VGA Graphics Controller for SRAM Memory with Rectangle Generator

TSTE12, version 0.1 DRAFT 090831

This section discusses the design of a graphics controller driving a VGA screen. It consists of three parts; a VGA screen controller, an SRAM controller, and a rectangle generator.

Included is a description of the timing for the signals that drive a VGA monitor and description of an VHDL module that will let you drive a monitor with a picture stored in the SRAM memory. The SRAM memory can of course be replaced by the any other of the memories depending on design choices.

The rectangle generator should be controlled by switches on the board, and invert the colors of an rectangle section of the image stored in the SRAM memory. The SRAM memory is only available to the graphics controller during the vertical retrace of the VGA controller.

The top level description of the design is shown below in

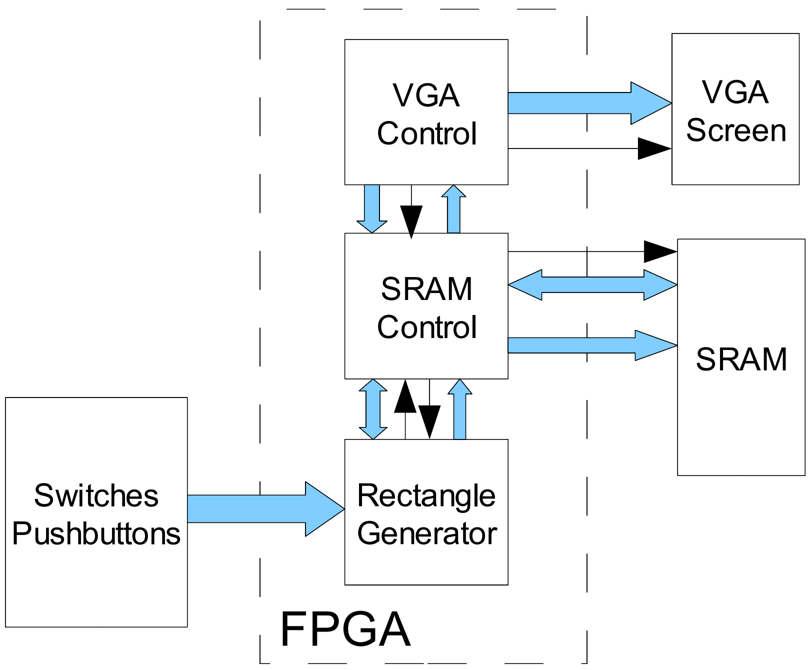

Figure 1: Overview of the complete system

1 SRAM Controller

The SRAM controller is responsible for reading from and writing to the SRAM. The other two units generates addresses which should be read/written. The SRAM controller should always give the VGA controller priority over the Rectangle Generator.

The SRAM controller should consist of a data path section connecting the SRAM address either to the VGA Controller or the Rectangle Generator. The Datapath should consist of loadable registers that store the data from the SRAM databus when enabled, or store data to be output on the SRAM databus.

The SRAM controller FSM (Finite State Machine) controls the SRAM operation. Input to the FSM is a reset signal, a VGA_ctrl signal indicating that the VGA controller wants access, a VGA_RD indicating that the VGA controller wants to read a memory cell, Rect_RD indicating the Rectangle generator want to read a memory cell, and, Rect_WR indicating that the Rectangle Generator want to write data to the memory.

Out from the SRAM controller FSM is enable signals controlling the register load, address multiplexer, SRAM OE, SRAM WE, and Rect_RD_Ack, Rect_WR_Ack.

The SRAM controller FSM will initially be in a wait state. In this state is the WR and OE signal high (no write, tristate driver in the SRAM), the Rect_Data_out is constantly stored in a register, the Rect_Data_in and VGA_Data_in are not loaded, the Rect_RD_Ack = 0, and Rect_WR_Ack = 0.

If VGA_RD = 1, then move to the next state, with Address select sending VGA_Addr to the SRAM address, SRAM_OE = 0, and store the SRAM Data in the next clock cycle. Return back to the idle state.

If Rect_RD = 1 and VGA_ctrl = 0 then move to the next state, with Address select sending the Rect_Addr to the SRAM address, SRAM_OE = 0, and store the SRAM Data in the next clock cycle. Send an Rect_RD_Ack for one clock cycle, and then return to the idle state.

If Rect_WR = 1 and VGA_ctrl = 0 then move to the next state, with Address select sending the Rect_Addr to the SRAM address, send the Rect_Data_out to the SRAM. In the following clock cycle set SRAM_WE = 0. The next cycle following that should return SRAM_WE = 1, and set Rect_WR_Ack = 1 for one clock cycle. Then return to the idle state.

2 VGA Screen Controller

The VGA screen is connected to the FPGA board through a 15 pin mini-DSUB connector. This connector contains three analog color information signal, ground signals, and two synchronization signal HSYNC and VSYNC. The image is stored in the SRAM memory.

2.1 VGA Color Signals

There are three signals -- red, green, and blue -- that send color information to a VGA monitor. These three signals each drive an electron gun that emits electrons which paint one primary color at a point on the monitor screen. Analog levels between 0 (completely dark) and 0.7 V (maximum brightness) on these control lines tell the monitor what intensities of these three primary colors to combine to make the color of a dot (or pixel) on the monitor's screen.

Each analog color input can be set to one of 2^{10} (=1024) levels by controlling the three digital inputs vga_r, vga_g, and vga_b on the digital-to-analog converters in the VGA DAC chip. The 1024 possible levels on each analog input are combined by the monitor to create a pixel with one of 1024x1024x1024 = 1 G different colors.

2.2 VGA Signal Timing

A frame of VGA video typically has 480 lines and each line usually contains 640 pixels. In order to paint a frame, there are deflection circuits in the monitor that move the electrons emitted from the guns both left-to-right and top-to-bottom across the screen. These deflection circuits require two synchronization signals in order to start and stop the deflection circuits at the right times so that a line of pixels is painted across the monitor and the lines stack up from the top to the bottom to form an image. The timing for the VGA synchronization signals is shown in Figure 2: VGA signal timing.

Negative pulses on the horizontal sync signal mark the start and end of a line and ensure that the monitor displays the pixels between the left and right edges of the visible screen area. The actual pixels are sent to the monitor within a 25.4 ms window. The horizontal sync signal drops low a minimum of 0.6 ms after the last pixel and stays low for 3.8 ms. A new line of pixels can begin a minimum of 1.9 ms after the horizontal sync pulse ends. So a single line occupies 25.4 ms of a 31.7 ms interval. The other 6.3 ms of each line is the horizontal blanking interval during which the screen is dark.

In an analogous fashion, negative pulses on a vertical sync signal mark the start and end of a frame made up of video lines and ensure that the monitor displays the lines between the top and bottom edges of the visible monitor screen. The lines are sent to the monitor within a 15.25 ms window. The vertical sync signal drops low a minimum of 0.32 ms after the last line and stays low for 63 ms. The first line of the next frame can begin a minimum of 1.02 ms after the vertical sync pulse ends. So a single frame occupies 15.25 ms of a 16.784 ms interval. The other 1.5 ms of the frame interval is the vertical blanking interval during which the screen is dark.

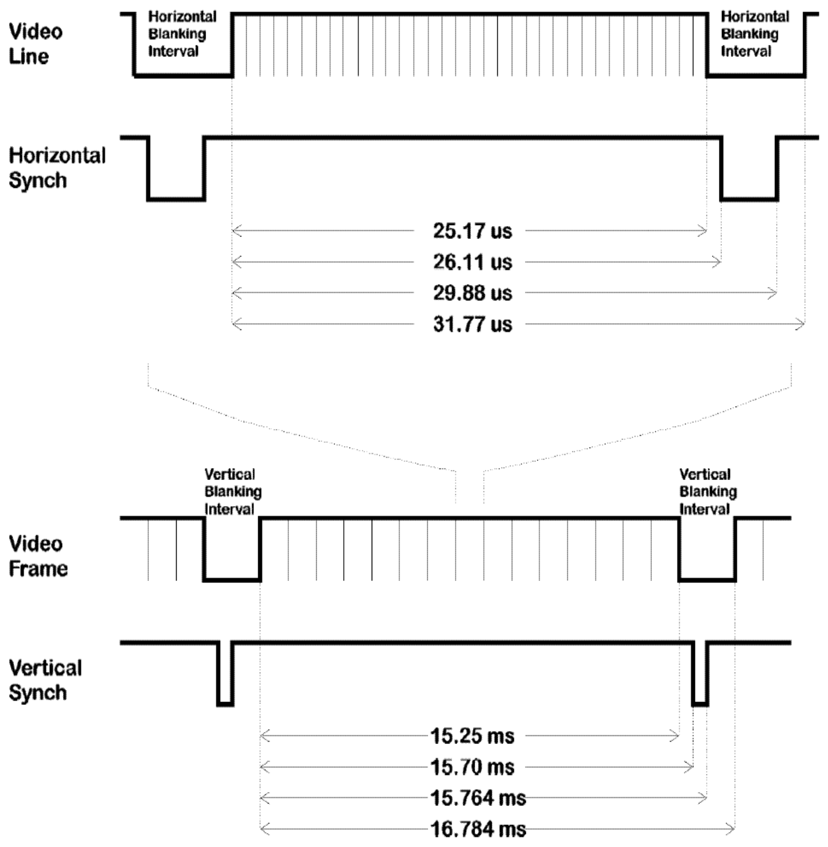

Figure 2: VGA signal timing

VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(Mhz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25 (640/c)
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36 (640/c)
SVGA(60Hz)	800x600	3.2	2.2	20	1	40 (800/c)
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49 (800/c)
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56 (800/c)

2.3 VGA Signal Generator Algorithm

VGA mode		Vertical Timing Spec			
Configuration	Resolution (HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10
VGA(85Hz)	640x480	3	25	480	1
SVGA(60Hz)	800x600	4	23	600	1
SVGA(75Hz)	800x600	3	21	600	1
SVGA(85Hz)	800x600	3	27	600	1

Figure 3: VGA timing modes

We now have to figure out a process that will send pixels to the monitor with the correct timing and framing. We can store a picture in the SRAM of the DE2 Board. Then we can retrieve the data from the SRAM, format it into lines of pixels, and send the lines to the monitor with the appropriate pulses on the horizontal and vertical sync pulses.

The pseudocode for a single frame of this process is shown in Figure 4. The pseudocode has two outer loops: one which displays the L lines of visible pixels, and another which inserts the V blank lines and the vertical sync pulse. Within the first loop, there are two more loops: one which sends the P pixels of each video line to the monitor, and another which inserts the H blank pixels and the horizontal sync pulse.

Within the pixel display loop, there are statements to get the next word from the SRAM. Each word contains two eight-bit pixels. A small loop iteratively extracts each pixel to be displayed from the lower eight bits of the word. Then the word is shifted by eight bits to the left so the next pixel will be in the right position during the next iteration of the loop. Since it has only eight bits, each pixel can store one of 256 levels of color or gray. The mapping from the eight-bit pixel value to the actual values required by the monitor electronics is done by the COLOR_MAP() routine. In this design we only make use of 256 levels of gray.

```
for line_cnt=1 to L              /* send L lines of video to the monitor */
    for pixel_cnt=1 to P             /* send P pixels for each line */
        data = RAM(address)          /* get pixel data from the memory */
        address = address + 1        /* SRAM data word contains 2 pixels */
        for d=1 to 2
            pixel = data & 0x00FF              /* mask off pixel in the lower eight bits */
            data = data>>8                     /* shift next pixel into lower eight bits */
            color = COLOR_MAP(pixel)           /* get the color for the right-bit pixel */
            send color to monitor
            d = d + 1                          /* increment by two pixels */
        pixel_cnt = pixel_cnt + 2
    for horiz_blank_cnt=1 to H           /* blank the monitor for H pixels */
        color = BLANK
        send color to monitor
        /* pulse the horizontal sync at the right time */
        if horiz_blank_cnt>HB0 and horiz_blank_cnt<HB1
            hsync = 0
        else
            hsync = 1
        horiz_blank_cnt = horiz_blank_cnt + 1
    line_cnt = line_cnt + 1
for vert_blank_cnt=1 to V            /* blank the monitor for V lines and insert vertical sync */
    color = BLANK
    send color to monitor
    /* pulse the vertical sync at the right time */
    if vert_blank_cnt>VB0 and vert_blank_cnt<VB1
        vsync = 0
    else
        vsync = 1
    vert_blank_cnt = vert_blank_cnt + 1
/* go back to start of picture in memory */
address = 0
```

Figure 4: VGA signal generation pseudocode

Figure 5 shows how to pipeline certain operations to account for delays in accessing data from the SRAM. The pipeline has four stages:

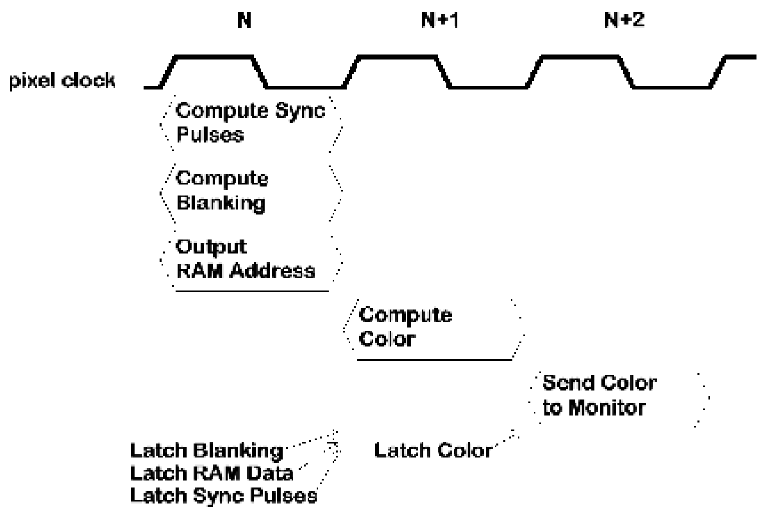

Figure 5: Pipelining of VGA signal generation tasks

The timing picture above shows that the blankning is calculated before the color, and it may therefore be necessary to compensate for this in the calculation of the blank and sync signals. Failure in this will create a screen image that is missing lines to the left or right.

2.4 Problem definition

Here are the definitions listed that are needed to complete the design.

2.4.1 Port definitions

The inputs and outputs of the circuit as defined are as follows:

Name	Type	Range	Description
fpga_clk	IN	std_logic	The system clock
reset_n	IN	std_logic	The circuit reset signal. Reset is active low, I.e., resert='0' gives reset
vga_clk	IN	std_logic	The DAC clock signal. Typically pixel clock signal.
vga_sync	OUT	std_logic	Not in use. Inactivate this with a logical '0'
vga_blank	OUT	std_logic	The blank signal from the design.
vga_r	OUT	std_logic_vector 9 downto 0	The red component of the display rgb signal. Always set the unused lower two bits to '0'
vga_g	OUT	std_logic_vector 9 downto 0	The green component of the display rgb signal. Always set the unused lower two bits to '0'
vga_b	OUT	std_logic_vector 9 downto 0	The blue component of the display rgb signal. Always set the unused lower two bits to '0'
vga_hsync	OUT	std_logic	The display horizontal sync pulse
vga_vsync	OUT	std_logic	The display vertical sync pulse
data	IN	std_logic_vector 15 downto 0	The display data from SRAM
adress	OUT	std_logic_vector 17 downto 0	The address to display data SRAM. Always set unused bits to '0'
we	OUT	std_logic	SRAM write enable. Set to '1' while reading image
oe	OUT	std_logic	SRAM output enable. Set to '0' while reading image
ce	OUT	std_logic	SRAM chip enable. Set to '0' in the VHDL code.
ub	OUT	std_logic	SRAM high-byte data mask. Set to '0' in VHDL code
lb	OUT	std_logic	SRAM low-byte data mask. Set to '0' in VHDL code

Pin location for these signals can be found in the board documentation available at /proj/tde/TSTE12/kursmaterial/DE2/DE2_user_manual/DE2_UserManual.pdf and /proj/tde/TSTE12/kursmaterial/DE2_70/DE2_70_user_manual/DE2_70 User Maual.

2.4.2 Interface

The synthesis tool needs to know how the FPGA is connected to the external world. An attribute description included in the the VHDL description will be used for this purpose. How to include this into the design was described in the keyboard exercise.

With guidance from the port definitions each group have to create their own attribute description. All necessary information is found in the documentation of the "DE2 education board - users manual".

2.4.3 Structure of the VGA Signal Generator

The pseudocode and pipeline timing in the last section will help us to understand the structure and create the VHDL code for a VGA signal generator.

2.4.3.1 Inputs and outputs

fpga_clk, vga_clk

The input for the 50 MHz clock of the DE2 Board. Derive the vga_clk signal from this clock. The vga_clk is used by the DAC to clock the individual pixels.

reset

Reset all the other circuitry to a known state. Connect this to one of the push buttons on the board.

vga_hsync, vga_vsync

The outputs for the horizontal and vertical sync pulses. The vga_hsync output should be declared as a buffer because it should also be referenced within the architecture section as an enable signal for the vertical line counter (count up one when detecting an edge on vga_hsync).

vga_blank

The output for the display blank signal.

vga_r, vga_g, vga_b

The outputs that control the red, green, and blue color guns. Each analog color input can be set to one of 2^{10} (=1024) levels by the three digital inputs vga_r, vga_g and vga_b using the video digital-to-analog converter. The 1024 possible levels on each analog input are combined by the monitor to create a pixel with one of 1024x1024x1024 = 1 G different colors. Always set the two unused lower bits to '0'. We only intend to use gray scale in the image. Thus send the data value read from SRAM to all rgb components.

addr, data

The outputs for driving the address lines of the SRAM and the inputs for receiving the data from the SRAM.

oe, we, ce, ub, lb select signals

Control signals for the SRAM interface. More information is available in the DE2 users manual and SRAM datasheet.

2.4.3.2 Signals

hcnt, vcnt

The counters that store the current horizontal position within a line of pixels and the vertical position of the line on the screen. We will call these the horizontal or pixel counter, and the vertical or line counter, respectively. The line period is 31.77 µs that is 794 clock cycles, so the pixel counter needs at least ten bits of resolution. Each frame is composed of 528 video lines (only 480 are visible, the other 48 are blanked), so a ten bit counter is needed for the line counter.

pixrg

The eight-bit register that stores the four pixels received from the SRAM.

hblank, vblank, pblank

The video blanking signal and its registered counterpart that is used in the next pipeline stage.

1...1 Processes

pixelclk_generator

Create the pixel clock signal used by other process. Note that this is only an enable signal, not a signal that will be used as a real clock. The only clock in your system must be the fpga_clk.

pixelcounter

This process describes the operation of the horizontal pixel counter. The counter is synchronously set to zero when the reset is applied. The counter increments on the rising edge of each pixel clock. The range for the horizontal pixel counter is [0,793]. When the counter reaches 793, it rolls over to zero on the next cycle. Thus, the counter has a period of 794 pixel clocks. With a pixel clock of 25 MHz, this translates to a period of 31.75 µs.

linecounter

This process describes the operation of the vertical line counter. The counter is synchronously set to zero when the reset input is low. The counter increments when the rising edge of the horizontal sync pulse is detected, that is, after a line of pixels is completed. The range for the vertical line counter is [0,524]. When the counter reaches 524, it rolls over to zero on the next cycle. Thus, the counter has a period of 525 lines. Refer to the the tables in the end of 2.2, “ VGA Signal Timing” for more information on the length of each subpart.

hsyncr

This process describes the operation of the horizontal sync pulse generator. The horizontal sync is set to its inactive high level when the reset is activated. During normal operations, the horizontal sync output is updated on every pixel clock. Refer to the the tables in the end of 2.2, “ VGA Signal Timing” for more information on the length of each subpart.

Here is also the hblank signal created. The video is blanked after 640 pixels on a line are displayed. The blanking signal is active high.

vsyncr

This process describes the operation of the vertical sync pulse generator. The vertical sync is set to its inactive high level when the reset is activated. During normal operations, the vertical sync output is updated after every line of pixels is completed. Refer to the the tables in the end of 2.2, “ VGA Signal Timing” for more information on the length of each subpart.

Here is also the vblank signal created. The video is blanked after 480 lines are displayed. The blanking signal is active high.

blank_syncr

This process describes the operation of the pipelined video blanking signal. Within the process, the blanking signal is stored in a register so it can be used during the next stage of the pipeline when the color is computed.

Computation of the combinatorial blanking signal. The total blank is calculated as blank = hblank OR vblank

SRAM_control

The SRAM is permanently selected and writing to the SRAM is disabled. It stores the video data and acts like a ROM. In addition, the outputs from the SRAM are disabled when the video is blanked since there is no need for pixels during the blanking intervals. This is necessary since there are other circuits trying to access the SRAM.

The image we will display is stored in a linear sequence of words inside the SRAM. The address to the SRAM is a linear counter updated every second tick of pixel clock when we do not have blank signal. The counter can be reset to 0 whenever the vertical blanking is started.

pixel_reg

This process describes the operation of the register that holds the byte of pixel data read from SRAM. The register is asynchronously cleared when the VGA circuit is reset. The register is updated on the rising edge of each pixel clock. The pixel register is loaded with data from the SRAM every second pixel clock cycle. The active pixel is always in the lower eight bits of the register. Each pixel in the SRAM data byte is shifted into the active position by right shifting the register eight bits on each rising clock edge. This may also be solved using a multiplexer and selecting upper or lower part of the SRAM data.

vga_gen

This process describes the process by which the current active pixel is mapped into the 30 bits that drive the red, green and blue color guns. The register is set to zero (which displays as the color black) when the reset input is low. The color register is clocked on the rising edge of the pixel clock since this is the rate at which new pixel values arrive. The value clocked into the register is a function of the pixel value and the blanking input.

When the pipelined blanking input is low (inactive), the color displayed on the monitor is a gray scale value depending the input value from SRAM. This means we send the same SRAM value to all the RGB inputs to achieve a grayscale image. When the pipelined blanking input is high, the color register is loaded with zero (black).

3 Rectangle Generator

The rectangle generator is a microprogrammed structure that reads data from the SRAM, modifies the value, and then writes the new data back to SRAM. The SRAM memory area affected by the modification is defined by the switches of the board. Two of the switches are used to select which coordinate the value should be entered into. The group SW6-SW0 is used to specify the value (0-128), with the switches SW8 and SW7 defines where the value should be stored. Use KEY2 to toggle the update the coordinate. Present the current value on the 7-segment LEDS, with HEX7 and HEX6 indicating the upper left X coordinate of the rectangle, HEX5 and HEX4 shows the upper left Y coordinate of the rectangle, HEX3 and HEX2 indicates the width of the rectangle, and SW3-SW0 specifies the height of the rectangle.

KEY3 should be used to trigger the start of the rectangle modification of the SRAM memory contents.

The dataflow part of the Rectangle Generator should consist of two loadable counters for the X And Y koordinate, with an increment enable input. The Rectangle generator address output should be calculated from the contents of these registers. There should also be two loadable counters that decrements towards zero. These should have an decrement enable signal, and a zero detect flag.

When updating the values in the rectangle should all bits in the pixel value be inverted.

The microprogram should wait for a toggle on the KEY2 or KEY3, and carry out the corresponding activity.

3.1 Tips and tricks

To aviod a lot of trouble in the design phase a few hints are needed

- Do not use multiple clock definitions, i.e. , use only one statement with ’EVENT defining a clock.
- The use of ’EVENT and ’LAST_VALUE are not useful in the context of synthesis except for defining the rising or falling edge of clock.
- Use modelsim with a clock of 50Mhz to measure the hsync and vsync timing. This is a very easy way to validate your design using the simulator.
- Create first the VGA Controller and SRAM Controller, and ground the Rectangle input signals to the SRAM Controller. This should produce a nice looking picture on the VGA screen.

3.2 Requirements

Here are the requirements to pass the laboratory

- Implement the design using hdldesigner
- Declare a symbol and corresponding VHDL view
- Create the model in hdldesigner with one block for each process. Call the top level VGA. Use at least three levels including the top level symbol VGA. For example, divide the unit into a sync generation part and a color generation part..
- It is not allowed to have more than one clock domain. The best way to check this is to not have more than one signal appearing on statements using ’event

- Synthesize the design and demonstrate the function on the DE2 FPGA board.

◦ *Synthesis flow (using Precision).*

Follow the steps lined up in the keyboard laboratory exercise.

◦ *Downloading the Design*

Follow the steps described in the keyboard laboratory exercise.