Filtro de ventana deslizante.

Presentación del trabajo práctico final de la materia Microarquitecturas y Softcores

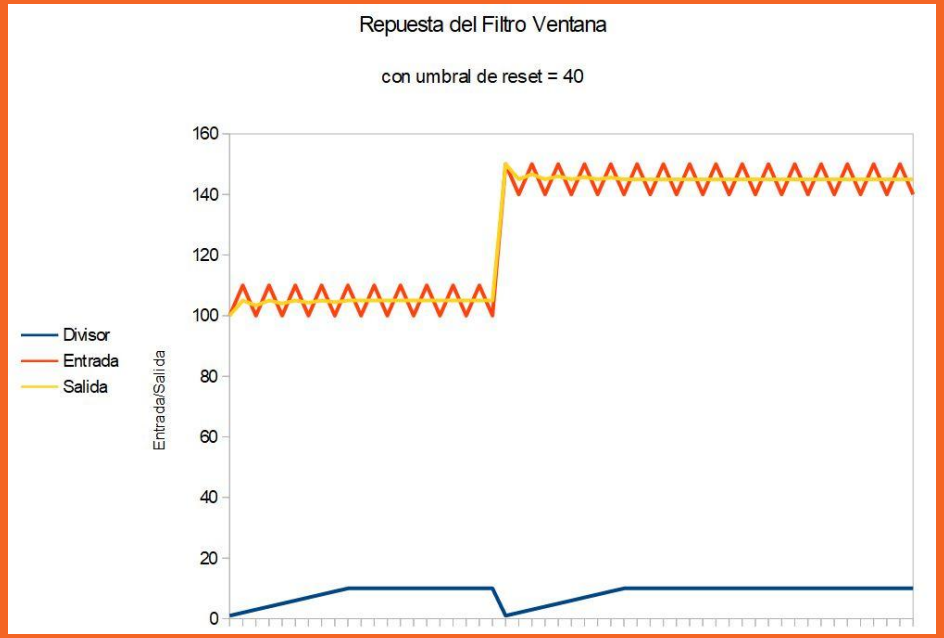
Autor: Ing. Gonzalo Sanchez - FIUBA
Junio 2022

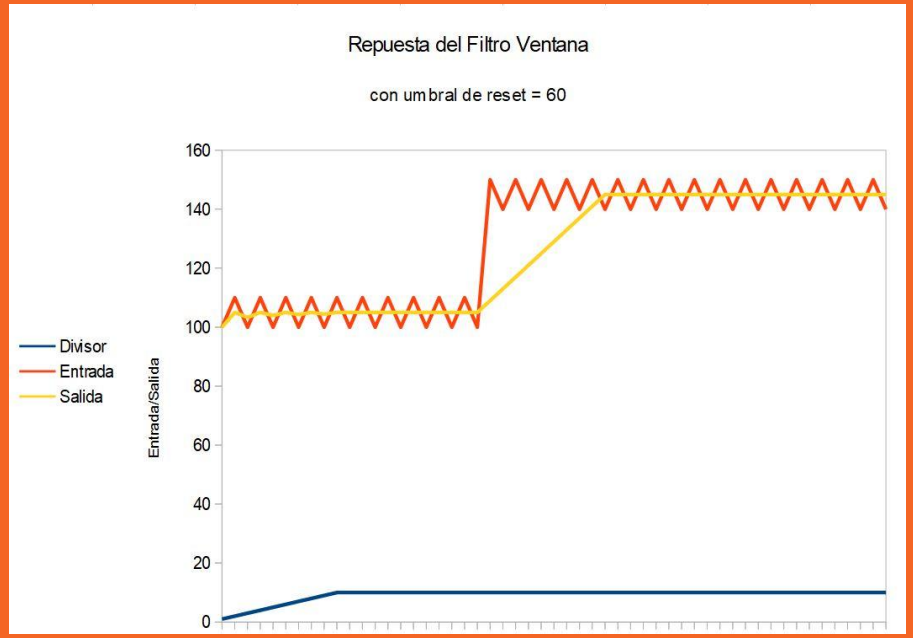
Características del proyecto

- Filtro de ventana deslizante de 50 muestras para filtrar lecturas de un ADC de 24 bits.
- Entrada de valores enteros de 32 bits.
- Valor de salida de 32 bits.
- Hasta que se llena la ventana solo se promedian las muestras leídas.
- Posee una entrada de 32 bits que fija un umbral de reseteo del filtro. Si la diferencia entre la muestra entrante y la muestra anterior es mayor que el umbral, el filtro se resetea automáticamente.

Valores ventana					Promedio calculado
1					**salida = 1 / 1**
2	1				**Salida = (1 + 2) /2**
3	2	1			**salida = (1 + 2 + 3) / 3**
4	3	2	1		**salida = (1 + 2 + 3 + 4) / 4**
50	49	.....	2	1	**salida = (1 + 2 + ... + 50) / 50**
51	50	49	.....	2	**salida = (2 + 3 + ...+ 51) / 50**
52	51	50	.....	3	**salida = (3 + 4 + ... + 52) / 50**

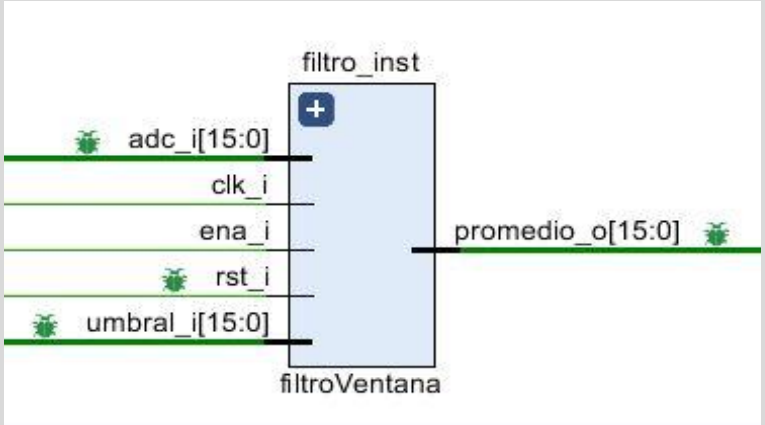
Comportamiento esperado





Entradas y Salidas del IP core

- Adc_i: valores de entrada std_logic_vector 32 bits
- Clk_i: entrada de clock, std_logic
- Ena_i: entrada habilitación, std_logic

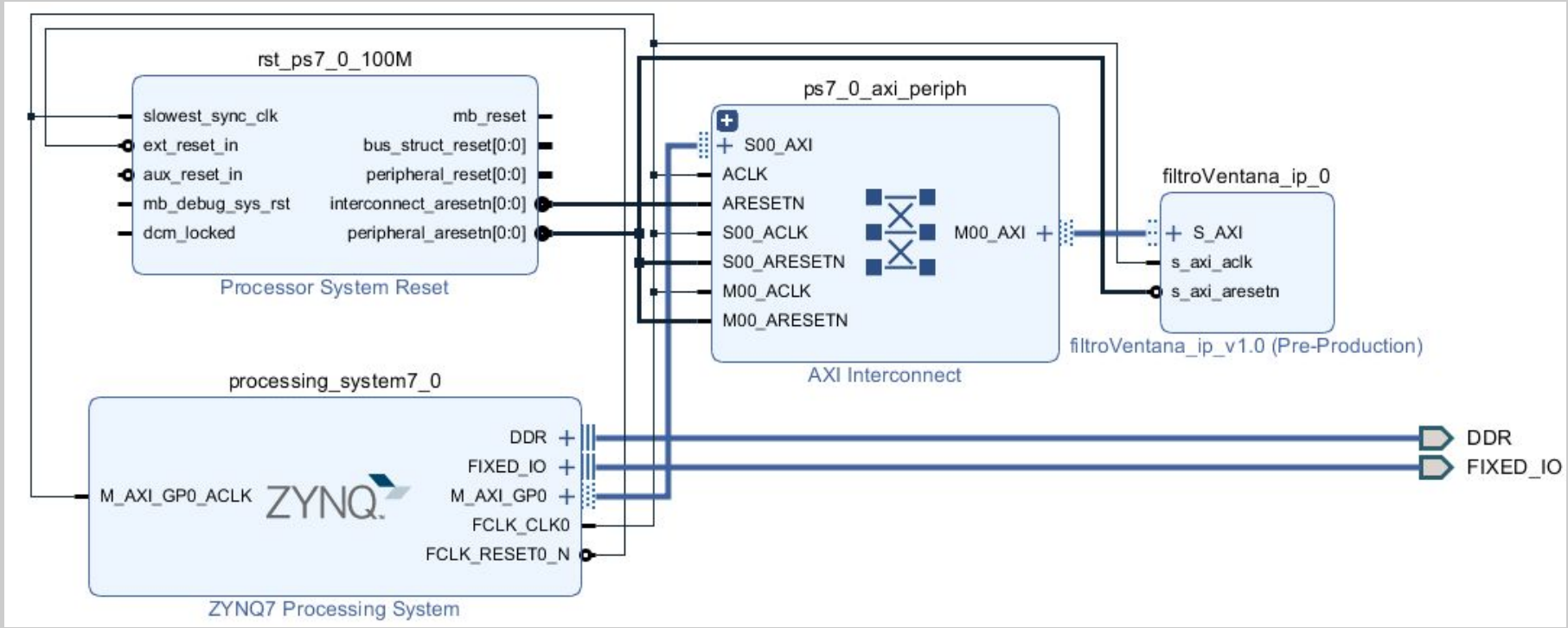


- Adc_i: valores de entrada std_logic_vector 32 bits
- Clk_i: entrada de clock, std_logic
- Ena_i: entrada habilitación, std_logic

```
  -- Add user logic here
inst_filtro: filtroVentana
        port map(
            adc_i =>        slv_reg0,
            umbral_i=>      slv_reg1,
            clk_i=>         slv_reg2(0),
            rst_i=>         slv_reg3(0),
            ena_i=>         slv_reg4(0),
            promedio_o=>    sal_filtro
        );
  -- User logic ends
```

- Dado que se debe hacer la promediación solo cuando se ingresa un nuevo valor al filtro, la señal de clk se conectó a un registro que al subirlo y bajarlo desde el software realiza el cálculo del promedio.
- Esto se podría mejorar modificando la lógica programable agregando una señal de conversión que ejecute un solo ciclo de clock y se autoresetee.

Agregado del IP al sistema



- Al igual que en el ejemplo del IP sumador, este IP no interactúa con las entradas o salidas de la placa sino que lo hace solamente con el Hardcore mediante el bus AXI. Es por esto que no se requiere agregar un VIO para probar su funcionamiento.

Software y simulación

- Para probar el funcionamiento se definió un valor umbral y un arreglo de valores aleatorios procurando generar algunos saltos mayores al umbral definido.
- Luego en un bucle se ingresaron al IP todos los valores del arreglo y se leyeron los valores de salida imprimiendolos en pantalla
- En otro bucle similar se subieron y bajaron las señales de reset y enable y se verificó su funcionamiento.

```
#define UMBRAL      500

static uint32_t datos_in[] =
{
        100,200,300,400,300,350,400,450,400,350,
        1100,1200,1300,1400,1350,1400,1450,1400,
        300,250,300,200,250,350,400,600,500,700
};
```

```
for(i = 0; i < cant; i++)
{
    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, IN_REG, datos_in[i]);
    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, CLK_FLAG);

    xil_printf("Dato in: %d \r\n", datos_in[i]);

    if(i > 0)
    {
        diff = calcular_abs(datos_in[i], datos_in[i-1]);

        if(diff > UMBRAL)
        {
            xil_printf("La diferencia entre %d y %d en mayor que el umbral %d \r\
            xil_printf("El filtro se resetea automaticamente\r\n");
        }

    }
    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, 0);
    res = FILTROVENTANA_IP_mReadReg(BASE_ADDRESS, OUT_REG);

    xil_printf("Salida Filtro: %d \r\n", res);
}
```

Reset por umbral

```
Inicio del programa para validar el uso del IP core
de un filtro de ventana deslizante con umbral de autoreset--

Fijo un umbral de reset igual a 500
Dato in: 100
Salida Filtro: 100
Dato in: 200
Salida Filtro: 150
Dato in: 300
Salida Filtro: 200
Dato in: 400
Salida Filtro: 250
Dato in: 300
Salida Filtro: 260
Dato in: 350
Salida Filtro: 275
Dato in: 400
Salida Filtro: 292
Dato in: 450
Salida Filtro: 312
Dato in: 400
Salida Filtro: 322
Dato in: 350
Salida Filtro: 325
Dato in: 1100
La diferencia entre 350 y 1100 en mayor que el umbral 500
El filtro se resetea automaticamente
Salida Filtro: 1100
Dato in: 1200
Salida Filtro: 1150
```

```
100 / 1 = 100
(100 + 200) / 2 = 150
(100 + 200 + 300) / 3 = 200
(100 + 200 + 300 + 400) / 4 = 250
...
```

```
Salida Filtro: 1200
Dato in: 1400
Salida Filtro: 1250
Dato in: 1350
Salida Filtro: 1270
Dato in: 1400
Salida Filtro: 1291
Dato in: 1450
Salida Filtro: 1314
Dato in: 1400
Salida Filtro: 1325
Dato in: 300
La diferencia entre 1400 y 300 en mayor que el umbral 500
El filtro se resetea automaticamente
Salida Filtro: 300
Dato in: 250
Salida Filtro: 275
Dato in: 300
Salida Filtro: 283
Dato in: 200
Salida Filtro: 262
Dato in: 250
Salida Filtro: 260
Dato in: 350
Salida Filtro: 275
```

```
300 / 1 = 300
(300 + 250) / 2 = 275
(300 + 250 + 300) / 3 = 283
(300 + 250 + 300 + 200) / 4 = 262
...
```

Prueba señal de Reset

```
Dato in: 350
Salida Filtro: 275
Dato in: 400
Salida Filtro: 292
Dato in: 600
Salida Filtro: 331
Dato in: 500
Salida Filtro: 350
Dato in: 700
Salida Filtro: 385
------------------------------------------------
Forzamos el reseteo del filtro
------------------------------------------------
Dato in: 100
Salida Filtro: 100
Dato in: 200
Salida Filtro: 150
Dato in: 300
Salida Filtro: 200
Dato in: 400
Salida Filtro: 250
Dato in: 300
Salida Filtro: 260
Dato in: 350
Salida Filtro: 275
Dato in: 400
Salida Filtro: 292
Dato in: 450
Salida Filtro: 312
Dato in: 400
```

```
100 / 1 = 100
(100 + 200) / 2 = 150
(100 + 200 + 300) / 3 = 200
(100 + 200 + 300 + 400) / 4 = 250
...
```

```
FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, RST_REG, 1);
xil_printf("------------------------------------------------\r\n");
xil_printf("Forzamos el reseteo del filtro \r\n");
xil_printf("------------------------------------------------\r\n");

FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, CLK_FLAG);
FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, 0);

FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, RST_REG, 0);


for(i = 0; i < cant/2; i++)
{
    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, IN_REG, datos_in[i]);

    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, CLK_FLAG);

    xil_printf("Dato in: %d \r\n", datos_in[i]);

    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, 0);

    res = FILTROVENTANA_IP_mReadReg(BASE_ADDRESS, OUT_REG);

    xil_printf("Salida Filtro: %d \r\n", res);
}
```

Prueba señal de Enable

```
Dato in: 1400
Salida Filtro: 1250
-----------------------------------------------
Bajamos la seal de Enable
-----------------------------------------------
Dato in: 1350
Salida Filtro: 1250
Dato in: 1400
Salida Filtro: 1250
Dato in: 1450
Salida Filtro: 1250
Dato in: 1400
Salida Filtro: 1250
Dato in: 300
Salida Filtro: 1250
Dato in: 250
Salida Filtro: 1250
Dato in: 300
Salida Filtro: 1250
Dato in: 200
Salida Filtro: 1250
Dato in: 250
Salida Filtro: 1250
Dato in: 350
Salida Filtro: 1250
Dato in: 400
Salida Filtro: 1250
Dato in: 600
Salida Filtro: 1250
Dato in: 500
Salida Filtro: 1250
Dato in: 700
Salida Filtro: 1250
Fin de la prueba
```

```
FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, EN_REG, 0);
xil_printf("-----------------------------------------------\r\n");
xil_printf("Bajamos la señal de Enable \r\n");
xil_printf("-----------------------------------------------\r\n");

for(i = cant/2; i < cant; i++)
{
    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, IN_REG, datos_in[i]);

    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, CLK_FLAG);

    xil_printf("Dato in: %d \r\n", datos_in[i]);

    FILTROVENTANA_IP_mWriteReg(BASE_ADDRESS, CLK_REG, 0);

    res = FILTROVENTANA_IP_mReadReg(BASE_ADDRESS, OUT_REG);

    xil_printf("Salida Filtro: %d \r\n", res);
}
```

¿Preguntas?