COE3DQ5-Project Report

Group 32

Areeb Khawaja: 400015882, Hassan Niazi: 400026586

Khawaa1@mcmaster.ca, niazih@mcmaster.ca

November 27th, 2017

Introduction

The objective of this project is to gain experience in digital system design by creating a hardware implementation of an image decompressor. The hardware image compression specification we are using for this project is the .mic11. The hardware implementation was done by taking a 320 x 240 pixel image and running it through the Altera DE2 board through the UART interface, from a personal computer and then storing it in the SRAM. The compressed data is read by the image decoding circuitry devised by us with the specifications in the project manual, and then store it back into the SRAM. From there, the VGA controller will read and display the image.

Design Structure

The completion of Milestone 1 required several modules. The pre-built modules we acquired from Lab 5 include: the SRAM controller, UART SRAM Interface, and VGA SRAM Interface. The SRAM controller was used because we needed to access data, and addresses in a specific order and hold it for a specific time. The SRAM controller from lab 5 was used for generating the properly timed signals and making the SRAM synchronous. In our project the data from the SRAM was read after 2 clock cycles. The UART SRAM Interface was needed because we wanted to read the serial data from the SRAM asynchronously. The VGA is a multiplexed display that would display our 320 x 240 pixel image. The module from lab 5 was compatible with our purposes so we decided to use it. One of the modules that we built was the interpolation component of Milestone 1. The interpolation component takes the downsampled YUV data from the SRAM read data, and using the 2 multiplexors along with one 8-bit Y register, six 8-bit U-registers, and six 8-bit V registers we completed the process of upsampling the YUV data. This upsampled YUV data is then fed into the colour space conversion component of our milestone. The colour space conversion component also uses 2 multiplexors to efficiently perform matrix calculation to convert the upsampled YUV data into upsampled RGB data, which is then written back into SRAM write data.

3. Implementation Details

Upsampling

The y values taken from the SRAM are passed to a 16-bit register to be later used for colour space conversion.

6-element array of 8-bit wide registers were used to store the downsampled values of u and v taken from the SRAM. The values in the register for u are passed to a single The are passed to a multiply-accumulate unit, and after a clock cycle save to a 32-bit register called ans1. It takes 6 clock cycles for the interpolation of the odd u prime value to fully finish. Which is then shifted 8 bits, the upsampled value is then saved to the first 8 bits of a 16-bit register called UPrime.

During the same 6 clock cycles, the odd v prime value is also calculated by a different multiplication circuit and saved to a 32-bit register called ans2. It is then shifted by 8 bits and passed to the first 8 bits of a 16-bit register called VPrime. The values that the downsampled u and v are multiplied with is are given by a multiplexer with the FSM controlling the output.

The even u and v prime values are taken directly from the u and v registers, no calculations are need, and passed to the last 8 bits of the 16-bit register UPrime and VPrime respectively.

Colour Space Conversion

The colour space conversion takes the upsampled YUV data and performs matrix multiplication on it to change it to RGB data. 9 registers are used for this (R0, G0, B0,) for even (R0, R1, G1) for odd and (R, G, B) to store the final value to be written to the SRAM. One multiplexer is used to pass the appropriate value Y/U/V value to the a multiply accumulative unit and another multiplexer passes the appropriate value to be multiplied with. The multiplexers are controlled by the FSM. The value that gets multiplied gets passed to the appropriate RGB registers (R0, G0, B0, R0, R1, G1).

They are later passed to the R, G, and B registers. Before they get passed to these registers, they are clipped. If the value is negative a 0 is passed. If the value is greater than 255, 255 is passed. Else, the value from 23 to 16 in the registers are passed.

Time/Clock cycles

For one row the lead in case takes 22 clock cycles, the common case takes 924 clock cycles and the lead out case takes 28 clock cycles.

The total clock cycle for one row is 974. The total clock cycle for milestone 1 is 974 times 240, 233760.

The total time that milestone 1 take is 4675200 ns.

During the lead in state, there is 61.36% utilization of the multipliers. In the common case there is 91.67% and in the lead out state there is 69.64%.

The total utilization of the four multipliers is:

$$(61.36*22 + 91.67*924 + 69.64*28)/974 = 90.35\%$$

We were able to meet the 85% utilization constraint given to milestone 1.

FSM

The top level FSM in project.v sets a flag called M1_en to 1 to enable milestone 1. Once milestone 1 has finished, it will set a flag indicating it finished back to the top level FSM.

Milestone 1 has four components: Idle, State, Lead-In State, Common Case State, and Lead-Out State.

Lead-In State: During this state, the U and V registers are filled with values to be used to calculate the first odd u prime and v prime values. The first four RGB values of the row are calculated using the matrix equation, and saved to the appropriate address in the SRAM. We use 22 states to determine the appropriate value for the first four RGB values. After the final Lead-in state, the program goes to the first common case state.

Common Case State: During this state, the SRAM address is determined by the sum of RGBaddress and RGB_counter. The RGB_counter is incremented by 1 in each state. The ans1 and ans2 register, which holds all U values and V values, is set to the signed value determined by the multiplication circuitry we devised. Ans1 and Ans2 will be used for color space conversion.

Lead-Out State: During this state the U prime and V prime registers are filled with values from the corresponding U and V locations. We used 28 lead-out states to determine the appropriate U prime and V prime values. At the end of the lead-out states the program goes back to the idle state.

A 10-bit register called Hori_count and a counter was used to keep track of the number of horizontal pixels in the one row. After ever pair of u and values were addressed from the SRAM, the counter will increment the Hori_count by 1. A comparator is used to determine when the Hori_count register reaches the value 311, once it does the FSM will leave the common case state and enter the lead out state. Once Hori_count reaches 319, milestone 1 will go to the idles state. Once in the idles state, another 10-bit register called Vert_count and a counter was used to keep track of the number of vertical pixels. A comparator is used to determine when the Vert_count register reaches the value 249, once it does, a flag is passed to the top level FSM to indicate the end of milestone 1.

Problems and Challenges

There are several problems we encountered throughout the program. At times we would accidentally mix up our registers, or mix up the values going into our registers. Sometimes we carried out arithmetic operations with unsigned values, when we should have been using signed values. We determined the problem by looking at the waves in Modelsim, and tracing back to see where the source of the error is. We would look through the colour space conversion states and see if there were any calculation errors. If there are none move on to the U and V interpolation calculations and see if the problem was there. If the operations there were correct, we would move on see if the right values were being passed to the registers. When we do find the error we would look into the code at the state giving the error and make the necessary corrections. Using this systematic method, we can identify most of the problems we encountered.

Week 1	• In week 1 we only read through the project guidelines and planned out what needed to be done for milestone 1. • We started to understand the process of upsampling and colour-space conversion. We met on Wednesday October 25th for 3 hours to determine the project guidelines. We also met on Thursday October 26th for 3 hours to understand the colour-space conversion matrix equation.
Week 2	• We started to make a state table on excel for milestone 1. • We met on Monday October 30th for 2 hours to start writing the state table. We continued on Wednesday November 1st for 3 hours. • By Friday November 3rd we were able to finish the state table for the common case state.
Week 3	• Hassan started writing the code for the common case state of milestone 1. Areeb assisted with common case state table. • Re-edited the state table on excel for the lead in state to better understand how to code the lead in state on Monday November 6th. • By the end of the week the lead in and common case states were fully coded.
Week 4	• Not much accomplished due to conflicting schedules and other priorities including midterms for other courses • Asked the TA's and the professor for clarifications on concepts for milestone 1. • We met on Friday November 17 for 3 hours to continue coding.
Week 5	• Hassan finished writing code for leading out case of milestone 1. • We began debugging milestone 1 on Wednesday November 22nd. • Areeb asked the TA's for further clarifications and help for bugs appearing in milestone 1. • We met on Saturday • By Saturday, milestone 1 had no mismatches on ModelSim. • Also on Saturday we started to writ the report for the project • On Sunday we run the code on Quartus and successively projected an image on the computer screen

Conclusion

This project was a great learning experience for us. Although we only completed Milestone 1, we had to draw upon knowledge we acquired throughout the course, especially from the labs. During the duration of this project we learned an assortment of technical skills and social skills. The social skills we developed include time management, communication and collaboration skills. Despite often conflicting schedules we ensured that we set aside time specifically to work on this project, and had well-defined tasks set out. The technical skills we learned include a general understanding of how hardware can be used to do image compression and decompression. We also specifically learned how to do colour-space conversion and how to implement upsampling and why that is important. Finally, this project was a good exercise in integrating the SRAM, UART and VGA together to accomplish a specific goal. Overall, this project was a great experience to learn about digital systems design.