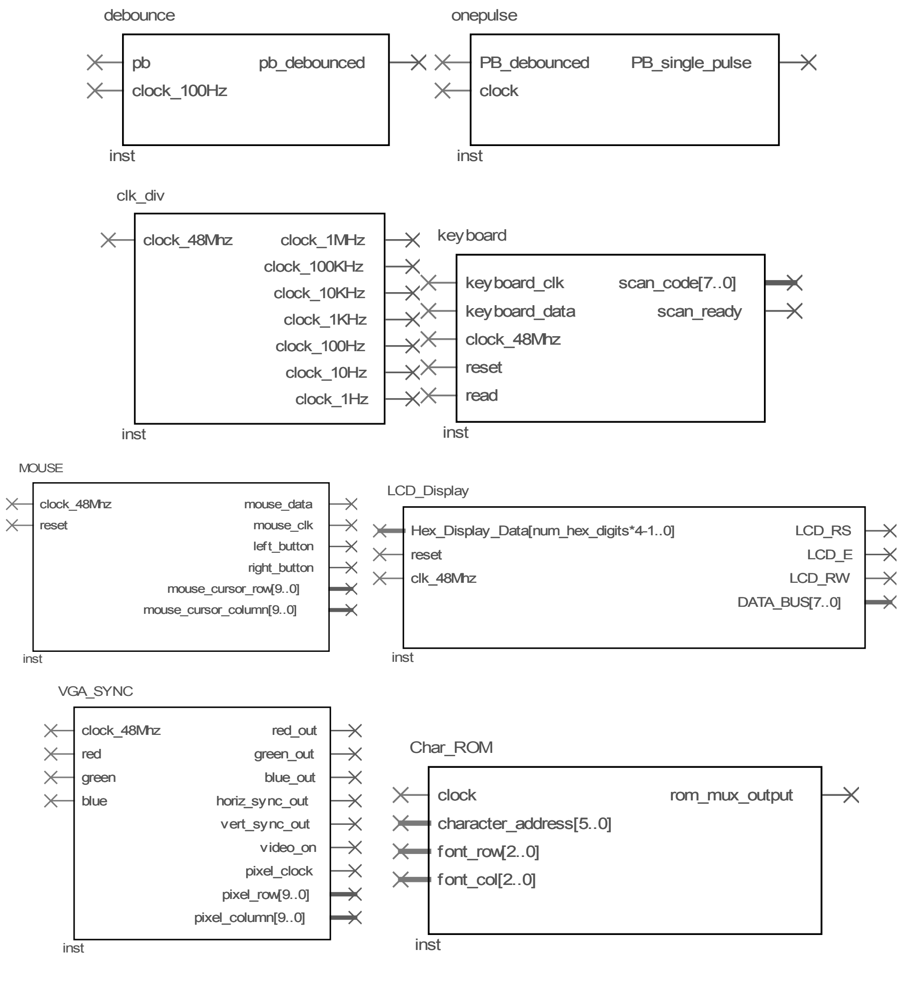
CHAPTER 5

FPGAcore Library Functions



5 FPGAcore Library Functions

In complex hierarchical designs, intellectual property (IP) cores are frequently used. An IP core is a previously developed synthesizable hardware design that provides a widely used function. Commercially licensed IP cores include functions such as microprocessors, microcontrollers, bus interfaces, multimedia and DSP functions, and communications controllers. IP cores promote design reuse and reduce development time by providing common hardware functions for use in a new design.

The FPGAcore functions listed in Table 5.1 are designed to simplify use of the FPGA board’s pushbuttons, keyboard, mouse, LCD display, seven-segment LEDs, and video output features. They can be used in schematic capture, VHDL, or Verilog based designs. Full source code is provided on the DVD.

Table 5.1 The FPGAcore Functions.

FPGAcore Name	Description
LCD_Display	Displays ASCII Characters and Hex Data on an LCD Panel
DEC_7SEG	Display Hex Data on a seven-segment LED Display
Debounce	Pushbutton Debounce Circuit
OnePulse	Pushbutton Single Pulse Circuit
Clk_Div	Clock Prescaler with 7 slower frequency outputs (1MHz to 1hz)
VGA_Sync	VGA Sync signal generator for FPGA that outputs pixel addresses
Video_PLL	Used by VGA Sync to generate the video pixel clock using a PLL
Char_ROM	Small Character Font ROM for video character generation
Keyboard	Reads keyboard scan codes from the board’s PS/2 connector
Mouse	Reads PS/2 mouse data and outputs cursor row and column address

FPGAcores can be used as symbols from the FPGAcore library, accessed via a VHDL package, or used as a component in other VHDL or Verilog files. An example of using the FPGAcore package in VHDL can be found in the file *board*\chap5\FPGApack.vhd available on the DVD. The use of FPGApack’s VHDL package saves retyping lengthy component declarations for the core functions in each VHDL-based design.

This section contains a one-page summary of each FPGAcore interface. VHDL source code is provided for all FPGAcores on the DVD. Additional documentation, examples, and interface details can be found in later chapters on video signal generation, the keyboard, and the mouse. The Clk_Div, LCD_Display, and Debounce functions were already used in the tutorial design example in Chapter 4.

For correct operation of the FPGAcore functions, I/O pin assignments must be made as shown in the description of each FPGAcore function. Clock inputs are also required on several of the FPGAcore functions. The Clk_Div FPGAcore is setup to provide the slower clock signals needed by some of the core functions.

Source code for the FPGAcore functions must be in the project directory or in the user's library search path. Review Section 4.2 for additional information on checking the library path. The VGA_Sync, Video_PLL, and LCD_Display core functions are often modified by the user to support different display resolutions and message options for each design. Be sure to select the right version of these core functions when adding file paths for a new project.

For UP2 and UP1 users, the same functionality is provided in the FPGAcore library functions found on the DVD in the **\UP2\Chap5** and **\UP1\Chap5** directories. Since the DE1 and UP2 do not have an LCD display, the seven segment display core, DEC_7SEG, must be used instead.

On the UP2 and UP1, only the 640 by 480 video mode is possible since the FPGA on these boards does not have a Phase Locked Loop (PLL) to generate the higher clock frequencies needed for other video display resolutions.

All of the Source code for the various FPGAcore functions can be found on the DVD in the *Board*\Chap5 subdirectories.

5.1 FPGAcore LCD_Display: LCD Panel Character Display

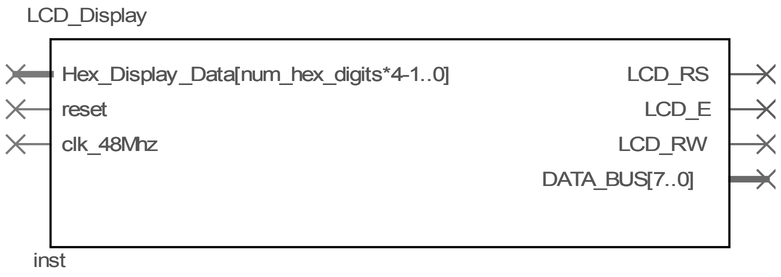


Figure 5.1 Symbol for LCD_Display FPGAcore.

The LCD_Display core is used to display static ASCII characters and changing hex values from hardware on the DE2's or UP3's 16 by 2 line LCD display panel. The core's VHDL code can be configured internally by the user to display different ASCII strings and hex data fields. Instructions can be found in comments in the core's VHDL code. A Generic, *Num_Hex_Digits*, is used to set the size of the Hex_Display_Data input (i.e., Each hex digit displayed requires a 4-bit signal).

The LCD controller datasheet contains information on graphics characters and LCD commands. A state machine is used to send data and commands to the LCD controller and to generate the required handshake signals. An ASCII to hex table can be found in Appendix D. See *LCD_Display.vhd* for more information.

5.1.1 VHDL Component Declaration

```
COMPONENT LCD_Display
   PORT(Hex_Display_Data:     IN   STD_LOGIC_VECTOR
                              ((Num_Hex_Digits*4)-1 DOWNTO 0);
          reset, clock_48MHz: IN STD_LOGIC;
          LCD_RS, LCD_E:      OUT    STD_LOGIC;
          LCD_RW:             INOUT STD_LOGIC;
         DATA_BUS:            INOUT  STD_LOGIC_VECTOR(7 DOWNTO 0));
END COMPONENT;
```

5.1.2 Inputs

Hex_Display_Data contains the 4-bit hexadecimal hardware signal values to convert to ASCII hex digits and send to the LED display. The Generic, Num_Hex_Digits, adjusts the size of the input hex data. Generics can be assigned a value in an HDL file or with a block's parameter assignment in a schematic. In a schematic, use **View ⇨Parameter assignment** to see the generic value and the symbol's properties parameters tab to set it.

5.1.3 Outputs

Outputs control an 8-bit tri-state bidirectional data bus to the LCD panel. Handshake lines are used to transfer ASCII data to the display. Pin assignments for the LCD module on the DE2 and UP3 boards are listed below. The DE1, UP2, and UP1 boards do not have a built-in LCD display module.

Table 5.2 The LCD Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
LCD_E	-	K3	50	-	Output	LCD Enable line
LCD_RW	-	K4	73	-	Output	LCD R/W control line
LCD_RS	-	K1	108	-	Output	LCD Register Select Line
LCD_DATA[0]	-	J1	94	-	Bidir.	LCD Data Bus
LCD_DATA[1]	-	J2	96(133)	-	Bidir.	LCD Data Bus
LCD_DATA[2]	-	H1	98	-	Bidir.	LCD Data Bus
LCD_DATA[3]	-	H2	100	-	Bidir.	LCD Data Bus
LCD_DATA[4]	-	J4	102(108)	-	Bidir.	LCD Data Bus
LCD_DATA[5]	-	J3	104	-	Bidir.	LCD Data Bus
LCD_DATA[6]	-	H4	106	-	Bidir.	LCD Data Bus
LCD_DATA[7]	-	H3	113	-	Bidir.	LCD Data Bus

5.2 FPGAcore DEC_7SEG: Hex to Seven-segment Decoder

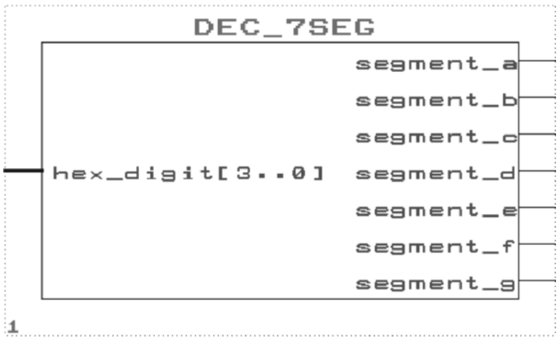


Figure 5.2 Symbol for DEC_7SEG FPGAcore.

The FPGAcore Dec_7seg shown in Figure 5.2 is a hexadecimal to seven-segment display decoder with active low outputs. This function is used to display hex numbers on an FPGA board's seven-segment LED displays.

5.2.1 VHDL Component Declaration

```
COMPONENT dec_7seg
        PORT( hex_digit         : IN    STD_LOGIC_VECTOR ( 3 DOWNTO 0 );
              seg_a, seg_b, seg_c,
              seg_d, seg_e, seg_f,
              seg_g             : OUT   STD_LOGIC );
END COMPONENT;
```

5.2.2 Inputs

Hex_digit is the 4-bit hexadecimal value to send to the LED display. Hex_digit[3] is the most-significant bit.

5.2.3 Outputs

Segments a through g are active low and should be connected as output pins to the corresponding pin on a seven-segment display. Table 5.3 lists the pin assignments for the first two hex displays on the FPGA boards. The UP3 does not contain seven segment displays, but it does have an LCD module. More than two seven segment displays are available on both the DE1 and DE2 boards, see the appropriate board's user manuals for a complete list of pin assignments.

Table 5.3 The Seven Segment Display Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
HEX0[0]	J2	AF10	-	6	Output	Seven Segment Display 0 LED Segment A (0=on)
HEX0[1]	J1	AB12	-	7	Output	Seven Segment Display 0 LED Segment B (0=on)
HEX0[2]	H2	AC12	-	8	Output	Seven Segment Display 0 LED Segment C (0=on)
HEX0[3]	H1	AD11	-	9	Output	Seven Segment Display 0 LED Segment D (0=on)
HEX0[4]	F2	AE11	-	11	Output	Seven Segment Display 0 LED Segment E (0=on)
HEX0[5]	F1	V14	-	12	Output	Seven Segment Display 0 LED Segment F (0=on)
HEX0[6]	E2	V13	-	13	Output	Seven Segment Display 0 LED Segment G (0=on)
HEX1[0]	E1	V20	-	17	Output	Seven Segment Display 1 LED Segment A (0=on)
HEX1[1]	H6	V21	-	18	Output	Seven Segment Display 1 LED Segment B (0=on)
HEX1[2]	H5	W21	-	19	Output	Seven Segment Display 1 LED Segment C (0=on)
HEX1[3]	H4	Y22	-	20	Output	Seven Segment Display 1 LED Segment D (0=on)
HEX1[4]	G3	AA24	-	21	Output	Seven Segment Display 1 LED Segment E (0=on)
HEX1[5]	D2	AA23	-	23	Output	Seven Segment Display 1 LED Segment F (0=on)
HEX1[6]	D1	AB24	-	24	Output	Seven Segment Display 1 LED Segment G (0=on)

5.3 FPGAcore Debounce: Pushbutton Debounce

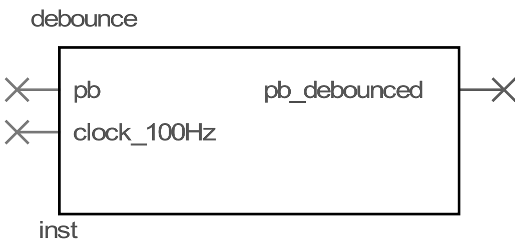


Figure 5.3 Symbol for Debounce FPGAcore.

The FPGAcore Debounce shown in Figure 5.3 is a pushbutton debounce circuit. This function is used to filter mechanical contact bounce in a switch or pushbutton. A shift register is used to filter out the switch contact bounce. The shift register takes several time spaced samples of the switch input and changes the output only after several sequential samples are the same value.

5.3.1 VHDL Component Declaration

```
COMPONENT  debounce
        PORT( pb, clock_100Hz        : IN          STD_LOGIC;
              pb_debounced           : OUT         STD_LOGIC;
END COMPONENT;
```

5.3.2 Inputs

PB is the raw pushbutton input. It should be tied to an input pin connected to a pushbutton or slide switch. See Chapter 2 for pushbutton and switch pin numbers. Clock is a clock signal of approximately 100Hz that is used for the internal 50ms switch debounce filter circuits. UP3, UP2, and UP1 board pushbuttons are not debounced in hardware, but DE2 and DE1 boards are.

5.3.3 Outputs

PB_debounced is the debounced pushbutton output. The output will remain Low until the pushbutton is released. If a pulse is needed to be only 1 clock period long, add the OnePulse core function to the debounced switch output.

Table 5.4 The Pushbutton Switch Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
KEY0	R22	G26	48	28 PB1	Input	Pushbutton KEY0 (debounced, 0 = button hit)
KEY1	R21	N23	49	29 PB2	Input	Pushbutton KEY1 (debounced, 0 = button hit)
KEY2	T22	P23	57	-	Input	Pushbutton KEY2 (debounced, 0 = button hit)
KEY3	T21	W26	62	-	Input	Pushbutton KEY3 (debounced, 0 = button hit)

5.4 FPGAcore OnePulse: Pushbutton Single Pulse

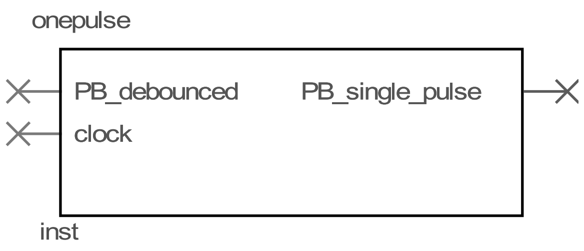


Figure 5.4 Symbol for OnePulse FPGAcore.

The FPGAcore OnePulse shown in Figure 5.4 is a pushbutton single-pulse circuit. Output from the pushbutton is High for only one clock cycle no matter how long the pushbutton is pressed. This function is useful in state machines that read external pushbutton inputs. In general, fewer states are required when it is known that inputs only activate for one clock cycle. Internally, an edge-triggered flip-flop is used to build a simple state machine.

5.4.1 VHDL Component Declaration

```
COMPONENT  onepulse
        PORT( PB_debounced, clock   : IN     STD_LOGIC;
              PB_single_pulse       : OUT    STD_LOGIC  );
END COMPONENT;
```

5.4.2 Inputs

PB_debounced is the debounced pushbutton input. It should be connected to a debounced pushbutton.

Clock is the user's state-machine clock. It can be any frequency. In some designs, the user may want to edit the VHDL code to add a reset input.

5.4.3 Outputs

PB_single_pulse is the output, which is High for only one clock cycle when a pushbutton is hit.

5.5 FPGAcore Clk_Div: Clock Divider

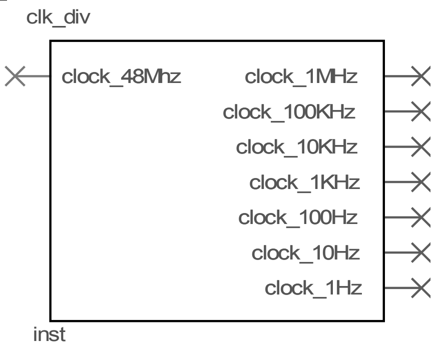


Figure 5.5 Symbol for Clk_Div FPGAcore.

The FPGAcore Clk_Div shown in Figure 5.5 is used to provide clock signals slower than the on-board clock oscillator. The output signals are obtained by dividing down the clock input signal. Multiple output taps provide clock frequencies in powers of ten.

5.5.1 VHDL Component Declaration

```
COMPONENT clk_div
        PORT(   clock_48MHz      : IN     STD_LOGIC;
                clock_1MHz       : OUT    STD_LOGIC;
                clock_100kHz     : OUT    STD_LOGIC;
                clock_10kHz      : OUT    STD_LOGIC;
                clock_1kHz       : OUT    STD_LOGIC;
                clock_100Hz      : OUT    STD_LOGIC;
                clock_10Hz       : OUT    STD_LOGIC;
                clock_1Hz        : OUT    STD_LOGIC );
END COMPONENT;
```

5.5.2 Inputs

A different frequency input clock is used on different FPGA boards, so each board has a slightly different version of this function. The UP3 version is shown. Clock_48MHz is an input pin that should be connected to the UP3 on-board 48MHz USB clock. The pin number for the UP3's 48MHz USB clock is 29. Make sure the JP3 jumper selects the 48MHz USB clock (default setting).

5.5.3 Outputs

Clock_1MHz through clock_1Hz provide output signals of the specified frequency. Based on a crystal oscillator, the actual frequency is 1.007 ± .005% times the listed value.

Table 5.5 The Crystal Oscillator Clock Pin Assignments

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
CLOCK	L1 50Mhz	N2 50Mhz	153 48Mhz	91 25Mhz	Input	25-50MHz Clock

5.6 FPGAcore VGA_Sync: VGA Video Sync Generation

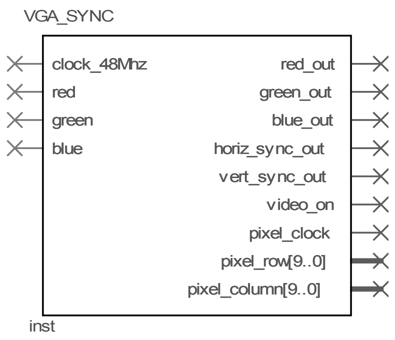


Figure 5.6 Symbol for VGA_Sync FPGAcore.

The FPGAcore VGA_Sync shown in Figure 5.6 provides horizontal and vertical sync signals to generate an 8-color 640 by 480 pixel VGA video image. For more detailed information on video signal generation see Chapter 9.

A table of the common screen resolutions and refresh rates along with the required pixel clocks and sync count values can be found at the end of the VGA_Sync IP core source code. When changing resolutions or refresh rates, use the MegaWizard edit feature to adjust the *video_pll.vhd* code to output a different pixel clock rate and change the horizontal and vertical sync counter limits to the six new values found in the table.

Video_pll.vhd must be present to compile VGA_Sync since it uses this component for the clock.

5.6.1 VHDL Component Declaration

```
COMPONENT vga_sync
   PORT( clock_48MHz, red, green, blue: IN   STD_LOGIC;
   red_out, green_out, blue_out,
   horiz_sync_out, vert_sync_out:           OUT STD_LOGIC;
   pixel_row, pixel_column:                 OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
END COMPONENT;
```

5.6.2 Inputs

Clock is an input pin that must be connected to the on-board clock. One of the FPGA's Phase Locked Loops (PLL) is used to generate the exact video clock rate required in the *video_pll.vhd* code. Red, Green, and Blue inputs provide the color information for the video signal. External user logic must generate the RGB input signals. One of the FPGA's PLLs is used to generate the required pixel clock on all boards except the UP2 and UP1, which only provide a 25Mhz clock for 640 by 480 modes. An external reference clock of 48 or 50MHz is used by the PLL for the input clock on the other FPGA boards.

5.6.3 Outputs

- Horiz_sync is an output pin that should be tied to the VGA horizontal sync.
- Vert_sync is an output pin that should be tied to the VGA vertical sync.
- Red_out is an output pin that should be tied to the red RGB signal.
- Green_out is an output pin that should be tied to the green RGB signal.
- Blue_out is an output pin that should be tied to the blue RGB signal.

An interface circuit on the UP2 & UP3 boards converts the digital red, green, and blue video color signals to the appropriate analog voltage for the monitor. On the UP3, set jumper JP3 to short pins 3-4 for the 48Mhz clock. Eight colors are possible using the three digital color signals. A four-bit signal is used on the DE1 for a total of 4096 colors and the DE2 has 10-bits per color. Just wiring up a signal to a couple of the high color bits on boards with more colors gives a solid color for simple designs that use only a few colors.

Pixel_clock, pixel_row, and pixel_column are outputs that provide the current pixel clock and the pixel address. Video_on indicates that pixel data is being displayed and a retrace cycle is not presently occurring. These outputs are used by user logic to generate RGB color input data.

Table 5.6 The VGA Video Display Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
CLOCK	L1 50Mhz	N2 50Mhz	153 48Mhz	91 25Mhz	Input	25-50MHz Clock - PLL input on DE*x* and UP3
VGA_RED	B7	E10	228	236	Output	VGA Red Video Signal (highest bit)
VGA_GREEN	A8	D12	122	237	Output	VGA Green Video Signal (highest bit)
VGA_BLUE	B10	B12	170	238	Output	VGA Blue Video Signal (highest bit)
VGA_VSYNC	B11	D8	226	239	Output	VGA Connector Vertical Sync Signal
VGA_HSYNC	A11	A7	227	240	Output	VGA Connector Horizontal Sync Signal

5.7 FPGAcore Char_ROM: Character Generation ROM

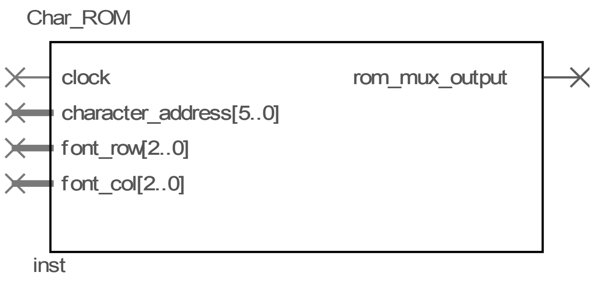


Figure 5.7 Symbol for Char_ROM FPGAcore.

The FPGAcore Char_ROM shown in Figure 5.7 is a character generation ROM used to generate text in a video display. Each character is represented by an 8 by 8 pixel font. For more information on video character generation see Chapter 10. Character codes are listed in Table 10.2 of Section 10.9. Font data is contained in the memory initialization file, tcgrom.mif. One Cyclone M4K memory block is required for the ROM that holds the font data.

5.7.1 VHDL Component Declaration

```
COMPONENT char_rom
        PORT( clock               : IN STD_LOGIC;
              character_address   : IN STD_LOGIC_VECTOR( 5 DOWNTO 0 );
              font_row, font_col  : IN STD_LOGIC_VECTOR( 2 DOWNTO 0 );
              rom_mux_output      : OUT STD_LOGIC);
END COMPONENT;
```

5.7.2 Inputs

Character_address is the address of the alphanumeric character to display. Font_row and font_col are used to index through the 8 by 8 font to address the single pixels needed for video signal generation. Clock loads the address register and should be tied to the video pixel_clock.

5.7.3 Outputs

Rom_mux_output is the pixel font value indexed by the address inputs. It is used by user logic to generate the RGB pixel color data for the video signal.

5.8 FPGAcore Keyboard: Read Keyboard Scan Code

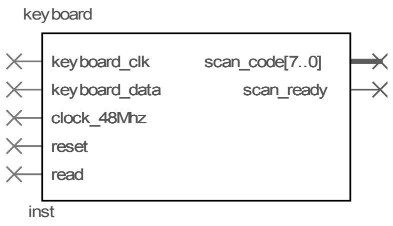


Figure 5.8 Symbol for Keyboard FPGAcore.

The FPGAcore Keyboard shown in Figure 5.8 is used to read the PS/2 keyboard scan code from a keyboard attached to the FPGA board's PS/2 connector. This function converts the serial data from the keyboard to parallel format to produce the scan code output. For detailed information on keyboard applications and scan codes see Table 11.2 in Chapter 11.

5.8.1 VHDL Component Declaration

```
COMPONENT keyboard
        PORT( keyboard_clk, keyboard_data, clock_48MHz ,
              reset, read       : IN    STD_LOGIC;
              scan_code         : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
              scan_ready        : OUT STD_LOGIC);
END COMPONENT;
```

5.8.2 Inputs

- Clock_48MHz is an input pin that must be connected to the on-board clock oscillator.
- Keyboard_clk and keyboard_data are PS/2 input data lines from the keyboard.
- Pin assignments for the FPGA boards are listed in Table 5.7.

Read is a handshake input signal. The rising edge of the read signal clears the scan ready signal. Reset is an input that clears the internal registers and flags used for serial-to-parallel conversion.

Table 5.7 The PS/2 Keyboard Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
PS2_CLK	H15	D26	12	30	Bidir.	PS2 Connector
PS2_DATA	J14	C24	13	31	Bidir.	PS2 Connector

5.8.3 Outputs

Scan_code contains the bytes transmitted by the keyboard when a key is pressed or released. See Table 11.2 in Chapter 11 for a listing of keyboard scan codes. Scan codes for a single key are a sequence of several bytes. A make code is sent when a key is hit, and a break code is sent whenever a key is released. When typing, it is normal to have several keys on a keyboard depressed at the same time on different fingers. Also, some features require holding down multiple keys, so several make codes may be seen before the break code for a particular key.

Scan_ready is a handshake output signal that goes High when a new scan code is sent by the keyboard. The read input clears scan_ready. The scan_ready handshake line should be used to ensure that a new scan code is read only once.

5.9 FPGAcore Mouse: Mouse Cursor

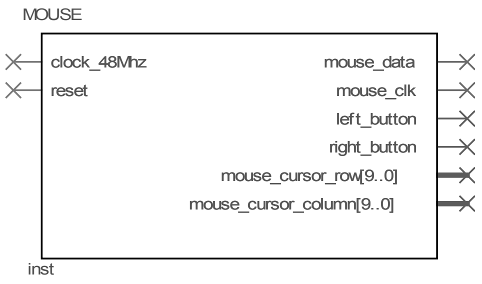


Figure 5.9 Symbol for Mouse FPGAcore.

The FPGAcore Mouse shown in Figure 5.9 is used to read position data from a mouse attached to the UP3's PS/2 connector. It outputs a row and column cursor address for use in video applications. The mouse must be attached to the FPGA board prior to downloading for proper initialization.

The internal operation of the core and more detailed information on mouse applications, commands, and data formats can be found in Chapter 11.

5.9.1 VHDL Component Declaration

```
COMPONENT mouse
        PORT( clock_48MHz, reset     : IN      STD_LOGIC;
              mouse_data             : INOUT   STD_LOGIC;
              mouse_clk              : INOUT   STD_LOGIC;
              left_button, right_button : OUT  STD_LOGIC;
              mouse_cursor_row       : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
              mouse_cursor_column    : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
END COMPONENT;
```

5.9.2 Inputs

Clock_48MHz is an input pin that must be connected to the on-board clock. On DE1 and DE2 boards a 50Mhz clock is used. Mouse_clk and mouse_data are bi-directional PS/2 signal lines from the mouse.

Table 5.8 The PS/2 Mouse Pin Assignments

Pin Name	**DE1**	**DE2**	**UP3**	**UP2, UP1**	**Pin Type**	**Function of Pin**
PS2_CLK	H15	D26	12	30	Bidir.	PS2 Clk Connector
PS2_DATA	J14	C24	13	31	Bidir.	PS2 Data Connector

5.9.3 Outputs

Mouse_cursor_row and mouse_cursor_column are outputs that contain the current address of the mouse cursor in the 640 by 480 screen area. The cursor is initialized to the center of the screen. Left_button and right_button outputs are High when the corresponding mouse button is pressed.

5.10 For additional information

The FPGA cores summarized in this chapter are used extensively in the textbook's design examples, and complete source code is provided on the DVD. They are provided to support any new FPGA designs that you may develop. Extensive lists of more complex commercial third-party IP cores available for purchase can be found at the major FPGA vendor web sites, www.altera.com and www.xilinx.com. Pricing on commercial cores can be expensive and access to source code may not be provided. An assortment of free open source IP cores for FPGAs is available at www.opencores.org.

Source Code for the FPGAcore functions is on the DVD.

In the Book's Design Examples, Additional materials

can be found in each \BOARD\\Chap*X* directory.