

iCE65 as XGA to WVGA Image scaler using Lanczos algorithm

Overview

This design document illustrates the implementation of a video image down-scaler using an iCE65 L04 device. The main downscaling algorithm used is the Lanczos2 algorithm.

Features

- Downscales Video Images from XGA (1024x768) to WVGA (800x480)
- Supports 64MHz input pixel clock and 32MHz output pixel clock
- Supports RGB 565 input and output Video data format
- No External Frame Buffer Required
- Built-in WVGA timing and data Controller
- Internal Line Buffer using iCE65 RAM blocks
- IP-XACT version 1.2 compliant

Resource Utilization

Table 1: Resource Utilization

LUTs	Registers	Memory	I/Os	GBs
3187	672	16	0	0

Note: Resource Utilization is based on iCEcube 2010.12.14671 release.

System Block Diagram

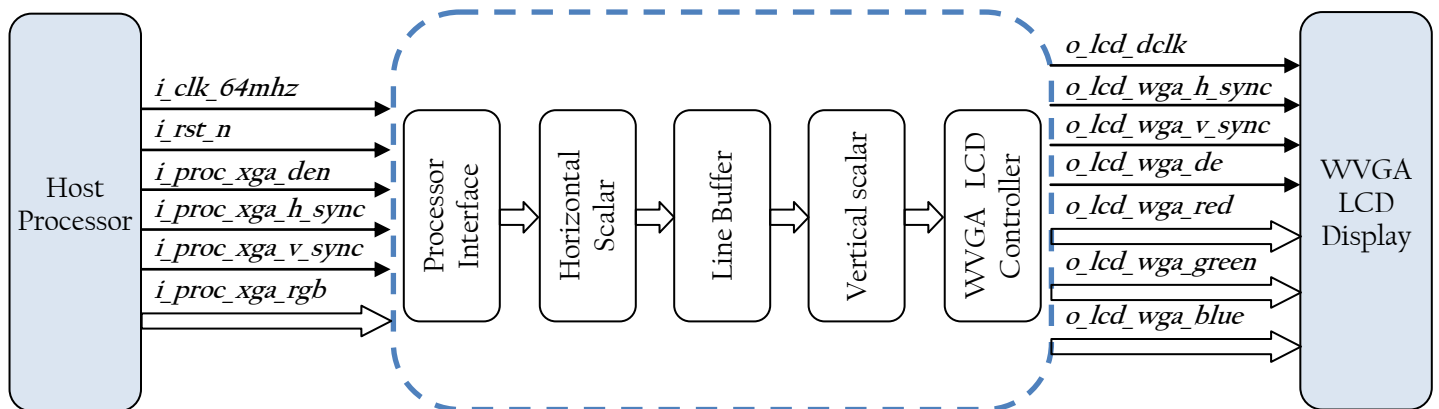


Figure 1: System Block Diagram

Functional Block Diagram

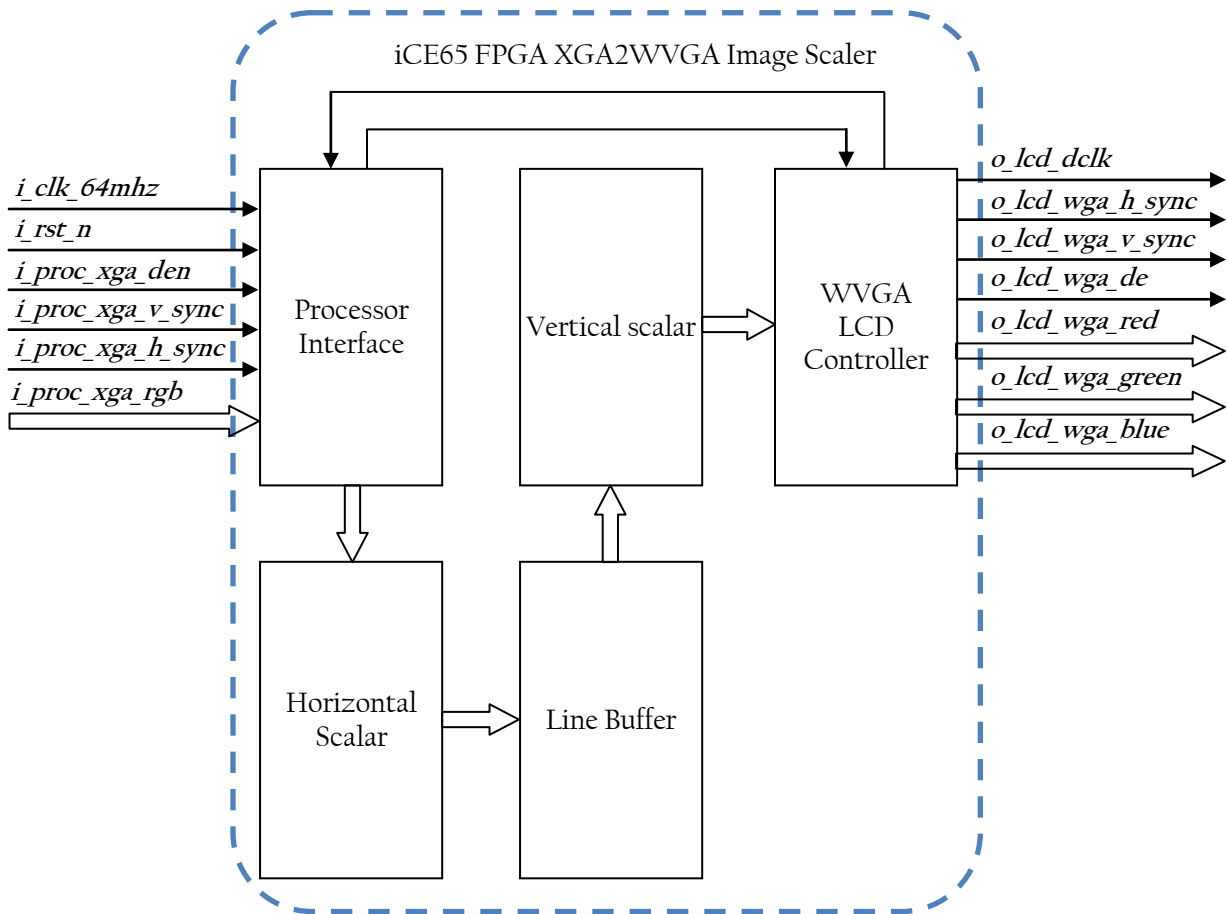


Figure 2: Functional Block Diagram

Design Interface

Table 2: Pin Description

Signal Name	Pin Type	Signal Description
i_clk_64mhz	Input	XGA clock signal, from host device
i_rst_n	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	Input	XGA Horizontal sync signal from host device
i_proc_xga_v_sync	Input	XGA Vertical sync signal from host device
i_proc_xga_de	Input	XGA data enable signal from host device
i_proc_xga_rgb	Input	XGA RGB active pixel data from host device
o_lcd_dclk	Output	WVGA LCD clock. This signal is generated by WVGA LCD controller.
o_lcd_wvga_h_sync	Output	WVGA horizontal timing control signal generated by WVGA LCD controller.
o_lcd_wvga_v_sync	Output	WVGA vertical timing control signal generated by WVGA LCD controller.
o_lcd_wvga_de	Output	WVGA pixel valid signal
o_lcd_wvga_red[4:0]	Output	WVGA red pixel data.
o_lcd_wvga_green[5:0]	Output	WVGA green pixel data.
o_lcd_wvga_blue[4:0]	Output	WVGA blue pixel data.

Configurable Parameters

None

Register Map

This design does not have any user accessible registers or memory..

Design Details

Processor Interface

A parallel processor interface is implemented between the processor and the Image scaler. This module bridges host device and image scaler module. The host device provides the XGA video/image to the processor interface in three data channels, mainly 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 downsampling from 768-line to 480-line.

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

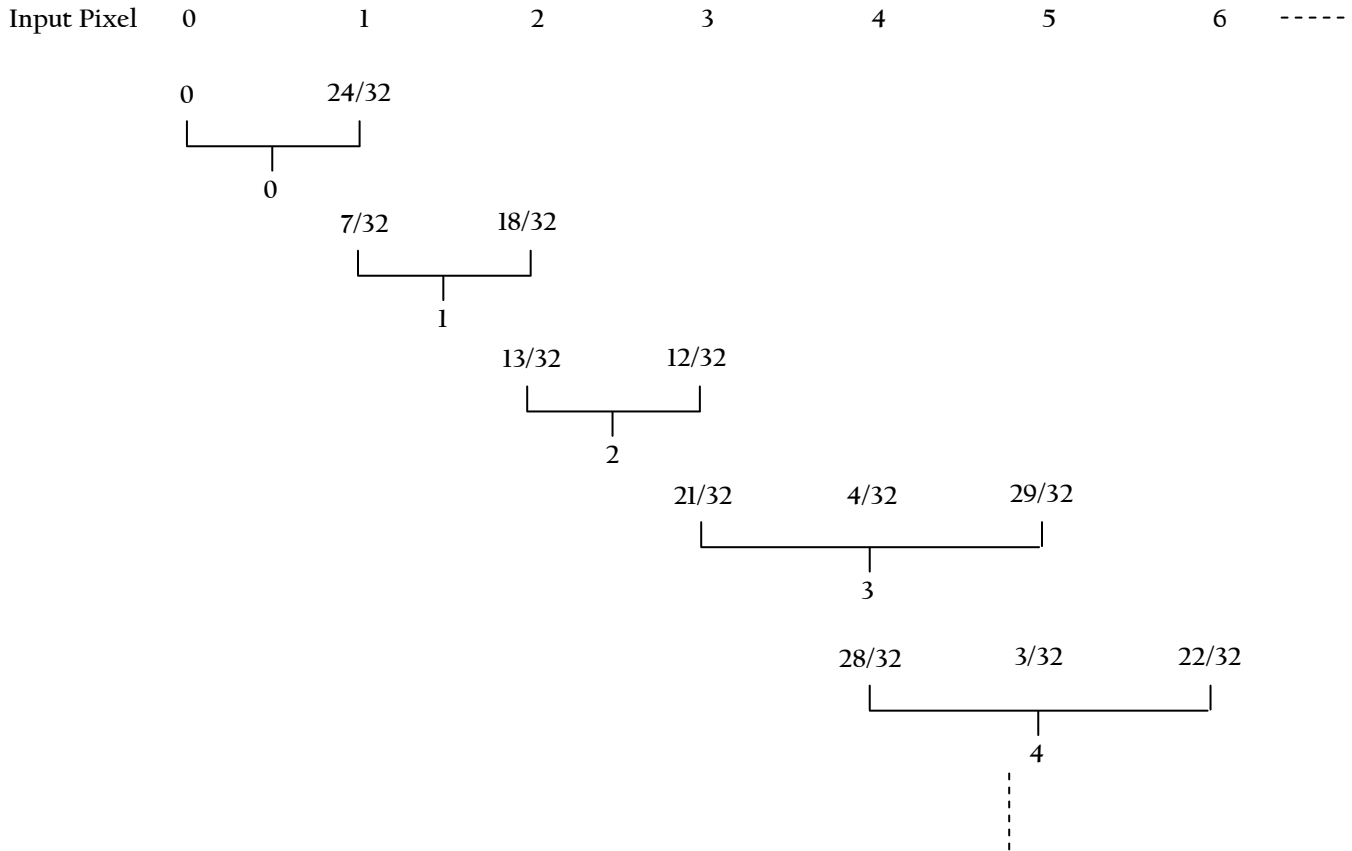


Figure 3: Lanczos Coefficients for the filter

The Lanczos2 mathematical formula is as shown below.

$$\text{Lanczos2} = \{\sin(x*\pi)*\sin(x*\pi/2)\}/(x*\pi*x*\pi/2)$$

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 used.

```
ScaleRectAvg(PIXEL *Target, PIXEL *Source, int SrcWidth, int SrcHeight, int TgtWidth, int TgtHeight, float threshold)
```

```
{
    int 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 the Figure 4

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

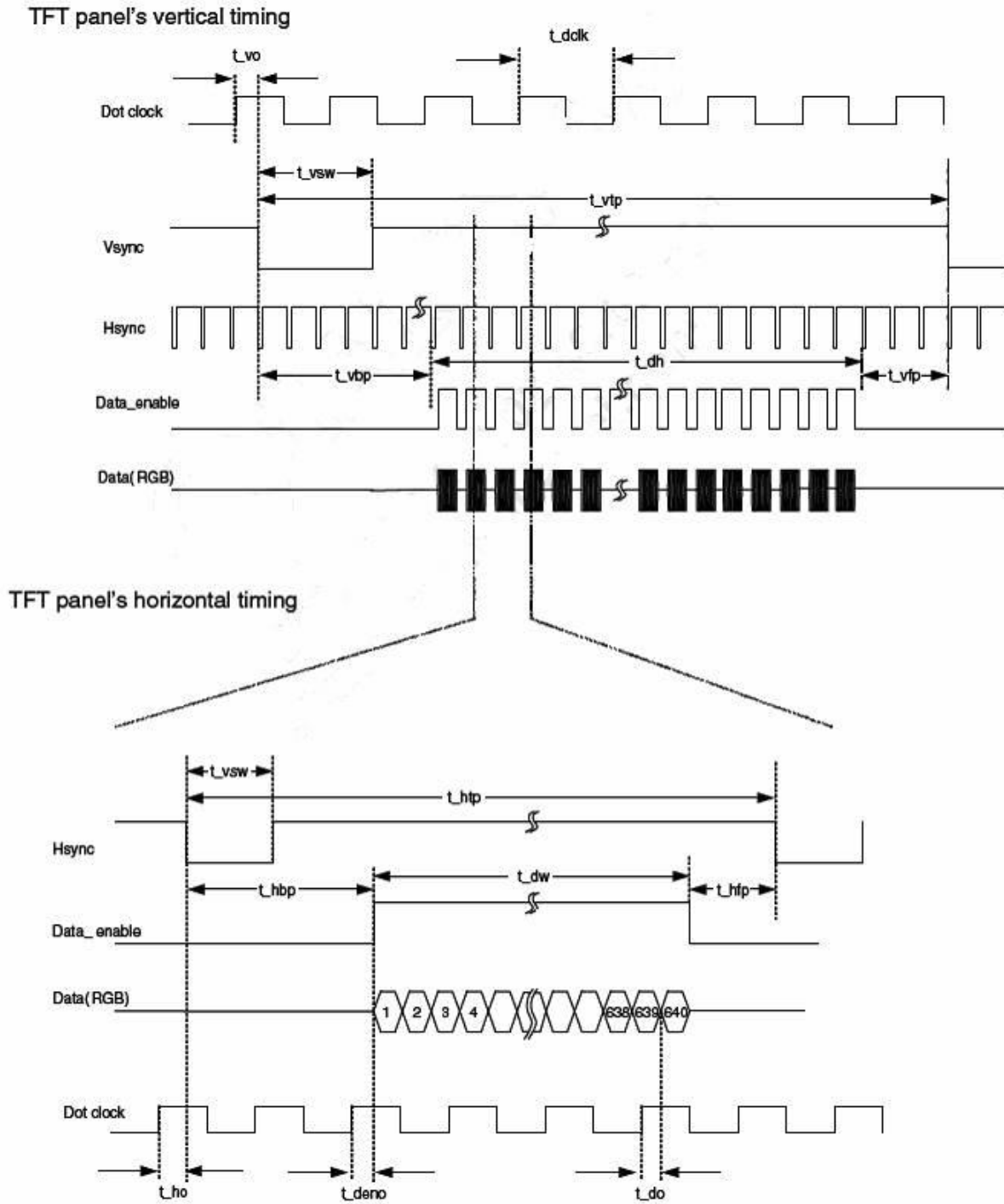


Figure 4: WVGA LCD Timing diagram

System Designer Flow

XGA to WVGA Image Scaler using Lanczos algorithm is compatible with System Designer/IP-XACT 1.2. The System Designer flow is as follows,

1. Launch the System Designer from Synplify Pro using menu 'Import -> Launch System Designer'.
2. Create a new project(open an existing old project, as necessary) and import the IP-XACT XML file
3. Drag and place the component from the 'Library' pane to the 'Design' pane
4. Click on the “Generate Files” button, which generates the necessary files required for synthesis and simulation.
5. Go to Synplify Pro and click on the “Run” button to synthesize the System Designer generated files. Synplify Pro generates all the necessary files for P&R in iCECube.

References

The following references were used in the creation of this design:

- SiliconBlue Technologies, Inc. “[iCE65 Ultra Low-Power mobileFPGA Family](#)” datasheet (26-MAY-2010).
- Image Scaling: http://en.wikipedia.org/wiki/Image_scaling

Revision History

Version	Date	Description
1.0	09-SEP-2010	Initial Draft Document
1.1	08-DEC-2010	IP-XACT Format Update

Disclaimer

Copyright © 2007–2009 by SiliconBlue Technologies LTD. All rights reserved. SiliconBlue is a registered trademark of SiliconBlue Technologies LTD in the United States. Specific device designations, and all other words and logos that are identified as trademarks are, unless noted otherwise, the trademarks of SiliconBlue Technologies LTD. All other product or service names are the property of their respective holders. SiliconBlue products are protected under numerous United States and foreign patents and pending applications, maskwork rights, and copyrights. SiliconBlue warrants performance of its semiconductor products to current specifications in accordance with SiliconBlue's standard warranty, but reserves the right to make changes to any products and services at any time without notice. SiliconBlue assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by SiliconBlue Technologies LTD. SiliconBlue customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



SiliconBlue Technologies Corporation

3255 Scott Blvd.,
Building 7, Suite 101
Santa Clara, CA 95054

Tel: 408-727-6101
Fax: 408-727-6085

www.SiliconBlueTech.com