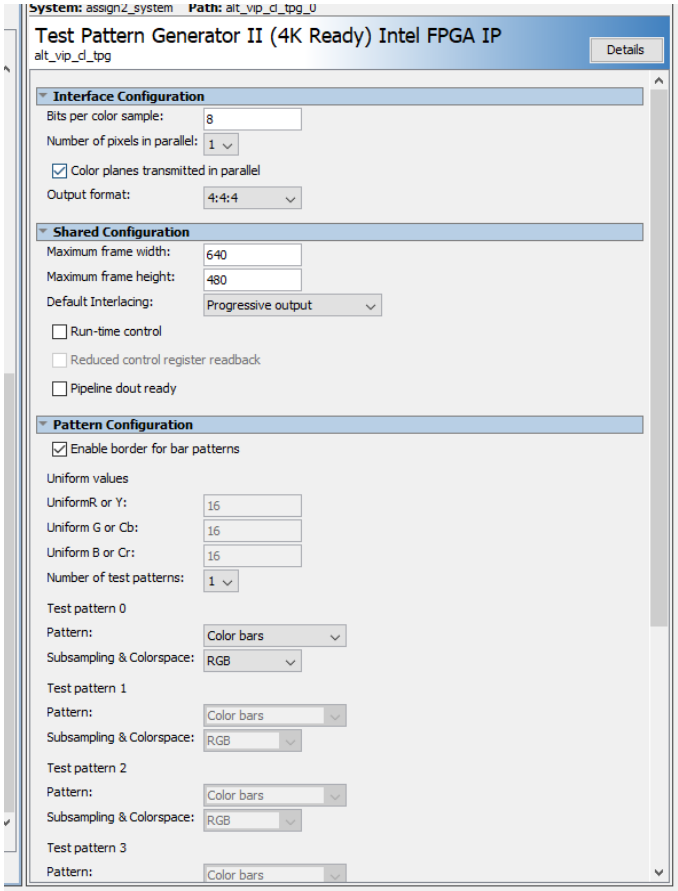
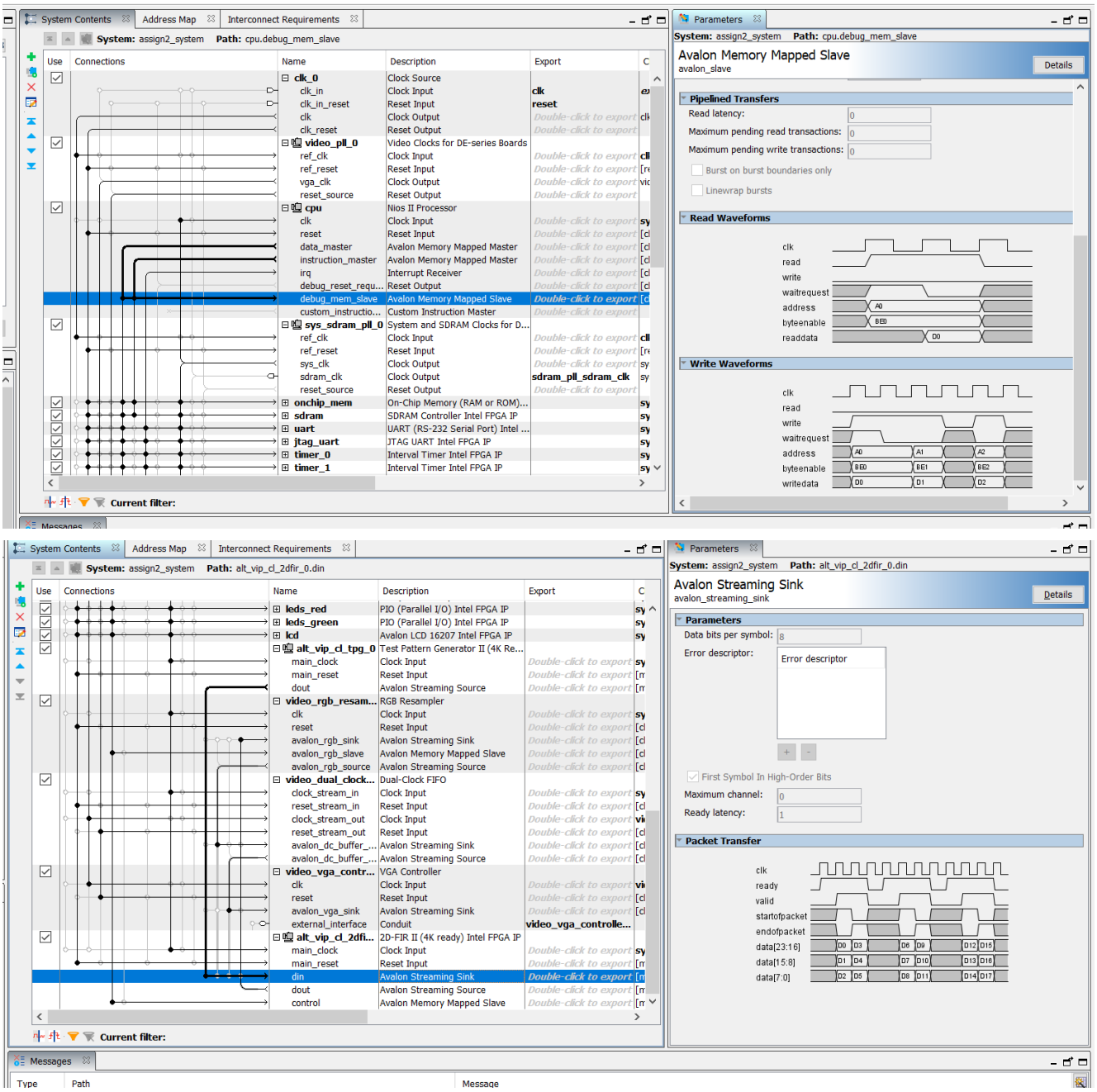
COMPSYS 303 A2 Team 10, smiy200, rnat697

Task 1

Q1. When we generated the HDL file using the qsys file given to us, added pin assignments, compiled the design and programmed it to the DE2_115 board, we saw this image above on the monitor.

Q2. The generated pattern can be changed using the pattern generator refer to image below for more info. Under the pattern configurations heading, we can change the pattern type which are Colour bars, Grayscale bars, Black and white bars, SDI pathological or Uniform background.





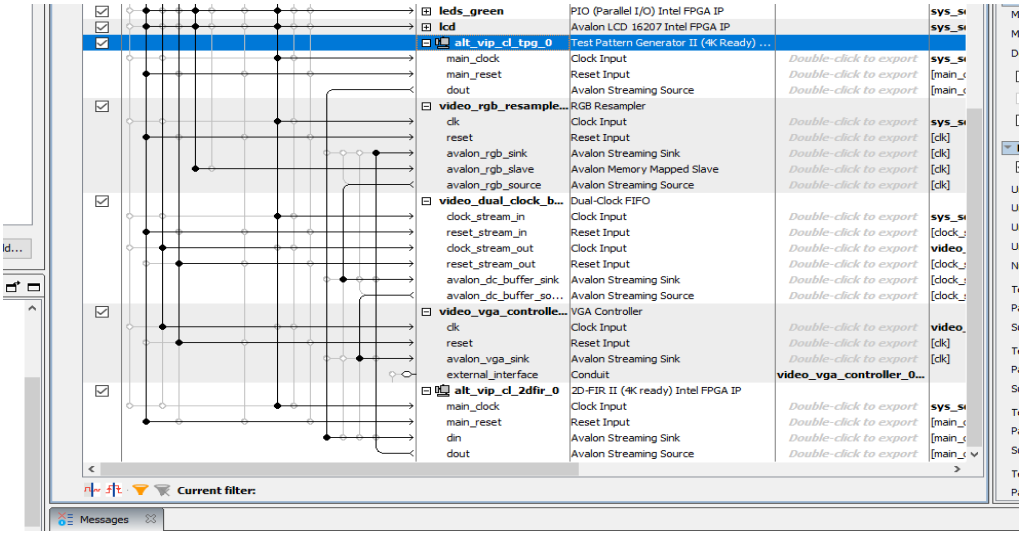
The two different Avalon bus types that are used in this system are Avalon Memory-Mapped bus and Avalon Streaming bus. The Avalon Memory Mapped buses are utilised through the Avalon Memory Mapped Master and Slave interfaces. They are used to connect the peripheral together with the switch fabric. This uses an address-based read/write interface with master and slave connections.

The Avalon Streaming buses are utilised through Avalon Streaming Sink and Source interfaces. This uses an unidirectional flow of data from source to sink and vice versa andis useful for direct memory transfer. In our case, it is used to transfer data between the pattern generator and 2D filter as well as the 2D filter and the video RGB resampler.

Task 2

Q1

This is when we added the 2D FIR II with a smoothing filter which uses the text file, Filter1.txt.



The image above is how we added the 2D FIR II component:

Figure1: The picture above is the 2D FIR II with a sobel edge X filter. This filter is defined in a text file called pokeX.txt.

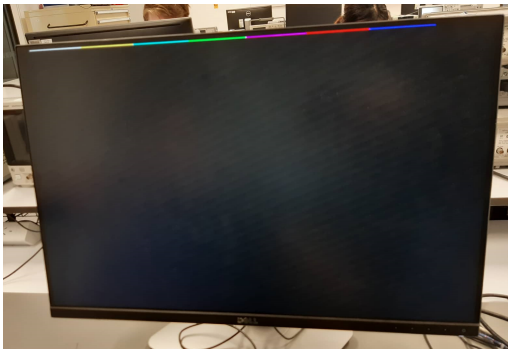


Figure 2. is the 2D FIR II with a sobel edge Y filter. This filter is defined in a text file called pokeY.txt

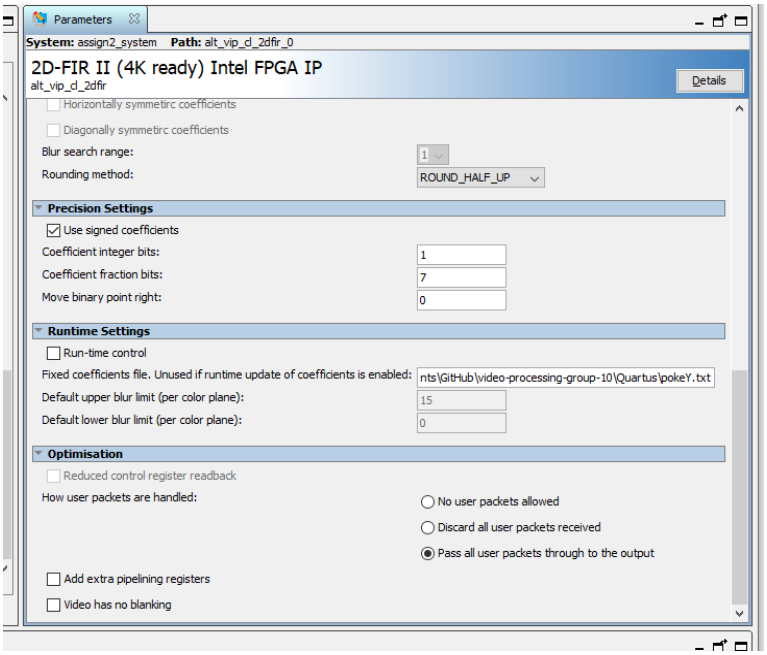
Q2. Using each filter resulted in the original colour bar having a matrix applied to it and distorting the image. The identity matrix resulted in the regular screen because the calculations applicable to the radius are all 0 and thus it is just the selected pixel value.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The Sobel X filter detects the edges in the x direction, and because in the centre the two 0's are in the y axis you can see in Figure 1 that the rest of the y bars have been deleted.

This applies to the sobel y filter where the horizontal centre has 0's and the upper centre is -2 while below centre is 2. Where it detects only the y edges. We can see that there is some colour at the top of the screen because when taking points around the upper part, the -2 is cancelled out so we are left with some colour value on each pixel.

Q3. Design is inflexible while only needing to change one line(filter location) to use another specific filter.The disadvantages include time taken to regenerate and compile the project after switching filters which is time expensive roughly(5-10mins). An alternative is to implement a control system in the nios processor that can be based on switches. This allows for multiple combinations of filters equal to the number of switches which is >10.



Task 3

Q1. We stored the filter coefficients in an array for the corresponding filters in binary or hexadecimal format. We also removed fraction points in the 2D FIR configuration, so we only use 3 bits. We then used a for loop that goes through the array and updates the FIR filter coefficients by using IOWR and including the 2D filter base and the address 6+i. The 6+i is to add the filter data at the corresponding tap.

```
for(int i = 1; i <= 9; i++) {
	IOWR(ALT_VIP_CL_2DFIR_0_BASE, 6+i, smoothing[i-1]);
}
```

Once we finished going through the array and updating the filter coefficients, we told NIOS we adjusted the coefficients by writing 6 and a value., `IOWR(ALT_VIP_CL_2DFIR_0_BASE, 6, 0b111111111);`. We chose 6 specifically to notify NIOS that we have committed the coefficient data.

What is interesting to note is that there seems to be a variance in compilation sometimes when using an integer as opposed to a binary into the register

Each filter has its own for loop and notifies NIOS of the update.

Address	Register	Description
3	`Blur search range`	Set this register to 1, 2, or 3 to override the default parameter setting for the edge detection range in edge-adaptive sharpen mode.
4	`Lower blur threshold`	This register updates the value of the lower blur threshold used in edge detection in edge-adaptive sharpen mode.
5	`Upper blur threshold`	This register updates the value of the upper blur threshold used in edge detection in edge-adaptive sharpen mode.
6	`Coefficient commit`	Writing any value to this register causes the coefficients currently in addresses 7 to (6+T) to be applied from the start of the next input.
7-(6+ T)	`Coefficient data`	Depending on the number of vertical taps, horizontal taps, and symmetry mode, T addresses are allocated to upload the T unique coefficient values required to fill the 2D coefficient array.

On switch 001, the filter that will be used is smoothing. On switch 010, the filter that will be used is Sobel Edge X. On switch 100, the filter that will be used is Sobel Edge Y and on any other switch value, the filter that will be used is identity.

Q2. To improve on our final system, we could add an interrupt for changes in switch instead of using polling.This would free up system resources that could be used elsewhere instead of polling for changes.

References:

[How Matrix Filters Work (sql4arc.com)]