interfering with the DIP switch logic levels. Set all the DIP switches to the OPEN position.

- Attach a VGA monitor to the DB-HD15 connector (J5).
- Download the VGA40.BIT file and a video test pattern into the XS40/XStend combination with the command: XSLOAD TESTPATT.HEX VGA40.BIT.
- Release the reset to the VGA circuitry with the command: XSPORT 0.
- Observe the color bars on the monitor screen.
- The steps for compiling and testing the design using an XS95 combined with an XStend Board are as follows:
  - Synthesize the VHDL code in the VGA95/XGA.VHD file for an XC95108 CPLD.
  - Compile the synthesized netlist using the VGA95.UCF constraint file (Listing 19).
  - Generate an SVF file for the design.
- Mount an XS95 Board in the XStend Board and attach the downloading cable from the XS95 to the PC parallel port. Apply 9VDC though jack J9 of the XS40. Place shunts on jumpers J4, J7, and J8 of the XStend Board to enable the LED displays. Remove the shunt on jumper J17 to keep the XStend codec serial output from interfering. Set all the DIP switches to the OPEN position.
- Attach a VGA monitor to the DB-HD15 connector (J5).
- Download the VGA95.SVF file and a video test pattern into the XS95/XStend combination with the command: XSLOAD TESTPATT.HEX VGA95.SVF.
- Release the reset to the VGA circuitry with the command: XSPORT 0.
- Observe the color bars on the monitor screen.

**Reading Keyboard Scan Codes Through the PS/2 Interface**

This example creates a circuit that accepts scan codes from a keyboard attached to the PS/2 interface of the XStend Board. The binary pattern of the scan code is displayed on the bargraph LEDs. In addition, if a scan code for one of the keys '0'—'9' arrives, then the numeral will be displayed on the right LED display of the XStend Board.

The format of the scan code transmissions from the keyboard are shown in Figure 10. The keyboard electronics drive the clock and data lines. The start of a scan code transmission is indicated by a low level on the data line on the falling edge of the clock. The eight bits of the scan code follow (starting with the least-significant bit) on successive falling clock edges. These are followed by an odd-parity bit and then a high-level stop bit.

When the clock line goes high after the stop bit, the receiver (in this case, the FPGA or CPLD on the XS Board inserted in the XStend Board) can pull the clock line low to inhibit any further transmissions. After the clock line is released and it returns to a high level, the
keyboard can send another scan code. If the receiver never pulls the clock line low, then
the keyboard will send scan codes whenever a key is pressed.

- **Figure 10**: Keyboard data transmission waveforms.

The VHDL code for this example is shown in . The inputs and outputs of the circuit as
defined in the entity declaration are as follows:

rst: This output drives the reset pin of the microcontroller on the XS Board.

oeb: This output drives the output-enable pin of the RAM on the XS Board.

kb_data: The scan code bits enter through this input.

kb_clk: The keyboard clock signal enters through this input.

db: These outputs drive the segments of the bargraph LED on the XStend Board.

rsb: These outputs drive the segments of the right LED digit on the XStend Board.

Within the main body of the architecture section, these operations occur:

**Lines 22 & 23**: The microcontroller reset pin and the RAM output-enable pin are driven
high so these chips interfere while receiving data from the keyboard.

**Lines 25 & 26**: The keyboard clock passes through an input buffer and then a global clock
buffer before it reaches the rest of the circuitry. (These buffers are declared on lines
18 and 19, respectively.) The global clock buffer distributes the clock signal with
minimal skew in the XS40 Board FPGA. These statements are not used with the
CPLD in the XS95 Board.

gather_scancode: On every falling edge of kb_clk, this process shifts the data bit on the
kb_data input into the most-significant bit of a 10-bit shift register. After 11 clock
cycles, the lower 8 bits of the register will contain the scan code, the upper 2 bits will
store the stop and parity bits, and the start bit will have been shifted through the entire
register and discarded.

**Line 38**: The value in the shift register is inverted and applied to the segments of the LED
bargraph. Since the bargraph segments are active-low, a segment will light for every
‘1’ bit in the shift register. The LED segment drivers are not registered so there will be
some flickering as the shift register contents change.
Lines 40-51: If the scan code in the shift register matches the codes for the digits 0-9, then the right LED digit segments will be activated to display the corresponding digit. If the scan code does not match one of these codes, the letter 'E' is displayed.

The steps for compiling and testing the design using an XS40 combined with an XStend Board are as follows:

- Synthesize the VHDL code in the KEYBRD40|KEYBRD.VHD for an XC4005XL FPGA.
- Compile the synthesized netlist using the KEYBRD40.UCF constraint file (Listing 21).
- Mount an XS40 Board in the XStend Board and attach the downloading cable from the XS40 to the PC parallel port. Apply 9VDC though jack J9 of the XS40. Place shunts on jumpers J4, J7, and J8 to enable the LEDs. Remove the shunt on jumper J17 to keep the XStend codec from interfering. Set all the DIP switches to the OPEN position.
- Attach a keyboard to the PS/2 connector of the XStend Board.
- Download the KEYBRD40.BIT file into the XS40/XStend combination with the command: XSLOAD KEYBRD40.BIT.
- Press keys on the keyboard and observe the results on the LED displays.

The steps for compiling and testing the design using an XS95 combined with an XStend Board are as follows:

- Synthesize the VHDL code in the KEYBRD95|KEYBRD.VHD for an XC95108 CPLD.
- Compile the synthesized netlist using the KEYBRD95.UCF constraint file (Listing 22).
- Generate an SVF file for the design.
- Mount an XS95 Board in the XStend Board and attach the downloading cable from the XS95 to the PC parallel port. Apply 9VDC though jack J9 of the XS95. Place shunts on jumpers J4, J7, and J8 to enable the LEDs. Remove the shunt on jumper J17 to keep the XStend codec from interfering. Set all the DIP switches to the OPEN position.
- Download the KEYBRD95.SVF file into the XS95/XStend combination with the command: XSLOAD KEYBRD95.SVF
- Press keys on the keyboard and observe the results on the LED displays.

- **Listing22**: VHDL code for receiving keyboard scan codes from the PS/2 interface.

```vhdl
001- LIBRARY IEEE;
002- USE IEEE.STD_LOGIC_1164.ALL;
003-
004- ENTITY kbd_read IS
005- PORT
006- (}
007-   rst: OUT STD_LOGIC; -- uC reset
008-   oeb: OUT STD_LOGIC; -- RAM output enable
009-   kb_data: IN STD_LOGIC; -- serial data from the keyboard
010-   kb_clk: IN STD_LOGIC; -- clock from the keyboard
011-   db: OUT STD_LOGIC_VECTOR(8 DOWNTO 1); -- bargraph LED
012-   rsb: OUT STD_LOGIC_VECTOR(6 DOWNTO 0) -- right LED digit
013-   );
014- END kbd_read;
015-
016- ARCHITECTURE kbd_read_arch OF kbd_read IS
017- SIGNAL scancode: STD_LOGIC_VECTOR(9 DOWNTO 0);
018- COMPONENT ibuf PORT(i: IN STD_LOGIC; o: OUT STD_LOGIC); END COMPONENT;
019- COMPONENT bufg PORT(i: IN STD_LOGIC; o: OUT STD_LOGIC); END COMPONENT;
020- SIGNAL buf_clk0, buf_clk1: STD_LOGIC;
021- BEGIN
022-   rst <= '1'; -- keep the uC in the reset state
023-   oeb <= '1'; -- disable the RAM output drivers
024-   b0: ibuf PORT MAP(i=>kb_clk,o=>buf_clk0); -- buffer the clock from
025-   b1: bufg PORT MAP(i=>buf_clk0,o=>buf_clk1); -- the keyboard
026-   -- shift keyboard data into the MSb of the scancode register
027-   -- on the falling edge of the keyboard clock
028-   gather_scancode:
029-     PROCESS(buf_clk1,scancode)
030-       BEGIN
031-       IF(buf_clk1'EVENT AND buf_clk1='0') THEN
032-          scancode <= kb_data & scancode(9 DOWNTO 1);
033-       END IF;
034-     END PROCESS;
035-   END IF;
036- END PROCESS;
037-   db <= NOT(scancode(7 DOWNTO 0)); -- show scancode on the bargraph
038-   -- display the key that was pressed on the right LED digit
039-   rsb <= "11011011" WHEN scancode(7 DOWNTO 0)="000101010" ELSE -- 1
040-      "01000110" WHEN scancode(7 DOWNTO 0)="000111101" ELSE -- 2
041-      "01001010" WHEN scancode(7 DOWNTO 0)="001001101" ELSE -- 3
042-      "10001011" WHEN scancode(7 DOWNTO 0)="001011101" ELSE -- 4
043-      "00110100" WHEN scancode(7 DOWNTO 0)="001010110" ELSE -- 5
044-      "00100000" WHEN scancode(7 DOWNTO 0)="001101101" ELSE -- 6
045-      "01011101" WHEN scancode(7 DOWNTO 0)="010001110" ELSE -- 7
046-      "00000000" WHEN scancode(7 DOWNTO 0)="000011110" ELSE -- 8
047-      "00010000" WHEN scancode(7 DOWNTO 0)="000000000" ELSE -- 9
048-      "00010000" WHEN scancode(7 DOWNTO 0)="000100000" ELSE -- 0
049-      "00100100" WHEN scancode(7 DOWNTO 0)="001001001" ELSE -- E
050-      "00100010" WHEN scancode(7 DOWNTO 0)="001000010" ELSE -- ON
051-      "00010001" WHEN scancode(7 DOWNTO 0)="000100011" ELSE -- 0
052-  END kbd_read_arch;

• Listing2t: XS40 UCF file for the PS/2 keyboard interface.

001- net rst loc=p36;
002- net oeb loc=p61;
003- net kb_data loc=p69;
004- net kb_clk loc=p68;
Inputting and Outputting Stereo Signals Through the Codec

The stereo codec on the XStand Board is capable of digitizing two analog signals to 20 bits of resolution while simultaneously generating two analog signals from 20-bit values. A high-level view of the codec chip is shown on the right-half of Figure 11. Two analog inputs (which are typically the left and right channels of a stereo audio signal) enter the codec and are digitized into two 20-bit values by analog-to-digital converters (ADCs). These values are loaded into shift registers, which are shifted out of a single pin of the codec under control of a shift clock and a left/right channel selector control input. At the same time, 20-bit values are alternately shifted into two shift registers in the codec, which feed digital-to-analog converters (DACs) that drive two analog outputs. Signals on these outputs are typically the left and right channels of a stereo audio signal.