		7	6	5	4	3	2	1	0
North IOB	00	Del	En	DfPB	S	S16[2:0]			-
	01	SCL[2:0]			-	S4[0]	-	-	-
	10	PUp	PDn	Slew	S4[2:1]		Clr[2:0]		

	Column Offset <1:0>	DATA BIT							
		7	6	5	4	3	2	1	0
South IOB	00	-	-	Clk[1:0]		N	DfPB	En	Del
	01	N16[2:0]			Clk[2]	N4[0]	Slew	PDn	PUp
	10	-	-	-	N4[2:1]		NCL[2:0]		

**Table 18: North/South IOB Configuration Registers**

The coding for the individual multiplexers is shown below:

North IOB				
Sel	Del	En	S	DfPB
0	Delay	N4In	PN4	DFromPad
1	No Delay	N4In	DFromPad	N16

North IOB				
Sel[2:0]	S4 Mux	S16 Mux	SCL Mux	Clr Mux
000	SCLOut		0	GClr
001	NCLIn		PN4	GClk
010	PN16	PN4	NCIn	G1
011	NCIn		MN	G2
100	MN		PN16	SOut
101	DToPadB	RegWord	RegWord	0
110	EnToPadB	DToPadB	EnToPadB	DelData
111	DelData			SCLOut

South IOB				
Sel	Del	En	N	DfPB
0	Delay	S4In	PS4	DFromPad
1	No Delay	S4In	DFromPad	S16

South IOB				
Sel[2:0]	N4 Mux	N16 Mux	NCL Mux	Clk Mux
000	NCLOut		0	NCLOut
001	SCLIn		PS4	NOut
010	PS16	PS4	SCIn	0
011	SCIn		MS	G1
100	MS		PS16	G2
101	DToPadB	RegWord	RegWord	GClr
110	EnToPadB	DToPadB	EnToPadB	GClk

South IOB				
Sel[2:0]	N4 Mux	N16 Mux	NCL Mux	Clk Mux
111	DelData			

**Table 19: North/South IOB Configuration Coding**

'RegWord' is the Word line which is asserted when state registers in the column corresponding to the IOB are read or written. This may be routed back into user designs as a means of detecting CPU reads and writes of state registers. This has many applications. For example it could be used to implement a 'Wait-Signal' semaphore system where a register must be read before it can be re-written. 'RegWord' is a pulse which starts after the first falling clock edge of a CPU cycle and ends after the second falling clock edge.

## North/South Switch Configuration

Column Offset <1:0>	DATA BIT							
	7	6	5	4	3	2	1	0
North Switch	00	-	-	-	-	PrimaryClk[1:0]		N[2]
	01	-	-	-	N16[2:0]		N4[1:0]	
	10	-	-	-	-	-	N4[3:2]	
	11	-	-	-	SecClkA[1:0]		SecClkB[1:0]	
South Switch	00	S[2]	S16[2:0]		-	-	-	-
	01	S4[3:2]		-	-	-	-	-
	10	S4[1:0]		Clr[2:0]		-	-	-

**Table 20: North/South Routing Configuration Registers**

The clock multiplexer in each North Switch is split into two sections (see [Figure 9](#) on [page 9](#)). The multiplexer which drives ClkOut is referred to as the primary clock multiplexer (PrimaryClk). The multiplexer whose output drives an input of the primary clock multiplexer is referred to as the secondary clock multiplexer (SecClk). Column Offset = 11 in the North Switch addresses the control lines for the secondary clock multiplexers. There is a separate secondary clock multiplexer for each column of cells in a 4x4 block. They are written to in pairs with Column[1:0] determining which pair is addressed. If Column[1:0] = 00 then SecClkA = Column1 (within 4x4 block), SecClkB = Column0. If Column[1:0] = 11 then SecClkA = Column2, SecClkB = Column3. Column[1:0] = 01 and 10 are illegal. Row[1:0] must be set to 11.

The length-16 FastLANE™ bits are only of relevance at 16x16 boundaries. The N[2] and S[2] bits are the MSBs controlling the cell neighbor output. The LSBs are the normal neighbor selection bits in the cell routing register. See ["Routing Switches" on page 8](#).

Sel[2:0]	N Mux	S Mux	N16 Mux	S16 Mux	Clr Mux
000	$\overline{F}$	$\overline{F}$	NCL	SCL	GClr
001	$\overline{N}$	$\overline{E}$	SCL	NCL	ClrIn
010	$\overline{E}$	$\overline{W}$	N16	S16	SCL
011	$\overline{W}$	$\overline{S}$	N4In	S4In	G2
100	$\overline{N4In}$	$\overline{S16}$	PS4	PN4	GClk
101	$\overline{N16}$	$\overline{PN4}$	NCOOut	SCOut	0
110	$\overline{PS4}$	$\overline{S4In}$	SCIn	NCIn	S
111	$\overline{MN}$	$\overline{MS}$	MN	MS	G1

Sel[3:0]	N4 Mux	S4 Mux	Sel[3:0]	N4 Mux	S4 Mux
0000	SCIn	MN	1000	$\overline{ClkIn}$	SCOut
0001	NCL	NCL	1001	$\overline{N4In}$	-
0010	N16	PN16	1010	-	-
0011	NCOOut	S4In	1011	$\overline{MNA}$	-
0100	-	S16	1100	PS16	$\overline{NCIn}$
0101	-	MS	1101	SCL	-
0110	-	MSA	1110	MN	-
0111	-	SCL	1111	MS	-

Sel[1:0]	PrimaryClk	SecClk
00	SecClk	GClr
01	ClkIn	NCL
10	$\overline{NOut}$	G1
11	GClk	G2

**Table 21: North/South Routing Configuration Coding**

## Address Mode 11 - Programming Control Registers

Gross features of the microprocessor interface are controlled by a number of Control Registers. We have already come across some of these in the form of the Map and Mask Registers. All the Programming Control Registers are mapped into the region of the device address space with the mode bits set to 11.

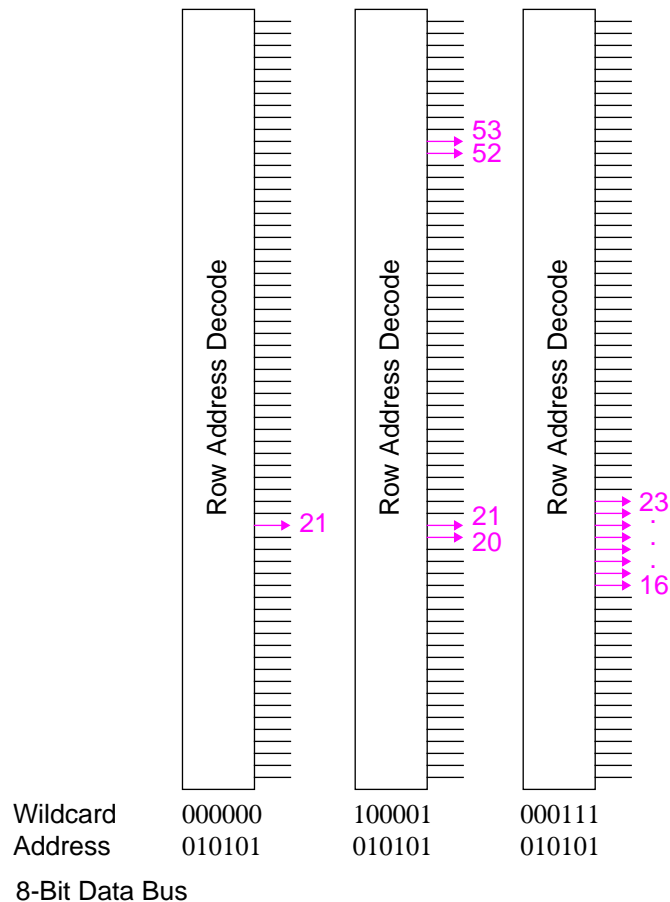


Figure 26. Row Wildcard Register

## Wildcard Registers

The FASTmap™ contains additional hardware sub-systems which can significantly reduce the processor overhead involved in configuring the device. These units are only active on write cycles. They are also inactive during writes to all Programming Control Registers.

The row address decoder is supplemented with a Wildcard Register which can be written through the FASTmap™. This register has one bit for each bit in the row address. During write cycles, logic one bits in the Wildcard Register indicate that the corresponding bit in the address is to be taken as 'don't-care': that is the address decoder matches addresses independent of this bit. When the device is reset this register is initialized to zero so all address bits are treated as significant. **Figure 26** shows some examples of the use of this unit assuming an 8-bit external data bus and a XC6216 device, so the cell array is eight words high. Outputs from the row address decoder enable bit line circuitry for the appropriate word.

The Wildcard Register allows many cell configuration memories within the same column of cells to be written simultaneously with the same data. This is used during

device testing to allow regular patterns to be loaded efficiently into the control memory, but is more generally useful, especially with regular bit sliced designs, because it allows many cells to be changed simultaneously. For example, a 16-bit 2:1 multiplexer could be built using cell routing multiplexers and switched between sources using a single control store access.

Similarly, the column address decoder has a Wildcard Register which allows several cells on the same row to be written with the same configuration. The column address decoder drives the word lines to enable particular columns of RAM cells. In this case the number of columns which can be written simultaneously is internally limited to 32: that is at most five don't care bits can be set. However, to guarantee that cells which are not being written do not overwrite each other, **a maximum of 4 columns should be written simultaneously.**

The row and column Wildcard Registers can be used simultaneously to rapidly configure regular structures onto the device.

The row Wildcard Register is ignored in state access mode, as the row decoding is controlled by the Map Register here. The column Wildcard Register may still be used to write to several banks of registers simultaneously.

The mask unit, described on [page 19](#), simplifies changing areas of the control store within a single word unit: for example, changing the source of one multiplexer within a cell without affecting the others. Consider changing the source for a cell's North multiplexer without this unit: the following operations are required:

1. Read control store at appropriate address.
2. Mask out bits corresponding to North register.
3. Get new value for north register bits 6 and 7.  
Make sure other bits are 0.
4. OR new value with value from stage 2.
5. Write back.

Using the mask unit the following steps suffice (using 8-bit transfers):

1. Write Mask Register with binary 00111111.
2. Write new value to control store at appropriate address.

The mask and wildcard units can be used together to perform complex operations such as changing the source of the West multiplexer on every second cell at offsets between 0 and 15.

## Device Configuration Register

This register controls global device functions and modes. The control functions of the bits are given in [Table 22](#).

'Config Speed' sets the baud rate of the serial programming interface. 'Bus Width' allows selection of external data bus width between 8,16 and 32 bits. The 'TTL/CMOS' bit globally controls the input logic threshold levels for all the I/Os. The default value of '0' causes TTL input thresholds to be used. '1' selects CMOS thresholds. 'Clock Enable' allows the user to stop the GClk signal to the cell array. The clock to the control circuitry is not stopped so that CPU cycles, etc. may continue. The default state is Clock Enable = 0, disabling the array GClk. This bit may be modified at any time. Internal circuitry makes sure that the clock is started and stopped in a safe, glitch free manner. Changes to Clock Enable take effect immediately after the write cycle to the Configuration Register.

Bit:	7	6	5	4	3	2	1	0
Function:	-	Clock Enable	-	TTL/CMOS	Bus Width		Config Speed	

**Table 22: Device Configuration Register**

Config. Reg [3:2]	Data Bus Width
00	8
01	16
10	32
11	Illegal

**Table 23: Data Bus Width Coding**

Config. Reg [1:0]	Config Speed
00	GClk/16
01	GClk/8
10	GClk/4
11	GClk/2

**Table 24: Configuration Speed Coding**

## Device Identification Register

The ID register is a 16-byte store which must be written with the correct pattern before the device outputs can be enabled. This serves as a check that the programming interface is operating correctly before allowing potentially damaging outputs to be driven. Each byte must be written with the ASCII code for the letters shown in Table 25. The only exception is the last byte. This must be written with a number which distinguishes this part from others within the XC6000 group. It is not an ASCII coded number. Details of the appropriate ID number for different family members are given in Table 27.

Until the ID register is correctly programmed, all the IO pad output drivers are disabled and the pull-up resistors are all enabled. Once a valid ID pattern has been written, the state of the IO pads depends on the  $\overline{OE}$  signal. If  $\overline{OE}$  is High then the output drivers remain disabled with their pull-ups on. When  $\overline{OE}$  is Low individual outputs may be enabled and pulled up depending on their individual control signals.

The ID register also provides a means of simultaneously disabling all the device outputs under software control. If any valid ID byte is changed then the outputs are all disabled. The outputs are all re-enabled as soon as the ID byte is written with the correct value.

The internal *ConfigOK* signal is available to user designs to determine when a valid pattern is present.

## Programming Control Register Memory Map

Table 25 lists the address for all the control registers for the XC6209 and XC6216. Each register may be written in 8-bit bytes. If 16 or 32 data bits are available then less writes are required. For example eight write cycles (to C010, C011,.....,C017) are required to change every bit of the Map Register with an 8-bit bus. Only two write cycles (to C010 and C014) are required with a 32-bit bus.

If only some bits are to be changed within a particular register then only the appropriate bytes need be written.

XC6236 and XC6264 have 18-bit address busses. The Control Register memory map for these parts is given in Table 26.

The full address decoding for the XC6200 family is summarized in Tables 28 to 31.

A[15:0]	Register	A[15:0]	Register
C000	Device Config	C031	ID (Byte1) (= 'i')
C004	Row Wildcard	C032	ID (Byte2) (= 'l')
C005	Column Wildcard	C033	ID (Byte3) (= 'i')
C008	Mask (Byte0)	C034	ID (Byte4) (= 'n')
C009	Mask (Byte1)	C035	ID (Byte5) (= 'x')
C00A	Mask (Byte2)	C036	ID (Byte6) (= ' ')
C00B	Mask (Byte3)	C037	ID (Byte7) (= 'X')
C010	Map (Byte0)	C038	ID (Byte8) (= 'C')
C011	Map (Byte1)	C039	ID (Byte9) (= '6')
C012	Map (Byte2)	C03A	ID (Byte10) (= '0')
C013	Map (Byte3)	C03B	ID (Byte11) (= '0')
C014	Map (Byte4)	C03C	ID (Byte12) (= '0')
C015	Map (Byte5)	C03D	ID (Byte13) (= ' ')
C016	Map (Byte6)*	C03E	ID (Byte14) (= ' ')
C017	Map (Byte7)*	<b>C03F</b>	<b>ID (Byte15) (=ID #)</b>
C030	ID (Byte0) (= 'X')		

**Table 25: Control Register Memory Map For XC6209 and XC6216**

\* Reserved locations in XC6209

A[17:0]	Register	A[17:0]	Register
30000	Device Config	3001D	Map (Byte13)*
30004	Row Wildcard	3001E	Map (Byte14)*
30005	Column Wildcard	3001F	Map (Byte15)*
30008	Mask (Byte0)	30030	ID (Byte0) (= 'X')
30009	Mask (Byte1)	30031	ID (Byte1) (= 'i')
3000A	Mask (Byte2)	30032	ID (Byte2) (= 'l')
3000B	Mask (Byte3)	30033	ID (Byte3) (= 'i')
30010	Map (Byte0)	30034	ID (Byte4) (= 'n')
30011	Map (Byte1)	30035	ID (Byte5) (= 'x')
30012	Map (Byte2)	30036	ID (Byte6) (= ' ')
30013	Map (Byte3)	30037	ID (Byte7) (= 'X')
30014	Map (Byte4)	30038	ID (Byte8) (= 'C')
30015	Map (Byte5)	30039	ID (Byte9) (= '6')
30016	Map (Byte6)	3003A	ID (Byte10) (= '0')
30017	Map (Byte7)	3003B	ID (Byte11) (= '0')
30018	Map (Byte8)	3003C	ID (Byte12) (= '0')
30019	Map (Byte9)	3003D	ID (Byte13) (= ' ')
3001A	Map (Byte10)	3003E	ID (Byte14) (= ' ')
3001B	Map (Byte11)	<b>3003F</b>	<b>ID (Byte15) (=ID #)</b>
3001C	Map (Byte12)*		

**Table 26: Control Register Memory Map For XC6236 and XC6264**

\* Reserved locations in XC6236

Device	ID Number
<b>XC6209</b>	<b>2</b>
<b>XC6216</b>	<b>1</b>
<b>XC6236</b>	<b>4</b>
<b>XC6264</b>	<b>3</b>

**Table 27: XC6200 Family ID Numbers**

Address Bus	Decode	
A[15:14] (Mode[1:0])	00	Cells
	01	East/West Switch or IOB
	10	North/South Switch or IOB
	11	Control Registers
A[13:8] (Column[5:0])	Cell Mode - Cell column	
	North/South Mode - Switch column	
	East/West Mode - Column[5:2] = 4x4 block number Column[1:0] decoded as:	
	00	West Switch
	01	West IOB (Column[5:2]=0000)
	10	East IOB (Column[5:2]=1011)
	11	East Switch
A[7:6] (Column Offset[1:0])	Cell Mode	
	00	Neighbor Routing
	01	Function Input Routing
	10	Function
	11	State Access (Cell Registers)
	North/South Switch Mode	
	00	N/S Switch or N/S IOB Reg 0
	01	N/S Switch or N/S IOB Reg 1
	10	N/S Switch or N/S IOB Reg 2
	11	Secondary Clock Mux (Column[1:0] = 00 or 11) && (Row[1:0]=11)
	East/West Switch Mode	
	00	E/W Switch or E/W IOB Reg 0
	01	E/W IOB Reg 1
A[5:0] (Row[5:0])	Cell Mode - Cell row	
	East/West Mode - Switch row	
	North/South Mode - Row[5:2] = 4x4 block number Row[1:0] decoded as:	
	00	South Switch
	01	South IOB (Row[5:2]=0000)
	10	North IOB (Row[5:2]=1011)
	11	North Switch

**Table 28: XC6209 Memory Map**

Address Bus	Decode	
A[15:14] (Mode[1:0])	00	Cells
	01	East/West Switch or IOB
	10	North/South Switch or IOB
	11	Control Registers
A[13:8] (Column[5:0])	Cell Mode - Cell column	
	North/South Mode - Switch column	
	East/West Mode - Column[5:2] = 4x4 block number Column[1:0] decoded as:	
	00	West Switch
	01	West IOB (Column[5:2]=0000)
	10	East IOB (Column[5:2]=1111)
	11	East Switch
A[7:6] (Column Offset[1:0])	Cell Mode	
	00	Neighbor Routing
	01	Function Input Routing
	10	Function
	11	State Access (Cell Registers)
	North/South Switch Mode	
	00	N/S Switch or N/S IOB Reg 0
	01	N/S Switch or N/S IOB Reg 1
	10	N/S Switch or N/S IOB Reg 2
	11	Secondary Clock Mux (Column[1:0] = 00 or 11) && (Row[1:0]=11)
	East/West Switch Mode	
	00	E/W Switch or E/W IOB Reg 0
	01	E/W IOB Reg 1
A[5:0] (Row[5:0])	Cell Mode - Cell row	
	East/West Mode - Switch row	
	North/South Mode - Row[5:2] = 4x4 block number Row[1:0] decoded as:	
	00	South Switch
	01	South IOB (Row[5:2]=0000)
	10	North IOB (Row[5:2]=1111)
	11	North Switch

**Table 29: XC6216 Memory Map**

Address Bus	Decode	
A[17:15] (Mode[1:0])	00	Cells
	01	East/West Switch or IOB
	10	North/South Switch or IOB
	11	Control Registers
A[14:9] (Column[6:0])	Cell Mode - Cell column	
	North/South Mode - Switch column	
	East/West Mode - Column[6:2] = 4x4 block number Column[1:0] decoded as:	
	00	West Switch
	01	West IOB (Column[6:2]=00000)
	10	East IOB (Column[6:2]=10111)
	11	East Switch
A[8:7] (Column Offset[1:0])	Cell Mode	
	00	Neighbor Routing
	01	Function Input Routing
	10	Function
	11	State Access (Cell Registers)
	North/South Switch Mode	
	00	N/S Switch or N/S IOB Reg 0
	01	N/S Switch or N/S IOB Reg 1
	10	N/S Switch or N/S IOB Reg 2
	11	Secondary Clock Mux (Column[1:0] = 00 or 11) && (Row[1:0]=11)
	East/West Switch Mode	
	00	E/W Switch or E/W IOB Reg 0
	01	E/W IOB Reg 1
A[6:0] (Row[6:0])	Cell Mode - Cell row	
	East/West Mode - Switch row	
	North/South Mode - Row[6:2] = 4x4 block number Row[1:0] decoded as:	
	00	South Switch
	01	South IOB (Row[6:2]=00000)
	10	North IOB (Row[6:2]=10111)
	11	North Switch

Table 30: XC6236 Memory Map

Address Bus	Decode	
A[17:15] (Mode[1:0])	00	Cells
	01	East/West Switch or IOB
	10	North/South Switch or IOB
	11	Control Registers
A[14:9] (Column[6:0])	Cell Mode - Cell column	
	North/South Mode - Switch column	
	East/West Mode - Column[6:2] = 4x4 block number Column[1:0] decoded as:	
	00	West Switch
	01	West IOB (Column[6:2]=00000)
	10	East IOB (Column[6:2]=11111)
	11	East Switch
A[8:7] (Column Offset[1:0])	Cell Mode	
	00	Neighbor Routing
	01	Function Input Routing
	10	Function
	11	State Access (Cell Registers)
	North/South Switch Mode	
	00	N/S Switch or N/S IOB Reg 0
	01	N/S Switch or N/S IOB Reg 1
	10	N/S Switch or N/S IOB Reg 2
	11	Secondary Clock Mux (Column[1:0] = 00 or 11) && (Row[1:0]=11)
	East/West Switch Mode	
	00	E/W Switch or E/W IOB Reg 0
	01	E/W IOB Reg 1
A[6:0] (Row[6:0])	Cell Mode - Cell row	
	East/West Mode - Switch row	
	North/South Mode - Row[6:2] = 4x4 block number Row[1:0] decoded as:	
	00	South Switch
	01	South IOB (Row[6:2]=00000)
	10	North IOB (Row[6:2]=11111)
	11	North Switch

Table 31: XC6264 Memory Map



## Serial Programming Interface

All the memory mapped locations in XC6200 may be written in parallel or serial mode. All the operations which can be carried out with the FastMAP™ parallel interface may also be done serially. The serial interface gives random access to all the XC6200 memory locations. It is designed to operate with any Xilinx serial PROM. A single serial PROM may be used to configure several FPGAs. In this case one of the FPGAs acts as a 'Master' and the others as 'Slaves'. The Master controls the serial PROM and the Slaves. This is illustrated in Figure 27.

The serial PROM interface consists of 6 dedicated I/O pins:

$\overline{Serial}$	Input which controls transitions between states in serial mode state machine. 0 => serial mode, 1 => parallel mode
$\overline{Wait}$	Input which controls transitions between states in serial mode state machine. 0 => continue loading, 1 => pause until $\overline{Wait}$ deasserted
$\overline{SERreset}$	Output from Master FPGA which resets serial PROM address counter.
$\overline{SECE}$	Output from Master FPGA which enables serial PROM output.
$\overline{SEClk}$	Output from Master FPGA which clocks serial PROM and slave FPGAs. $\overline{SEData}$ is clocked into the FPGAs on the rising edge of $\overline{SEClk}$ .
$\overline{SEData}$	Serial data input to FPGA. This is sampled in the FPGA by $\overline{SEClk}$ and retimed by the FPGA's own $\overline{GClk}$ .

In a multi-FPGA configuration a user I/O also has to be available to provide the  $\overline{Wait}$  input to the next device in the chain.

On Reset each FPGA examines its  $\overline{Serial}$  and  $\overline{Wait}$  inputs. Any FPGA which sees both these signals Low at this time assumes it is the master and drives  $\overline{SERreset}$ ,  $\overline{SECE}$  and  $\overline{SEClk}$ . All User I/Os are held in a high-impedance state (with pull-up) until a valid configuration is loaded. In the Figure 27 example the User I/O's are pulled High on Reset, hence the  $\overline{Wait}$  input to the Slaves is High and they configure as Slaves. A valid configuration is assumed when the device ID register is loaded with the correct ID. Programmable I/Os can only be enabled when this is present.

Serial data is loaded in address/data pairs. Once an address/data pair has been shifted into the FPGA, the data word is parallel written to the corresponding address inside the FPGA, just as though a parallel CPU write had occurred. This means it is possible to do all the things which can be accomplished with the FastMAP™. e.g. use of the Mask Register, writes to cell state registers, etc.

The write operation is pipelined so there need be no interruption in the serial data stream. The first address/data pair must be preceded by a Synchronization Byte = 1111\_1110. There are no start/stop bits, checksums or error check/correction bits. A full 8-bit Synchronization byte is not actually required by the FPGA. Three or more ones followed by a single zero is interpreted as a valid synchronization pattern.

The address and data are shifted in MSB first. The address is always 16-bits. The data word is initially 8-bits but may be increased to 16 or 32 bits by loading the Device Configuration Register with the appropriate code. The bits are shifted in on the rising edge of  $\overline{SEClk}$ . The  $\overline{SEClk}$  rate may also be increased by writing the appropri-

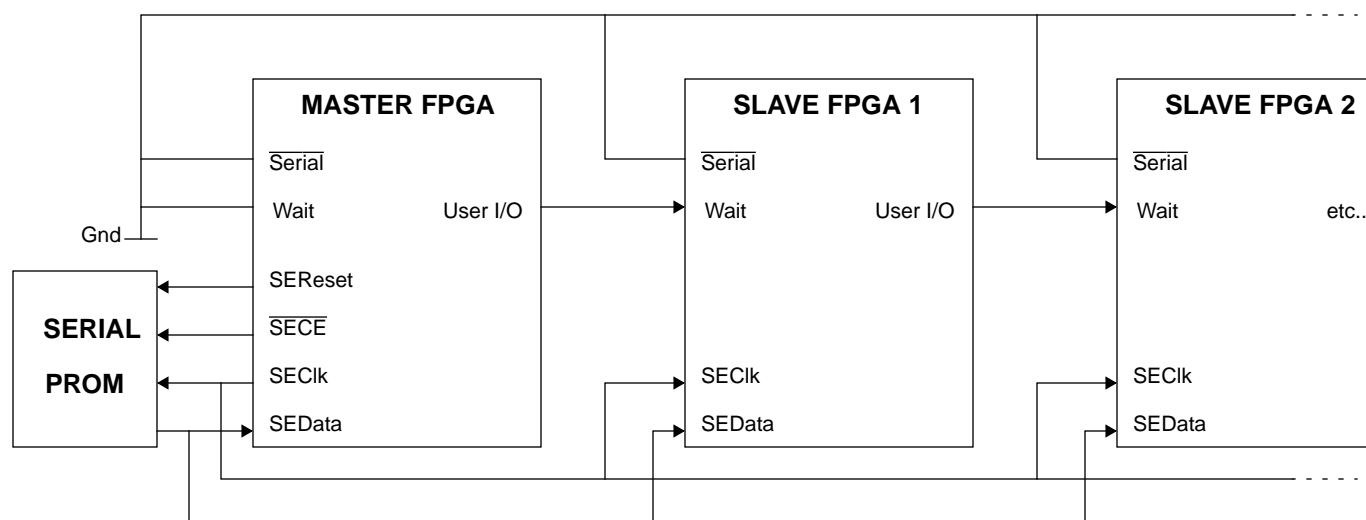


Figure 27 Master-Slave Serial Configuration

ate code to the Device Configuration Register. Initially  $SEClk$  is 1/16 GClk frequency. It can also be set to 1/8, 1/4 or 1/2 GClk.

An example is shown in **Figure 28**. Data1 is loaded into Addr1 after the address lsb has been shifted in. In this example the first write was to the Device Configuration Register and the data bus width was changed from 8 to 32 bits. Data word 2 starts immediately after Addr1 has been shifted in. Due to the new data bus width, 32 data bits are shifted in. If the width had not been changed data word 2 would also have been 8 bits. Data continues to be loaded until  $\overline{Serial}$  goes High or  $Wait$  goes High.

## Serial Interface State Machine

The serial interface is controlled by a state machine. This is a synchronous state machine with the state transitions occurring on rising edges of GClk. The transitions between states are mainly controlled by the  $\overline{Serial}$  and  $Wait$  inputs. These signals are internally retimed, so it is the value sampled on the rising GClk edge **prior** to the edge causing the state transition which is used to determine the next state. A state diagram is shown in **Figure 29**. Transitions which show the state machine remaining in a state are not shown. The behavior is described below:

## Reset

Master = False - i.e. initially this FPGA is not the Master.

Device remains in this state for  $T_{MRR}$  after  $\overline{Reset}$  has been deasserted. The 4th rising GClk edge after  $T_{MRR}$  causes the transition to State 0. It is also this 4th clock edge which samples  $\overline{Serial}$  and  $Wait$  to determine which state to enter after State 0.

Goto State 0

**State 0** : Initial State - Determine if this is the Master FPGA.

$\overline{SECE}$  inactive,  $SEClk$  Low.

```

if ( $\overline{Serial}$  = 1) then
    goto State 1    (Parallel)
else if ( $Wait$  = 1) then
    goto State 4    (Wait)
else
    goto State 2    (Master Initialize)
    
```

**State 1** : Parallel - Ready to load parallel data via CPU interface.

If Master,  $\overline{SECE}$  inactive,  $SEClk$  Low.

If  $Wait=1$  in this state, and the outputs are disabled (either because  $\overline{OE}=1$  or the ID registers have not yet been written), all the FPGA I/Os enter Leakage Test mode. This mode is for

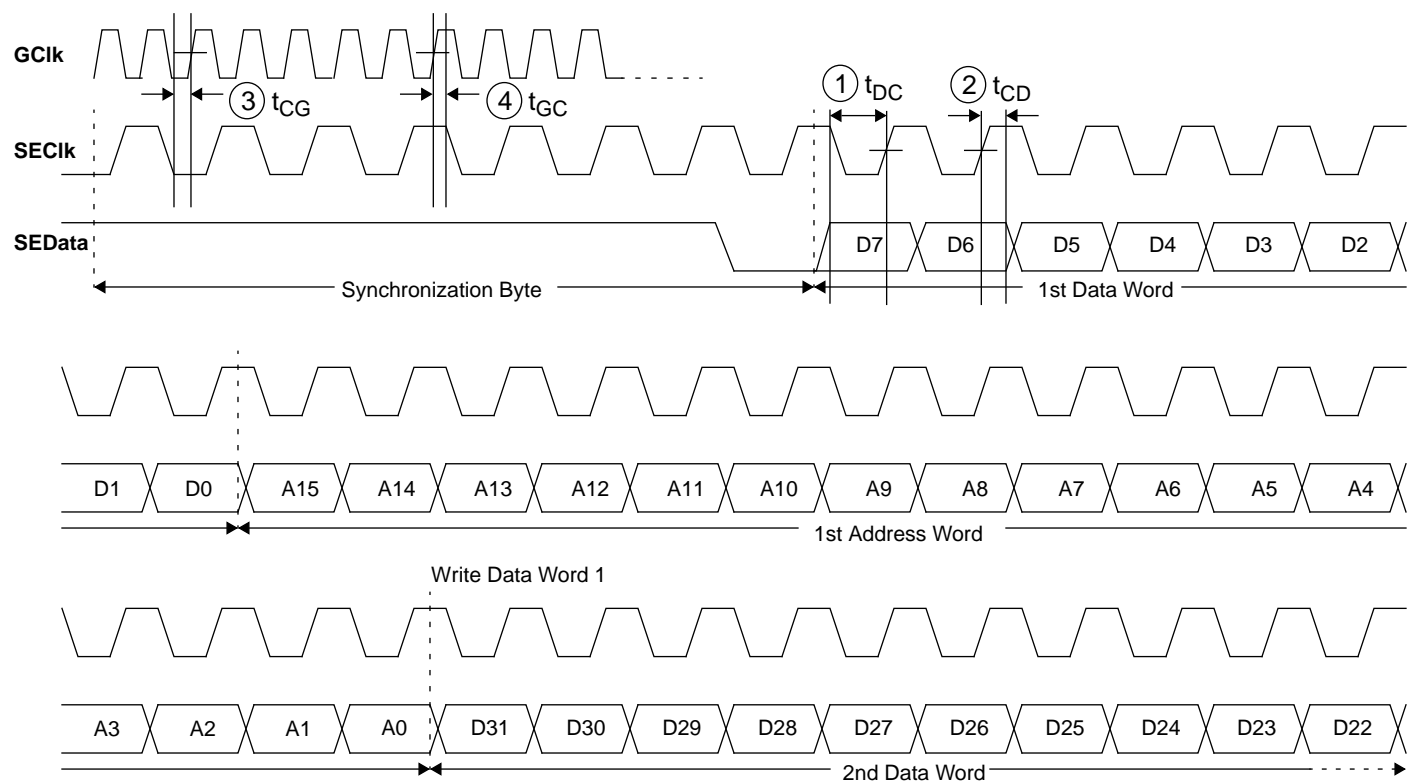


Figure 28. Serial Configuration Timing

factory testing and should not be used. All pull-up's and pull-down's are disabled in this mode.

```

if ( $\overline{Serial}$  = 1) then
    goto State 1      (Remain in this state)
else if ( $Wait$  = 1) then
    goto State 4      (Wait)
else if (Master) then
    goto State 5      ( $\overline{SECE}$  Delay -  $\overline{Serial}$  = 0 &  $Wait$  = 0)
else
    goto State 6      (Sync. Wait -  $\overline{Serial}$  = 0 &  $Wait$  = 0)

```

**State 2** : Master Initialize - FPGA assumes it is the Master and resets Serial PROM.

$\overline{SECE}$  inactive,  $SEClk$  Low.

Master = True.

Assert SEReset.

Wait 16 GClk cycles (Note - this must be greater than the minimum reset time specified in the Xilinx databook Serial PROM section ( $T_{HOE}$ )).

Deassert SEReset.

```

if ( $Wait$  = 1) then
    goto State 4      (Wait)
else
    goto State 5      ( $\overline{SECE}$  Delay)

```

**State 3** : Load Data - Shift in serial data and load.

If Master then  $\overline{SECE}$  active,  $SEClk$  toggles.

```

if ( $Data\_Loaded$  = 0) then      ( $Data\_Loaded$  is asserted at
                                end of each addr/data pair)
    goto State 3                (Wait until complete addr/
                                data pair loaded)
else if ( $\overline{Serial}$  = 1) then
    goto State 1                (Parallel)
else if ( $\overline{Serial}$  = 0 &  $Wait$  = 1) then
    goto State 4                (Wait)

```

**State 4** : Wait - Still in serial mode but idle.

If Master then  $\overline{SECE}$  active,  $SEClk$  toggles.

```

if ( $\overline{Serial}$  = 1) then
    goto State 1                (Parallel)
else if ( $Wait$  = 1) then
    goto State 4                (Remain in this state until Wait
                                deasserted)
else

```

goto State 6 (Sync. Wait)

**State 5** :  $\overline{SECE}$  Delay - Assert  $\overline{SECE}$  and wait for serial PROM to output data (Master only).  $SEClk$  Low.

Assert  $\overline{SECE}$

Wait 9 GClk cycles (Note - this must be greater than the minimum  $\overline{CE}$  to Data time specified in the Xilinx databook ( $T_{CE}$ ))

goto State 6 (Sync. Wait)

**State 6** : Sync. Wait - Wait for valid synchronization byte to be shifted in.

If Master then  $\overline{SECE}$  active,  $SEClk$  toggles.

The FPGA actually looks for the pattern '1110' so it doesn't matter if the first few bits are missed.

```

if ( $\overline{Serial}$  = 1) then      (Can escape out of this state by
                            returning to parallel mode)
    goto State 1            (Parallel)
else if (Valid_Sync_Pattern_In) then
    goto State 3            (Load Data)
else
    goto State 6            (Wait for valid sync byte)

```

The Master toggles  $SEClk$  during States 3, 4 and 6, otherwise it is held Low.  $\overline{SECE}$  is driven Low by the Master in States 3, 4, 5 and 6.

## Typical Serial Load Sequence For Chain Of FPGAs

Central to the operation of serial loads is XC6200's ability to control the values seen on 'dedicated' I/O pins, such as  $Wait$ , by simple writes to IOB addresses. The following sequence is just one possible sequence. XC6200 allows the serial interface to write to any memory mapped RAM location, therefore it is completely flexible.

- 1) All FPGAs are reset and enter State 0.
- 2) FPGAs interrogate  $\overline{Serial}$  and  $Wait$  inputs to see if they are the Master.
- 3) Master FPGA enters State 2, Slaves State 4.
- 4) Master resets serial PROM. Slaves wait in State 4.
- 5) Master enters State 5 and enables serial PROM.
- 6) Master enters State 6 and waits for a Sync pattern.
- 7) Master enters State 3 and starts loading data. The last data word to be loaded modifies the Master's

*Wait* IOB so that *Wait* goes High. This write also modifies the configuration to deassert the signal driving the *Wait* input of the next FPGA in the chain, causing the first Slave to enter State 6.

Note that if *SEClk* is set to *GClk/2* rate then one more word is always loaded after the write which causes the transition from State 3 to 4. In this case the second last data word asserts the Master's *Wait* input and the last data word modifies the configuration to deassert the signal driving the *Wait* input of the next FPGA in the chain, causing the first Slave to enter State 6.

- 8) Master enters State 4 but still drives *SEClk* and *SECE*. Slave waits for Sync pattern.
- 9) Slave receives Sync pattern, enters State 3 and starts loading data.
- 10) First Slave completes loading data. The last data word modifies the Slave's own *Serial* IOB so it enters State 1 after the next word. The last data word also causes the *Wait* input of the next Slave in the chain to be de-asserted. Again, if *SEClk*=*GClk/2*, this is performed with two separate writes.
- 11) First Slave enters State 1 and is ready to receive parallel CPU interface cycles. Next Slave enters

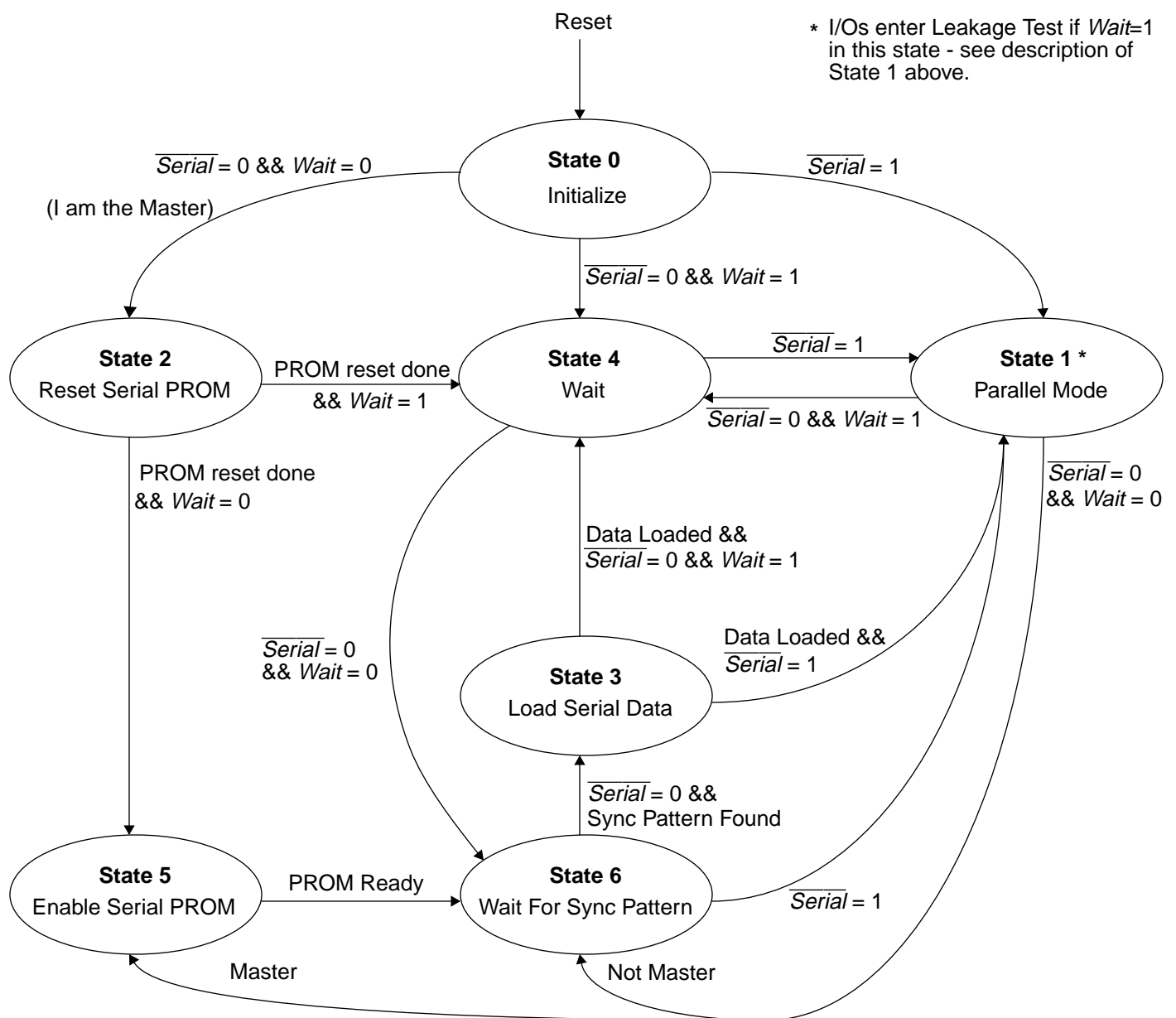


Figure 29. Serial Interface State Diagram

State 6 and waits for Sync pattern. The above sequence from step 8 is repeated until the last FPGA in the chain is reached.

- 12) The last FPGA could simply return to parallel mode (State 1) and leave the Master in State 4. If the Master is also to be returned to parallel mode, the last Slave must cause the Master's *Serial* input to go High. There are a number of ways of doing this. One way is for one of the last Slave's I/Os to be routed back to the Master and used to control the Master's *Serial* IOB. The last write to the last Slave would deassert this signal, causing the Master's *Serial* IOB to go High.
- 13) Master enters State 1, stops *SEClk* and deasserts *SECE*.

## Reset And Initialization

When the XC6200 is powered up or after a reset, all configuration memory is cleared and the cell state registers are cleared. XC6200 does not require the *Reset* pin to be active during or after power-up to initialize itself. To avoid potential high current random configurations, the power-up reset is carried out automatically. The automatic power-up initialization takes  $T_{PRR}$  (see [page 39](#)). All the XC6200 I/O pads are disabled during this time and it is impossible to access the device. The XC6200 may be re-initialized at any time by asserting the *Reset* input for a minimum of  $T_{WMR}$ . This acts as a signal to the chip to initialize itself. This initialization occurs **after** *Reset* has been deasserted. Therefore there is a  $T_{MRR}$  reset recovery time when no device accesses are possible. The 5th rising *GClk* edge after this reset recovery period brings the state machine (on [page 33](#)) into State 1, 2 or 4, depending on whether Serial or Parallel Mode is selected. It is not possible to start parallel write cycles until State 1 is entered.

XC6200 devices initialize to the following configuration:

- 1) All control registers are cleared to 0. This disables all row and column wildcarding. *GClk* to the cell array is disabled. Inputs have TTL logic thresholds. The data bus width is 8 bits. The Map Register contains all 0s, therefore the eight data bus bits are applied to the entire column of state registers addressed. The serial configuration speed is *GClk*/16 (the slowest possible).
- 2) Cell function units are configured as registers with register protect off. Registers are cleared to '0'. The register clock inputs are held at a constant value to reduce power consumption. Thus the output of all function units is a constant 0.
- 3) All neighbor multiplexers select the function unit output so all cell neighbor outputs are constant.
- 4) Cell X1, X2 and X3 multiplexers select the South input. With the exception of the top row of cells, this is a cell neighbor output, and hence constant. In the top row of cells this signal comes from the North IOB and is connected to PN4In. This is the N4 output from the top cell in the array which is also a constant.
- 5) Cell magic outputs are sourced from X3 and hence are constant. Magic signals at the edge of the array are grounded.
- 6) Array N4 multiplexers select SC which is a cell output and hence constant. The N4 multiplexer in the South IOB selects NCL which is 0.
- 7) Array S4 multiplexers select MN which is a magic signal and hence constant. The S4 multiplexer in the North IOB selects SCL which is 0.
- 8) Array E4 multiplexers select MW which is a magic signal and hence constant. The E4 multiplexer in the West IOB selects ECL which is 0.
- 9) Array W4 multiplexers and the W4 multiplexer in the East IOB select WCL which is 0.
- 10) Chip-Length wires from the IOBs all select '0'.
- 11) N16 multiplexers in the array and the South IOB select NCL which is 0.
- 12) S16 multiplexers in the array and the North IOB select SCL which is 0.
- 13) E16 multiplexers in the array and the West IOB select ECL which is 0.
- 14) W16 multiplexers in the array and the East IOB select WCL which is 0.
- 15) Clear multiplexers in the array and North IOB select *GClr*. *GClr* is asserted by initialize logic independent of the *GClr* pin.
- 16) Clock multiplexers in the array and South IOBs select NCL = 0. This means the state register clocks are constant.
- 17) Enable, PUp and PDwn signals to the I/O pads are not acted upon until the device identification register has a valid pattern, so initial values are irrelevant.
- 18) All I/O pads function as inputs. Pull-up resistors on pads are active until the device identification register is correctly loaded so undriven control inputs are pulled High.



Packaging

This section contains pin identification for the XC6200 family. Devices are available in small and large packages. The small packages are useful where board area is at a premium and the design can make use of the wireless I/O FastMAP™ CPU interface to determine the state of internal nodes. The large package options give a very high user programmable I/O count where this is a prime requirement. The available packaging options for the XC6216 are summarized in Table 32. These options are advance information and subject to change. Please confirm availability with Xilinx.

Package	Pins	Signal Pins	Max Data Bus Pins	Unshared User I/O
PC	84	68	16	22
HT	144	120	16	72
BG	225	198	32	134
HQ	240	199	32	137
PG	299	242	32	180
HQ	304	242	32	180

Table 32: XC6216 Package Options

Signal pins are all the non-supply pins which drive into the array or control circuitry. Some of these pins are shared between control signals and user I/O. The unshared user I/Os do not share a pin with a control signal. The number of user I/Os available is somewhere between the number of signal pins and the number of un-shared I/Os, depending on how many of the FPGA control signals are actually required. For example, if only an 8-bit data bus and no serial interface were required, the number of user I/Os would go up by 24+6=30 in a PGA299 package.

Precise pin-out information for all package types is given in the Device Pin-Outs section starting on page 45.

Pin Descriptions

The pins are labeled as follows:

V<sub>CC</sub> - Supply

Connections to the nominal +5V supply. All must be connected.

GND - Supply

Connections to ground. All must be connected.

$\overline{CS}$  - Input

Chip Select enables the programming circuitry and initiates address decoding. When  $\overline{CS}$  is Low data can be read from or written to the control memory. This signal is intended to be used in conjunction with address decoding circuitry to select one part within a larger array for programming.

D<d:0> - Bidirectional

(d+1)-bit bidirectional data bus. Used for device configuration and direct cell register access.

A<a:0> - Input

Address bus for CPU access of internal registers and configuration memory. 'a' varies between family members.

Rd $\overline{Wr}$  - Input

When  $\overline{CS}$  is Low this signal determines whether data is read from or written to the control memory. If Rd $\overline{Wr}$  is High then a read cycle takes place. If Rd $\overline{Wr}$  is Low then a write cycle takes place.

GClk, GC $\overline{I}r$ , G1, G2 - Inputs

Global signals. GClk should be used for global user clocks, GC $\overline{I}r$  for global user clears and G1 and G2 for other global, low-skew signals. The GClk pin is **always** configured as an input and **cannot** be used as a fully flexible User I/O like the majority of other control signals.

All the usual IOB input functionality is available except it is not possible to route a user signal onto the GClk Control Signal using DfPB (Figure 15 and Table 18) as the GClk is routed to the built-in control logic directly from the pad. This differs from the GClk signal supplied to the user array, which comes from the GClk IOB and can have a user signal routed onto it in the usual way. See Figure 18 for details.

$\overline{Reset}$  - Input

When  $\overline{Reset}$  is driven Low the programming registers (mask unit and address wildcard unit) are re-initialized, resulting in the XC6200 appearing as a conventional SRAM. The control store of the cell array is initialized into a low power consumption configuration. All programmable output pad enable signals are forced inactive. All the IO-pad pull-up resistors are also enabled. As with GClk, this pin is **always** configured as an input and **cannot** be

used as a fully flexible User I/O like the majority of other control signals.

---

#### $\overline{OE}$ - Input

When this signal is High the outputs of all programmable I/O pads are forced into a high impedance state (independent of the contents of the control store). All the IO-pad pull-up resistors are also enabled. As with GClk, this pin is **always** configured as an input and **cannot** be used as a fully flexible User I/O like the majority of other control signals.

---

#### $\overline{Serial}$ - Input

Input which controls transitions between states in serial state machine.  $\overline{Serial}$  **must** be Low initially to enter Serial mode or High to enter Parallel mode. After configuration,  $\overline{Serial}$  may be driven internally from the logic array to leave the IOB free for use as a general purpose I/O.

---

#### Wait - Input

Input which controls transitions between states in serial state machine. In Parallel mode *Wait* **must** be Low whilst initially configuring. This avoids putting the I/Os in Leakage Test mode (see “[Serial Interface State Machine](#)” on page 31). After configuration, *Wait* may be driven internally from the logic array to leave the IOB free for use as a general purpose I/O.

0 => continue loading, 1 => pause until Wait deasserted

---

#### SEReset - Output

Output from Master FPGA which resets serial PROM address counter.

---

#### $\overline{SECE}$ - Output

Output from Master FPGA which enables serial PROM output.

---

#### SEClk - Bidirectional

Output from Master FPGA which clocks serial PROM and slave FPGAs. On Slave FPGAs this is an input which is used to sample *SEData*.

---

#### SEData - Input

Serial data input to FPGA. This is sampled in the FPGA by *SEClk* and retimed by the FPGA's own GClk.

---

#### ConfigOK - Internal

Signal is active (High) when a valid pattern is present in the ID register and inactive when the pattern is invalid. ConfigOK cannot be directly routed onto the S31 pin as the Control Enable for S31 is permanently disabled. However it is available as 'DToPadB' in [Table 19](#) and can be routed onto the N4 or N16 output from the S31 IOB.

---

#### ShiftDOut - Internal

MSB of input shift chain during serial loads. Control Enable for this signal is permanently disabled, so it can only be observed by routing through another IOB as with *ConfigOK*.

---

#### $N_x$ - Bidirectional

North I/Os. Connections to I/O Blocks on the north of the array.

---

#### $S_x$ - Bidirectional

South I/Os. Connections to I/O Blocks on the south of the array.

---

#### $E_x$ - Bidirectional

East I/Os. Connections to I/O Blocks on the east of the array.

---

#### $W_x$ - Bidirectional

West I/Os. Connections to I/O Blocks on the west of the array

## Electrical Parameters

The XC6200 series is fabricated in triple-metal n-well CMOS process.

As with all CMOS devices, care must be exercised when handling, since inputs can be damaged by static discharge, although standard circuit design procedures have been used to minimize this risk.

Internal to the XC6200 devices,  $V_{CC}$  and ground are distributed on grids, which ensure minimal voltage drops between supply pads and internal circuitry. Power to the I/O cells is distributed in separate  $V_{CC}$  and ground rings.

All  $V_{CC}$  and GND pins must be connected to their respective PC-board plane. At least one low-inductance decoupling capacitor is recommended for each  $V_{CC}$  pin. A typical value is 0.1  $\mu F$ . It must be placed very close to the  $V_{CC}$  lead, with a very short connection to the ground plane.



The power-supply current of an XC6200 device can vary between a few tens to several hundreds of milliamps, depending on the implemented design and the input activity. Power consumption is almost exclusively dynamic, and caused by the switching activity of the internal logic and the device outputs. For each internal node, power consumption is proportional to the switching frequency of that node.

The user must consider power consumption, package thermal characteristics, and ambient temperature to calculate the maximum chip (junction) temperature. Above 85°C, the internal delays increase by 0.35% per degree. See the Packages and Thermal Characteristics section of the Xilinx Data Book.

Simultaneous switching of many outputs can cause ground-bounce effects. The Product Technical Information section of the Xilinx Data Book discusses this.

Notice: The information contained in this data sheet pertains to products just entering production. These specifications are subject to change without notice. Verify with your local Xilinx sales office that you have the latest data sheet before finalizing a design. All the parameters given in the following tables are for the **XC6216 only**.

## Absolute Maximum Ratings

Symbol	Parameter	Value	Units
$V_{CC}$	Supply voltage with respect to GND	-0.5 to 7.0	V
$V_{IN}$	DC Input voltage with respect to GND	-0.5 to $V_{CC}+0.5$	V
$V_{TS}$	Voltage applied to 3-state output with respect to GND	-0.5 to $V_{CC}+0.5$	V
$T_{STG}$	Storage temperature	-65 to +150	°C
$T_{SOL}$	Maximum soldering temperature (10s @ 1/16 in. = 1.5 mm)	+260	°C
		<b>Advance</b>	

**Warning:** Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Recommended Operating Conditions is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability.

## Recommended Operating Conditions

Symbol	Parameter	Min	Max	Units
$V_{CC}$	Supply voltage relative to GND    Commercial $T_A = 0^\circ\text{C}$ to $85^\circ\text{C}$ junction	4.75	5.25	V
	Supply voltage relative to GND    Industrial $T_A = -40^\circ\text{C}$ to $100^\circ\text{C}$ junction	4.50	5.50	V
$V_{ILT}$	Low-level input voltage    - TTL configuration	0	0.80	V
$V_{IHT}$	High-level input voltage    - TTL configuration	2.0	$V_{CC}$	V
$V_{ILC}$	Low-level input voltage    - CMOS configuration	0	20%	$V_{CC}$
$V_{IHC}$	High-level input voltage    - CMOS configuration	70%	100%	$V_{CC}$
$T_{IN}$	Input signal transition time	-	250	ns
		<b>Advance</b>		

## DC Characteristics Over Recommended Operating Conditions

Symbol	Parameter	Test Conditions	Min	Max	Units
$V_{OH}$	High-level output voltage	$I_{OH} = -8.0 \text{ mA}$ $V_{CC} = \text{Min}$	3.86	-	V
$V_{OL}$	Low-level output voltage	$I_{OL} = 8 \text{ mA}$ $V_{CC} = \text{Max}$	-	0.4	V
$I_{IL}$	Input leakage current	$V_{CC} = \text{Max}$ $V_{IN} = \text{GND or } V_{CC}$	-1	1	$\mu\text{A}$
$C_{IN}$	Input capacitance for Input and I/O pins	$V_{IN} = \text{GND}$	-	15	pF
$I_{CC}^{(2)}$	Quiescent Supply Current	$V_{IN} = V_{CC} \text{ or GND}$ $V_{CC} = 5 \text{ V}$	-	12	mA
Advance					

Notes: 1. Sample tested.

2. Measured with no output loads, no active input pull-up resistors and all package pins at  $V_{CC}$  or GND.

## Power-up/Reset Timing Parameters

		Speed Grade: -2			Units
Symbol	Parameter	Min	Typ	Max	
$T_{WMR}$	Master Reset input Low pulse width	5	-	-	ns
$T_{MRR}$	Recovery time after Master Reset deasserted	460	-	-	ns
$T_{PRR}$	Recovery time after power up <sup>(1)</sup>	-	-	1.5	$\mu\text{s}$
Advance					

Notes: 1. Power up is defined as the time when  $V_{CC}$  has reached a stable level within the Min/Max limits specified in the "Recommended Operating Conditions" table.

## AC Characteristics

### Global Buffer Switching Characteristic Guidelines

		Speed Grade:		Units
Symbol	Parameter	Min	Max	
T <sub>PGClk</sub>	From pad through <b>GClk</b> buffer to any register clock	-	10	ns
T <sub>PG</sub>	From pad through <b>G1,G2,GClr</b> buffers to any register clock	-	10	ns
T <sub>PClr</sub>	From pad through <b>global</b> buffers to any register clear	-	12	ns
T <sub>PCkS</sub>	Skew between any pair of register clocks using the same <b>global</b>	-	0.9	ns
T <sub>PCIS</sub>	Skew between any pair of register clears using the same <b>global</b>	-	0.9	ns
		Advance		

Notes: 1. Typical loading values are used.

### Guaranteed Input and Output Parameters (Pin-to-Pin)

All values listed below are tested directly and guaranteed over all the operating conditions. The same parameters can also be derived indirectly from the IOB and Global Buffer specifications. The delay calculator software uses this indirect method. When there is a discrepancy between these two methods, the directly tested values listed below should be used and the derived values should be ignored.

Speed Grade:		-2				Units
Symbol	Parameter	Best I/O <sup>(5)</sup>		Worst I/O		
		Min	Max	Min	Max	
T <sub>ICKOF</sub>	Global Clock (GClk) to Output (fast)	-	16	-	19	ns
T <sub>ICKO</sub>	Global Clock (GClk) to Output (slew limited)	-	17	-	20	ns
T <sub>PSUF</sub>	Input Set-up Time (fast)	-	-	4	-	ns
T <sub>PSU</sub>	Input Set-up Time with delay	-	-	10	-	ns
T <sub>PHF</sub>	Input Hold Time (fast)	-	-	6	-	ns
T <sub>PH</sub>	Input Hold Time with delay	-	-	0	-	ns
T <sub>OEHz</sub>	$\overline{OE}$ deasserted (High) to Pad begin hi-Z (slew rate independent)	-	TBA	-	TBA	ns
T <sub>OEONF</sub>	$\overline{OE}$ asserted (Low) to Pad active and valid (fast)	-	TBA	-	TBA	ns
T <sub>OEONS</sub>	$\overline{OE}$ asserted (Low) to Pad active and valid (slew rate limited)	-	TBA	-	TBA	ns
		Advance				

- Notes: 1. All appropriate ac specifications tested using +/-8mA test load.  
2. These parameters are tested directly and guaranteed over the operating conditions.  
3. As the parameters vary between I/Os, values are given for best and worst I/Os. The parameters for other I/Os are somewhere between these two extremes. The delay calculator software calculates the correct value for each I/O used.  
4. All parameters assume the cell register is the closest one to the IOB.  
5. The difference between Best I/O and Worst I/O is caused by differing IOB to Pad delays.

## IOB Switching Characteristic Guidelines

Speed Grade:		-2				
Symbol	Parameter	Best I/O		Worst I/O		Units
		Min	Max	Min	Max	Units
T <sub>PID</sub>	<b>INPUT</b> Pad to Neighbor data (NO-DELAY)	-	TBA	-	5	ns
T <sub>PID4</sub>	Pad to L4 FastLANE™ (MID-DELAY)	-	TBA	-	6	ns
T <sub>PDID4</sub>	Pad to L4 FastLANE™ with delay	-	TBA	-	14	ns
T <sub>OPF</sub>	<b>OUTPUT</b> Neighbor data to Output (fast)	-	TBA	-	6.5	ns
T <sub>OPS</sub>	Neighbor data to Output (slew rate limited)	-	TBA	-	8	ns
T <sub>TSHZ</sub>	3-state to Pad begin hi-Z (slew rate independent)	-	TBA	-	4	ns
T <sub>TSO NF</sub>	3-state to Pad active and valid (fast)	-	TBA	-	5	ns
T <sub>TSO NS</sub>	3-state to Pad active and valid (slew rate limited)	-	TBA	-	7	ns
		Advance				

- Notes: 1. As the parameters vary between I/Os, values are given for best and worst I/Os. The parameters for other I/Os are somewhere between these two extremes. The delay calculator software calculates the correct value for each I/O used.  
2. Typical loading values are used.

## Cell Switching Characteristic Guidelines

		Speed Grade:		-2		Units
Symbol	Parameter	Min	Max	Min	Max	
T <sub>ILO1</sub>	X1 change to Function Output <sup>(1)</sup>	-	2			ns
T <sub>ILO23</sub>	X2/X3 change to Function Output <sup>(2)</sup>	-	2.5			ns
T <sub>ICK1</sub>	Internal Register Set-Up Time @ X1 <sup>(1)</sup>	3	-			ns
T <sub>ICK23</sub>	Internal Register Set-Up Time @ X2/X3 <sup>(2)</sup>	3	-			ns
T <sub>IHCK1</sub>	Internal Register Hold Time @ X1 <sup>(1)</sup>	0	-			ns
T <sub>IHCK23</sub>	Internal Register Hold Time @ X2/X3 <sup>(2)</sup>	0	-			ns
T <sub>CH</sub>	Clock High Time <sup>(3)</sup>	2	-			ns
T <sub>CL</sub>	Clock Low Time <sup>(3)</sup>	2	-			ns
T <sub>CLW</sub>	Clear Pulse Width <sup>(3)</sup>	1	-			ns
T <sub>CKO</sub>	Clock to Function Output	-	2.5			ns
T <sub>CKLO</sub>	Clock to Function Output via X2/X3 feedback multiplexers	-	3.5			ns
<b>Advance</b>						

- Notes: 1. Data input measured at input to X1 routing multiplexer. Clock input measured at register.  
2. Data input measured at input to X2/X3 routing multiplexers. Clock input measured at register.  
3. Measured at the actual register in the cell.  
4. Typical loading values are used.

## Internal Routing Delays

		Speed Grade:		Units
Symbol	Parameter	Min	Max	
$T_{NN}$	Route Neighbor In to Neighbor Out	-	1	ns
$T_{Magic}$	Route X2/X3 to Magic Out	-	1.5	ns
$T_{L4}$	Length-4 FastLANE™ delay	-	1.5	ns
$T_{L16}$	Length-16 FastLANE™ delay	-	2	ns
$T_{CL64}$	Chip-Length (64) FastLANE™ delay	-	3	ns
Advance				

Notes: 1. Delays vary depending on direction. Typical figures are given here. The delay calculator software calculates the correct delay for each direction.  
2. Typical loading values are used.

## CPU Interface Timing

		Speed Grade:		Units
Symbol	Parameter	Min	Max	
1 $T_{su\overline{CS}}$	$\overline{CS}$ set up before Clock <sup>(1)</sup>	8	-	ns
2 $T_{h\overline{CS}}$	$\overline{CS}$ hold after Clock <sup>(1)</sup>	0	-	ns
3 $T_{suRd\overline{Wr}}$	$Rd\overline{Wr}$ set up before Clock	3	-	ns
4 $T_{hRd\overline{Wr}}$	$Rd\overline{Wr}$ hold after Clock	0	-	ns
5 $T_{suA}$	Address Bus set up before Clock	2	-	ns
6 $T_{hA}$	Address Bus hold after Clock	0	-	ns
7 $T_{suD}$	Data Bus set up before Clock	3	-	ns
8 $T_{hD}$	Data Bus hold after Clock	0	-	ns
9 $T_{WC(ram)}$	Configuration SRAM Write cycle time <sup>(2)</sup>	30	-	ns
10 $T_{RC(ram)}$	Configuration SRAM Read cycle time <sup>(2)</sup>	40	-	ns
11 $T_{WC(reg)}$	State Write cycle time <sup>(2,3)</sup>	30	-	ns
12 $T_{RC(reg)}$	State Read cycle time <sup>(2,3)</sup>	56	-	ns
13 $T_{CKD}$	Clock to Valid Data	-	16	ns
14 $T_{CKDZ}$	Clock to Data high impedance <sup>(4)</sup>	-	18	ns
15 $T_{\overline{CS}DZ}$	$\overline{CS}$ to Data high impedance <sup>(4)</sup>	-	18.5	ns
16 $T_{MPST}$	Map Register Settling Time <sup>(5)</sup>	-	140	ns
Advance				

Notes: 1.  $\overline{CS}$  must be correctly sampled Low at the start of the cycle ( $t_1$ ) and sampled High at the end of the cycle ( $t_2$ ). Other signals only require to be correctly sampled at  $t_1$ .  
2. The minimum time for a read or write cycle is two CPU clock periods, although the cycles shown do not start and finish at the start of a clock period. A 50% GCLK duty cycle is assumed.  
3. The cycle time for state accesses differs from the time for configuration SRAM accesses.  
4. Data is removed from the bus  $T_{CKDZ}$  after  $t_3$  unless  $\overline{CS}$  is still asserted at this time. In this case, data is removed from the bus asynchronously  $T_{\overline{CS}DZ}$  after  $\overline{CS}$  goes High.  
5. After a modification to any of the Map Register bits, it is not safe to perform register writes or reads for  $T_{MPST}$ . This is measured from the first falling clock edge of the Map Register Write Cycle to the **second** rising clock edge of the next register access ( $t_2$  in [Figures 30 and 31](#)).



## Serial Interface Timing

Speed Grade:			-2		
Symbol		Description	Min	Max	Units
1	$T_{DC}$	SEData setup before SEClk rising	3	-	ns
2	$T_{CD}$	SEData hold after SEClk rising	0	-	ns
3	$T_{CG}$	SEClk setup before GClk rising <sup>(2)</sup>	5	-	ns
4	$T_{GC}$	SEClk hold after GClk rising <sup>(2)</sup>	0	-	ns
5	$T_{CKSE}$	GClk rising to SEClkOut <sup>(1)</sup>	12	-	ns
6	$T_{suSerial}$	Serial setup before GClk rising	3	-	ns
7	$T_{hSerial}$	Serial hold after GClk rising	0	-	ns
8	$T_{suWait}$	Wait setup before GClk rising	4	-	ns
9	$T_{hWait}$	Wait hold after GClk rising	0	-	ns
			Advance		

Notes: 1. Only for Master device.

2. Only for Slave device.

## Timing Diagrams

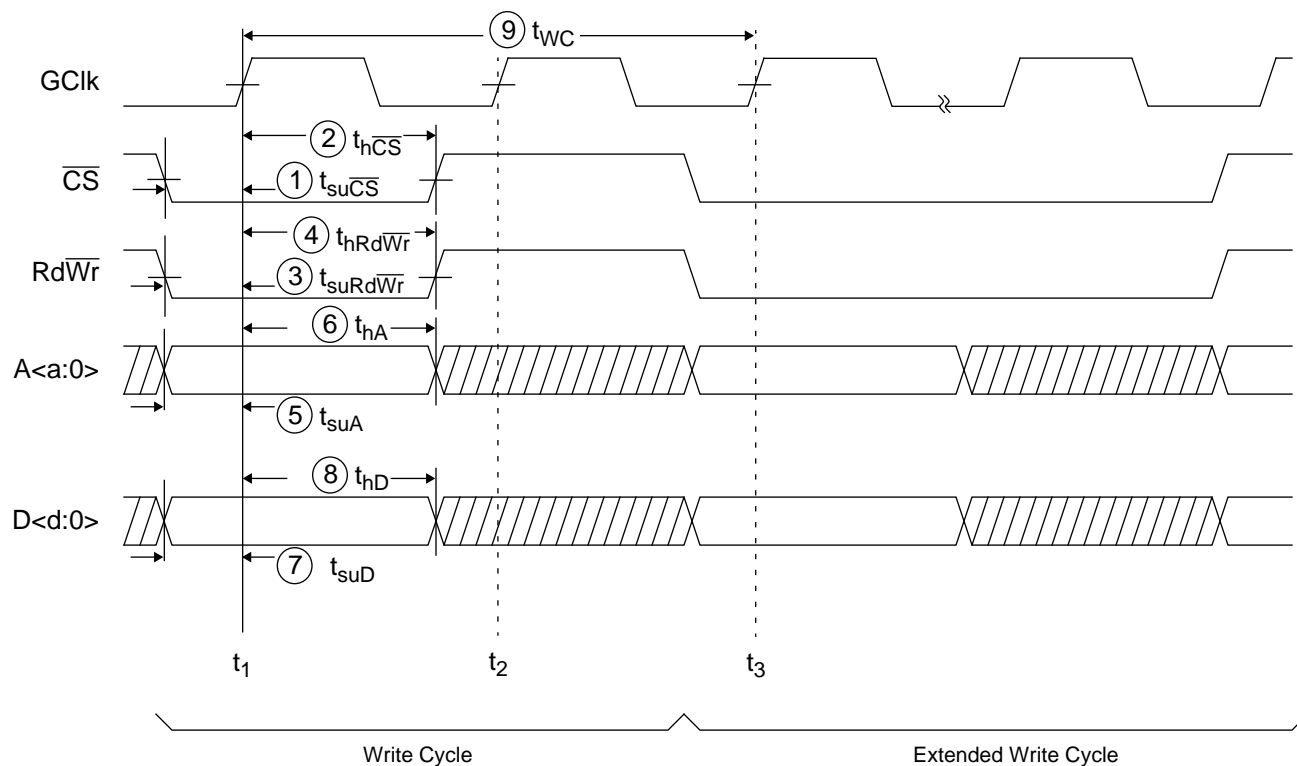


Figure 30. Configuration Memory Write Cycles

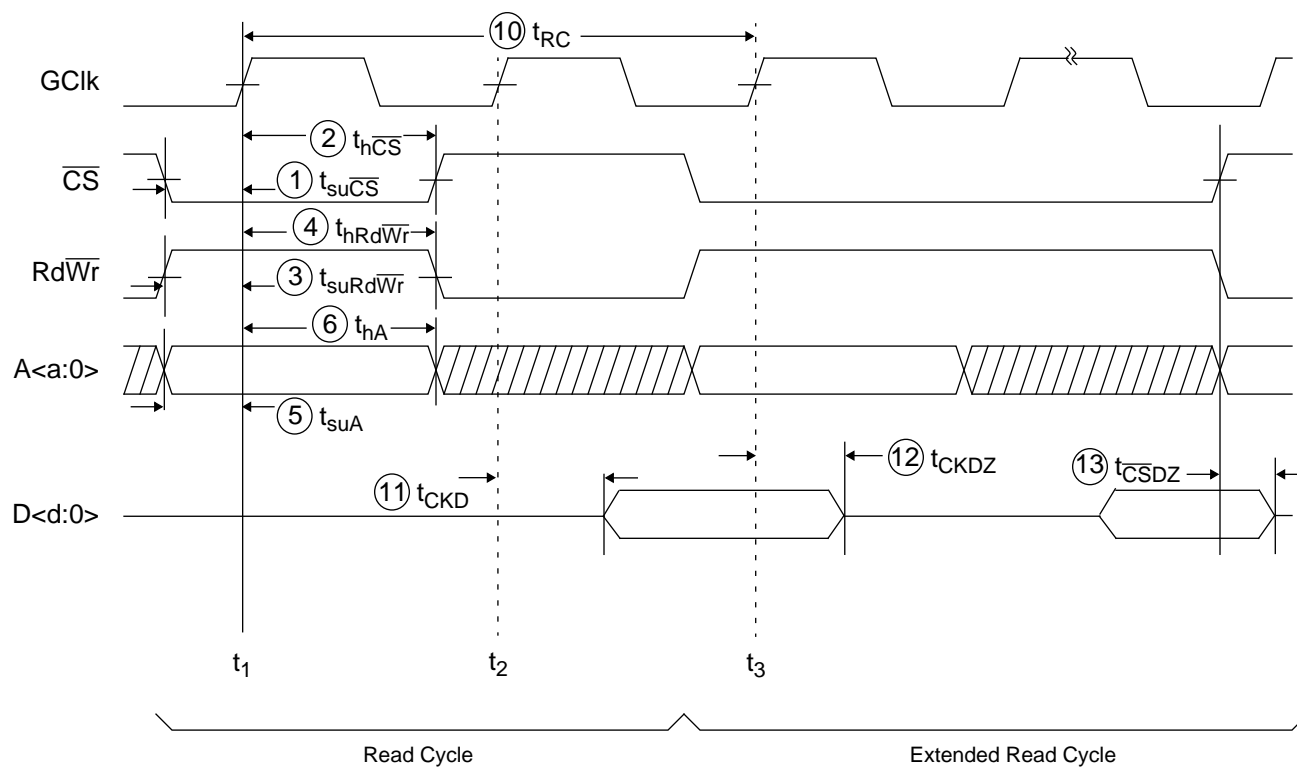


Figure 31. Configuration Memory Read Cycles

## XC6216 Pinouts - West Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
D0/W <sub>1</sub> <sup>(2)</sup>	P32	P36	P2	P60	C18	P229
<b>GND</b>	<b>P31</b>	<b>P35</b>	<b>H10</b>	<b>P59</b>	<b>A19</b>	<b>P230</b>
W <sub>14</sub>	-	P34	N3	P58	A20	P231
NC	-	-	P1 <sup>(1)</sup>	P57 <sup>(1)</sup>	C17 <sup>(1)</sup>	P232 <sup>(1)</sup>
D1/W <sub>3</sub>	P30	P33	K6	P56	D16	P233
W <sub>16</sub>	-	P32	N2	P55	E15	P234
D2/W <sub>5</sub>	P29	P31	M3	P54	B18	P235
W <sub>18</sub>	-	-	N1	P53	B17	P236
D3/W <sub>7</sub>	P28	P30	L4	P52	C16	P237
NC	-	-	-	-	-	P254 <sup>(1)</sup>
NC	-	-	-	-	D15 <sup>(1)</sup>	P238 <sup>(1)</sup>
NC	-	-	-	-	A18 <sup>(1)</sup>	P239 <sup>(1)</sup>
W <sub>20</sub>	-	-	M2	P51	E14	P240
D4/W <sub>9</sub>	P27	P29	K5	P50	C15	P241
W <sub>22</sub>	-	-	M1	P49	B16	P242
W <sub>24</sub>	-	-	L3	P48	D14	P243
W <sub>26</sub>	-	-	K4	P47	A17	P244
D5/W <sub>11</sub>	P26	P28	L2	P46	C14	P245
NC	-	-	-	-	E13 <sup>(1)</sup>	P246 <sup>(1)</sup>
NC	-	-	-	-	B15 <sup>(1)</sup>	P247 <sup>(1)</sup>
<b>GND</b>	-	<b>P27</b>	<b>H9</b>	<b>P45</b>	<b>A15</b>	<b>P248</b>
W <sub>28</sub>	-	P26	L1	P44	D13	P249
W <sub>30</sub>	-	P25	J6	P43	B14	P250
W <sub>32</sub>	-	-	K3	P42	C13	P251
D6/W <sub>13</sub>	P25	P24	K2	P41	A14	P252
<b>V<sub>CC</sub></b>	-	-	-	<b>P40</b>	<b>A16</b>	<b>P253</b>
D16/W <sub>33</sub>	-	P23	K1	P39	B13	P255
W <sub>34</sub>	-	-	J5	P38	E12	P256
<b>GND</b>	-	-	-	<b>P37</b>	<b>E16</b>	-
NC	-	-	-	-	D12 <sup>(1)</sup>	P257 <sup>(1)</sup>
NC	-	-	-	-	C12 <sup>(1)</sup>	P258 <sup>(1)</sup>
NC	-	-	-	-	A13 <sup>(1)</sup>	P259 <sup>(1)</sup>
NC	-	-	-	-	B12 <sup>(1)</sup>	P260 <sup>(1)</sup>
D7/W <sub>15</sub>	P24	P22	J4	P36	A12	P261
D17/W <sub>35</sub>	-	-	J3	P35	D11	P262
W <sub>36</sub>	-	-	J1	P34	E11	P263
D18/W <sub>37</sub>	-	P21	J2	P33	C11	P264
D8/W <sub>17</sub>	P23	P20	H5	P32	B11	P265
W <sub>38</sub>	-	P19	H4	P31	B10	P266

Note: 1. Pin not connected.

2. Pins with a dual function have the 'Control' signal shown first. See section "I/O Architecture" on page 10 for details.

## XC6216 Pinouts - West Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>V<sub>CC</sub></b>	<b>P22</b>	<b>P18</b>	<b>H1</b>	<b>P30</b>	<b>A11</b>	<b>P267</b>
<b>GND</b>	<b>P21</b>	<b>P17</b>	<b>H2</b>	<b>P29</b>	<b>A10</b>	<b>P268</b>
D19/W <sub>39</sub>	-	P16	H3	P28	C10	P269
W <sub>40</sub>	-	-	-	-	D10	P270
D9/W <sub>19</sub>	P20	P15	G5	P27	A9	P271
D20/W <sub>41</sub>	-	P14	G1	P26	E10	P272
W <sub>42</sub>	-	-	-	-	B9	P273
D21/W <sub>43</sub>	-	-	G2	P25	C9	P274
D10/W <sub>21</sub>	P19	P13	G3	P24	A8	P275
W <sub>44</sub>	-	-	-	-	B8	P276
W <sub>46</sub>	-	-	-	-	D9	P277
D22/W <sub>45</sub>	-	-	G4	P23	A7	P278
<b>GND</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P22</b>	<b>-</b>	<b>-</b>
W <sub>48</sub>	-	-	F1	P21	E9	P279
W <sub>50</sub>	-	-	F2	P20	C8	P280
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P19</b>	<b>A6</b>	<b>P282</b>
D24/W <sub>49</sub>	-	P12	F3	P18	B7	P283
D11/W <sub>23</sub>	P18	P11	F4	P17	C7	P284
D23/W <sub>47</sub>	-	P10	E1	P16	D8	P285
D25/W <sub>51</sub>	-	P9	F5	P15	B6	P286
<b>GND</b>	<b>-</b>	<b>P8</b>	<b>G7</b>	<b>P14</b>	<b>A5</b>	<b>P287</b>
W <sub>52</sub>	-	-	-	-	B5	P288
W <sub>54</sub>	-	-	-	-	E8	P289
W <sub>56</sub>	-	-	E2	P13	C6	P290
D12/W <sub>25</sub>	P17	P7	E3	P12	D7	P291
W <sub>58</sub>	-	-	D1	P11	A4	P292
D26/W <sub>53</sub>	-	-	E4	P10	C5	P293
D27/W <sub>55</sub>	-	-	G6	P9	B4	P294
D13/W <sub>27</sub>	P16	P6	D2	P8	E7	P295
W <sub>60</sub>	-	-	-	-	D6	P296
W <sub>62</sub>	-	-	-	-	A3	P297
NC	-	-	-	-	-	P281
D28/W <sub>57</sub>	-	-	C1	P7	C4	P298
D14/W <sub>29</sub>	P15	P5	D3	P6	D5	P299
D29/W <sub>59</sub>	-	-	E5	P5	E6	P300
D15/W <sub>31</sub>	P14	P4	C2	P4	B3	P301
D30/W <sub>61</sub>	-	P3	B1	P3	B2	P302
D31/W <sub>63</sub>	P13	P2	D4	P2	D4	P303
<b>GND</b>	<b>P12</b>	<b>P1</b>	<b>A1</b>	<b>P1</b>	<b>B1</b>	<b>P304</b>

Note: 1. Pin not connected.

## XC6216 Pinouts - South Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>V<sub>CC</sub></b>	<b>P33</b>	<b>P37</b>	<b>R1</b>	<b>P61</b>	<b>B20</b>	<b>P228</b>
RdWr/S <sub>1</sub>	P34	P38	M4	P62	D17	P227
$\overline{\text{CS}}$ /S <sub>3</sub>	P35	P39	R2	P63	B19	P226
W <sub>12</sub> /S <sub>0</sub>	-	P40	P3	P64	C19	P225
$\overline{\text{OE}}$ /S <sub>5</sub>	P36	P41	L5	P65	F16	P224
W <sub>10</sub> /S <sub>2</sub>	-	P42	N4	P66	E17	P223
Reset/S <sub>7</sub>	P37	P43	R3	P67	D18	P222
W <sub>8</sub> /S <sub>4</sub>	-	P44	P4	P68	C20	P221
<b>NC</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P205<sup>(1)</sup></b>
W <sub>6</sub> /S <sub>6</sub>	-	-	K7	P69	F17	P220
W <sub>4</sub> /S <sub>8</sub>	-	-	M5	P70	G16	P219
W <sub>2</sub> /S <sub>10</sub>	-	-	R4	P71	D19	P218
W <sub>0</sub> /S <sub>12</sub>	-	-	N5	P72	E18	P217
S <sub>14</sub>	-	-	P5	P73	D20	P216
S <sub>16</sub>	-	-	L6	P74	G17	P215
S <sub>18</sub>	-	-	-	-	F18	P214
S <sub>20</sub>	-	-	-	-	H16	P213
S <sub>22</sub>	-	-	-	-	E19	P212
S <sub>24</sub>	-	-	-	-	F19	P211
<b>GND</b>	<b>-</b>	<b>P45</b>	<b>K8</b>	<b>P75</b>	<b>E20</b>	<b>P210</b>
S <sub>26</sub>	-	P46	R5	P76	H17	P209
S <sub>33</sub>	-	P47	M6	P77	G18	P208
Serial/S <sub>9</sub>	P38	P48	N6	P78	G19	P207
Wait/S <sub>11</sub>	P39	P49	P6	P79	H18	P206
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P80</b>	<b>F20</b>	<b>P204</b>
S <sub>28</sub>	-	-	R6	P81	J16	P203
S <sub>35</sub>	-	-	M7	P82	G20	P202
<b>GND</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P83</b>	<b>-</b>	<b>-</b>
S <sub>30</sub>	-	-	-	-	J17	P201
S <sub>32</sub>	-	-	-	-	H19	P200
S <sub>34</sub>	-	-	-	-	H20	P199
S <sub>36</sub>	-	-	-	-	J18	P198
GClk/S <sub>13</sub>	P40	P50	N7	P84	J19	P197
S <sub>38</sub>	-	-	P7	P85	K16	P196
GClr/S <sub>15</sub>	P41	P51	R7	P86	J20	P195
S <sub>37</sub>	-	-	L7	P87	K17	P194
S <sub>40</sub>	-	P52	N8	P88	K18	P193
S <sub>39</sub>	-	P53	P8	P89	K19	P192
<b>V<sub>CC</sub></b>	<b>P42</b>	<b>P54</b>	<b>R8</b>	<b>P90</b>	<b>L20</b>	<b>P191</b>

Note: 1. Pin not connected.

## XC6216 Pinouts - South Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>GND</b>	<b>P43</b>	<b>P55</b>	<b>M8</b>	<b>P91</b>	<b>K20</b>	<b>P190</b>
G1/S <sub>17</sub>	P44	P56	L8	P92	L19	P189
G2/S <sub>19</sub>	P45	P57	P9	P93	L18	P188
S <sub>42</sub>	-	-	R9	P94	L16	P187
S <sub>41</sub>	-	-	N9	P95	L17	P186
S <sub>44</sub>	-	-	-	-	M20	P185
S <sub>43</sub>	-	-	-	-	M19	P184
S <sub>46</sub>	-	-	-	-	N20	P183
S <sub>48</sub>	-	-	-	-	M18	P182
E <sub>0</sub> /S <sub>50</sub>	-	P58	M9	P96	M17	P181
E <sub>2</sub> /S <sub>52</sub>	-	P59	L9	P97	M16	P180
<b>GND</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P98</b>	<b>-</b>	<b>-</b>
E <sub>4</sub> /S <sub>54</sub>	-	-	R10	P99	N19	P179
S <sub>45</sub>	-	-	P10	P100	P20	P178
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P101</b>	<b>T20</b>	<b>P177</b>
ShiftDOOut/S <sub>21</sub>	P46	P60	N10	P102	N18	P175
SEData/S <sub>23</sub>	P47	P61	K9	P103	P19	P174
E <sub>6</sub> /S <sub>56</sub>	-	P62	R11	P104	N17	P173
S <sub>47</sub>	-	P63	P11	P105	R19	P172
<b>GND</b>	<b>-</b>	<b>P64</b>	<b>J7</b>	<b>P106</b>	<b>R20</b>	<b>P171</b>
NC	-	-	-	-	N16 <sup>(1)</sup>	P170 <sup>(1)</sup>
NC	-	-	-	-	P18 <sup>(1)</sup>	P169 <sup>(1)</sup>
E <sub>8</sub> /S <sub>58</sub>	-	-	M10	P107	U20	P168
S <sub>49</sub>	-	-	N11	P108	P17	P167
S <sub>51</sub>	-	-	R12	P109	T19	P166
S <sub>53</sub>	-	-	L10	P110	R18	P165
S <sub>55</sub>	-	-	-	-	P16	P164
S <sub>57</sub>	-	-	-	-	V20	P163
E <sub>10</sub> /S <sub>60</sub>	-	-	P12	P111	R17	P162
E <sub>12</sub> /S <sub>62</sub>	-	-	M11	P112	T18	P161
NC	-	-	-	-	-	P176 <sup>(1)</sup>
SECE/S <sub>25</sub>	P48	P65	R13	P113	U19	P160
SEReset/S <sub>27</sub>	P49	P66	N12	P114	V19	P159
S <sub>59</sub>	-	P67	P13	P115	R16	P158
S <sub>61</sub>	-	P68	K10	P116	T17	P157
SEClk/S <sub>29</sub>	P50	P69	R14	P117	U18	P156
ConfigOK/S <sub>31</sub>	P51	P70	N13	P118	X20	P155
<b>GND</b>	<b>P52</b>	<b>P71</b>	<b>H8</b>	<b>P119</b>	<b>W20</b>	<b>P154</b>
S <sub>63</sub>	P53	P72	P14	P120	V18	P153

Note: 1. Pin not connected.



## XC6216 Pinouts - East Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>V<sub>CC</sub></b>	<b>P54</b>	<b>P73</b>	<b>R15</b>	<b>P121</b>	<b>X19,T16</b>	<b>P152</b>
E <sub>14</sub>	-	P74	M12	P122	U17	P151
A0/E <sub>1</sub>	P55	P75	P15	P123	W19	P150
E <sub>16</sub>	-	P76	N14	P124	W18	P149
A1/E <sub>3</sub>	P56	P77	L11	P125	T15	P148
E <sub>18</sub>	-	-	M13	P126	U16	P147
A2/E <sub>5</sub>	P57	P78	N15	P127	V17	P146
E <sub>20</sub>	-	-	M14	P128	X18	P145
NC	-	-	-	-	U15 <sup>(1)</sup>	P144 <sup>(1)</sup>
NC	-	-	-	-	T14 <sup>(1)</sup>	P143 <sup>(1)</sup>
NC	-	-	-	-	-	P128 <sup>(1)</sup>
A3/E <sub>7</sub>	P58	P79	J10	P129	W17	P142
E <sub>22</sub>	-	-	L12	P130	V16	P141
E <sub>24</sub>	-	-	M15	P131	X17	P140
E <sub>26</sub>	-	-	L13	P132	U14	P139
A4/E <sub>9</sub>	P59	P80	L14	P133	V15	P138
E <sub>28</sub>	-	-	K11	P134	T13	P137
NC	-	-	-	-	W16 <sup>(1)</sup>	P136 <sup>(1)</sup>
NC	-	-	-	-	W15 <sup>(1)</sup>	P135 <sup>(1)</sup>
<b>GND</b>	-	<b>P81</b>	<b>J8</b>	<b>P135</b>	<b>X16</b>	<b>P134</b>
E <sub>30</sub>	-	-	L15	P136	U13	P133
E <sub>32</sub>	-	-	K12	P137	V14	P132
A5/E <sub>11</sub>	P60	P82	K13	P138	W14	P131
A16/E <sub>33</sub> <sup>(2)</sup>	-	P83	K14	P139	V13	P130
<b>V<sub>CC</sub></b>	-	-	-	<b>P140</b>	<b>X15</b>	<b>P129</b>
E <sub>34</sub>	-	P84	K15	P141	T12	P127
A17/E <sub>35</sub> <sup>(2)</sup>	-	P85	J12	P142	X14	P126
<b>GND</b>	-	-	<b>J9</b>	<b>P143</b>	-	-
NC	-	-	-	-	U12 <sup>(1)</sup>	P125 <sup>(1)</sup>
NC	-	-	-	-	W13 <sup>(1)</sup>	P124 <sup>(1)</sup>
NC	-	-	-	-	X13 <sup>(1)</sup>	P123 <sup>(1)</sup>
NC	-	-	-	-	V12 <sup>(1)</sup>	P122 <sup>(1)</sup>
A6/E <sub>13</sub>	P61	P86	J13	P144	W12	P121
E <sub>36</sub>	-	-	J14	P145	T11	P120
E <sub>37</sub>	-	-	J15	P146	X12	P119
E <sub>38</sub>	-	P87	J11	P147	U11	P118
A7/E <sub>15</sub>	P62	P88	H13	P148	V11	P117
E <sub>39</sub>	-	P89	H14	P149	W11	P116
<b>V<sub>CC</sub></b>	<b>P63</b>	<b>P90</b>	<b>H15</b>	<b>P150</b>	<b>X10</b>	<b>P115</b>

Note: 1. Pin not connected.

2. A16/A17 only listed for designers wishing to be upwardly compatible with XC6264.

## XC6216 Pinouts - East Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>GND</b>	<b>P64</b>	<b>P91</b>	<b>H6</b>	<b>P151</b>	<b>X11</b>	<b>P114</b>
A8/E <sub>17</sub>	P65	P92	H12	P152	W10	P113
E <sub>40</sub>	-	P93	H11	P153	V10	P112
A9/E <sub>19</sub>	P66	P94	G14	P154	T10	P111
E <sub>41</sub>	-	P95	G15	P155	U10	P110
E <sub>42</sub>	-	-	G13	P156	X9	P109
E <sub>43</sub>	-	-	G12	P157	W9	P108
NC	-	-	-	-	X8 <sup>(1)</sup>	P107 <sup>(1)</sup>
E <sub>44</sub>	-	-	-	-	V9	P106
E <sub>46</sub>	-	-	-	-	U9	P105
E <sub>48</sub>	-	-	-	-	T9	P104
<b>GND</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P158</b>	<b>T5</b>	<b>-</b>
A10/E <sub>21</sub>	P67	P96	G11	P159	W8	P103
E <sub>50</sub>	-	P97	F15	P160	X7	P102
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P161</b>	<b>X5</b>	<b>P101</b>
E <sub>45</sub>	-	P98	F14	P162	V8	P99
E <sub>52</sub>	-	-	F13	P163	W7	P98
A11/E <sub>23</sub>	P68	P99	G10	P164	U8	P97
E <sub>47</sub>	-	-	E15	P165	W6	P96
<b>GND</b>	<b>-</b>	<b>P100</b>	<b>H7</b>	<b>P166</b>	<b>X6</b>	<b>P95</b>
E <sub>54</sub>	-	-	-	-	T8	P94
E <sub>56</sub>	-	-	-	-	V7	P93
E <sub>58</sub>	-	-	E14	P167	X4	P92
E <sub>49</sub>	-	-	F12	P168	U7	P91
A12/E <sub>25</sub>	P69	P101	E13	P169	W5	P90
E <sub>51</sub>	-	-	D15	P170	V6	P89
E <sub>53</sub>	-	-	F11	P171	T7	P88
E <sub>55</sub>	-	-	D14	P172	X3	P87
NC	-	-	-	-	-	P100 <sup>(1)</sup>
A13/E <sub>27</sub>	P70	P102	E12	P173	U6	P86
E <sub>57</sub>	-	-	C15	P174	V5	P85
E <sub>60</sub>	-	-	-	-	W4	P84
E <sub>62</sub>	-	-	-	-	W3	P83
A14/E <sub>29</sub>	P71	P103	D13	P175	T6	P82
E <sub>59</sub>	-	P104	C14	P176	U5	P81
A15/E <sub>31</sub>	P72	P105	F10	P177	V4	P80
E <sub>61</sub>	-	P106	B15	P178	X1	P79
E <sub>63</sub>	P73	P107	C13	P179	V3	P78
<b>V<sub>CC</sub></b>	<b>P74</b>	<b>P108</b>	<b>B14</b>	<b>P180</b>	<b>W1</b>	<b>P77</b>

Note: 1. Pin not connected.

## XC6216 Pinouts - North Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>V<sub>CC</sub></b>	<b>P11</b>	<b>P144</b>	<b>B2</b>	<b>P240</b>	<b>A2,E5</b>	<b>P1</b>
N <sub>1</sub>	P10	P143	C3	P239	C3	P2
N <sub>3</sub>	P9	P142	A2	P238	D3	P3
N <sub>0</sub>	-	P141	F6	P237	E4	P4
N <sub>2</sub>	-	P140	B3	P236	F5	P5
N <sub>4</sub>	-	-	C4	P235	C2	P6
N <sub>6</sub>	-	-	A3	P234	D2	P7
N <sub>8</sub>	-	-	-	-	E3	P8
N <sub>10</sub>	-	-	-	-	F4	P9
NC	-	-	-	-	-	P11 <sup>(1)</sup>
NC	-	-	-	-	-	P24 <sup>(1)</sup>
N <sub>5</sub>	P8	P139	D5	P233	C1	P10
N <sub>7</sub>	P7	P138	B4	P232	G5	P12
N <sub>12</sub>	-	-	E6	P231	F3	P13
N <sub>14</sub>	-	-	A4	P230	E2	P14
N <sub>16</sub>	-	-	C5	P229	G4	P15
N <sub>18</sub>	-	-	D6	P228	D1	P16
N <sub>20</sub>	-	-	-	-	G3	P17
N <sub>22</sub>	-	-	-	-	H5	P18
<b>GND</b>	<b>-</b>	<b>P137</b>	<b>F8</b>	<b>P227</b>	<b>F1</b>	<b>P19</b>
N <sub>24</sub>	-	P136	B5	P226	F2	P20
N <sub>26</sub>	-	P135	A5	P225	H4	P21
N <sub>33</sub>	-	-	F7	P224	G2	P22
N <sub>28</sub>	-	-	C6	P223	H3	P23
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P222</b>	<b>E1</b>	<b>P25</b>
N <sub>30</sub>	-	-	-	-	G1	P26
N <sub>32</sub>	-	-	-	-	H2	P27
N <sub>34</sub>	-	-	-	-	J5	P28
N <sub>36</sub>	-	-	-	-	J4	P29
N <sub>35</sub>	P6	P134	B6	P221	J3	P30
N <sub>9</sub>	P5	P133	A6	P220	H1	P31
<b>GND</b>	<b>-</b>	<b>-</b>	<b>G9</b>	<b>P219</b>	<b>-</b>	<b>-</b>
N <sub>11</sub>	-	-	E7	P218	J2	P32
N <sub>38</sub>	-	-	D7	P217	J1	P33
N <sub>37</sub>	-	P132	C7	P216	K4	P34
N <sub>40</sub>	-	P131	A7	P215	K5	P35
N <sub>39</sub>	P4	P130	B7	P214	K3	P36
N <sub>13</sub>	P3	P129	E8	P213	K2	P37
<b>V<sub>CC</sub></b>	<b>P2</b>	<b>P128</b>	<b>D8</b>	<b>P212</b>	<b>K1</b>	<b>P38</b>

Note: 1. Pin not connected.

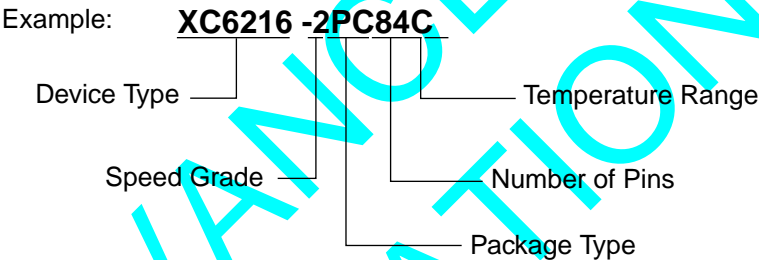
## XC6216 Pinouts - North Side

Pin Description	PC84	HT144	BG225	HQ240	PG299	HQ304
<b>GND</b>	<b>P1</b>	<b>P127</b>	<b>A8</b>	<b>P211</b>	<b>L1</b>	<b>P39</b>
N <sub>15</sub>	P84	P126	B8	P210	L2	P40
N <sub>17</sub>	P83	P125	C8	P209	L3	P41
N <sub>19</sub>	-	P124	E9	P208	L4	P42
N <sub>42</sub>	-	P123	A9	P207	M1	P43
N <sub>41</sub>	-	-	B9	P206	L5	P44
N <sub>44</sub>	-	-	C9	P205	M2	P45
<b>GND</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P204</b>	<b>-</b>	<b>-</b>
N <sub>43</sub>	P82	P122	D9	P203	M3	P46
N <sub>21</sub>	P81	P121	A10	P202	N1	P47
N <sub>46</sub>	-	-	-	-	N2	P48
N <sub>48</sub>	-	-	-	-	M4	P49
N <sub>50</sub>	-	-	-	-	P1	P50
N <sub>52</sub>	-	-	-	-	M5	P51
<b>V<sub>CC</sub></b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>P201</b>	<b>R1</b>	<b>P52</b>
N <sub>23</sub>	-	-	B10	P200	N3	P54
N <sub>54</sub>	-	-	C10	P199	P2	P55
N <sub>45</sub>	-	P120	D10	P198	P3	P56
N <sub>56</sub>	-	P119	A11	P197	N4	P57
<b>GND</b>	<b>-</b>	<b>P118</b>	<b>G8</b>	<b>P196</b>	<b>T1</b>	<b>P58</b>
NC	-	-	-	-	R2 <sup>(1)</sup>	P59 <sup>(1)</sup>
N <sub>47</sub>	-	-	-	P195	T2	P60
N <sub>58</sub>	-	-	E10	P194	N5	P61
N <sub>49</sub>	-	-	B11	P193	R3	P62
N <sub>51</sub>	-	-	C11	P192	P4	P63
N <sub>53</sub>	-	P117	A12	P191	U1	P64
N <sub>55</sub>	-	-	D11	P190	T3	P65
N <sub>57</sub>	-	-	F9	P189	U2	P66
N <sub>60</sub>	-	-	-	-	P5	P67
N <sub>62</sub>	-	-	-	-	R4	P68
NC	-	-	-	-	-	P53 <sup>(1)</sup>
N <sub>59</sub>	P80	P116	B12	P188	V1	P69
N <sub>25</sub>	P79	P115	A13	P187	U3	P70
N <sub>27</sub>	-	P114	C12	P186	T4	P71
N <sub>61</sub>	-	P113	E11	P185	R5	P72
N <sub>63</sub>	P78	P112	B13	P184	V2	P73
N <sub>29</sub>	P77	P111	A14	P183	W2	P74
<b>GND</b>	<b>P76</b>	<b>P110</b>	<b>D12</b>	<b>P182</b>	<b>X2</b>	<b>P75</b>
N <sub>31</sub>	P75	P109	A15	P181	U4	P76

Note: 1. Pin not connected.

For a detailed description of the device architecture, see [page 3](#).  
For a detailed description of the device timing, see [“Electrical Parameters” on page 36](#).  
For package physical dimensions and thermal data, see package section of Xilinx databook.

Ordering Information



Speed Options

- 2      Faster option
- 3      Slower option

Packaging Options

- PC84    84-Pin PLCC
- HT144   144-Pin TQFP
- BG225   225-Pin Ball-Grid-Array
- HQ240   240-Pin PQFP
- HQ304   304-Pin PQFP
- PG299   299-Pin-Grid-Array

Temperature Options

- C      Commercial,     $T_J = 0^{\circ}\text{C}$  to  $85^{\circ}\text{C}$  (Junction)
- I      Industrial,      $T_J = -40^{\circ}\text{C}$  to  $100^{\circ}\text{C}$  (Junction)
- M      Military,         $T_C = -55^{\circ}\text{C}$  to  $125^{\circ}\text{C}$  (Case)