**Technion**

*Electrical Engineering Department*

High Speed Digital System Lab

Symbol Generator

Project

**Version**: 1

**Last update**:

**Written By**: Yoav Shvartz and Olga Liberman

Table of content

[Table of Changes 5](#_Toc314329213)

[0 Terminology 6](#_Toc314329214)

[1 Intro 6](#_Toc314329215)

[1.1 Abstract 6](#_Toc314329216)

[1.2 Project Goal 7](#_Toc314329217)

[1.3 Project Requirements 7](#_Toc314329218)

[2 General Description 8](#_Toc314329219)

[2.1 Data Flow 8](#_Toc314329220)

[2.2 Project Components Scheme 8](#_Toc314329221)

[2.2.1 Implementation in this project 9](#_Toc314329222)

[2.3 Abbreviations 9](#_Toc314329223)

[3 Implementation 10](#_Toc314329224)

[3.1 Top Architecture 10](#_Toc314329225)

[3.2 SDRAM Controller 11](#_Toc314329226)

[3.2.1 General Description 11](#_Toc314329227)

[3.2.2 Supported commands 11](#_Toc314329228)

[3.2.3 Resources 11](#_Toc314329229)

[3.2.4 Burst Write – Wave 11](#_Toc314329230)

[3.2.5 Burst Read – Wave 11](#_Toc314329231)

[3.2.6 SDRAM Documentation 11](#_Toc314329232)

[3.3 UART Transmitter and Receiver 12](#_Toc314329233)

[3.3.1 UART Characteristics 12](#_Toc314329234)

[3.3.2 Resources 12](#_Toc314329235)

[3.3.3 RX Operation – Wave 12](#_Toc314329236)

[3.3.4 TX Operation - Wave 12](#_Toc314329237)

[3.3.5 UART Documentation 12](#_Toc314329238)

[3.4 Message Pack Description 13](#_Toc314329239)

[3.4.1 Message Pack Decoder 13](#_Toc314329240)

[3.4.2 Message Pack Encoder 13](#_Toc314329241)

[3.4.3 Checksum 13](#_Toc314329242)

[3.4.4 Resources 13](#_Toc314329243)

[3.4.5 Message Pack Decoder - Wave 13](#_Toc314329244)

[3.4.6 Message Pack and Checksum Documentation 13](#_Toc314329245)

[3.5 VESA Generator Controller 14](#_Toc314329246)

[3.5.1 DVI Operation 14](#_Toc314329247)

[3.5.2 VESA Generator Operation 15](#_Toc314329248)

[3.5.3 Frame 15](#_Toc314329249)

[3.5.4 Defenition of Terms 16](#_Toc314329250)

[3.5.5 Resources 16](#_Toc314329251)

[3.5.6 Complete Frame (Image) - Wave 16](#_Toc314329252)

[3.5.7 VESA Generator Documentation 16](#_Toc314329253)

[3.6 Global Nets 17](#_Toc314329254)

[3.6.1 Resources 17](#_Toc314329255)

[3.6.2 Global Nets - Wave 17](#_Toc314329256)

[3.6.3 Clock Block 17](#_Toc314329257)

[3.6.4 Reset Block 17](#_Toc314329258)

[3.6.5 Clock and Reset Documentation 17](#_Toc314329259)

[3.7 Top Architecture Details 18](#_Toc314329260)

[3.7.1 Memory Block 18](#_Toc314329261)

[3.7.2 Image Block 18](#_Toc314329262)

[3.7.3 TX Block 19](#_Toc314329263)

[3.7.4 RX Block 20](#_Toc314329264)

[3.7.5 Interconnects 20](#_Toc314329265)

[3.7.6 Wishbone Cycles 21](#_Toc314329266)

[3.8 Project's Directory Structure 21](#_Toc314329267)

[4 Testing 23](#_Toc314329268)

[4.1 Simulation 23](#_Toc314329269)

[4.2 Lab Examination 23](#_Toc314329270)

[4.2.1 Matlab's GUI 23](#_Toc314329271)

[4.2.2 Algorithm Example from Matlab 24](#_Toc314329272)

[5 Summary 25](#_Toc314329273)

[5.1 Project Usage 25](#_Toc314329274)

[5.2 Project Status 25](#_Toc314329275)

[5.3 Problems during the project 25](#_Toc314329276)

[5.4 Project Conclusions 26](#_Toc314329277)

[5.5 Coding Conclusions 26](#_Toc314329278)

Table of Tables

Table 1 – Table of changes 4

Tables of Figures

# 

# Table of Changes

|  |  |  |
| --- | --- | --- |
| Version | Date of Change | Description |
| 1.0 |  | Creation of documentation |

Table 1 – Table of Changes

# 

# Intro

## 1.1 Abstract

Generating symbols on display screens is an essential operation these days. It is commonly used in varies applications, such as mobile phones, televisions and computer screens. The need to minimize frame transmission time and resources becomes more and more crucial with the progress of technology.

## Project Goal

Implement a software symbol generator (using Matlab) with a HW extractor (FPGA) using VHDL. There is a set of known symbols (such as letters, digits, icons, etc), all having in common the same dimensions, and are generated on the screen.

The user chooses the appearance of the frame on the screen: where to place symbols, from where to delete them or both. For this purpose, he uses a graphic interface implemented in Matlab.

The purpose is to save time, resources and bandwidth. Therefore, the program transfers the HW only the change of the wanted frame from the current one, using a command for the HW, instead of sending the whole frame.

## Project Requirements

1. Transmit 640x480 BMP grayscale picture, using Matlab, to the DE2 board.
2. Size of the symbols: 32x32 pixels.
3. The frame is divided into 20x15 blocks (row x column). Each block is size 32x32 pixels. The blocks are aligned and do not run over each other.
4. A black screen will be displayed, until the first frame will be generated and transmitted.
5. Data will be sent from host to the DE2 board using UART 115,200 bits/sec.
6. UART protocol:
   1. Line not active = '1'
   2. Parity bit will not be used, since checksum is being calculated
   3. 8 bits will be wrapped by *start bit*, represented by '0', and *stop bit*, represented by '1'.
7. Message Pack Structure:
   1. SOF – Start of Frame
   2. Type – Message Type (Data, Debug…)
   3. Address – registers address
   4. Data (payload) Length – number of bytes – 1 in data block
   5. Data (payload)
   6. CRC (Checksum will be used instead CRC)
   7. EOF – End of Frame
8. Main types ('b' in the Message Pack Structure):
   1. Compressed Data
   2. Last Packet Summary (End of packet) – contains the number of bytes, which had been transmitted for this picture (Size of compressed image in bytes)
   3. Reading / Writing registers
   4. Reading and writing from SDRAM for debug
   5. More types will be added later, in case such are required
9. SDRAM – Reading / Writing in up to 256 words burst.
10. Required size in SDRAM: (num. of symbols) x 32 x 32 bytes
11. In case an error has occurred during transmission, next picture will be written to the same bank in memory, and the last fine picture will keep being transmitted to the screen. The corrupted picture will be discarded.
12. SDRAM Arbiter priority: Writing, then reading, in order to prevent data loss.
13. Clocks in systems:
    1. 40 MHz for VESA
    2. 133MHz for SDRAM and the Symbol Generator block
14. Checksum will be calculated on the TYPE, ADDRESS, LENGTH and DATA blocks, in that order.
15. Two FIFOs shall be implemented (FIFO\_A and FIFO\_B), each FIFO size: 640 x 8 bits (row x column). One FIFO will load a video row from the SDRAM, while the other FIFO will transmit the previous video row to the VESA.

# General Description

## Data Flow

The user chooses the wanted display using the **GUI,** and the display is encoded into a wrapped message. Then, the **Host** transmits the wrapped message, composed of the changes in the current display relatively to the previous one, to the **RX Path**. Message is decoded (from the SOF, CRC, EOF), transmitted to the **Symbol Generator** and decoded (which symbol to add to display, and in what coordinate on the screen). The symbols are read from the **SDRAM** and stored in FIFOs in the **Symbol Generator**. Then, data from the FIFOs is transmitted to the **VESA Generator** and to the display.

Status and debug signals can be sent through the **TX Path** to the **Host**.

## Project Components Scheme

Figure 3 - Detailed Scheme

### Implementation in this project

The user chooses the desired display, using the GUI (the symbols and their location on the screen). This information is decoded in *Matlab* environment and transmitted, as a wrapped message, to the Altera DE2 board, through **UART** protocol.

The FPGA decodes the wrapped message (**Message Pack Decoder**), validates **Checksum** and correctness of message length, and transfers the message to the **Symbol Generator**. Then, the message is decoded: the desired symbols and their location on the screen.

## Abbreviations

Video row – the pixels of a specific row in a video frame

Symbol row – the pixels of a specific row in a symbol

SDRAM – Synchronous Dynamic Random Access Memory

RAM – Random Access Memory

TX – Transmission

RX – Receive

FIFO – First in First out

PLL – Phased Locked Loop

TB – Test bench

SOF – Start of frame

EOF – End of frame

CRC - Cyclic redundancy check

MP – Message Pack

TB – Test Bench

UART – Universal Asynchronous Receiver Transmitter

VESA - Video Electronics Standards Association

VGA - Video Graphics Array

DVI - Digital Visual Interface

IP – Intellectual Property

LSB – Least Significant Bit

MSB – Most Significant Bit

MHz – Mega Hz

# Implementation

## Top Architecture

## SDRAM Controller

### General Description

### Supported commands

### Resources

### Burst Write – Wave

### Burst Read – Wave

### SDRAM Documentation

## UART Transmitter and Receiver

### UART Characteristics

### Resources

#### RX

#### TX

### RX Operation – Wave

### TX Operation - Wave

### UART Documentation

## Message Pack Description

### Message Pack Decoder

### Message Pack Encoder

### Checksum

### Resources

### Message Pack Decoder - Wave

### Message Pack and Checksum Documentation

## VESA Generator Controller

### DVI Operation

A DVI video signal contains 6 signals:

* Clock – Pixel Clock
* HSync – New Line signal
* VSync – New Frame (image) signal
* Blank – Blanking signal to the screen (ignore the RGB inputs)
* RED – Analog signal, used to control the color
* GREEN – Analog signal, used to control the color
* BLUE – Analog signal, used to control the color

Colors are produced by changing the analog levels of the three RGB signals.

The screen refresh process begins in the top left corner and paints 1 pixel at a time from left to right. At the end of the first row, the row increments and the column address is reset to the first column. Once the entire screen has been painted, the refresh process begins again.

The video signal must redraw the entire screen 60 times per second to provide motion in the image and to reduce flicker. This period is called the refresh rate.

In 800X600 pixel mode, with a 60 Hz refresh rate, this is approximately 25 ns per pixel, which requires a 40 MHz.

The **VSync** signal commands the monitor to start displaying a new frame, and the monitor starts in the upper left corner with pixel (0, 0).

The **HSync** signal commands the monitor to refresh another row of 800 pixels.

After 600 rows of pixels are refreshed with 600 HSync signals, a VSync signal resets the monitor to the upper left corner and the process continues.

During the time when pixel data is not being displayed (**Blanking**) and the beam is returning to the left column to start another horizontal scan, the RGB signals should all be set to black color (all zero).

### VESA Generator Operation

VESA Generator is a VESA Non-Interlaced Controller, which transmit to the DVI screen the following data and signals:

* (R, G, B) Pixels
* Horizontal and Vertical Sync
* Blanking

**IMPORTANT:** The DVI should be supplied by pixel-clock. The VESA generator does notsupplies the pixel-clock. The user shall ensure that the DVI has the same clock input as the VESA generator.

The (R, G, B) Pixels should be supplied to the VESA generator by an outside component.

Back Porch, Front Proch, Left - Right - Upper - Lower borders, Sync Time and Active pixels / lines should be defined, according to the VESA standard, using the generic parameters.

The R, G and B port sizes may be changed individually, using a generic parameter.

The R, G and B values will be transmitted, one by one, in the each pixel-clock. See clock explanation in the DVI operation (40MHz in the example).

### Frame

A generic-color frame can wrap the transmitted image, using the left, right, lower and upper input registers.

Figure 11 – Frame of Image

### Defenition of Terms

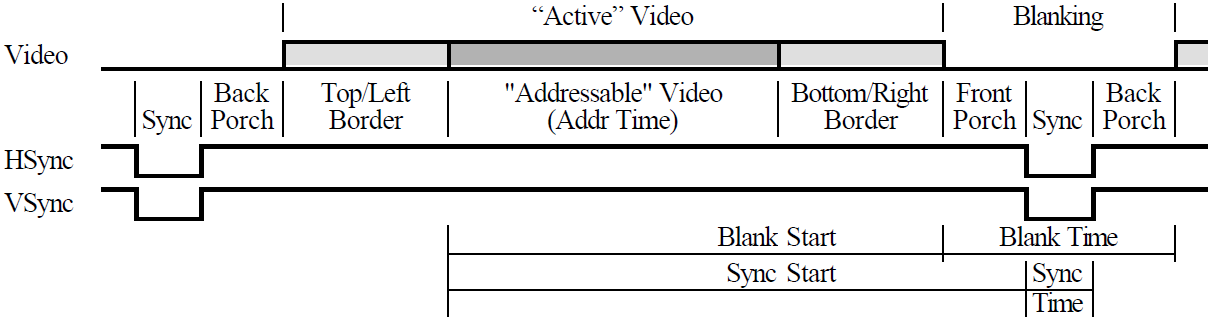


Figure 12 – Definition of Terms

### Resources

### Complete Frame (Image) - Wave

### VESA Generator Documentation

## Global Nets

### Resources

### Global Nets - Wave

### Clock Block

### Reset Block

#### Reset Block - Wave

#### Reset Filter

#### Reset Filter Working Method

#### Sync Reset Generator

#### Sync Reset Generator Working Method

### Clock and Reset Documentation

## Top Architecture Details

### Memory Block

#### Registers in Memory Block

#### Write Target

#### RAMs in the Memory Block

#### SDRAM Bank Select

#### Write and Read Counters

#### Write to SDRAM

#### Read from SDRAM

#### SDRAM Arbiter

### Image Block

#### Registers in the Image Block

#### Receive Data

#### Synthetic Picture Generator

### TX Block

#### Registers in the TX Block

#### Operation

#### Read from SDRAM

#### Read from Register Block

#### UART Transmission

### RX Block

#### Write to Registers

#### Internal Registers in RX Block

### Interconnects

#### INTERCON (Z)

#### INTERCON (Y)

#### Registers in the INTERCON (Y)

#### INTERCON (X)

### Wishbone Cycles

#### General Rules

#### Wishbone BURST (Block) Read Transaction

#### Burst Write Transaction – Start of Cycle

#### Burst Write Transaction – End of Cycle

## Project's Directory Structure

Figure 25– Directory Structure

# Testing

## Simulation

## Lab Examination

### Matlab's GUI

### Algorithm Example from Matlab

# Summary

## Project Usage

## Project Status

## Problems during the project

## Project Conclusions

## Coding Conclusions