.UBAfiuba
FACULTAD DE INGENIERÍA

Trabajo Práctico Final

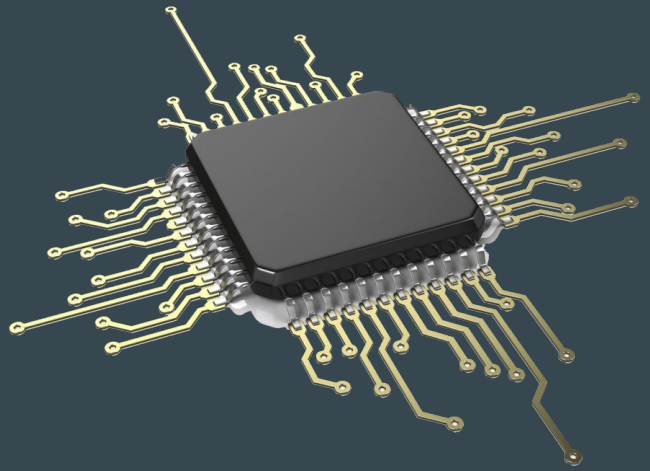
Microarquitecturas y Softcores

•••

Especialización en sistemas embebidos - 15va Cohorte

Alumno: Pablo Javier Morzán
Fecha: 13/06/2022

Introducción



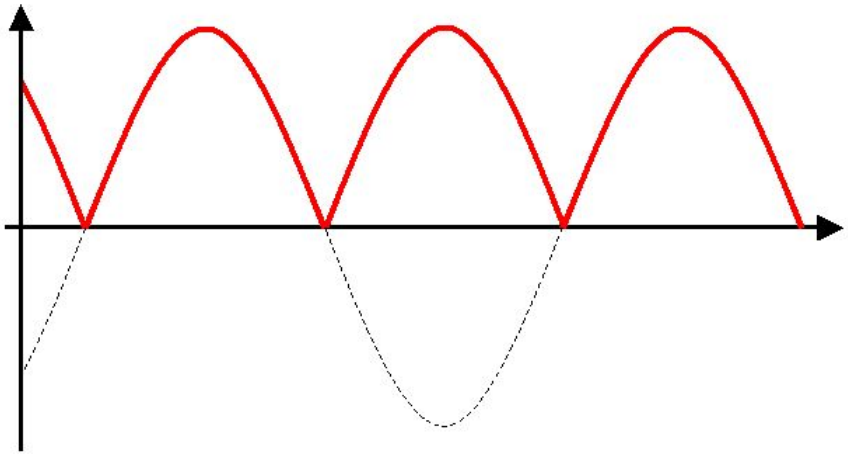
- **Objetivos:**
  - Aplicación de conceptos de la asignatura
  - Familiarización con software utilizado
  - Experiencia de instancia de exposición grupal.

- **Entorno utilizado:**
  - Software: VSCode, Xilinx SDK, GTK Wave y Vivado.
  - Hardware: Arty Z7-10 (Conexión Remota)

Descripción del proyecto

Se utiliza un filtro de promedios móviles para calcular el valor eficaz de una señal senoidal.

Nota: Se contempla la señal ya rectificada y multiplicada por 1.111



$$Vef = Vmed * (\pi / 2\sqrt{2}) = Vmed * 1.111$$

Código VHDL : generador de datos

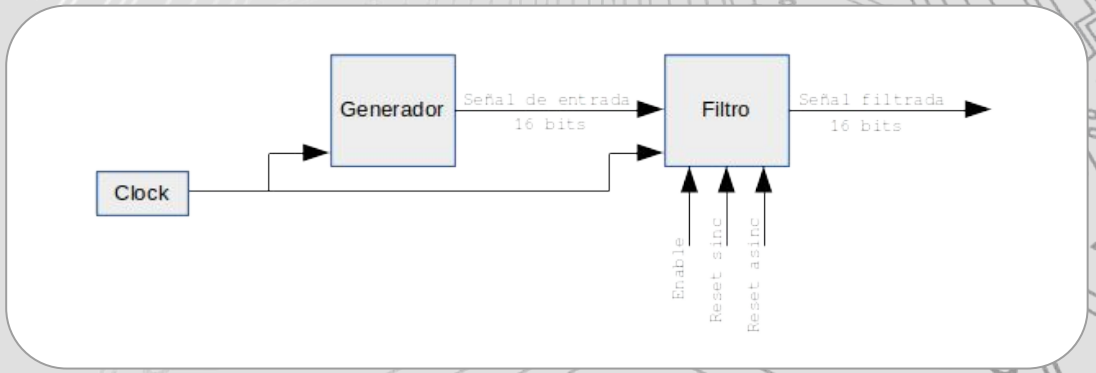
```
begin


    rom : process (clk)
    variable i: natural := 0;
    variable j: natural := 0;
    begin
        if rising_edge(clk) then

            if (3 = j) then
                ena_fil <= '1';
                data_out <=mem(i);
                if i = 15 then       -- 15 hardco
                    i := 0;
                else
                    i := i + 1;
                end if;
                j:=0;
            else
                ena_fil <= '0';
            end if;

        j:=j+1;
        end if;
    end process rom;

end architecture synth;
```

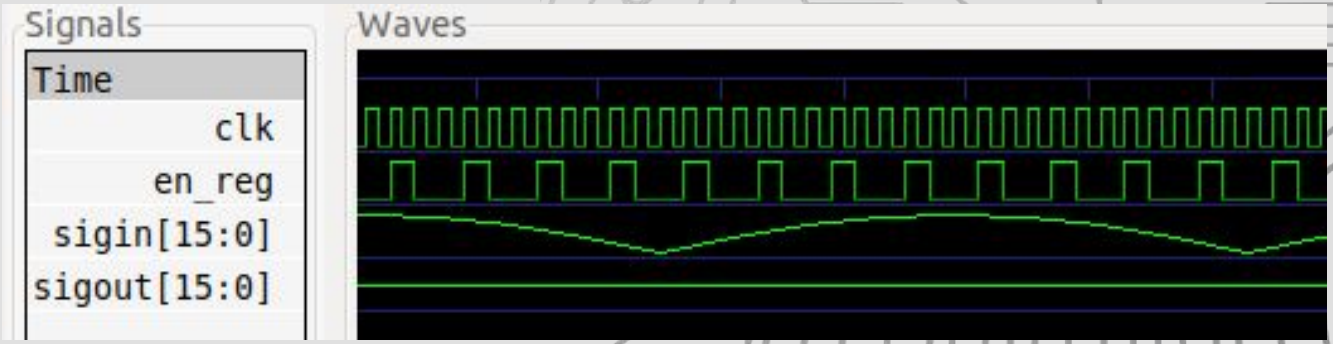
```
constant mem : mem_type :=
    (0 => X"0000" ,
    1 => X"348e" ,
    2 => X"611b" ,
    3 => X"7ee0" ,
    4 => X"8955" ,
    5 => X"7ee0" ,
    6 => X"611b" ,
    7 => X"348e" ,
    8 => X"0000" ,
    9 => X"348e" ,
    10 => X"611b" ,
    11 => X"7ee0" ,
    12 => X"8955" ,
    13 => X"7ee0" ,
    14 => X"611b" ,
    15 => X"348e");
```

Código VHDL : Filtro MA

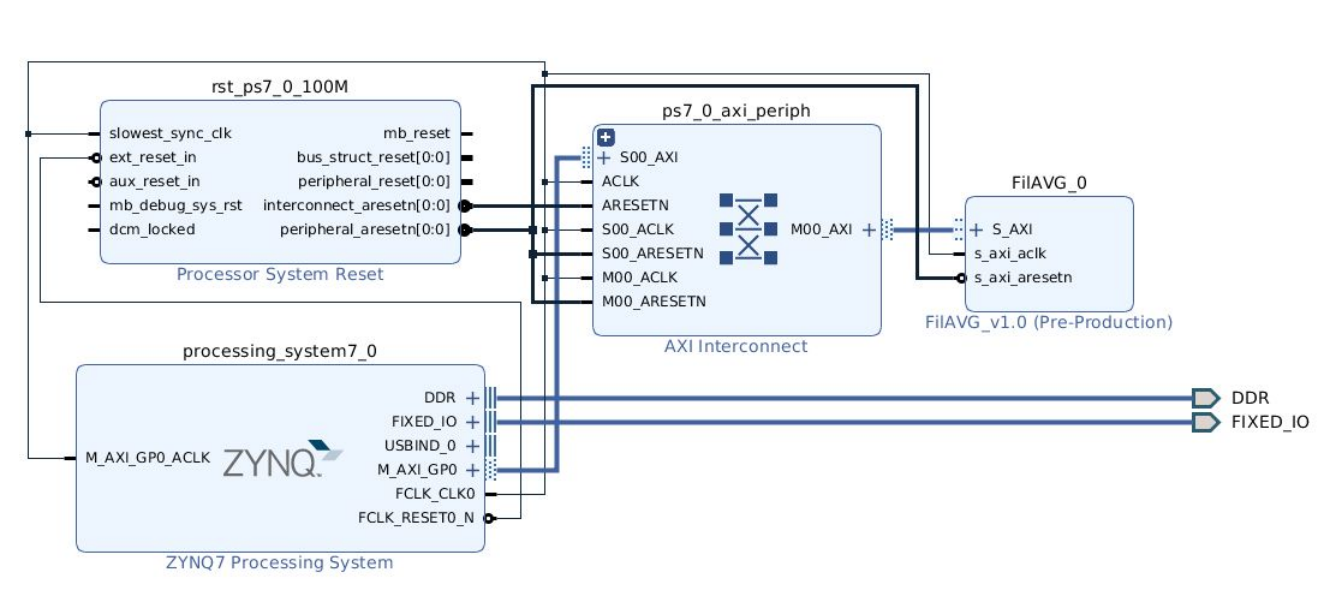
```
begin
    if (a_rst = '1') then
        en_reg <= '0';
        fil_buff <= (others => (others => '0'));
        sum_aux := (others => '0');
        sigout <= (others => '0');
    elsif rising_edge(clk) then
        if (s_rst = '1') then
            en_reg <= '0';
            fil_buff <= (others => (others => '0'));
            sum_aux := (others => '0');
            sigout <= (others => '0');
        else
            en_reg <= ena;
            en_reg_aux2 := en_reg_aux;
            en_reg_aux := en_reg;

            if (not en_reg_aux2 and en_reg_aux)='1' then
                fil_buff(0) <= unsigned(sigin);
                for i in 1 to fil_w-1 loop
                    fil_buff(i) <= fil_buff(i-1);
                end loop;
                sum_aux := (others => '0');
                for i in 0 to fil_w-1 loop
                    sum_aux := sum_aux + resize(fil_buff(i), sum_aux'length);
                end loop;
            sigout <= std_logic_vector(sum_aux(sig_w+bitadd(fil_w)-1 downto bitadd(fil_w)));
            end if;
        end if;
    end if;
end process;
end architecture;
```

Simulación GTK Wave



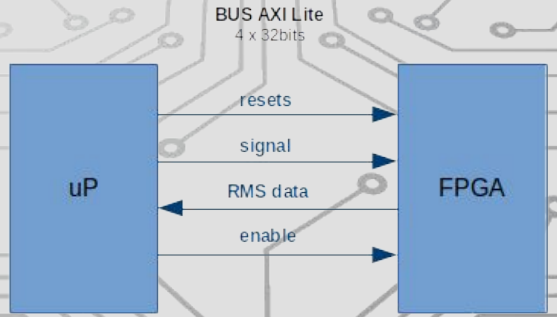


Creación de IP Core y conexión con sistema de procesamiento



Código de enlace AXI de IP Core

```
component FilAvg is
    generic (
        sig_w : integer := 16;
        fil_w : integer := 8
    );
    port (
        clk     : in std_logic;
        a_rst   : in std_logic;
        s_rst   : in std_logic;
        ena     : in std_logic;
        sigin   : in std_logic_vector(sig_w-1 downto 0);
        sigout  : out std_logic_vector(sig_w-1 downto 0)
    );
end component;

constant sig_w_aux: natural := 16;
constant fil_w_aux: natural := 8;
signal salFil: std_logic_vector(31 downto 0);
```




```
process (slv_reg0, slv_reg1, salFil, slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
    -- Address decoding for reading registers
    loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
    case loc_addr is
      when b"00" =>
        reg_data_out <= slv_reg0;
      when b"01" =>
        reg_data_out <= slv_reg1;
      when b"10" =>
        reg_data_out <= salFil;
      when b"11" =>
        reg_data_out <= slv_reg3;
      when others =>
        reg_data_out  <= (others => '0');
    end case;
end process;
```

```
    -- Add user logic here
FilAvgInst: FilAvg
    generic map(
        sig_w   => sig_w_aux,
        fil_w   => fil_w_aux
    )
    port map(
        clk     => S_AXI_ACLK,
        a_rst   => slv_reg0(0),
        s_rst   => slv_reg0(1),
        ena     => slv_reg3(0),
        sigin   => slv_reg1(15 downto 0),
        sigout  => salFil(15 downto 0)
    );
    -- User logic ends
```

Programación de sistema de procesamiento : Declaración de variables

```c
typedef union {
    struct {
        uint8_t b0:1;
        uint8_t b1:1;
        uint32_t baux:30;
    };
    struct {
        uint32_t bitsData:32;
    };
} Bits_t;

typedef union {
    struct {
        uint16_t signal:16;
        uint16_t sigaux:16;
    };
    struct {
        uint32_t signalData:32;
    };
} Signal_t;

//tabla datos de señal senoidal rectificada
uint16_t seno16[] = {0x0000 ,0x348e ,0x611b ,0x7ee0 ,0x8955 ,0x7ee0 ,0x611b ,0x348e ,0x0000 ,0x348e ,0x611b ,0x7ee0 ,0x8955 ,0x7ee0 ,0x611b ,0x348e};

//valor de entrada para múltiplicar la señal
uint8_t valor = 1;

Bits_t reset_data = {.bitsData=0};
Bits_t enable_data = {.bitsData=0};
Signal_t input_data = {.signalData=0};
Signal_t output_data = {.signalData=0};
```

Programación de sistema de procesamiento : Declaración de variables

```
while(1){
    //setear enable
    enable_data.b0 = 1;
    FILAVG_mWriteReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG3_OFFSET, enable_data.bitsData);

    //enviar un dato
    input_data.signal = (uint16_t)(seno16[i]/div);
    FILAVG_mWriteReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG1_OFFSET, input_data.signalData);

    //sacar dato
    uint32_t res = FILAVG_mReadReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG2_OFFSET);
    xil_printf("Valor RMS: %d V\r\n", res/100);

    //borrar enable
    enable_data.b0 = 0;
    FILAVG_mWriteReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG3_OFFSET, enable_data.bitsData);

    i = (i + 1) % 16;

    //reset cada 20 muestras
    count--;
    if(count==0){
        count = 20;
        reset_data.b0 = 1;

        FILAVG_mWriteReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG0_OFFSET, reset_data.bitsData);
        usleep(5000);
        reset_data.b0 = 0;
        FILAVG_mWriteReg(XPAR_FILAVG_0_S_AXI_BASEADDR, FILAVG_S_AXI_SLV_REG0_OFFSET, reset_data.bitsData);
        xil_printf("Reset...\r\n");
    }

    sleep(1);

}
```

Resultados


artyz7-user00@lse-server-pc:~
Archivo Editar Ver Buscar Terminal Ayuda
OPCIONES: I18n
Compilado en Jul 28 2020, 00:00:00.
Port /dev/ArtyZ7-Board01, 07:07:36
Presione CTRL-A Z para obtener ayuda sobre teclas especiales
-- Inicio del programa.... --
-- Para voltaje 220V presione A
-- Para voltaje 110V presione otro
Valor ingresado: 220 V
Valor RMS: 0 V
Valor RMS: 0 V
Valor RMS: 0 V
Valor RMS: 16 V
Valor RMS: 47 V
Valor RMS: 88 V
Valor RMS: 132 V
Valor RMS: 173 V
Valor RMS: 204 V
Valor RMS: 220 V
Valor RMS: 220 V
Valor RMS: 220 V


artyz7-user00@lse-server-pc:~
Archivo Editar Ver Buscar Terminal Ayuda
-- Inicio del programa.... --
-- Para voltaje 220V presione A
-- Para voltaje 110V presione otro
Valor ingresado: 110 V
Valor RMS: 0 V
Valor RMS: 0 V
Valor RMS: 0 V
Valor RMS: 8 V
Valor RMS: 23 V
Valor RMS: 44 V
Valor RMS: 66 V
Valor RMS: 86 V
Valor RMS: 102 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V
Valor RMS: 110 V


¿PREGUNTAS?