XGA to WVGA Lanczos Image Scaler

April 2013 Reference Design RD1161

Introduction

Image scaling is the process of resizing a digital image. Scaling is a non-trivial process that involves a trade-off between efficiency, smoothness and sharpness. As the size of the image is increased, the pixels which comprise the image become increasingly visible, making the image appear “soft”. Conversely, reducing an image will tend to enhance its smoothness and sharpness.

This design document illustrates the implementation of a video image downscaler. The downscaling algorithm used is the Lanczos2 algorithm.

The design is implemented in VHDL. The Lattice iCEcube2™ Place and Route tool integrated with the Synopsys Synplify Pro® synthesis tool is used for the implementation of the design. The design can be targeted to other iCE40™ FPGA product family devices.

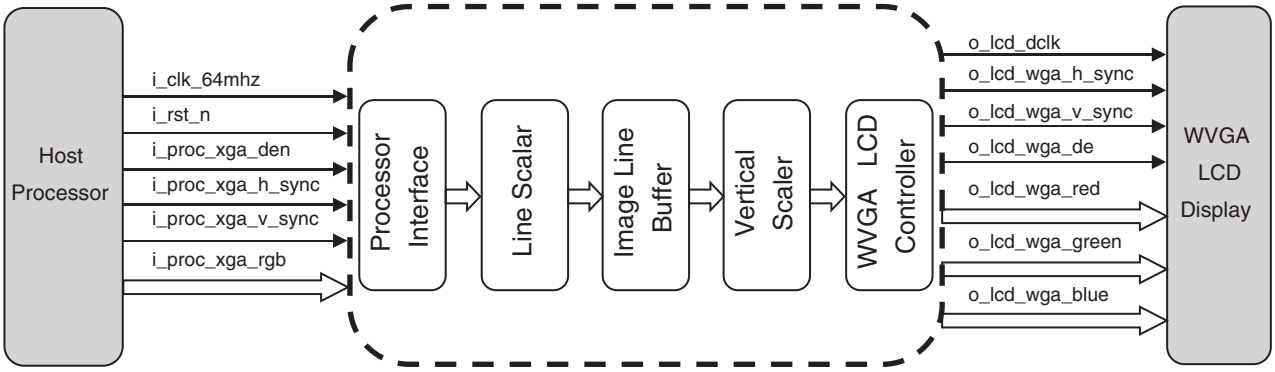
Figure 1 shows the System Block Diagram for the Image Scaler.

Features

- Downscales video images from XGA (1024x768) to WVGA (800x480)
- Supports 64MHz input pixel clock and 32MHz output pixel clock
- Supports RGB565 input and output video data format
- No external frame buffers required
- Built-in WVGA timing and data controller
- Internal image line buffer using iCE40 RAM block

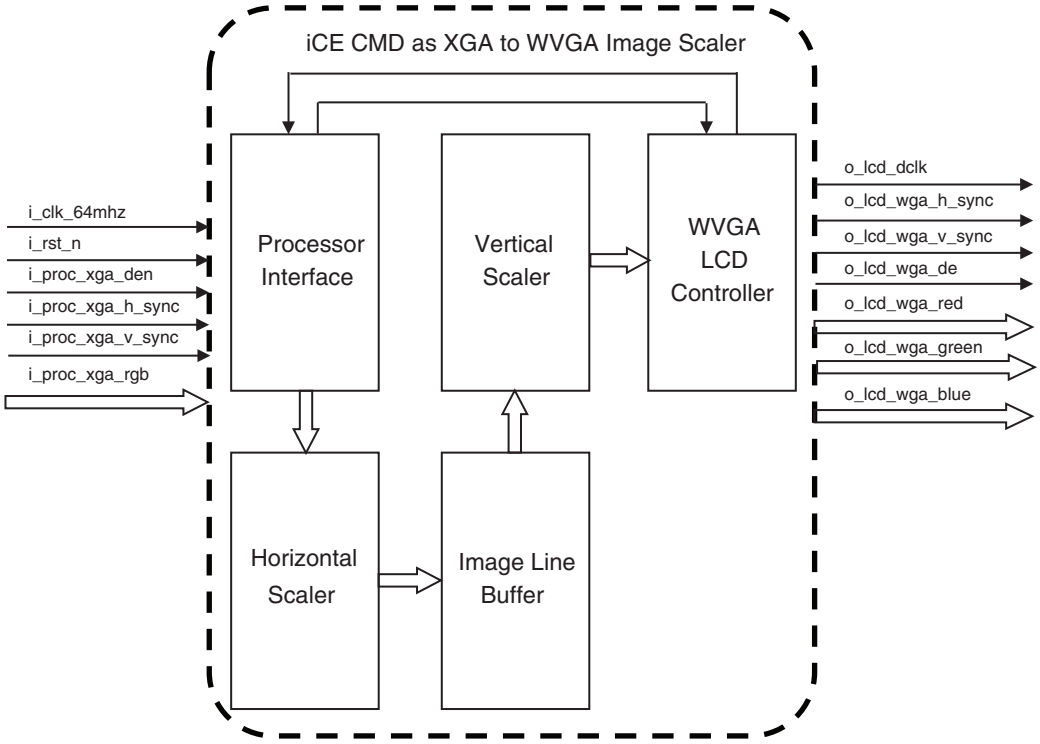
System Block Diagram

Figure 1. System Block Diagram



Functional Block Diagram

Figure 2. Functional Block Diagram



Processor Interface Module

A parallel Processor Interface is implemented between the processor and the Image Scaler. This module bridges the host device and the Image Scaler module. The host device provides the XGA video/image data to the Processor Interface in three data channels namely Red, Green and Blue (RGB). This interface also passes on the three data and timing control signals such as Data Enable (de), Horizontal Sync (h_sync), and Vertical Sync (v_sync) signals to the scaler. The h_sync signal determines the start and the end of a horizontal pixel line and the v_sync signal determines the end of a frame. The Processor Interface extracts incoming active pixels from the RGB Data channels and transfers them to the Image Scaler.

Image Scaler

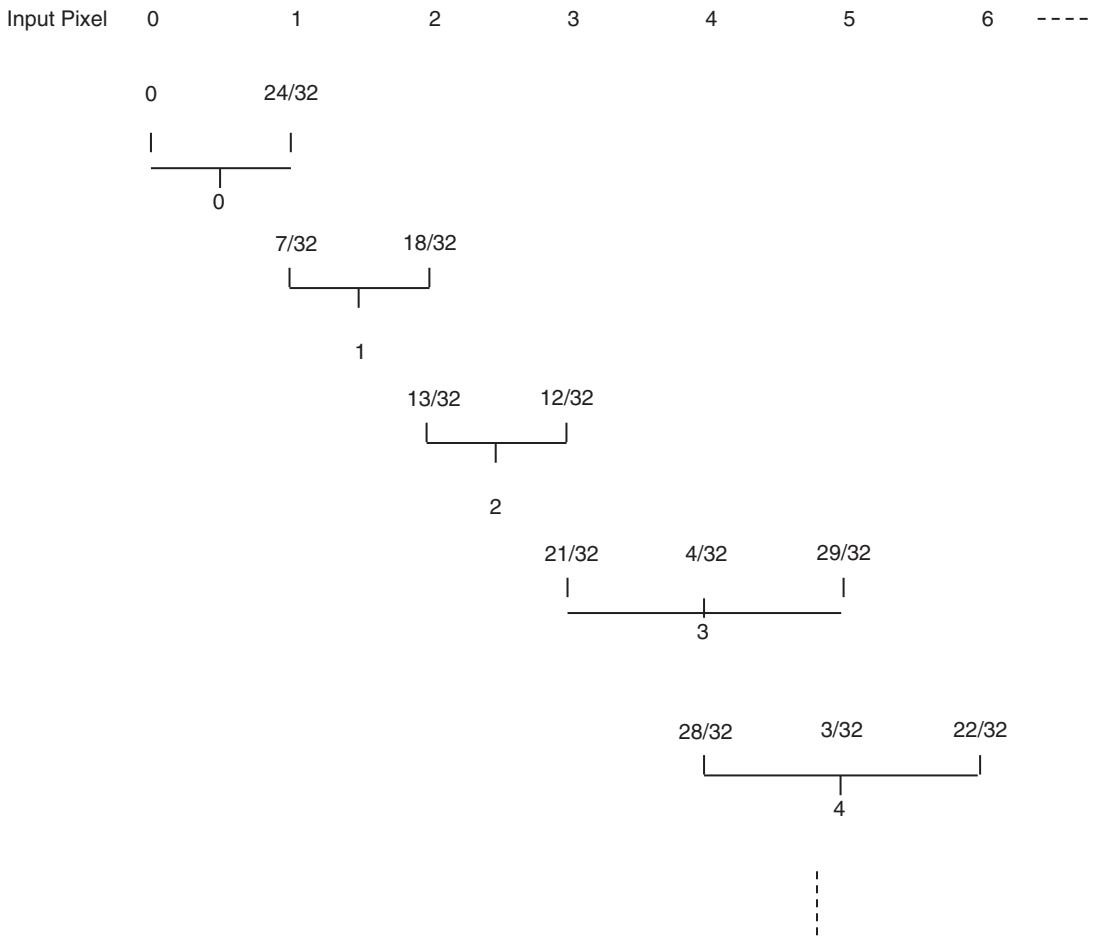
In this design the Image Scaler is implemented in two modules – the Horizontal Scaler and the Vertical Scaler.

Horizontal Scaler Module

The Horizontal Scaler downscales the incoming horizontal line of 1024 active pixels to 800 active pixels using the Lanczos2 scaling algorithm as illustrated below. The downscaled 800-pixel lines are stored in the line buffer first and then shifted into Vertical Scaler for downscaling from 768-line to 480-line.

Figure 3 below shows the pixel position and the computed Lanczos2 coefficients for the filter or scale

Figure 3. Lanczos Coefficients for the Filter



The Lanczos2 mathematical formula is as shown below:

```
Lanczos2 = {sin(x*pi)*sin(x*pi/2)}/(x*pi*x*pi/2)
```

Vertical Scaler Module

The Vertical Scaler takes in pixels from two different lines shifted out of the line buffer and vertically scale them to the targeted 480 lines. The algorithm involves averaging the two pixels from the two lines.

Below is the code snippet of the vertical scaling algorithm.

```
ScaleRectAvg(PIXEL *Target, PIXEL *Source, int SrcWidth, int SrcHeight, int Tgt-
Width, int TgtHeight, float threshold)
{
     iint NumPixels = TgtHeight;
     int IntPart = (SrcHeight / TgtHeight) * SrcWidth;
     int FractPart = SrcHeight % TgtHeight;
     int Mid = TgtHeight * threshold;
     int E = 0;
     int skip;
     PIXEL *ScanLine, *ScanLineAhead;
     PIXEL *PrevSource = NULL;
     PIXEL *PrevSourceAhead = NULL;
     skip = (TgtHeight < SrcHeight) ? 0 : TgtHeight / (2*SrcHeight) + 1;
     NumPixels -= skip;
     ScanLine = (PIXEL *)malloc(TgtWidth*sizeof(PIXEL));
     ScanLineAhead = (PIXEL *)malloc(TgtWidth*sizeof(PIXEL));
     while (NumPixels-- > 0) {
     if (Source != PrevSource) {
     if (Source == PrevSourceAhead) {
     PIXEL *tmp = ScanLine;
     ScanLine = ScanLineAhead;
     ScanLineAhead = tmp;
     } else {
     ScaleLineAvg(ScanLine, Source, SrcWidth, TgtWidth, threshold);
     }
     PrevSource = Source;
     }
     if (E >= Mid && PrevSourceAhead != Source+SrcWidth) {
     int x;
     ScaleLineAvg(ScanLineAhead, Source+SrcWidth, SrcWidth, TgtWidth, threshold);
     for (x = 0; x < TgtWidth; x++)
     ScanLine[x] = average(ScanLine[x], ScanLineAhead[x]);
     PrevSourceAhead = Source + SrcWidth;
     }
     memcpy(Target, ScanLine, TgtWidth*sizeof(PIXEL));
     Target += TgtWidth;
     Source += IntPart;
     E += FractPart;
     if (E >= TgtHeight) {
     E -= TgtHeight;
     Source += SrcWidth;
     }
     }
     if (skip > 0 && Source != PrevSource)
     ScaleLineAvg(ScanLine, Source, SrcWidth, TgtWidth, threshold);
     while (skip-- > 0) {
     memcpy(Target, ScanLine, TgtWidth*sizeof(PIXEL));
     Target += TgtWidth;
```

```
    }
    free(ScanLine);
    free(ScanLineAhead);
}
```

LCD Controller

The LCD Controller module generates the necessary timing and data control signals to drive a WVGA LCD display. During operation, the Processor interface module provides a synchronizing signal to the LCD Controller module to lock the start of frame. Essentially, this module uses the required back porch and front porch timing of WVGA frame to generate the h_sync, v_sync and de signals to the WVGA LCD display.

The timing and data control signals and data with respect to the pixel clock are shown in Figure 4.

Signal Description

Table 1. Signal Description

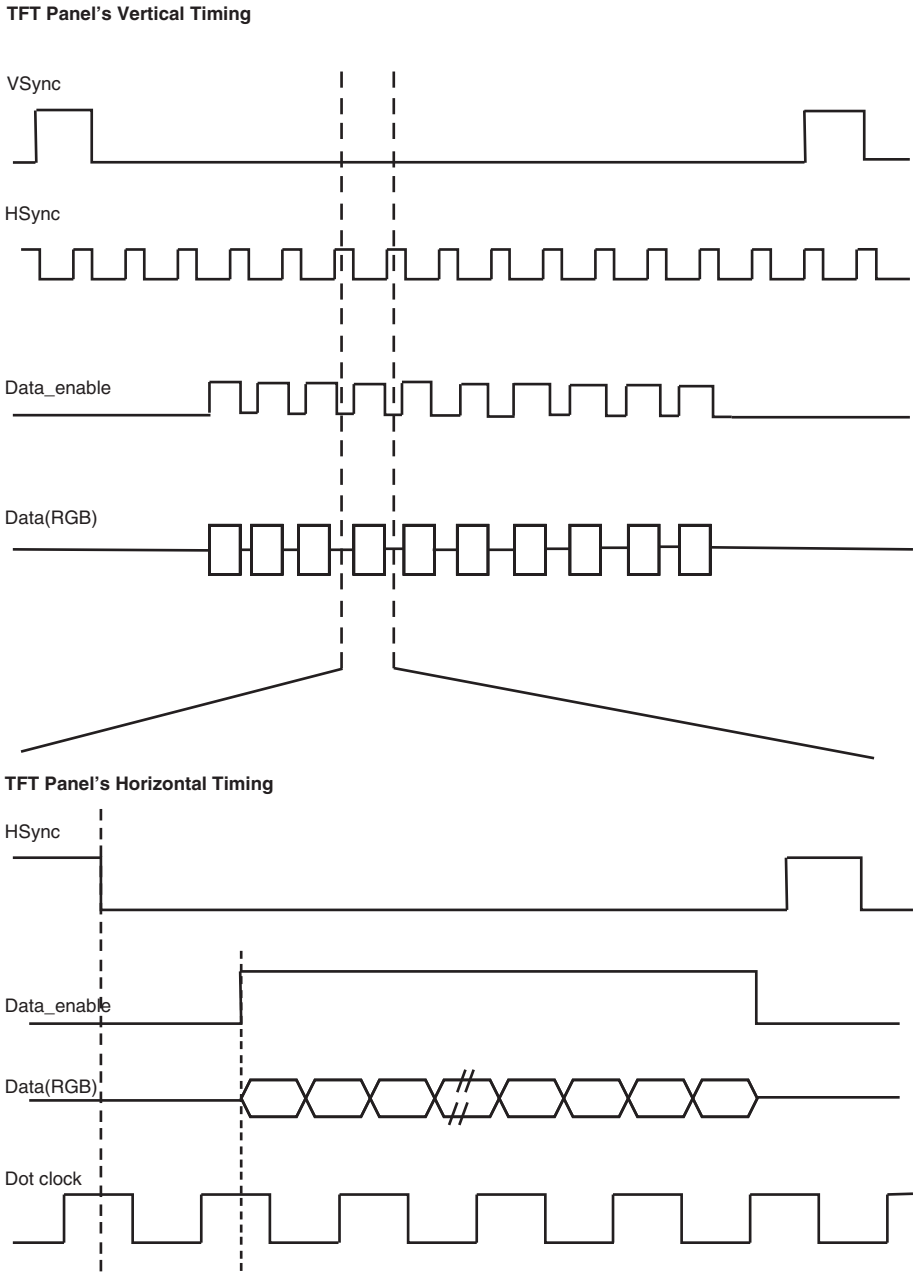
Signal	Width	Type	Description
i_clk_64mhz	1	Input	XGA clock signal from host device
i_rst_n	1	Input	Active low asynchronous reset signal. This signal is used to initialize the internal state machine to a known state.
i_proc_xga_h_sync	1	Input	XGA Horizontal sync signal from host device
i_proc_xga_v_sync	1	Input	XGA Vertical sync signal from host device
i_proc_xga_de	1	Input	XGA data enable signal from host device
i_proc_xga_rgb	1	Input	XGA RGB active pixel data from host device
o_lcd_dclk	1	Output	WVGA LCD clock. This signal is generated by WVGA LCD controller
o_lcd_wvga_h_sync	1	Output	WVGA horizontal timing control signal generated by WVGA LCD controller
o_lcd_wvga_v_sync	1	Output	WVGA vertical timing control signal generated by WVGA LCD controller
o_lcd_wvga_de	1	Output	WVGA pixel valid signal
o_lcd_wvga_red	5	Output	WVGA red pixel data
o_lcd_wvga_green	6	Output	WVGA green pixel data
o_lcd_wvga_blue	5	Output	WVGA blue pixel data

Initialization Conditions

An active low reset signal assertion is required to initialize the scalers and LCD controller state machines to a known operating state. No register configuration is necessary.

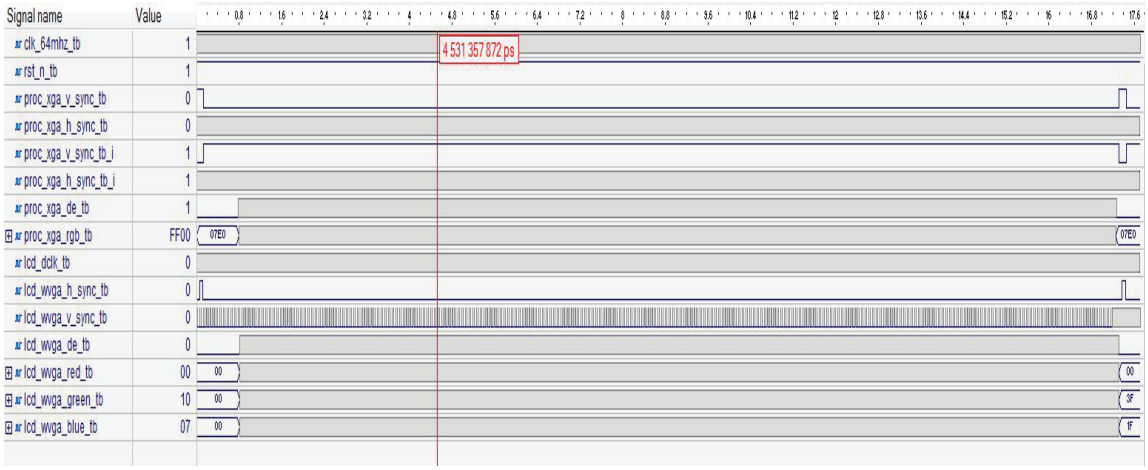
Timing Diagram

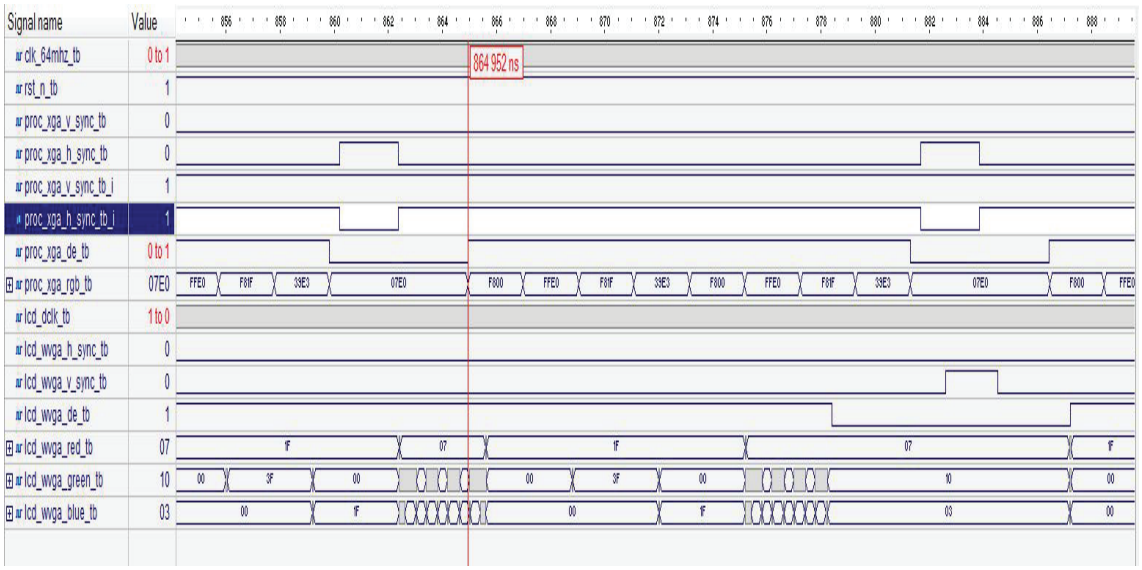
Figure 4. Timing Diagram



Simulation Waveforms

Figure 5. Simulation Waveforms





Operation Sequence

1. Test bench instantiates a wrapper which will in turn instantiate the design top level.
2. It will generate clock and reset signals to the design.
3. It has counters for counting pixels, lines, and pulse width period of HSync and VSync.
4. It generates XGA control signals like HSync, VSync and DE.
5. It feeds XGA data to the image scaler design during the active region of the XGA image.
6. The output red, green and blue data are fed to a .tiff image writer.
7. A module tiff.v will take the WVGA pixel data in and add a header to make it a .tiff image.
8. Current simulation directory provides the resulting WVGA image. This can be viewed by using any image viewing software.

Implementation

This design is implemented in VHDL. When using this design in a different device, density, speed or grade, performance and utilization may vary.

Table 2. Performance and Resource Utilization

Family	Language	Utilization (LUTs)	f_{MAX} (MHz)	I/Os
iCE40[1]	VHDL	3184	>50	40

1. Performance and utilization characteristics are generated using iCE40HX8K-CM225 with iCEcube2 design software.

References

- iCE40 Family Handbook

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
April 2013	01.0	Initial release.