VLSI Plus

NLM

Data Sheet

Revision 0

Features

- Efficient implementation of the NLM noise reduction algorithm
- Two clock per pixel
- Low power consumption
- Search window size and number of bits per pixel are parameters
- Deliverables:
    - obfuscated Verilog RTL
    - Bit-exact C model (executable only)
    - Verilog Test Bench

The NLM Algorithm

NLM (Non Local Mean) is a noise reduction algorithm wherein the value of each pixel is determined a rectangle of pixels around that pixel (a "center patch") with all identically-sized rectangles of pixels ("search patches") within a relatively large rectangle around the pixel (the "search window"). A weight is then given to the center pixel of each "search patch" according to the distance of the corresponding search patch from the center patch, and the output pixel is derived by calculating the weighted average of all the pixels in the search window.

There are two variants to the evaluation of the distance between the center patch and a search patch – sum of the squares of the absolute values of the differences between respective pixels in the two patches. Mathematically:

Let $B(p,r)$ indicate a neighborhood (patch) centered at p of size $(2r+1)\times(2r+1)$ pixels, for the case of 3X3 patches we have $B(p,1)$ which is a 3X3 patch centered at pixel p.

Then:

SSD (Sum of Square Differences):

$$d_{SSD}\big(B(p,f),B(q,f)\big)=\frac{1}{(2f+1)^2}\sum_{j\in B(0,f)}\big(u(p+j)-u(q+j)\big)^2$$

SAD (Sum of Absolute Differences):

$$d_{SAD}\big(B(p,f),B(q,f)\big)=\frac{1}{(2f+1)^2}\sum_{j\in B(0,f)}|u(p+j)-u(q+j)|$$

The weight of each patch is then calculated as a function of its distance from the center pixel, according to the equation:

SSD (Sum of Square Differences):

$$w(p,q) = e^{-\frac{d_{SSD}}{h^2}}$$

SAD (Sum of Absolute Differences):

$$w(p,q) = e^{-\frac{d_{SAD}}{h}}$$

h is a parameter which is generally proportional to the expected noise of the image.

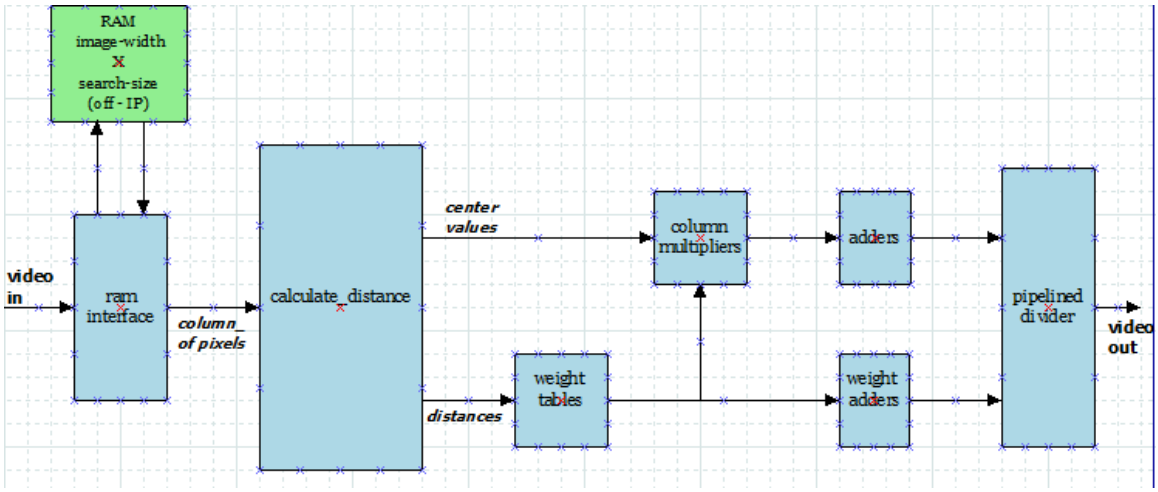
For more information about the NLM algorithm, the reader is referred to:

1. https://dx.doi.org/10.1109%2FCVPR.2005.38
2. http://123seminarsonly.com/Seminar-Reports/029/42184313-10-1-1-100-81.pdf
3. http://www.ipol.im/pub/art/2011/bcm_nlm/

NLM implementations may vary in the following aspects:

- (See above) - Distance calculation may be done by summing the squares or absolute values.
- The distance of the center patch is, by definition, 0. Heuristically, however, choosing 0 often disrupts noise reduction. Therefore, implementations often prefer to use a lower value for the weight of the center patch**.**
- Calculation done for pixels at the edges and near the edges of the image, where there is no complete search window.
- Accuracy of the calculations – from the input pixels, through the various stages and up to and including the output pixels.

The NLM01 Block Diagram



The figure above depicts a simplified block diagram of the V-NLM01 IP. Incoming pixels are handled by a RAM-interface unit, which stores previous rows in an off-IP RAM, and outputs a column of N-1 pixels (N is the height of the search window).

Next, the column of pixels enter the Calculate Distance unit. A Column Shift Register stores the values of the previous n columns and, at all times, the values of all pixels in the search window. In parallel, a group of column summers concurrently sums the absolute differences for all the patches in the search window.

Distances out of the Calculate Distances unit are input to a Weight Tables unit, which calculates weight estimates using a piece-wise linear approximation of the exponent function.

Both the search window pixel values and the corresponding weights are input to a Column Multipliers unit, which multiplies the values of the pixels by the corresponding weights. The products are then added by the Adders unit, which produces a sum-of-products value for each input pixel. At the same time, the weights of all pixels of the search window are summed in a Weight Adders unit. The sum of products and the sum of weights enter a Pipeline Divider unit, which produces the output pixel by division and rounding to the nearest.

IP Pads

The following table depicts the inputs and outputs of the RTL model. Pads like power, boundary scan and similar are not included, and should be added by the customer as part of the back-end design.

All signals except rst_n are synchronous to the clock input.

Pad	I/O	Description
Clock, Reset		
clock	input	Single clock input. Frequency should be at least twice the pixel rate
rst_n	input	Asynchronous active-low reset input
Modes, Parameters		
bypass_mode	Input	Used for debug. When this active-high input is asserted, the V-NLM01 will assign zero weights to all pixels in the search window except the center pixel, so that the output pixel stream will be identical to the input pixel stream.
one_over_h[11:0]	Input	This input should be driven with the value of 5232/h, where h is as defined in the NLM algorithm (see above)
Video Input		
pixel_in [NUM_BITS-1:0]	Input	Pixel input stream.
pixel_in_valid	Input	Signal indicating valid pixel input. There must be a gap of at least one clock cycle between every two clocks with pixel_in_valid=1
line_start_in	Input	Single clock period pulse, indicating the beginning of an input video line.
line_end_in	Input	Single clock period pulse, indicating the end of an input video line.
frame_start_in	Input	Single clock period pulse, indicating the beginning of an input video frame.
frame_end_in	Input	Single clock period pulse, indicating the end of an input video frame.
Video Output		
pixel_out [NUM_BITS-1:0]	Output	De-noised pixel output stream
pixel_out_valid	Output	Signal indicating valid pixel output. There will be a gap of at least one clock cycles between every two clocks with pixel_out_valid=1

Pad	I/O	Description
line_start_out	Output	Single clock period pulse, indicating the beginning of an output video line
line_end_out	Output	Single clock period pulse, indicating the end of an output video line.
frame_start_out	Output	Single clock period pulse, indicating the beginning of an output video frame.
frame_end_out	Output	Single clock period pulse, indicating the end of an output video frame.
RAM interface(*)		
ram_ceb	Output	RAM enable, active low.
ram_web	Output	RAM write, active low.
ram_addr [NUM_BITS-1:0]	Output	RAM Address.
ram_din [NUM_BITS*(N-1)-1:0]	Output	RAM write data.
ram_bweb [NUM_BITS*(N-1)-1:0]	Output	RAM write mask, active low.
ram_dout [NUM_BITS*(N-1)-1:0]	Input	RAM read-data.

(*) It is assumed that a one clock cycle synchronous RAM is used, but arbitration logic between the V-NLM-01 and the SRAM adds one clock cycle delay to all outputs to the SRAM, and one extra clock cycle delay to the ram_dout input from the SRAM.

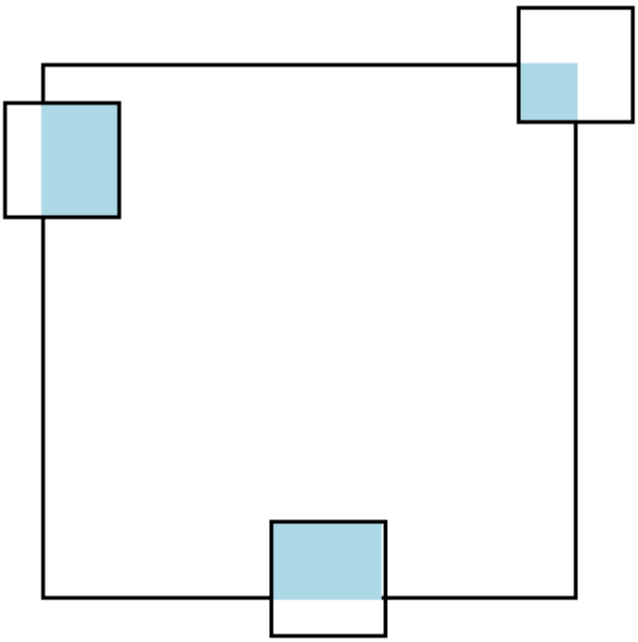
Handling of Boundary Pixels

Boundary pixels are pixels located near or at the edge of the image, and, consequently, are not surrounded by full search windows.

The V-NLM01 handles boundary pixels as follows:

Pixels at one of the 4 edges – those pixels not only lack a significant portion of the search window, but they also do not have a center patch. Hence it is impossible to determine their distance and their weight. Those pixels are transferred as are from input to output.

Pixels close to the edge – pixels which are (N-1)/2-1 or less pixels apart from an image edge (N is the search window size) have a partial search window only, as depicted in the figure below:

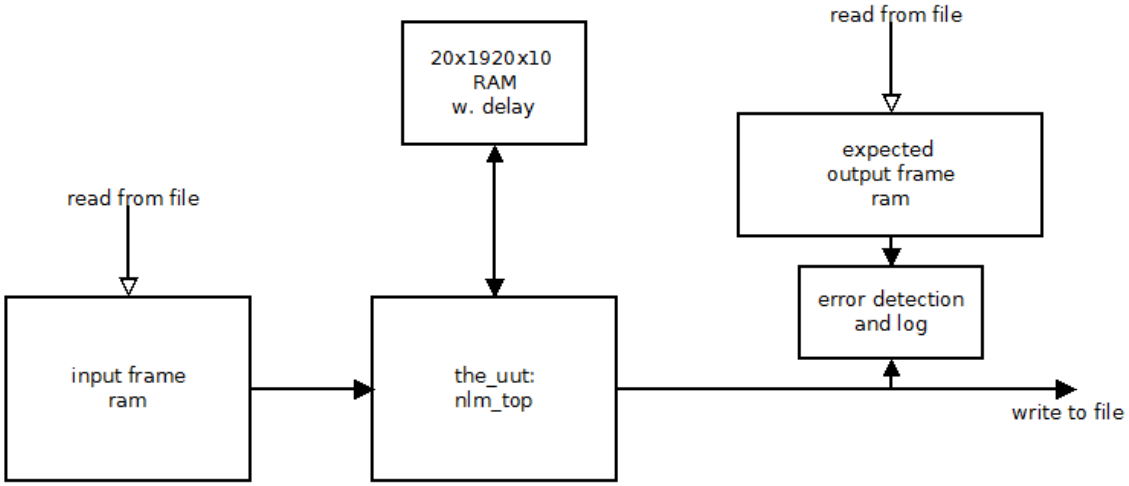


Effective search windows are colored grey. The value of the pixel is evaluated by calculating the weighted average of the patches in the grey area only.

The C model

A bit-exact C model of the VNLM01 RTL is provided (executable only), along with a .bat script file .

Verilog Test Bench



A Verilog test bench is provided with the VNLM01 (configured for 1920 pixel lines; search window=21x21 in the figure above). It reads an input image from a file, has `defined WIDTH, HEIGHT, bit-per-pixel, search-window size and one_over_h. It feeds the UUT with the pixels, and with the video control signals. It generates randomly spaced pixel_valid signals. The output from the UUT is stored in an output image. Using `define switches, it is possible to make it compare its output to a given expected-results file, and stop when / if any pixel fails.

The test bench also include a model of the off-IP memory, with delay as defined by the customer.