XILINX

XC6200 Field Programmable Gate Arrays

June 1, 1996 (Version 1.0)

Advance Product Specification

Features

- *Advanced Processor-Compatible Architecture*
  - FastMAP™ interface allows 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, 32-bits)
  - Easily interfaced to most microcontrollers and microprocessors
- High-Performance Sea-of-Gates FPGA
  - Up to 16K configurable cells
  - Abundant registers, gates and routing resources
  - Extremely high gate count for structured logic or datapath designs
- High Capacity Distributed RAM
  - High speed SRAM control store
  - 2 bytes of synchronous RAM per cell
- High Speed Flexible Interconnect Architecture
  - Low delay hierarchical routing scheme gives large number of fast 'longlines' (Fastlanes)
  - Any cell can be connected to any other
  - Suited to both structured synchronous cell data path type designs or irregular random logic
  - Completely flexible clocks and clears for registers
  - 4 Global low-skew signals
- >110MHz 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 Dynamic Reconfiguration Capability
  - High speed reconfiguration via parallel CPU interface
  - Unlimited reprogrammability
  - Full or partial reconfiguration/context switching possible
  - Ideal for custom computing applications
- *Flexible Pin Configuration*
  - All User I/O's programmable as in, out, bidirect, tri-state or open drain.
  - Configurable pull-up/down resistors
  - CMOS or TTL logic levels
  - 8, 16, 32-bit CPU interface
- Testability
  - Pre-tested volume part
  - JTAG capability with library macrocells
- XACT*step* Series 6000 Development System
  - Implement designs using familiar tools like Viewlogic and Synopsys
  - 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 [1]	XC6216	XC6236 [1]	XC6264 [1]
Max Logic Gates	13,000	24,000	55,000	100,000
Typical Gate Range	9,000 - 13,000	16,000 - 24,000	36,000 - 55,000	64,000 - 100,000
Number of Cells	2304	4096	9216	16384
Max. No. Registers	2304	4096	9216	16384
Number IOB's	192	256	384	512
Cell Rows x Columns	48x48	64x64	96x96	128x128
Max. RAM (bits)	36K	65K	147K	262K

Notes: 1. Planned Product

Description

The XC6200 family is a new type of high performance Field Programmable Gate Array (FPGA) from Xilinx.

The XC6200 series 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 for standard processors.

XC6200 FPGAs can provide extremely high gate counts for data path or regular array type designs. In these cases the actual gate count may be considerably higher 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 cells is supplemented with wires of length 4, 16 and chip-length (CL) cells, called FastLane-4, 16 and CL respectively, 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.

An XC6200 part is configured by the content of 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 RAM interface. A host processor 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 virtualised 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 correct internal state in their registers. Sections of the device can be reconfigured without disturbing circuits running in other sections. Thus an XC6200 FPGA in a coprocessor application can be time-shared by several processes running on the host computer.

Design entry and verification are carried out with Xilinx XACT*step* Series 6000 software using industry-standard schematic capture, synthesis and simulation packages such as Viewlogic, Synopsys and Mentor Graphics. 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 a number of CAD tools ranging from simple symbolic editors to sophisticated cell-compilation tools. These tools will ensure that the captured design is laid out efficiently with no user intervention. Node delays can then be back-annotated to the top level logic simulator. The tools also 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.

Detailed Functional Description

Logical and Physical Organization

The XC6200 architecture may be viewed as a hierarchy. At the lowest level lies a large array of basic cells (Figure 1). This is the 'sea-of-gates'. Each cell is individually programmable to implement a D-type register and a 2-input 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.

Cells, Blocks and Tiles

First generation fine-grain architectures implemented only nearest-neighbor interconnection and had no hierarchical routing (Figure 1). The XC6200 architecture 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) that themselves form a cellular array, communicating with neighboring 4x4 cell Blocks. A 4x4 array of these 4x4 Blocks forms a 16x16 cell Tile (Figure 3). In the XC6216 part, a 4x4 array of these 16x16 Tiles forms the central 64x64 cell array which is then surrounded by I/O pads (Figure 4).

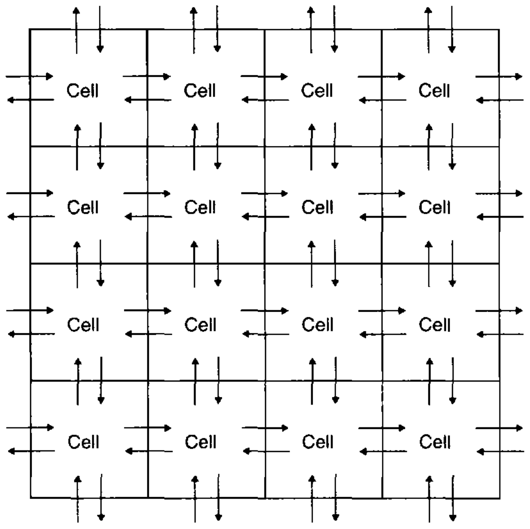

Figure 1: Nearest-Neighbor Interconnect Array Structure

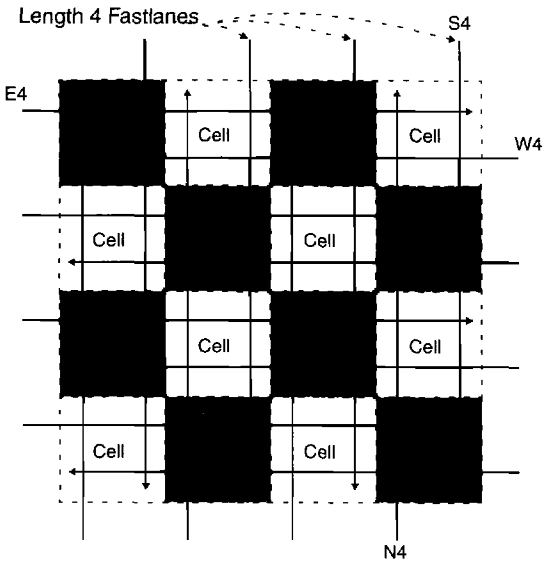

Figure 2: XC6200 4x4 Cell Block

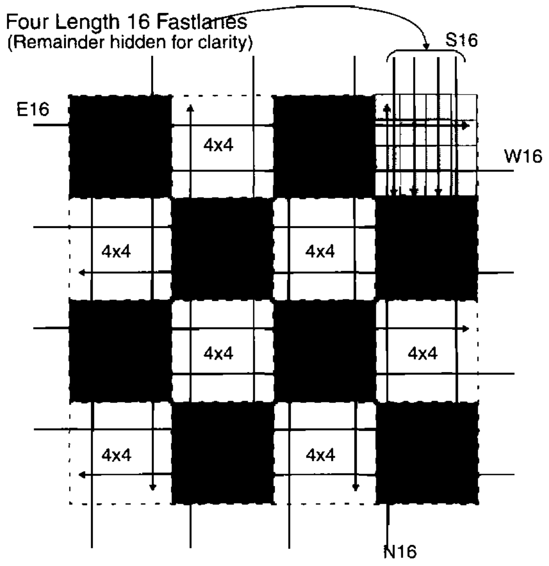

Figure 3: XC6200 16x16 Cell Tile

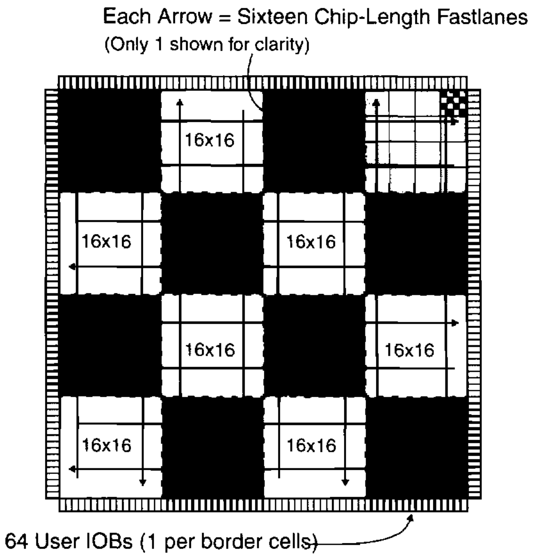
Each Arrow = Sixteen Chip-Length Fastlanes
(Only 1 shown for clarity)
16x16
64 User IOBs (1 per border cells)

Figure 4: XC6216 Device

Routing Resources

Each level of hierarchy (basic cells, 4x4 cell Blocks, 16x16 cell Tiles, 64x64, etc.) has its own associated routing resources. Basic cells can route to their nearest neighbors or through the neighbor cell to its neighbor. Note that cells used for interconnect in this manner can still be used to provide a logic function. Wires of length four are provided to allow 4x4 cell blocks to route across themselves without using basic cell resources. Similarly 16x16 cell tiles provide additional wires of length 16 cells and the 64x64 array provides wires of length 64. 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 tiles. Switches at the edge of the blocks and tiles 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' because 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 labelled 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 CL 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 tile 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 length CL Fastlanes.

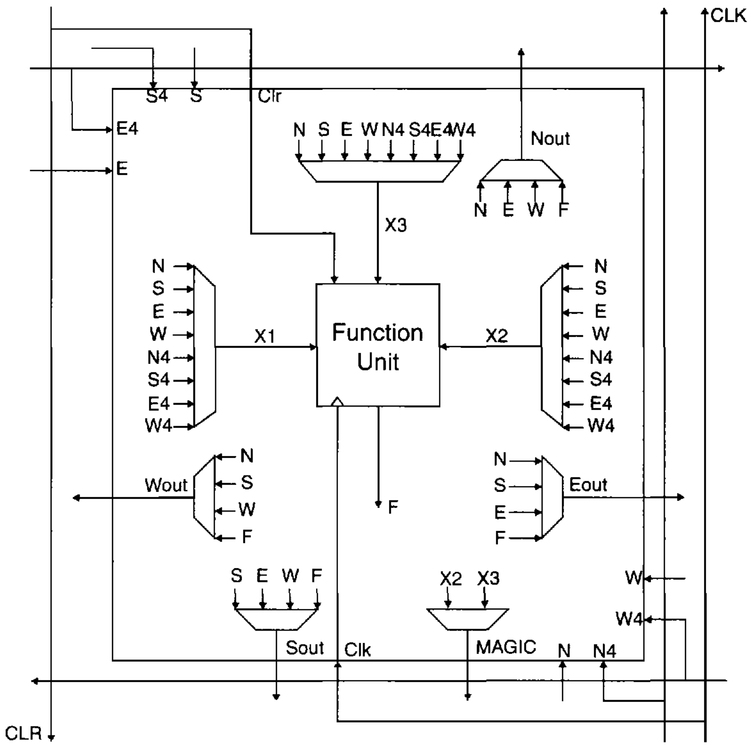

Figure 5: XC6200 Basic Cell

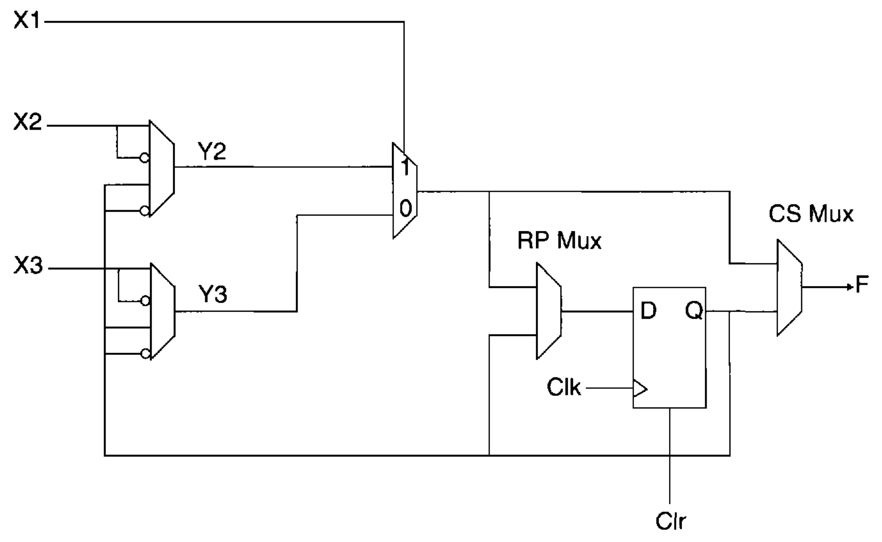

Figure 6: XC6200 Function Unit

Magic Wires

The majority of interconnections are routed using the nearest-neighbor and Fastlane wires described above. Each cell has a further output (labelled '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 257.

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, GClk and GClr) 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 function block 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 GClk as a global Clear signal, however for minimum delay, it is recommended that GClk be used for global clocks and GClr for global clears. GClk and GClr 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 GClk and GClr.

Function Unit

Figure 5 shows the basic XC6200 cell in detail. The inputs from neighboring cells are labelled 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 labelled 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 possible. The Magic routing output can only be used if the signal to be routed can be placed on X2 or X3.

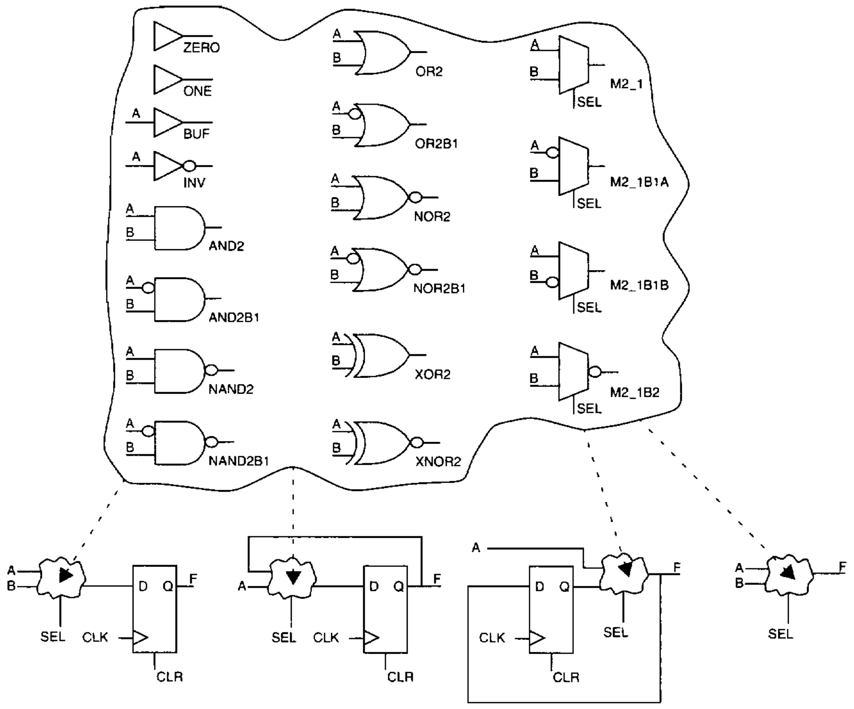

Figure 7: Cell Logic Functions

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 combinatorial 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 will 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 flavor 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 combinatorial '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, however the place and route software will generally optimize this out as 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. $\overline{A}$•*B* (AND2B1) is shown but *A*•$\overline{B}$ (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.

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 Q to Y3. *A* is routed to X1 and *B* to X2. X2 is assigned to Y2. When *A* changes to

Table 2: Function Derivation

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

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, and 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 Fastlanes. 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 CL Fastlanes) to be connected to length-4 Fastlanes.

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 labelled M and MA. The Magic wires allow cell outputs to jump to the edge of the 4x4 block and hence onto Fastlanes 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 Figure 9 through Figure 12. 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 will be one of NIn, E, W, or F depending on the least-significant two bits of the NOut multiplexer select lines. Hence NC will be 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 will be one of F, NIn, E or W depending on which signal is routed to NOut. The 'C' signal will be one of the upper four inputs to the 8:1 multiplexers shown in Figure 9 through Figure 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 Previous). e.g. PS4. This feature allows Fastlane wires to perform U-turns.

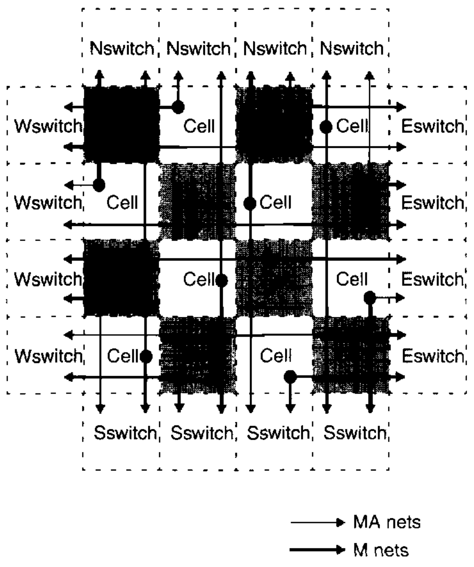

Figure 8: Routing Switches at 4x4 Block Boundary

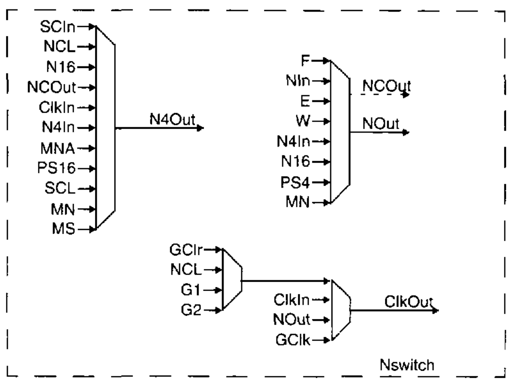

Figure 9: Contents of Nswitch

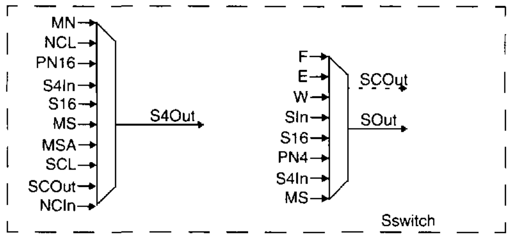

Figure 10: Contents of Sswitch

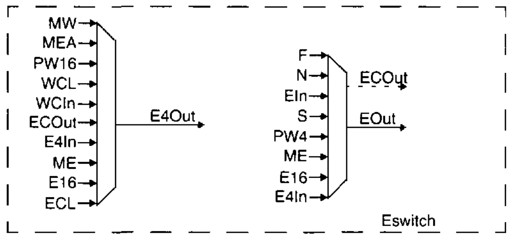

Figure 11: Contents of Eswitch

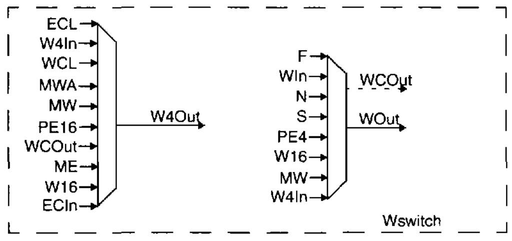

Figure 12: Contents of Wswitch

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. 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. 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.

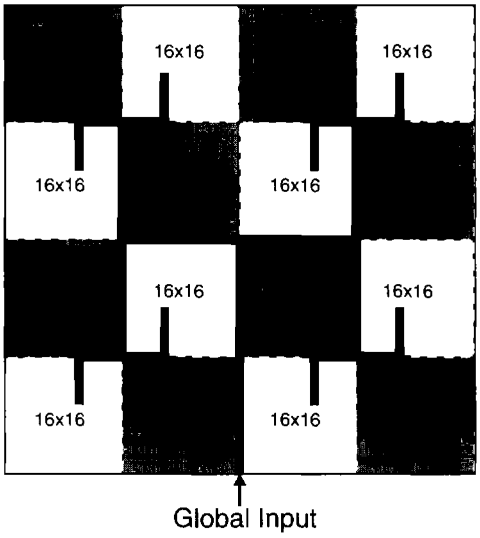

Figure 13: Low Skew 'H' Distribution Of Global Signals (XC6216)

It is also possible to route North chip-length Fastlanes onto the Clock lines. This allows up to 64 (for a XC6216 device) locally used clocks to be provided that 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.

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 Fastlane, since limiting fast clock distribution to the area of the device where it is required will reduce 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 tile, 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.

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.

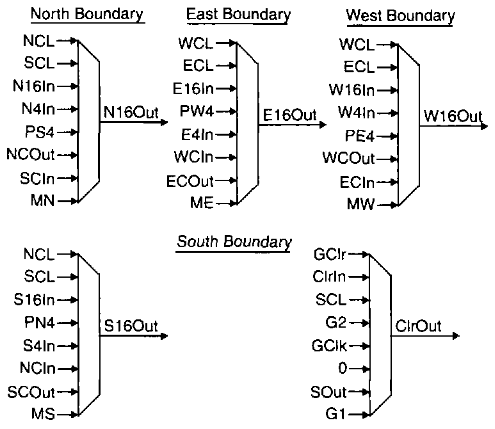

Figure 14: Additional Switches at 16x16 Boundaries

Input/Output Blocks (IOBs)

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 map-

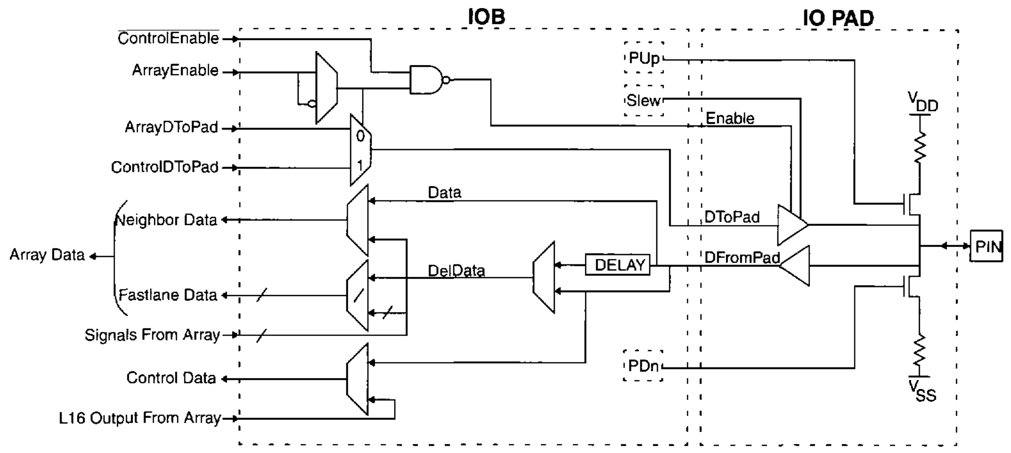

Figure 15: Input/Output Architecture

ping of IOBs to device pins is given in the pinout tables starting on page 282.

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 PAL's or discrete logic.

Each IOB has an array data input and a control data input, labelled 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.

There are less real control signals than IOBs, hence the three control signals on some IOBs are not connected to the device control logic. These spare control signals are used to route data to and from the padless IOBs mentioned above. The control signals on the padless IOB are not used. This is illustrated in Figure 16.

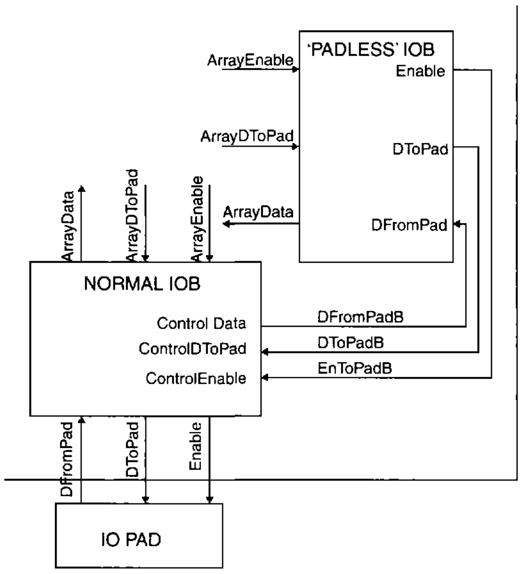

Figure 16: 'Padless' IOB Configuration

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

Three configuration RAM bits within each IOB control the programmable aspects of its IO pad. These RAM bits have no effect for padless IOBs. 'PUp' 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.

The Array Enable, Array Data and Control Data multiplexers are also 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.

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 $\overline{\text{OE}}$ 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 on page 268. The $\overline{\text{OE}}$ signal provides a quick way of disabling all the output drivers and may be activated at any time. Only when $\overline{\text{OE}}$ 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.

I/O 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 Fastlane output from the same cell.

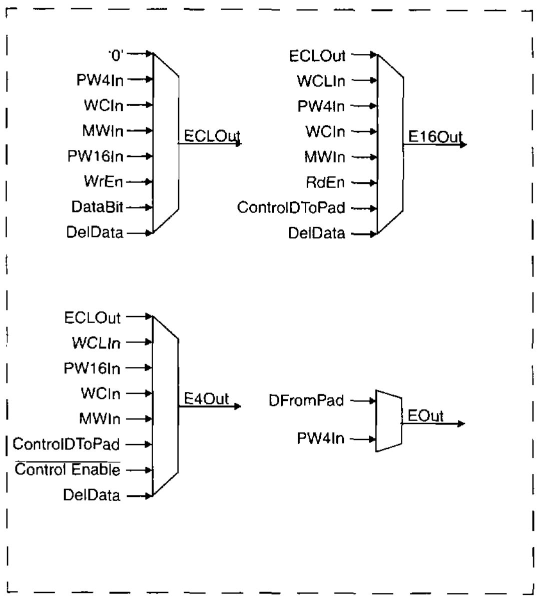

Figure 17: Array Data Sources In West IOBs

The Array Data multiplexer in Figure 15 is actually a collection of multiplexers that 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. This is illustrated for a West IOB in Figure 17. 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. 'DataBit' 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 Programming Interface" on page 268. 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 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 Array Enable inputs to IOBs.

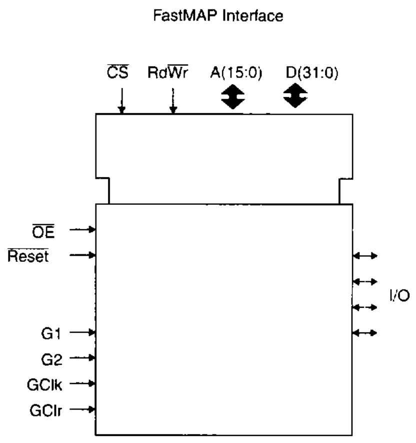

Figure 18: XC6216 Logic Symbol

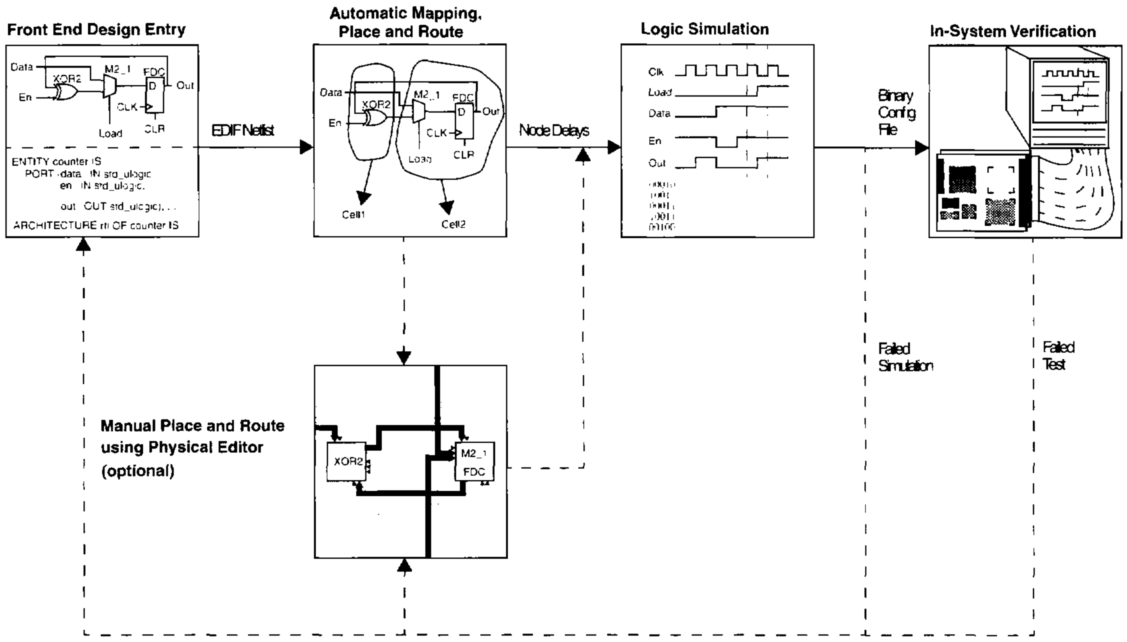

Figure 19: XC6200 Logic Design Flow

Designing with XC6200

The designing of XC6200 FPGAs into systems may be partitioned into three distinct activities: board design, logic design, and software design.

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 will have conventional SRAM data, address and control signals. In other cases, it may only require the user defined I/O signals of an ASIC. Packaging information for the part is shown in Table 6. The number of user I/O signals will depend 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 devices makes this particularly easy.

The configuration RAM bits in the IOBs allow for a number of different programmable options to make interfacing to other ICs easier.

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 19. 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 that 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 that 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 provides 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 20 is mapped onto the XC6200 architecture. Figure 21 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 workstations. 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 solid 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.

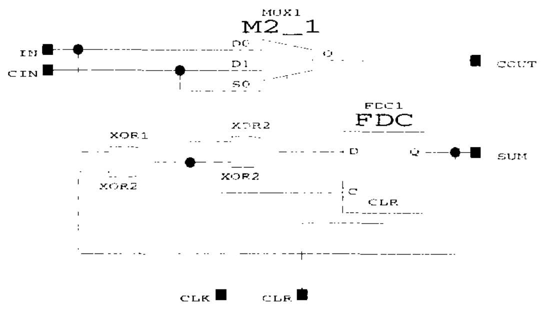

Figure 20: Accumulator Schematic

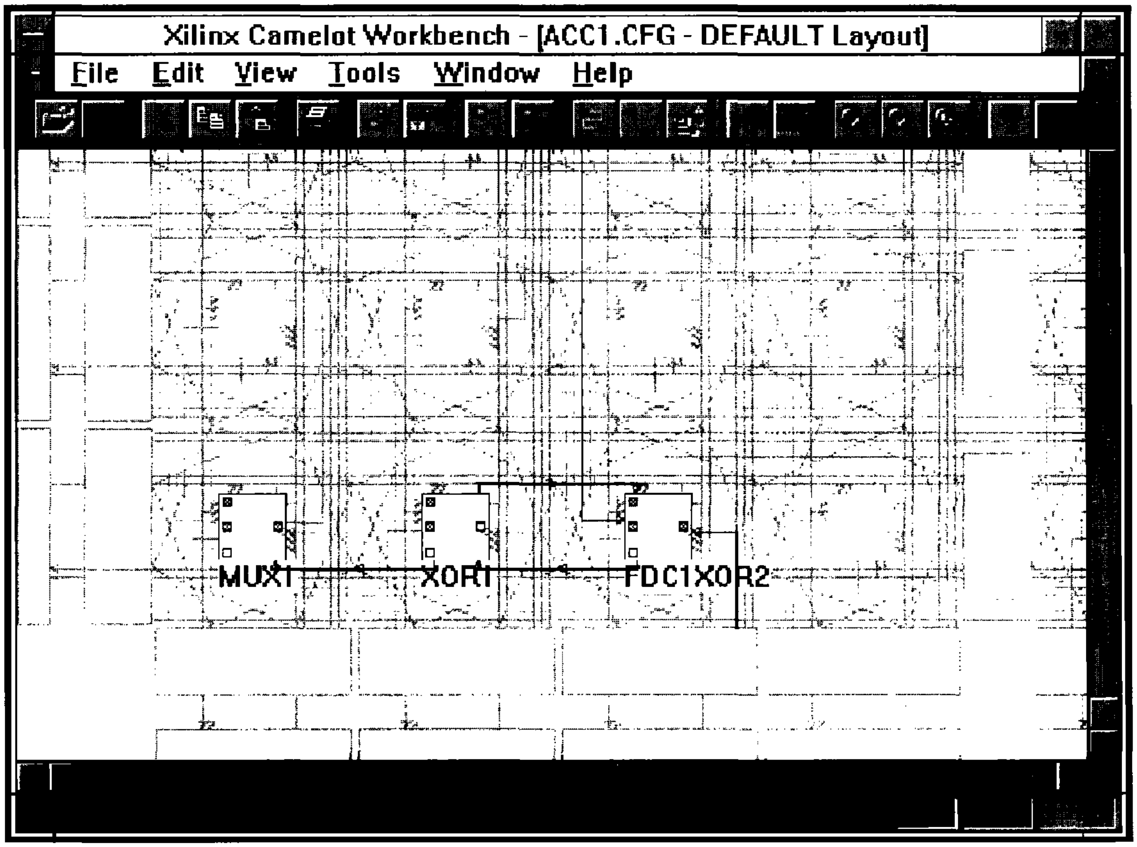

Figure 21: Accumulator Physical Editor View

The inputs and outputs to the function unit are connected to the edges of the cell box. The CAD plot shows all the routing resources available. To the left of each cell the six wires running in a Southerly direction are: SCL, S16, Clr, S4, Magic and S. The signal direction is indicated with arrow heads. S, S4 and Clr are shown entering the cells on the left edge of the cell boxes. The two outputs on the left edge of the cell box are Magic and SOut. The inputs and outputs on the remaining three cell box edges follow a similar pattern. The Clk input on the right hand cell box edge is denoted with a clock '<' symbol.

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.

Software Design with XC6200

This is the design of a program for the host processor which interacts with a design running on the XC6200 FPGA. 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 reconfigured as required. Predefined device drivers and an efficient run-time library are available to make optimal use of the high speed reconfiguration capabilities with minimal development time.

Register Access

The XC6200 architecture 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 the inverse of F in Figure 6 (= Q or $\overline{D}$).

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.

In many applications this access to internal nodes will be 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 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.

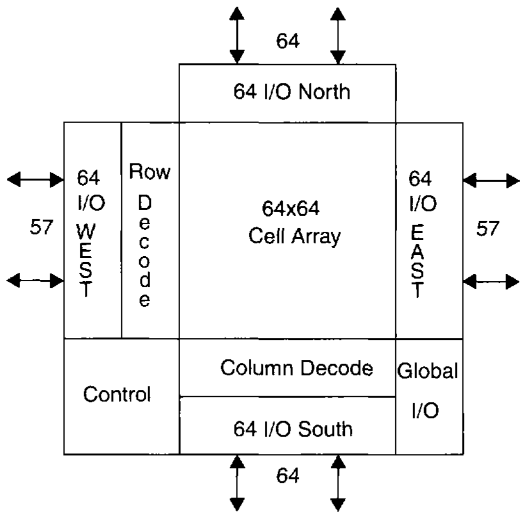

Figure 22: XC6216 Block Diagram

Figure 22 is a block diagram of the XC6216 part, showing the row and column address decoders. Figure 23 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 that are to be accessed word-wide: they must be on the same column of cells within the array.

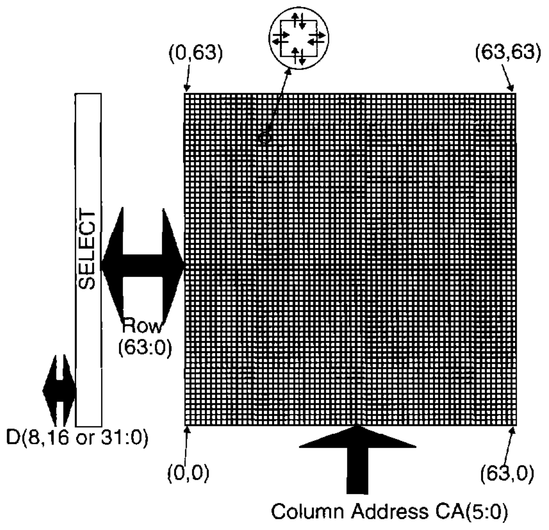

Figure 23: Memory Mapped I/O

Map Register

The XC6200 architecture also provides a mechanism for mapping the 64 possible cell outputs onto the 8,16 or 32-bit external data bus, selecting only those cells that 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 mechanism provided takes the form of a 64-bit **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 will connect 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 will be set once to indicate the placement of the user I/O register that will then be 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.

If more '0's 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 '0's and the CPU writes FFh to a particular column. All the state registers in that column will be written with a '1'. The default state of the map register is all '0's.

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 less '0's in the map register than data bus bits, the upper data bus bits, which are not mapped, will be read as '1's during CPU reads and ignored during CPU writes.

An example of map register operation is shown in Figure 24. The position of the user-defined register within the cell array is defined by the '0's in the 64-bit map register. Similar registers could be defined for every column in the array if desired.

There is a delay after a write to the map register before the change takes effect. No state accesses should be carried out during this time.

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 '1's in the Mask Register will retain their values when written. On a write operation the corresponding bit line will not be enabled and the state information for that bit will not be changed. When the device is reset the Mask Register will contain all logic 0's corresponding to all data bus bits valid.

During CPU reads, valid register bits which are disabled will be 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.

Programming

The binary data for configuring an XC6200 device, 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 Programming Interface

The XC6200 FPGA has a conventional programming interface for static RAM, 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 28 and 29. 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 24. This is an important point, as only registers clocked directly by GClk can be reliably read or written using this method.

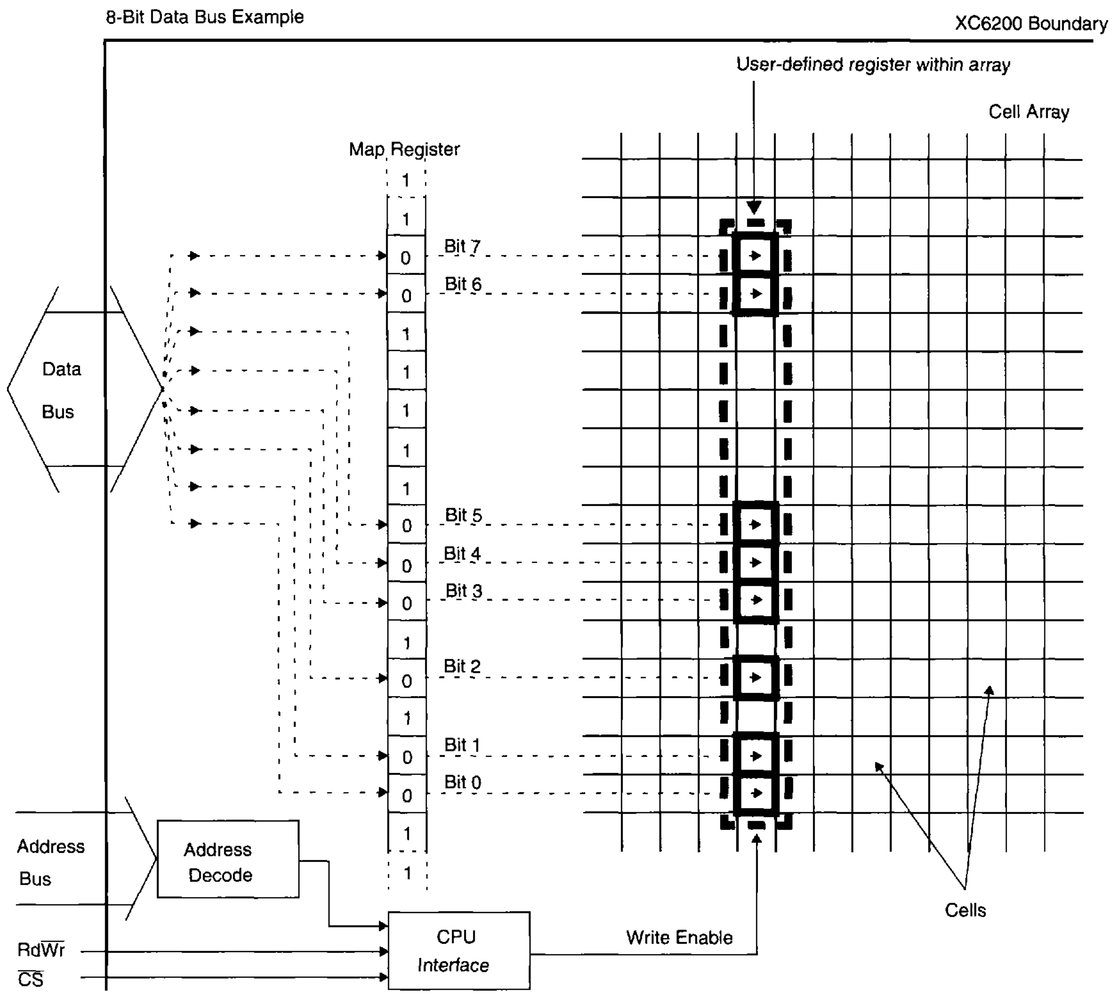

Figure 24: Internal Register Access

Figure 29 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 28).

$\overline{CS}$ must always be sampled as a '1' before the next cycle can begin. In Figure 29 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 a block transfer operation.

The format of the address bus to the XC6216 device is shown in Table 3. Larger XC6200 devices have proportionally more bits allocated to row and column addresses.

Table 3: Address Bus Format (XC6216)

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

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 4.

Table 4: Address Mode Selection

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

Wildcard Registers

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, since 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 limited to 32: that is at most five don't care bits can be set.

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

The address decoding for the XC6216 FPGA is summarized in Table 5.

Table 5: XC6216 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	N/S Secondary Clock Mux (Column[1:0] = 00 or 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

Serial Programming Interface

All the memory mapped locations in an XC6200 device may be written in parallel or serial mode. All the operations which can be carried out with the parallel interface may also be done serially. The serial interface gives random access to all the XC6200 memory locations. The serial interface 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 25.

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

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

In a multi-FPGA configuration a user I/O also will have to be available to provide the Wait input to the next device in the chain.

On Reset each FPGA examines its $\overline{\text{Serial}}$ and Wait inputs. Any FPGA that sees both these signals low at this time assumes it is the master and drives SEReset, $\overline{\text{SECE}}$ and SEClk. All User I/Os are held in a high-impedance state (with pull-up) until a valid configuration is loaded. In Figure 25, the User I/Os will be pulled high on Reset, hence the Wait input to the Slaves will be high and they will 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 parallel interface, e.g. use of the mask register, writes to cell state registers, etc.

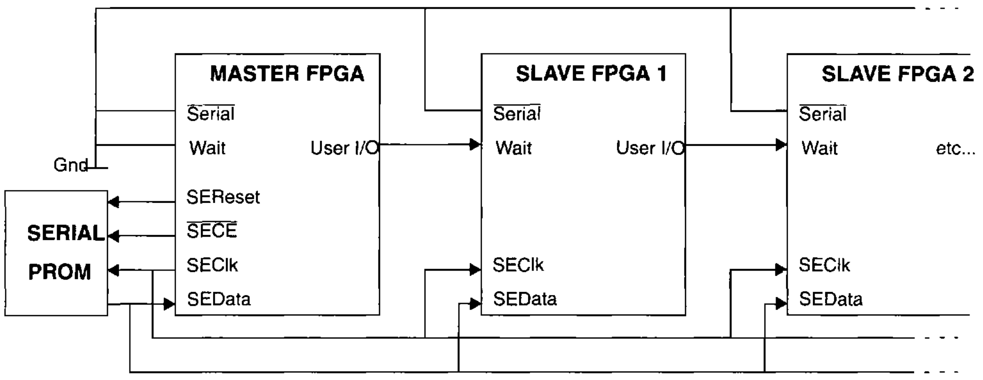

Figure 25: Master-Slave Serial Configuration

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.

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 SEClk. The SEClk rate may also be increased by writing the appropriate 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 25. 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 will be shifted in. If the width had not been changed data word 2 would also have been 8 bits. Data will continue to be loaded until $\overline{\text{Serial}}$ goes high or Wait goes low.

Reset And Initialization

When the XC6200 FPGA is powered up or after a reset, all configuration memory is cleared and the cell state registers are cleared. The $\overline{\text{Reset}}$ pin is not required to be active during or after power-up to initialize the FPGA. To avoid potential high current random configurations, the power-up reset is carried out automatically. The automatic power-up initialization takes 2.5 µs. 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 $\overline{\text{Reset}}$ input for a minimum of 20 ns. This acts as a signal to the chip to initialize itself. This initialization occurs after $\overline{\text{Reset}}$ has been deasserted. Therefore there is a 1.0 µs (typ.) reset recovery time when no device accesses are possible.

Pin Descriptions

The pins are labelled as follows:

V_{DD}

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

GND

Connections to ground. All must be connected.

$\overline{CS}$

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>

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

A<a:0>

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

Rd$\overline{Wr}$

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, GClr, G1, G2

Global signals. GClk should be used for global user clocks, GClr 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.

$\overline{Reset}$

When $\overline{Reset}$ is taken low the programming registers (mask unit and address wildcard unit) are re-initialized, resulting in the XC6200 device 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. This signal should be taken low immediately after power up to initialize the device. 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.

$\overline{OE}$

When this signal is low 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. 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 which controls transitions between states in serial mode state machine.

Wait

Input which controls transitions between states in serial mode state machine.

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

SEReset

Output from Master FPGA which resets serial PROM address counter.

$\overline{SECE}$

Output from Master FPGA which enables serial PROM output.

SEClk

Output from Master FPGA which clocks serial PROM and slave FPGAs.

SEData

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

ConfigOK

Signal is active (high) when a valid pattern is present in the ID register and inactive when the pattern is invalid.

N_x

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

S_x

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

E_x

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

W_x

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

Electrical Parameters

The XC6200 series is fabricated in 0.65 micron triple metal n-well CMOS. Foundry sources for this part have been chosen to meet or exceed relevant military standards and industry practice.

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

The power consumption of a XC6200 device can vary from a few tens to several hundreds of milliamps depending on its configuration and the data applied to it. The most significant sources of power consumption are I/O blocks and dynamic dissipation within the array, both of which are largely under user control. Dynamic power dissipation is of most concern where the XC6200 device is used to implement highly concurrent computations. Power dissipation must be considered carefully, not only because excessive dissipation could result in device failure but also because operating speed is reduced at high temperature.

A 0.22µF decoupling capacitor across V_{CC} and *GND* per part is recommended. Surface mounted, radial, plastic or ceramic capacitors are suitable.

Where possible, user designs that could result in many output pads making a simultaneous transition in the same direction should be avoided. This is especially important on heavily loaded connections to non-XC6200 parts. To minimize power dissipation, redundant connections in user designs (which may arise in hierarchical design styles to promote sub-block re-use) should be deleted by CAD programs prior to programming XC6200 devices. As a general guideline, users should attempt to minimize the number of cell resources used. Where buffers must drive heavy external loads it may be helpful to choose I/O blocks near *GND* pads.

XC6200 parts automatically reset themselves after power up, since the random values in the control store at this time could correspond to a relatively high power dissipation configuration.

The XC6200 part distributes power using a redundant scheme which ensures minimal voltage drop between pads and internal circuitry. Power for pads is distributed on a separate power and ground ring.

The maximum power consumption of the XC6200 is limited by two factors: the metal conductors supplying the part and heat dissipation. The metal conductors can handle up to 100mA each. Heat dissipation is generally a much more serious consideration: a full discussion of thermal characteristics for the different package options is given in section 4 of this data book.

XC6200 Switching Characteristics

Notice: The information contained in this data sheet pertains to products in the initial production phases of development. These specifications are subject to change without notice. Verify with your local Xilinx sales office that you have the latest sheet before finalizing a design.

XC6200 Operating Conditions

Symbol	Parameter			Min	Max	Units
V_{CC}	Supply voltage relative to GND	*Commercial*	$T_J = 0^o$ C to 85^o C *junction*	4.75	5.25	V
	Supply voltage relative to GND	Industrial	$T_J = -40^o$ C to 100^o 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
V_{IHC}	High-level input voltage – CMOS configuration			70%	100%	V
T_{IN}	Input signal transition time				250	ns

XC6200 DC Characteristics Over Operating Conditions

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{OH}	High-level output voltage	I_{OH} = -8.0 mA V_{DD} = Min	3.86		V
V_{OL}	Low-level output voltage	I_{OL} = 8mA V_{DD} = Min		0.4	V
I_{IL}	Input leakage current	V_{DD} = Max V_{IN} = GND or V_{CC}	10	10	µA
I_{OZ}	Output high-Z leakage current	V_{DD} = Max V_O = GND or V_{CC}		TBA	µA
C_{IN}	Input capacitance for Input and I/O pins	V_{IN} = GND f = 1.0 MHz		15	pF
I_{CC}^2	Quiescent Supply Current	$V_{IN} = V_{CC}$ or GND V_{DD} = 5 V f = 1.0 MHz @ 25°C		TBA	mA

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.

XC6200 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

***Warning**: Stresses beyond those listed under XC6200 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 XC6200 Operating Conditions is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability.*

XC6200 Power-on/Reset Timing Parameters

Symbol	Parameter	Min	Typ	Max	Units
T_{WMR}	Master Reset input Low pulse width		20		ns
T_{MRR}	Recovery time after Master Reset deasserted		1		µs
T_{PRR}	Recovery time after power up		2.5		µs
		Advance			

XC6200 Serial Configuration Timing

	Description	Symbol		Min	Max	Units
SEClk	SEData setup	1	T_{DC}	TBA	-	ns
	SEData hold	2	T_{CD}	TBA	-	ns
	Setup before GClk	3	T_{CG}	TBA	-	ns
	Hold after GClk	4	T_{GC}	TBA	-	ns
				Advance		

XC6200 Global Buffer Switching Characteristic Guidelines

	Speed Grade:	-2		
Symbol	Parameter	Min	Max	Units
T_{PGClk}	From pad through **GClk** buffer to any register clock		12	ns
T_{PG}	From pad through **G1,G2,GClr** buffers to any register clock		12	ns
T_{PClr}	From pad through **global** buffers to any register clear		11	ns
T_{PCkS}	Skew between any pair of register clocks using the same **global**		0.9	ns
T_{PClS}	Skew between any pair of register clears using the same **global**		0.9	ns
		Advance		

Note: Typical loading values are used.

XC6200 Cell Switching Characteristic Guidelines

	Speed Grade:	-2		
Symbol	Parameter	Min	Max	Units
T_{ILO1}	X1 change to Function Output(1)		2	ns
T_{ILO23}	X2/X3 change to Function Output(2)		3	ns
T_{ICK1}	Internal Register Set-Up Time @ X1(1)	3.5		ns
T_{ICK23}	Internal Register Set-Up Time @ X2/X3(2)	4		ns
T_{IHCK1}	Internal Register Hold Time @ X1(1)	-1.5		ns
T_{IHCK23}	Internal Register Hold Time @ X2/3(2)	-2		ns
T_{CH}	Clock High Time(3)	4.5		ns
T_{CL}	Clock Low Time(3)	4.5		ns
f_{TOG}	Export Control Max. flip-flop toggle rate		111	MHz
T_{CLW}	Clear Pulse Width(3)	1		ns
T_{CKO}	Clock to Function Output		2	ns
T_{CKLO}	Clock to Function Output via X2/X3 feedback mux's		3	ns
		Advance		

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.

XC6200 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**				
		Best I/O		**Worst I/O**		
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**
T_{ICKOF}	Global Clock (GClk) to Output (fast)				19	ns
T_{ICKO}	Global Clock (GClk) to Output (slew limited)				22.6	ns
T_{PSUF}	Input Set-up Time (fast)			-4		ns
T_{PSU}	Input Set-up Time with delay			5		ns
T_{PHF}	Input Hold Time (fast)			6.5		ns
T_{PH}	Input Hold Time with delay			-2.5		ns
		Advance				

Notes: All appropriate AC specifications tested using Figure 27 as test load circuit.
These parameters are tested directly and guaranteed over the operating conditions.
As the parameters vary between I/Os, values are given for best and worst I/Os. The parameters for other I/Os will be somewhere between these two extremes. The delay calculator software will calculate the correct value for each I/O used.
All parameters assume the cell register is the closest one to the IOB.

XC6200 IOB Switching Characteristic Guidelines

	Speed Grade:	**-2**				
Symbol	**Parameter**	**Best I/O**		**Worst I/O**		**Units**
		Min	**Max**	**Min**	**Max**	
T_{PID}	**INPUT** Pad to Neighbor data				4	ns
T_{PID4}	Pad to L4 Fastlane				5	ns
T_{PDID4}	Pad to L4 Fastlane with delay				13	ns
T_{OPF}	**OUTPUT** Neighbor data to Output (fast)				4	ns
T_{OPS}	Neighbor data to Output (slew rate limited)				8	ns
T_{TSHZ}	3-state to Pad begin hi-Z (slew rate independent)				5	ns
T_{TSONF}	3-state to Pad active and valid (fast)				5.2	ns
T_{TSONS}	3-state to Pad active and valid (slew rate limited)				9	ns
		Advance				

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

XC6200 Internal Routing Delays

	Speed Grade:	-2		
Symbol	Parameter	Min	Max	Units
T_{FN}	Function Output to Neighbor		1	ns
T_{NN}	Route Neighbor In to Neighbor Out		1.5	ns
T_{Magic}	Route X2/X3 to Magic Out		2.5	ns
T_{L4}	Length-4 Fastlane delay		2	ns
T_{L16}	Length-16 Fastlane delay		2.5	ns
T_{L64}	Chip-Length Fastlane delay		5	ns
		Advance		

Notes: Delays vary depending on direction. Worst case figures are given here. The delay calculator software will calculate the correct delay for each direction.

Typical loading values are used.

XC6200 CPU Interface Timing

		Speed Grade:	-2		
Symbol		Parameter	Min	Max	Units
1	$T_{su\overline{CS}}$	$\overline{CS}$ set up before Clock[1]	6		ns
2	$T_{h\overline{CS}}$	$\overline{CS}$ hold after Clock[1]	0		ns
3	$T_{suRd\overline{Wr}}$	Rd$\overline{Wr}$ set up before Clock	6		ns
4	$T_{hRd\overline{Wr}}$	Rd$\overline{Wr}$ hold after Clock	0		ns
5	T_{suA}	Address Bus set up before Clock	6		ns
6	T_{hA}	Address Bus hold after Clock	0		ns
7	T_{suD}	Data Bus set up before Clock	6		ns
8	T_{hD}	Data Bus hold after Clock	0		ns
9	T_{WC}	Write cycle time[2]	40		ns
10	T_{RC}	Read cycle time[2]	40		ns
11	T_{CKD}	Clock to Valid Data		8	ns
12	T_{CKDZ}	Clock to Data high impedance[3]		9	ns
13	$T_{\overline{CS}DZ}$	$\overline{CS}$ to Data high impedance[3]		9	ns
			Advance		

Notes:
1. $\overline{CS}$ must be correctly sampled as a '0' at the start of the cycle (t_1) and sampled as a '1' 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.
3. 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.

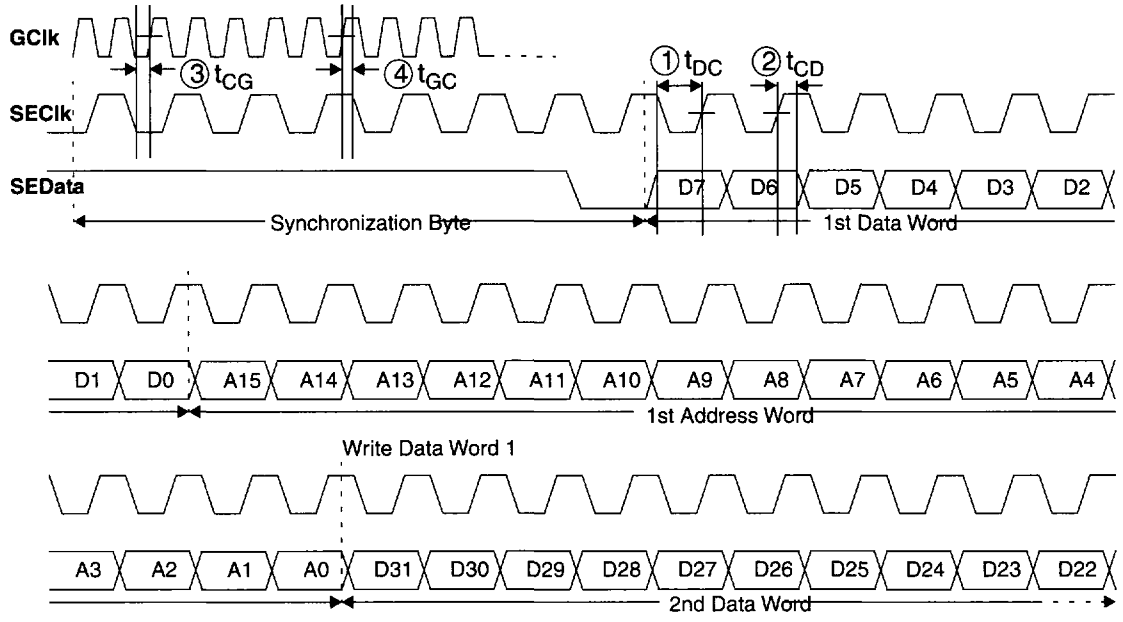

Figure 26: Serial Configuration Timing

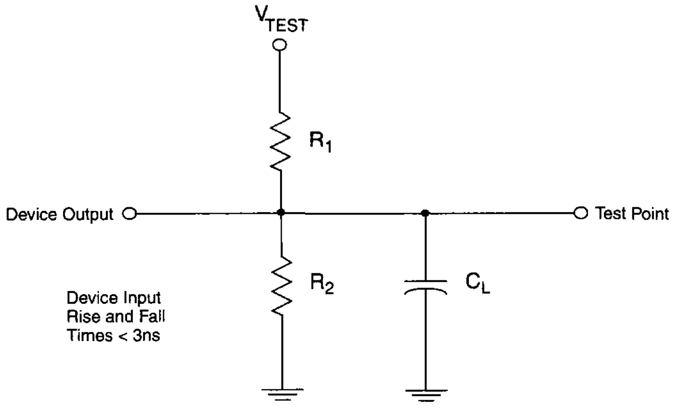

Figure 27: AC Load Circuit

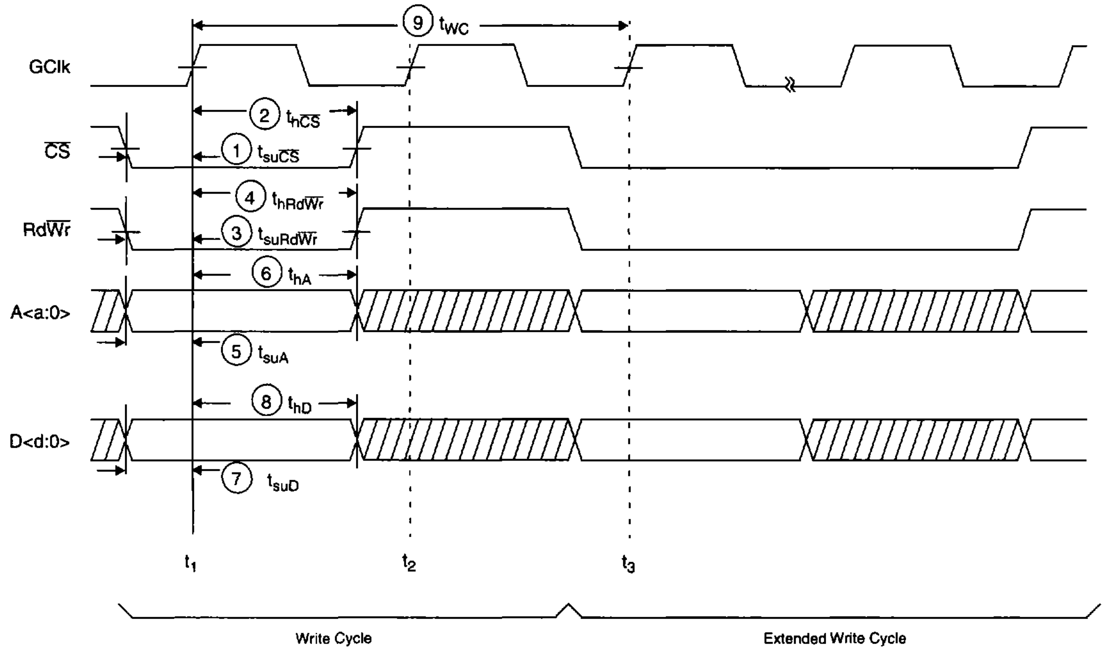

Figure 28: Configuration Memory Write Cycles

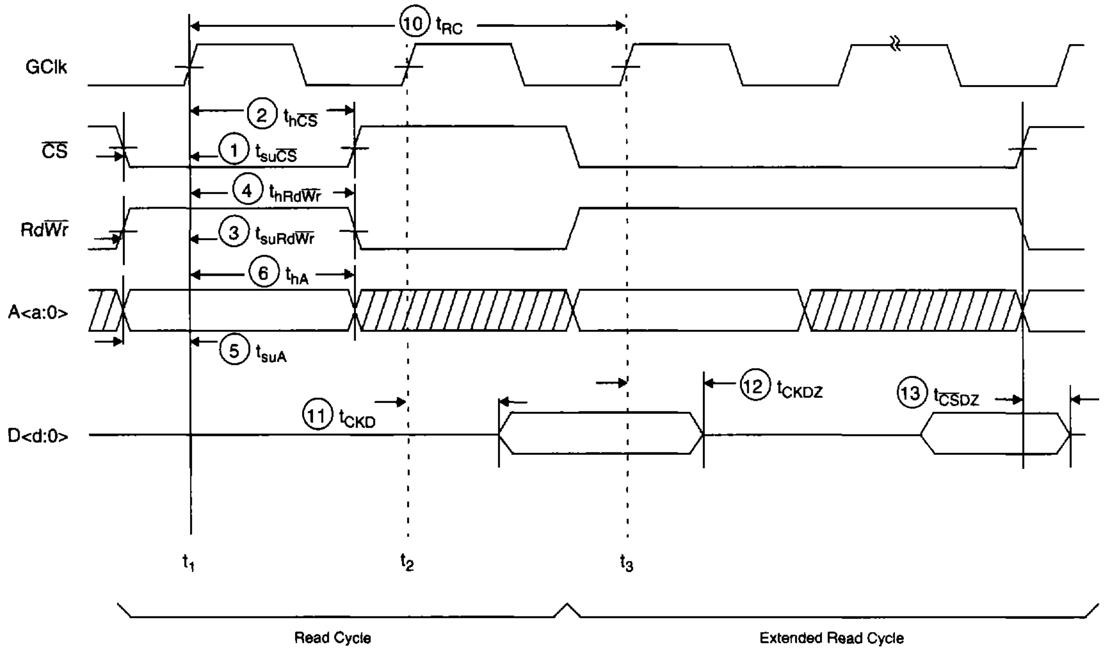

Figure 29: Configuration Memory Read Cycles

XC6200 Pinout Tables

XC6216 Pinouts - West Side

Pin Description	PQ240	PG299		Pin Description	PQ240	PG299
D0/W_1[2]	60	C18		V_{CC}	30	A11
GND	59	A19		GND	29	A10
W_{14}	58	A20		D19/W_{39}	28	C10
NC	57[1]	C17[1]		W_{40}	-	D10
D1/W_3	56	D16		D9/W_{19}	27	A9
W_{16}	55	E15		D20/W_{41}	26	E10
D2/W_5	54	B18		W_{42}	-	B9
W_{18}	53	B17		D21/W_{43}	25	C9
D3/W_7	52	C16		D10/W_{21}	24	A8
NC	-	-		W_{44}	-	B8
NC	-	-		W_{46}	-	D9
NC	-	D15[1]		D22/W_{45}	23	A7
NC	-	A18[1]		GND	22	-
W_{20}	51	E14		W_{48}	21	E9
D4/W_9	50	C15		W_{50}	20	C8
W_{22}	49	B16		V_{CC}	19	A6
W_{24}	48	D14		D24/W_{49}	18	B7
W_{26}	47	A17		D11/W_{23}	17	C7
D5/W_{11}	46	C14		D23/W_{47}	16	D8
NC	-	E13[1]		D25/W_{51}	15	B6
NC	-	B15[1]		GND	14	A5
GND	45	A15		W_{52}	-	B5
W_{28}	44	D13		W_{54}	-	E8
W_{30}	43	B14		W_{56}	13	C6
W_{32}	42	C13		D12/W_{25}	12	D7
D6/W_{13}	41	A14		W_{58}	11	A4
V_{CC}	40	A16		D26/W_{53}	10	C5
D16/W_{33}	39	B13		D27/W_{55}	9	B4
W_{34}	38	E12		D13/W_{27}	8	E7
GND	37	-		W_{60}	-	D6
NC	-	D12[1]		W_{62}	-	A3
NC	-	C12[1]		NC	-	-
NC	-	A13[1]		NC	-	-
NC	-	B12[1]		D28/W_{57}	7	C4
D7/W_{15}	36	A12		D14/W_{29}	6	D5
D17/W_{35}	35	D11		D29/W_{59}	5	E6
W_{36}	34	E11		D15/W_{31}	4	B3
D18/W_{37}	33	C11		D30/W_{61}	3	B2
D8/W_{17}	32	B11		D31/W_{63}	2	D4
W_{38}	31	B10		GND	1	B1

Notes: 1. Pin not connected.
2. Pins with a dual function have the 'Control' signal shown first. See section "Input/Output Blocks (IOBs)" on page 261 for details.

XC6216 Pinouts - South Side

Pin Description	PQ240	PG299	Pin Description	PQ240	PG299
V_{CC}	61	B20	GND	91	K20
$Rd\overline{Wr}/S_1$	62	D17	$G1/S_{17}$	92	L19
CS/S_3	63	B19	$G2/S_{19}$	93	L18
W_{12}/S_0	64	C19	S_{42}	94	L16
$\overline{OE}/S_5$	65	F16	S_{41}	95	L17
W_{10}/S_2	66	E17	S_{44}	-	M20
$\overline{Reset}/S_7$	67	D18	S_{43}	-	M19
W_8/S_4	68	C20	S_{46}	-	N20
NC	-	-	S_{48}	-	M18
NC	-	-	E_0/S_{50}	96	M17
W_6/S_6	69	F17	E_2/S_{52}	97	M16
W_4/S_8	70	G16	GND	98	-
W_2/S_{10}	71	D19	E_4/S_{54}	99	N19
W_0/S_{12}	72	E18	S_{45}	100	P20
S_{14}	73	D20	V_{CC}	101	T20
S_{16}	74	G17	S_{21}	102	N18
S_{18}	-	F18	$SEData/S_{23}$	103	P19
S_{20}	-	H16	E_6/S_{56}	104	N17
S_{22}	-	E19	S_{47}	105	R19
S_{24}	-	F19	GND	106	R20
GND	75	E20	NC	-	N16[1]
S_{26}	76	H17	NC	-	P18[1]
S_{33}	77	G18	E_8/S_{58}	107	U20
$\overline{Serial}/S_9$	78	G19	S_{49}	108	P17
$Wait/S_{11}$	79	H18	S_{51}	109	T19
V_{CC}	80	F20	S_{53}	110	R18
S_{28}	81	J16	S_{55}	-	P16
S_{35}	82	G20	S_{57}	-	V20
GND	83	-	E_{10}/S_{60}	111	R17
S_{30}	-	J17	E_{12}/S_{62}	112	T18
S_{32}	-	H19	NC	-	-
S_{34}	-	H20	NC	-	-
S_{36}	-	J18	$\overline{SECE}/S_{25}$	113	U19
$GClk/S_{13}$	84	J19	$SEReset/S_{27}$	114	V19
S_{38}	85	K16	S_{59}	115	R16
$GClr/S_{15}$	86	J20	S_{61}	116	T17
S_{37}	87	K17	$SEClk/S_{29}$	117	U18
S_{40}	88	K18	$ConfigOK/S_{31}$	118	X20
S_{39}	89	K19	GND	119	W20
V_{CC}	90	L20	S_{63}	120	V18

Note: 1. Pin not connected.

XC6216 Pinouts - East Side

Pin Description	PQ240	PG299	Pin Description	PQ240	PG299
V_{CC}	121	X19	GND	151	X11
E_{14}	122	U17	$A8/E_{17}$	152	W10
$A0/E_1$	123	W19	E_{40}	153	V10
E_{16}	124	W18	$A9/E_{19}$	154	T10
$A1/E_3$	125	T15	E_{41}	155	U10
E_{18}	126	U16	E_{42}	156	X9
$A2/E_5$	127	V17	E_{43}	157	W9
E_{20}	128	X18	NC	-	X8[1]
NC	-	U15[1]	E_{44}	-	V9
NC	-	T14[1]	E_{46}	-	U9
NC	-	-	E_{48}	-	T9
NC	-	-	GND	158	-
$A3/E_7$	129	W17	$A10/E_{21}$	159	W8
E_{22}	130	V16	E_{50}	160	X7
E_{24}	131	X17	V_{CC}	161	X5
E_{26}	132	U14	E_{45}	162	V8
$A4/E_9$	133	V15	E_{52}	163	W7
E_{28}	134	T13	$A11/E_{23}$	164	U8
NC	-	W16[1]	E_{47}	165	W6
NC	-	W15[1]	GND	166	X6
GND	135	X16	E_{54}	-	T8
E_{30}	136	U13	E_{56}	-	V7
E_{32}	137	V14	E_{58}	167	X4
$A5/E_{11}$	138	W14	E_{49}	168	U7
E_{33}	139	V13	$A12/E_{25}$	169	W5
V_{CC}	140	X15	E_{51}	170	V6
E_{34}	141	T12	E_{53}	171	T7
E_{35}	142	X14	E_{55}	172	X3
GND	143	-	NC	-	-
NC	-	U12	NC	-	-
NC	-	W13	$A13/E_{27}$	173	U6
NC	-	X13	E_{57}	174	V5
NC	-	V12	E_{60}	-	W4
$A6/E_{13}$	144	W12	E_{62}	-	W3
E_{36}	145	T11	$A14/E_{29}$	175	T6
E_{37}	146	X12	E_{59}	176	U5
E_{38}	147	U11	$A15/E_{31}$	177	V4
$A7/E_{15}$	148	V11	E_{61}	178	X1
E_{39}	149	W11	E_{63}	179	V3
V_{CC}	150	X10	V_{CC}	180	W1

Note: 1. Pin not connected.

XC6216 Pinouts - North Side

Pin Description	PQ240	PG299		Pin Description	PQ240	PG299
V_{CC}	240	A2		GND	211	L1
NC	-	-		N_{15}	210	L2
N_1	239	C3		N_{17}	209	L3
N_3	238	D3		N_{19}	208	L4
N_0	237	E4		N_{42}	207	M1
N_2	236	F5		N_{41}	206	L5
N_4	235	C2		N_{44}	205	M2
N_6	234	D2		GND	204	-
N_8	-	E3		N_{43}	203	M3
N_{10}	-	F4		N_{21}	202	N1
NC	-	-		N_{46}	-	N2
NC	-	-		N_{48}	-	M4
N_5	233	C1		N_{50}	-	P1
N_7	232	G5		N_{52}	-	M5
N_{12}	231	F3		V_{CC}	201	R1
N_{14}	230	E2		N_{23}	200	N3
N_{16}	229	G4		N_{54}	199	P2
N_{18}	228	D1		N_{45}	198	P3
N_{20}	-	G3		N_{56}	197	N4
N_{22}	-	H5		GND	196	T1
GND	227	F1		NC	-	R2[1]
N_{24}	226	F2		N_{47}	195	T2
N_{26}	225	H4		N_{58}	194	N5
N_{33}	224	G2		N_{49}	193	R3
N_{28}	223	H3		N_{51}	192	P4
V_{CC}	222	E1		N_{53}	191	U1
N_{30}	-	G1		N_{55}	190	T3
N_{32}	-	H2		N_{57}	189	U2
N_{34}	-	J5		N_{60}	-	P5
N_{36}	-	J4		N_{62}	-	R4
N_{35}	221	J3		NC	-	-
N_9	220	H1		NC	-	-
GND	219	-		N_{59}	188	V1
N_{11}	218	J2		N_{25}	187	U3
N_{38}	217	J1		N_{27}	186	T4
N_{37}	216	K4		N_{61}	185	R5
N_{40}	215	K5		N_{63}	184	V2
N_{39}	214	K3		N_{29}	183	W2
N_{13}	213	K2		GND	182	X2
V_{CC}	212	K1		N_{31}	181	U4

Note: 1. Pin not connected.

Product Availability

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 parallel 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 6. (These options are advance information and subject to change. Please confirm availability with Xilinx.

Signal pins are all the non-supply pins that drive into the array or control circuitry. Some of these pins are shared between control signals and user I/O. The un-shared user I/Os do not share a pin with a control signal. The number of user I/Os available will be 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.

Table 6: XC6216 Package Options

Package	Pins	Signal Pins	Max Data Bus Pins	Unshared User I/O
PLCC	84	68	16	22
PQFP	240	199	32	137
PGA	299	242	32	180