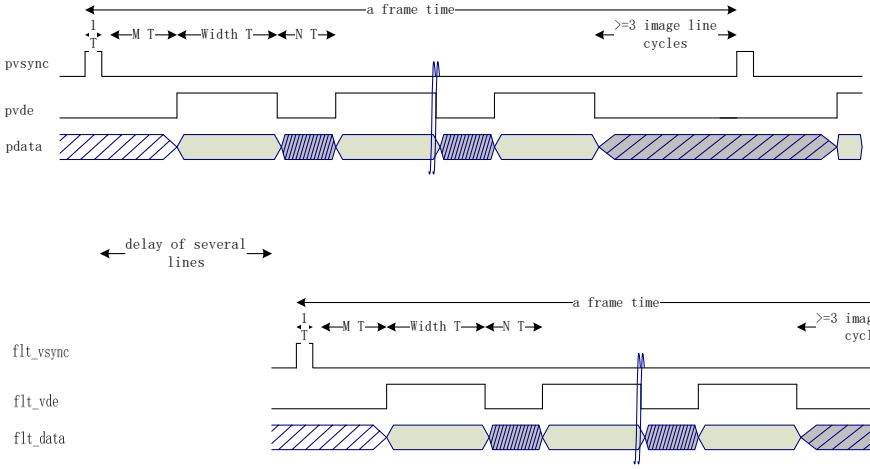
Guass Filter Specification

1. Function Description

Design a module of Guass Filter function for image processing. Input image format is YUV 4:2:2 with 8bit resolution. Image will be inputted pixel by pixel forms an image line, and then line by line forms an image frame. Each cycle will input data of a pixel, that's either {y[7:0],u[7:0]} or {y[7:0],v[7:0]}. Output is in the same format, just with delay of several image lines. Sample waveform:



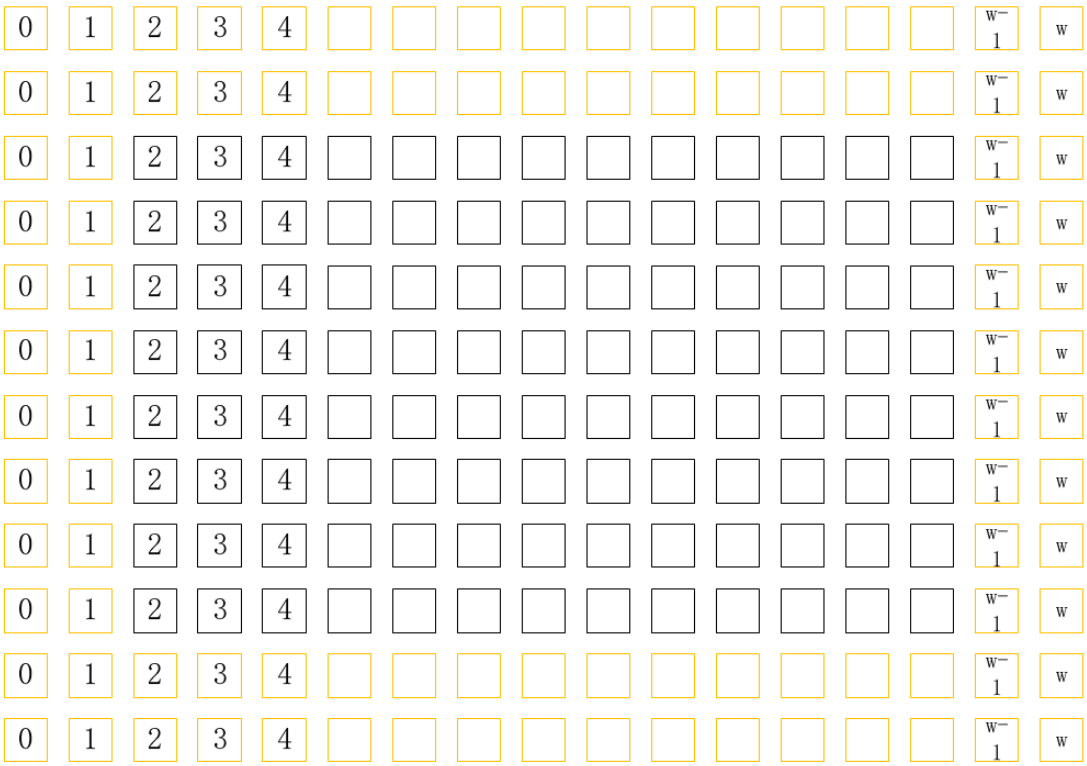
Let's take image of 1920x1080 (width x height) for example:

1) each line has 1920 pixels, taking 1920 cycles;
2) 200 idle cycles after each lines;
3) there is 3 idle image lines after all the valid lines of a frame, and before next high pulse of pvsync;

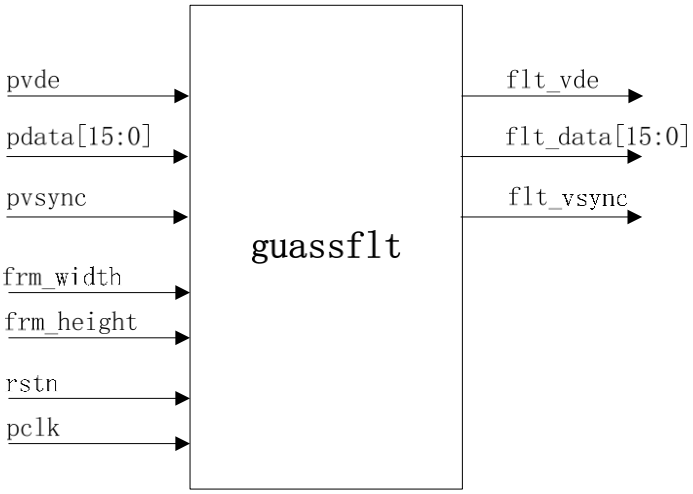
Define of Guass 5x5 filtering kernel:

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

And just filtering Y component, bypass UV components. While when outputting filtered frame, you need to align Y and UV component to make a pixel. At the boundary of each frame, that's x-coordinate equal to 0/1/frm_width-1/frm_width or y-coordinate equal to 0/1/frm_height-1/frm_height, just output the value of original input. These pixels are shown as orange color in following figure:



Block diagram of the design:



2. IO Define

Name	Direction	Bits	Description
pclk	I	1	clock input;
rstn	I	1	low active asynchronous reset;
frm_width	I	11	frame width of input image, count from 0. If 1919, the width of input image is 1920 pixels.
frm_width	I	11	frame height of input image, count from 0. If 1079, the height of input image is 1080 lines.
pvde	I	1	1'b1: there's valid pixel for this cycle; 1'b0: no valid cycle for this cycle;
pdata	I	16	{y[7:0],u[7:0]} or {y[7:0],v[7:0]};
pvsyc	I	1	1T high pulse indicate the beginning of a frame;
flt_vsync	O	1	1T high pulse indicate the beginning of a frame;
flt_vde	O	1	1'b1: there's valid pixel for this cycle; 1'b0: no valid cycle for this cycle;
flt_data	O	16	image value after filtering, {y_flt[7:0],u[7:0]} or {y_flt[7:0],v[7:0]};

3. Design Considerations:

1) Boundary processing method defined in the algorithm;
2) Can you write a C/Matlab/Python/… model for the algorithm first?
3) How many lines we need buffer? 5 or 4? (consider area of SRAM)
4) Using two-port or single port SRAM? (consider area of SRAM)
5) The filtering coefficients are symmetric. Can we using this character to save number of add/multiplier operation, then save area;
6) How to pipeline the 5x5 filtering kernel to maximal the clock frequency?