Παναγιώτης Κρεμμύδας 1435

10/27/2016

# Lab exercize 3:

# VGA controller

###### CE430: Digital Circuits Lab

###### Electrical& Computer Engineering

###### University of Thessaly

# Contents

0: Introduction

1: VRAM implementation: The display data to be accessed by the controller are initialized into SPARTANS 3E onboard BRAM.

2: HSYNC signal & timing: Controlling the Data flow to each horizontal pixel rotationally in the given framerate

3: VSYNC signal & timing: Controlling the Data flow to each horizontal pixel rotationally in the given framerate

4: Timer driven rotating message: Displaying characters of a preconfigured message rotationally on the LED units using a specified time interval

5: Conclusion: Challenges & Solutions in the designing, testing and implementing of the above

# Introduction:

Implementation of a Video Graphics Array Controller/Driver.

The goal is to successfully drive a typical monitor and display an image in it.

For the purpose of continues representation through the VGA port, part of the internal RAM of the FPGA unit used will be assigned as Video RAM (VRAM) of the driver. The suggested sample image for the testing of the aforementioned driver is the typical red, blue, green,black horizontal stripes separated repeatedly by white stripes. The black stripes part is also repeatedly vertically overlapped by a red,green,blue vertical stripe.

Video data typically comes from a video refresh memory with one or more bytes assigned

to each pixel location. The Spartan-3E Starter

Kit board uses three bits per pixel, producing

one of the eight possible colors shown in

Ta b l e 6 - 1

. The controller indexes into the video

data buffer as the beams move across the display. The controller then retrieves and applies

video data to the display at precisely the time the electron beam is moving across a given

pixel

As shown in Figure 6-2, the VGA controller generates the horizontal sync (HS) and vertical

sync (VS) timings signals and coordinates the delivery of video data on each pixel clock.

The pixel clock defines the time available to display one pixel of information. The VS signal

defines the

refresh

frequency of the display, or the frequency at which all information on the

display is redrawn. The minimum refresh frequency is a function of the display’s phosphor

and electron beam intensity, with practical refresh frequencies in the 60 Hz to 120 Hz

range.

Generally, a counter clocked by the pixel clock controls the horizontal timing. Decoded

counter values generate the HS signal. This

counter tracks the current pixel display

location on a given row.

A separate counter tracks the vertical timing

. The vertical-sync coun

ter increments with

each HS pulse and decoded values generate the VS signal. This counter tracks the current

display row.

These two continuously running counters form an address that is used by a video display buffer (a module that takes that address and loads memory from the block and signal the data through the specified pins). For example, the on-board DDR SDRAM provides an ideal display buffer.

No time relationship is specified between the

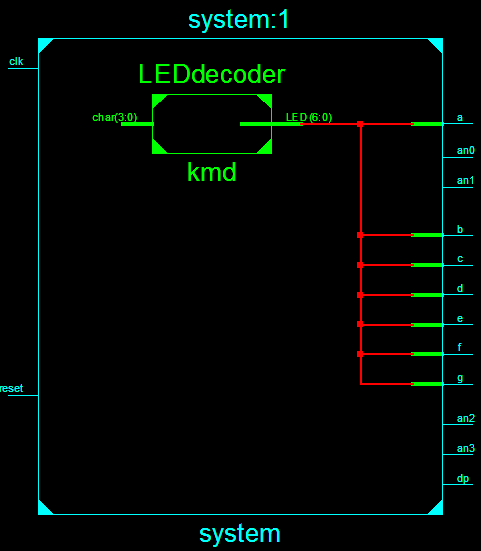
onset of the HS pulse and the onset of the VS

pulse. Consequently, the counters can be arranged to easily form video RAM addresses, or

to minimize decoding logic for sync pulse generation.

# Part 1 - Seven segment decoder:

## Implementation



LEDdecoder: The decoder itself is the above simple module taking 4 bits as the given character and outputting 0 to the segments of the LED unit that need to be activated and 1 to the ones that do not

<https://github.com/kmd178/FPGACODE_LAB1_7SegmentDisplay/commit/7d080337de7385e7a3e88d468b9c4b0cfa63854a>

## Verification

Simple 4 bit inputs covering the range of the specified alphanumeric characters where used in the testbench in order to crosscheck the resulting waveform with the necessary partial activation of the segments to light up the specified character.

Errors: 2 characters have been mistyped and later corrected.

## Experiment/Resulting implementation

FPGA board testing was not necessary for this part of the assignment

# Part 2 - Four digit Led driver:

## Implementation

DCM module: A slower clock was created using a preconfigured DCM module that the FPGA board itself distinctly supports in its hardware. This is necessary to easilly manipulate the slow LED anodes. The anodes need a lot of time to charge and discharge their high capacitances . They also only take input data from the same 8bits to activate the corresponding anodes, so they need to take turns doing so.

ledDataFeed module: Controls the data flow to the seven segments of all 4 Led units and activation of the corresponding unit in the given timeframe.

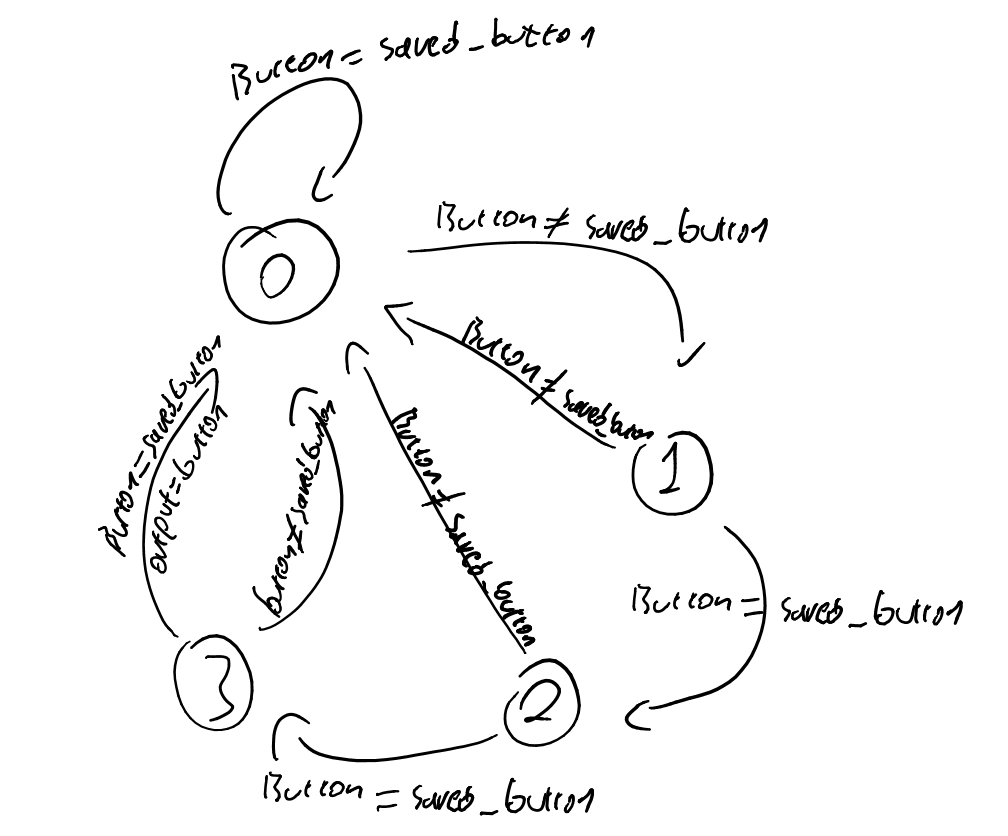
Anode activation FSM: Every 4 states in the 16 state FSM a LED unit is activated and the rest are deactivated. The flickering between activation and deactivation is not visible to the human eye because of the high capacitance and high frequency of the sequence. The anode 7segment registers need to be preassigned within 1 DCM cycle delay of the LED anode activation due to capacitancies delay phenomenons.

SystemCounter: The DCM drives a 4 bit counter that controls the 16 states which are necessary for the activation and data feed of a the assigned LED unit and the deactivation of the others.

Anti\_bounce\_reset& Anti\_bounce modules: Modules that stabilize the reset button itself or any other button pressed to avoid undesired effects on the rest of the system due to mechanical bouncing of the physical buttons. The bounce is usually lasting around 20ms but for the following verilog implementation the tolerance is increased to 80miliseconds. A different module is utilized for Buttons than the Reset button itself in order for the system to regain partial functionality in the a case of a shortcircuited or stuck button. The anti\_bounce module will effectively nulify the malfunctioning button’s effect when the reset button is pressed.

Counting 20ms time using AND logic gate: When clk\_hits 1048576 cycles=(20b'1111111111111111111111) clock periods CHECK will give me a posedge through the logic AND of all its bits. normally for a 50mhz clock (period=2e-8) and for a 20ms stabilizing time a 20bit counter that delays 2^20cycles is enough . Depending on the clock the FPGA system has different amounts of bits for the counter are used.

Stabilization FSM: The buttons are stabilized using the following FSM. The progress of the FSM is triggered by the AND of the bits of the 20bit counter, which means once every 20ms



<https://github.com/kmd178/FPGACODE_LAB1_7SegmentDisplay/commit/c54e1fc688d8a0d2d44b6b1cea3ceca740e98141>

## Verification

<https://github.com/kmd178/FPGACODE_LAB1_7SegmentDisplay/blob/c54e1fc688d8a0d2d44b6b1cea3ceca740e98141/tb.v>

The input button and reset are flactuating in short time intervals before set on or off for mechanical button simulation . For easy waveform crosschecking a short 6bit counter for short intervals is used to control faster if the button pressed and if the stabilized signal should pass as an output. Output waveforms on segment and anode registers are performing as expected.

## Experiment/Resulting implementation

First trial: Although the waveforms were performing as expected the 16 states which are necessary for the activation and data feed of a the assigned LED unit and the deactivation of the others anodes were formed as latches because not all conditions of the case statement were initialized. The output bitfile ignoring the above warning in its generation is producing dimly lit eights on the FPGA board. The above is corrected by assigning value to all anodes in every state and not only in the state that they are supposed to change. (Common error that happens using software programming way of thought )

Second trial: The LED driver is giving the expected 4 stable characters output on the second attempt as intented.

# Part 3 - Button triggered message rotation:

## Implementation

ledDataFeed module: The 16 state FSM now also controls in a second always statement when data is fed from memory to the segments of the LED unit to be activated. The stage which this is happening has to be placed DCM cycle before the LED anode activation due to capacitancies delay phenomenons on the 7segment registers

Memory Counter: Method of triggering is button press. The signal from the button is stabilized the the anti-bounce module. With every button press the assigned anodes are progressing through the memory to the next character to be displayed in their own sequence

Message: Memory is initiallized to display the message: 1435abCd

<https://github.com/kmd178/Digital_Systems_lab1_7SegmentDisplay/commit/1e4563356a4db4587280bf4259d301fb0052370e>

## Verification

Testbench is signalling the input button for rotation of the message to appear in the waveforms. Reset function is tested as well. The input button and reset are flactuating in short time intervals before set on or off for mechanical button simulation .

For easy waveform crosschecking a short 6bit counter for short intervals is used to control faster if the button pressed and if the stabilized signal should pass as an output.

Output waveforms on segment and anode registers are performing as expected.

## Experiment/Resulting implementation

First trial: One of the Led units could remain lit during a reset button press. This was the effect of the anode FSM not being connected to a reset signal that drives the anodes to deactivation.

Second trial: The FPGA board performs as expected. Using the reset button an effect might be noticed roughly 1/10 times just for some miliseconds. One of the LED units is distinctively emmiting more light. Apart from it all the LED units are shutting off normally when the reset is pressed in every attempt and the rotated message starts from the beggining. The effect is probably due to some power electronic phenomenon and does not effect the usuabillity of the system.

# Part 4 - Timer triggered message rotation:

## Implementation

ledDataFeed module Method of triggering is changed to timer. Every second the message is rotating by itself. A new 22bit counter is created for that reason.

check\_signal\_every\_second : Counting time using AND. When clk\_hits 4194304Cycles=(22b'1111111111111111111111111) clock periods CHECK will give me a posedge through the logic AND of all its bits. For 50mhz clock divided by 16 on the DCM(period=(2e-8)/16) and for a 1340ms distance between posedges we need 2^22cycles.

https://github.com/kmd178/Digital\_Systems\_lab1\_7SegmentDisplay/commit/1e4563356a4db4587280bf4259d301fb0052370e

## Verification

For easy waveform crosschecking a short 6bit counter for short intervals is used to control faster if the button pressed and if the stabilized signal should pass as an output.

Output waveforms on segment and anode registers are performing as expected.

## Experiment/Resulting implementation

First trial: The FPGA board performs as expected. Using the reset button an effect might be noticed roughly 1/10 times just for some miliseconds. One of the LED units is distinctively emmiting more light. Apart from it all the LED units are shutting off normally when the reset is pressed in every attempt and the rotated message starts from the beggining. The effect is probably due to some power electronic phenomenon and does not effect the usuabillity of the system.

# Conclusions:

Bad practices:

Using same registers in different always blocks,

Using signals without posedge creating latches,

Missing condition in case statement resulting in latch,

Assignment of memory outside an always block ,

Including in the same always blocks different groups of registers,

Using (\*) in always blocks.