Design Explanation for Image Filter

Karan Aggarwal

2019CS10699

1 Design Overview

Inputs:

1. Clock clk
2. Button b1 : Starts the filtering process.
3. Button b2 : A switch indicating whether smoothening or sharpening is required.

There are a total of 4 states in the design:

1. S0 - Initial state
2. S1 - A transition is made to this state when Button b1 is pressed (Also corresponds to the left and top most element (of the 9 elements currently being filtered) for each pixel of the filtered image.
3. S2 - This handles the rest of the 8 elements of the filtering process for each pixel of the filtered image.
4. S3 - The result for each pixel of the output is written to the corresponding address in memory here.

Data Signals used:

1. ram_read_address : The address to be read from the RAM at any moment
2. ram_write_address : The address to be written to the RAM at any moment
3. rom_address : The address to be read from the ROM at any moment
4. curr_address : Address of a pixel for which neighboring 8 elements are also used during filtering
5. I : Row number of the pixel for which neighboring 8 elements are used during filtering
6. J : Column number of the pixel for which neighboring 8 elements are used during filtering at any moment
7. a : Row number of a 3x3 matrix
8. b : Column number of a 3x3 matrix

S0
At state S0, some signals are initialised to default values.
When button b1 is pressed:
If b2 is off, rom_address is set to "00000" (0) (Smoothening).
Else rom_address is set to "01001" (9) (Sharpening).

S1
Now it is in state S1 with default values I = 1, J = 1. Here a = 0, b = 0. Also, the ram_read control is enabled in this state.
Here, in the state, ram_read_address stores the value of the left most and top most pixel out of the 9.
At the clock edge, ram_read_address is updated, rom_read_address is updated and a transition to S2 is made.

S2
When the transition to S2 is made for the first time, ram_out holds the value of the address of the of the left most and top most pixel out of the 9 neighboring in the original image.
In state S2, the mac_control is enabled. At every clock edge of this state, the sum in the MAC is updated with the product corresponding to the previous pixel. This continues for the remaining 8 pixels.

Also, at every edge, MAC updates the value of the previous state, not the present one, because of the delay as MAC architecture is written in a process. When all the 9 pixels are added to the sum, it transits to state S3.

S3
Here ram_read is disabled and ram_write is enabled. At the transition edge, state becomes S1 and the sum is written to the RAM.

This continues for the next pixel (I=1, J=2) now.

When I reaches 159, the FSM exits S1 and goes to S0.

Further details in ASM chart and code comments