DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Filtro de imágenes

Diseño de Sistemas Digitales con FPGA

Rosende Federico

federico-rosende@hotmail.com

Objetivo

- En un primer momento se apuntó a realizar un filtro de imágenes para procesar los cuadros de una cámara web en tiempo real.
- Para llegar al objetivo, se empezó realizando un filtro que sea aplicado sobre imágenes estáticas.

El filtro

La idea del filtro es aplicar una convolución simple sobre la imagen para reducir el ruido y suavizarla. [fil,] En particular, para cada canal p'_c de cada pixel p' que no está en el borde, se aplica

$$new\_p'_c = \frac{1}{n}(p'_c + \sum_{p \in N(p')} p_c)$$

Donde $N(p')$ es el vecindario de p' y $n = |N(p')| + 1$. Es decir, se calcula el promedio considerando $N(p')$.

El vecindario

Para definir el vecidario se considera una matriz de píxeles centrada en el pixel que se está suavizando. En este caso, se opta por la siguiente matriz de convolución de 3×3

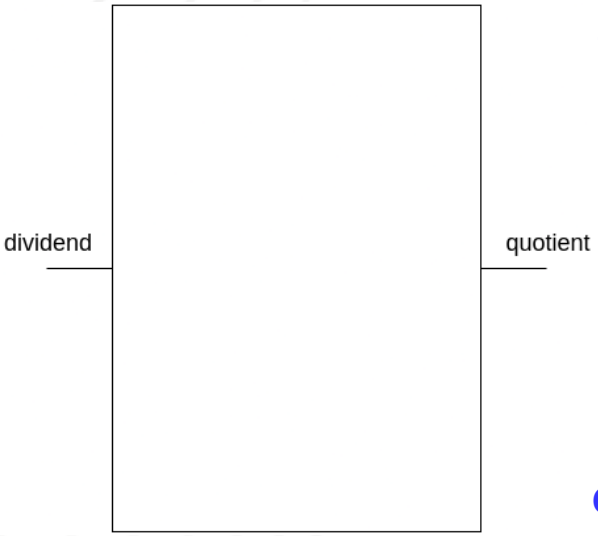
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Propuesta

Para avanzar con la especificación, se proponen dos entidades principales:

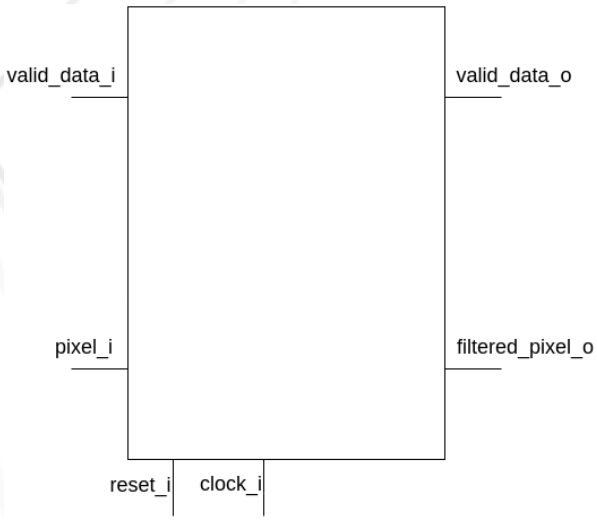
- Un divisor que recibe un número y devuelve el resultado de la divisón entera por 9 (no interesa la parte decimal).
- Un filtro que aplica la convolución sobre un pixel, considerando su vecindario y haciendo uso del divisor.

En detalle
Divisor




```
entity divider_entity IS
  port (
    -- 2295 = 255*9
    dividend: in integer range 0 to 2295;
    quotient: out std_logic_vector
                                (7 downto 0)
  );
end divider_entity;
```

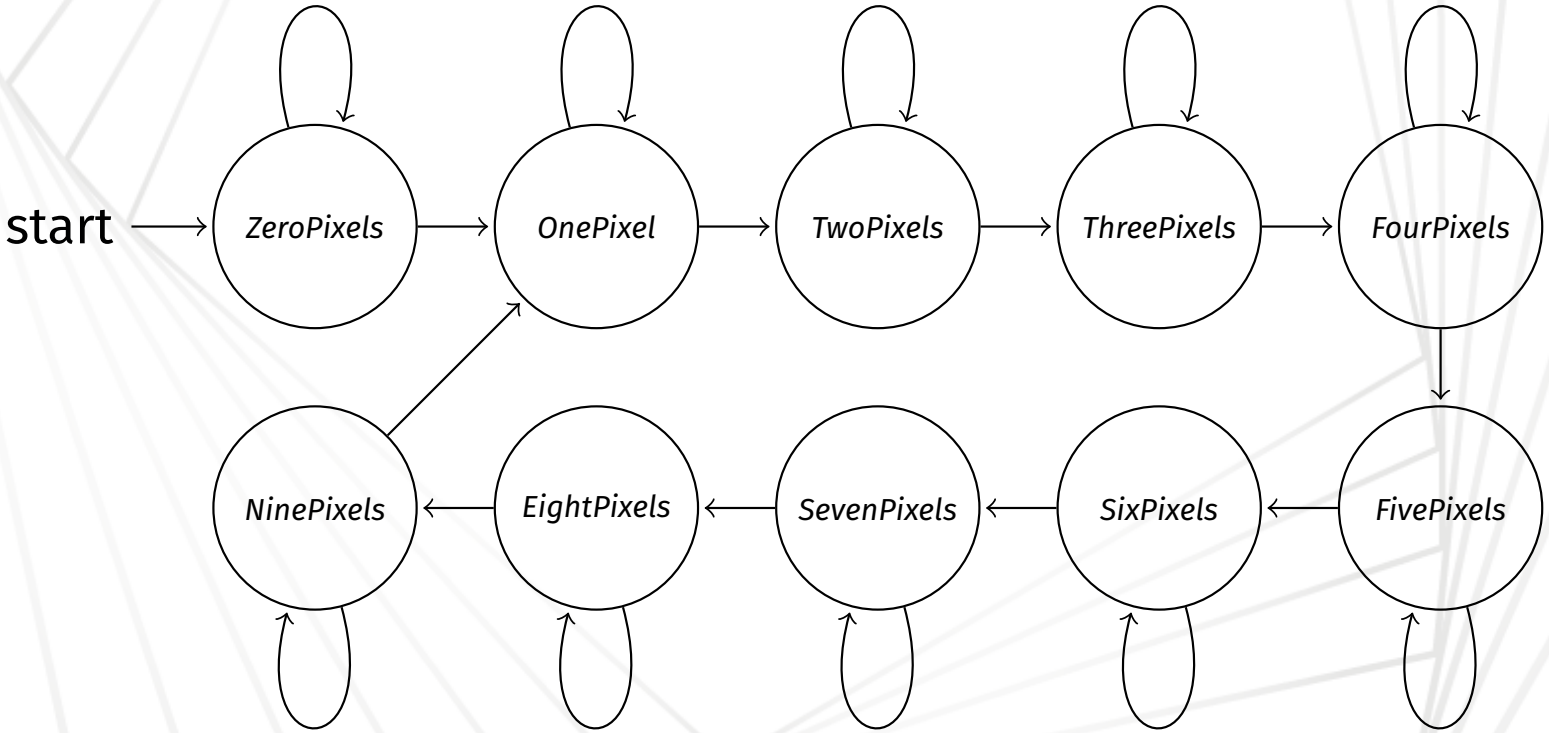
En detalle
Filtro




```
entity pixel_filter_entity is
  port(
    clock_i : in std_logic;
    reset_i : in std_logic;
    pixel_i : in std_logic_vector
                          (31 downto 0);
    valid_data_i : in std_logic;
    filtered_pixel_o : out std_logic_vector
                                (31 downto 0);
    valid_data_o : out std_logic
  );
end entity;
```

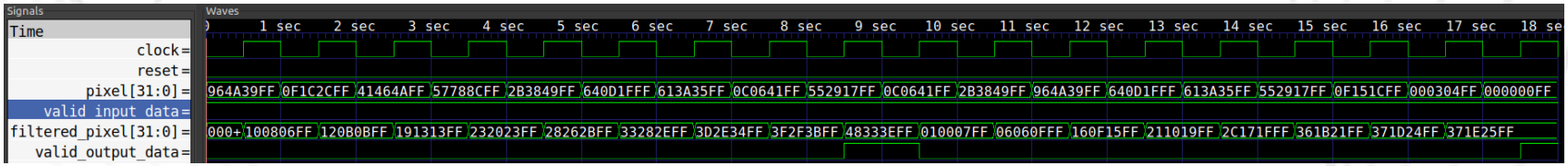
Estados del filtro

El filtro utiliza una máquina de estados como representación interna. Las transiciones entre estados distintos se dan a través de *valid_data_i*, mientras que los bucles ocurren con *not valid_data_i*. Cuando el filtro alcanza el estado *NinePixels*, levanta la señal *valid_data_o* indicando que *filtered_pixel_o* contiene el resultado final para el pixel que se procesa.



Ejemplo

Para ver un ejemplo, se cuenta con un test bench que carga los píxeles de una imagen de 3×4. Al observar las señales, se aprecia que computa correctamente el valor de los dos píxeles internos suavizados.

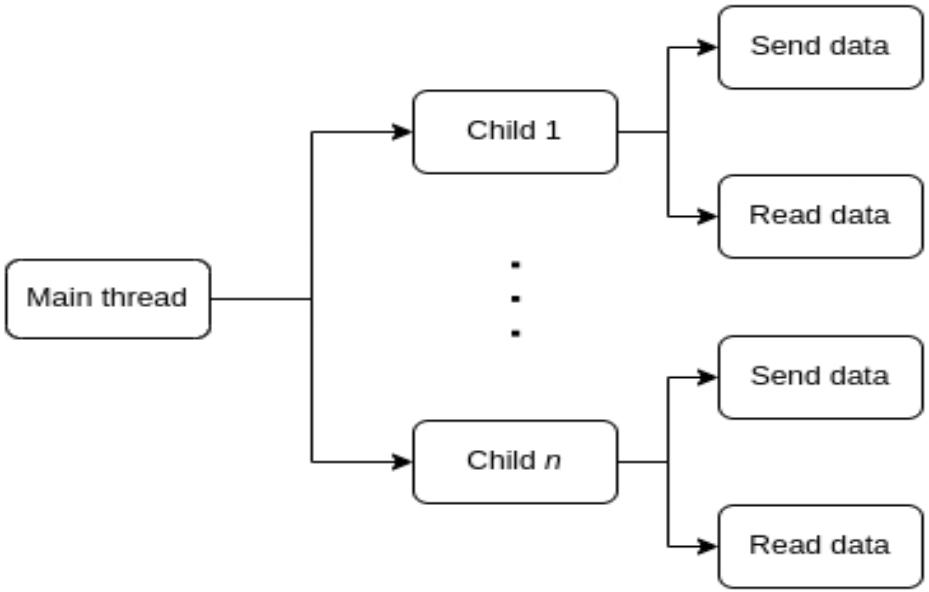


Desde VHDL

Se busca combinar las entidades nuevas con *top_level* (provista por la cátedra) que permite la utilización de FPGALink para interactuar con una placa desde un programa en Python. Esta entidad utiliza FIFOs para enviar y recibir datos, y el filtro se integra al esquema de forma que queda conectado entre las FIFOs de entrada (*FIFO8to32*) y salida (*FIFO32to8*). Puede pensarse el conjunto *FIFO8to32* + *Filtro* + *FIFO32to8* como una agregación lógica, de forma que se cuenta con dos de ellas y cada una escribe y lee de un único canal.

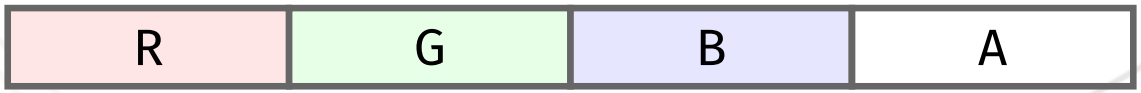
Desde Python

Para procesar una imagen se plantea un filtrado concurrente de la misma. El *thread* principal envía la imagen a procesar a *n threads* y cada uno de estos vuelve a dividir la tarea entre dos *threads*. Uno se encarga de enviar el pixel a la placa (a través de un canal dedicado para el *thread* padre), mientras que el otro lee el pixel suavizado que es devuelto. En la implementación actual, *n* toma un valor de 2.



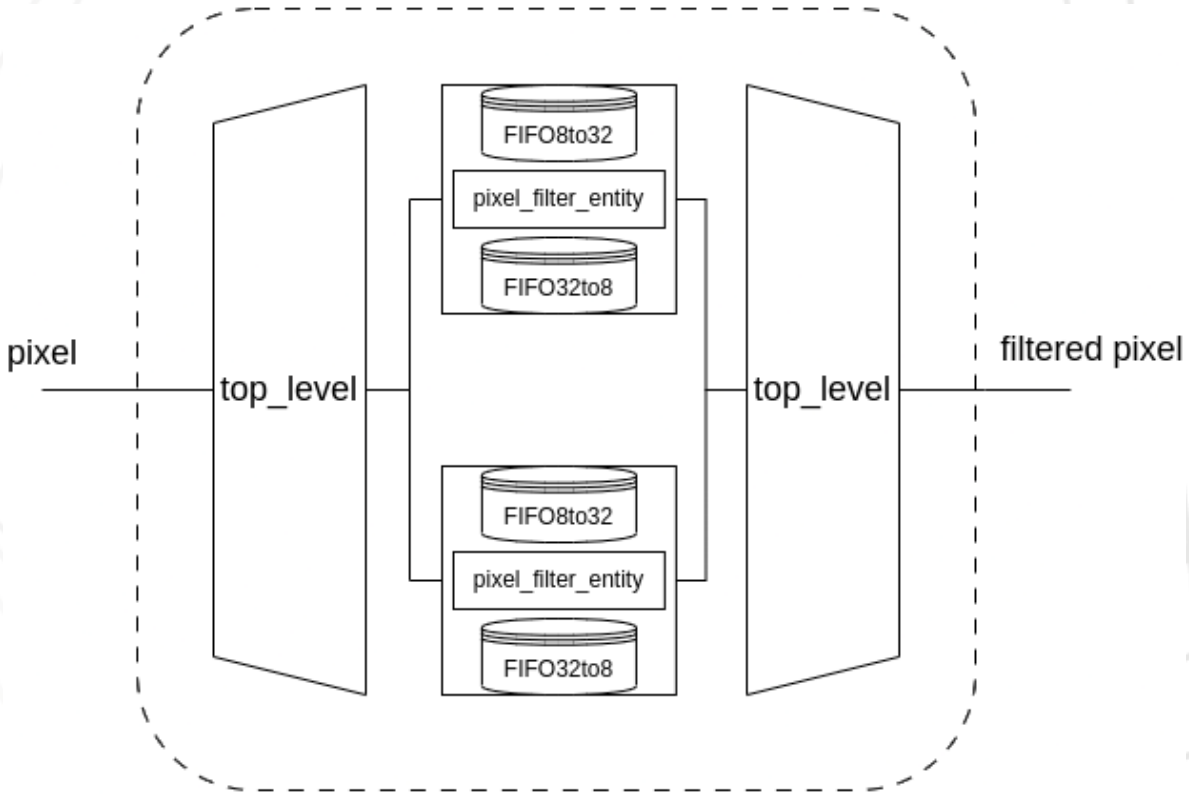
Tamaño de los datos

La representación de la imagen se da en formato *RGB* y cada pixel ocupa 24 bits. La herramienta utilizada no permite la generación de las entidades *FIFO8to24* y *FIFO24o8* que serían adecuadas para el caso de uso. Por ese motivo, se extiende la representación a *RGBA* pero el canal *alpha* no se utiliza durante el filtrado y posteriormente se descarta en el *thread* encargado de lectura.



Esquema completo

Por lo tanto, el esquema general que se obtiene es el siguiente



Problemas

A partir de este punto se requería el uso de una placa para validar la especificación. Si bien mediante el uso de *PlanAhead* es posible generar el bitstream, no se pudieron hacer pruebas sobre la placa y quedan como trabajo futuro. A su vez, esto limitó el alcance del proyecto que se restringió a procesar una imagen puntual en lugar de hacer un procesamiento en tiempo real de los cuadros entregados por una cámara web.

Trabajo futuro

- Integrar el filtro y las FIFOs en una única entidad
- Lograr conectarse con una FPGA para validar la arquitectura propuesta
- Procesar los cuadros de una cámara web en tiempo real
- Aplicar otro tipo de filtro que resulte de interés
- Considerar vecindarios de 5×5 y comparar con la versión de 3×3
- Permitir *n* número de *threads*

Referencias

 *Eliminación de ruido aplicando el filtro.*