UNIVERSIDAD CATÓLICA DE CÓRDOBA

Laboratorio AD_SAR

Electrónica Digital

Nieves Aramayo, Pablo Agustín

16/06/2020

Profesor: Agustín Laprovitta

Documento con el desarrollo del laboratorio del conversor analógico a digital usando la herramienta Quartus 2 con código descriptivo VHDL.

Índice

Índice 1
AD_SAR: Elementos 2
Modulo: saadc_fsm 2
Modulo: shiftreg 3
Modulo: approx_reg 4
Modulo: SAR 5
Modulo: latch 6
Modul: AD_SAR 7

AD_SAR: Elementos

El AD_SAR está compuesto de dos grandes bloques, el SAR y el latch. Mientras que el latch puede ser descripto fácilmente como un único elemento el SAR está compuesto de varios elementos internos. Estos últimos son: saadc_fsm, shiftreg, approx_reg, y una OR. La OR estará implementada en el mismo VHDL que se juntaran todos los elementos del SAR, y los demás tendrán su propio VHDL que serán módulos que el SAR llamara.

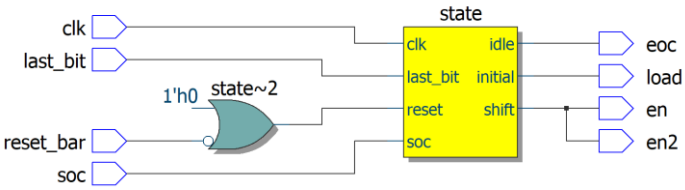
En los siguientes puntos se mostrara el código VHDL, junto con su RTL y después su test bench.

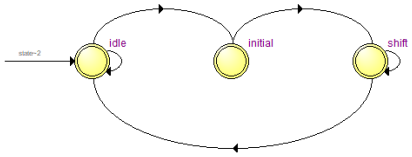
Modulo: saadc_fsm

Codigo:

```
library ieee;
use ieee.std_logic_1164.all;

entity saadc_fsm is

   port(
      clk         : in  std_logic;
      reset_bar   : in  std_logic;
      soc         : in  std_logic;
      last_bit    : in  std_logic;
      load        : out std_logic;
      en          : out std_logic;
      en2         : out std_logic;
      eoc         : out std_logic
   );

end entity;

architecture rtl of saadc_fsm is


   type state_type is (idle, initial, shift);

   signal state   : state_type;

begin


   process (clk, reset_bar, last_bit, soc)
   begin
      if reset_bar = '0' then
         state <= idle;
      elsif (rising_edge(clk)) then
         case state is
            when idle=>
               if soc = '1' then
                  state <= initial;
               else
                  state <= idle;
               end if;
            when initial=>
                  state <= shift;
            when shift=>
               if last_bit = '1' then
                  state <= idle;
               else
                  state <= shift;
               end if;
         end case;
      end if;
   end process;


   process (state)
   begin
      case state is
         when idle =>
            en    <=    '0';
            en2   <=    '0';
            eoc   <=    '1';
            load  <=    '0';
            when initial =>
            en    <=    '0';
            en2   <=    '0';
            eoc   <=    '0';
            load  <=    '1';
         when shift =>
            en    <=    '1';
            en2   <=    '1';
            eoc   <=    '0';
            load  <=    '0';
      end case;
   end process;

end rtl;
```

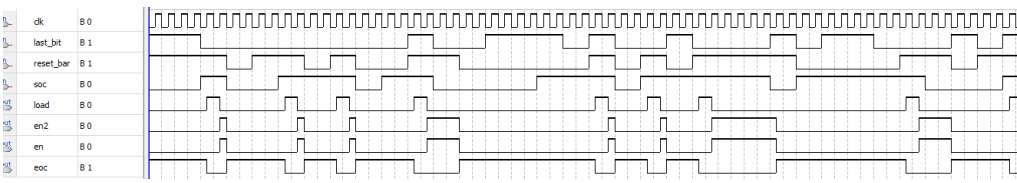
RTL:





Máquina de estado

Test Bench:

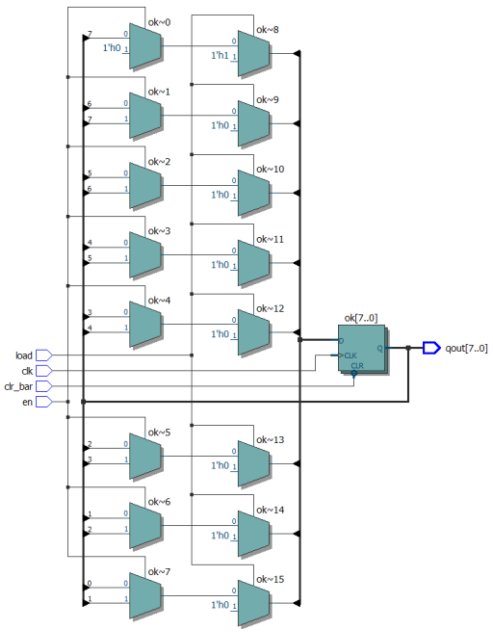


Modulo: shiftreg

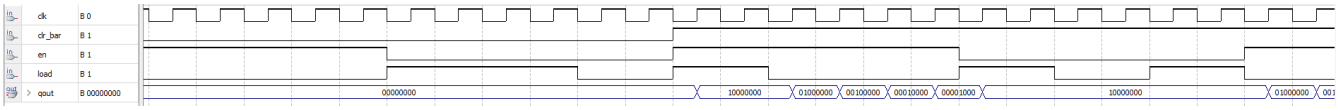
Codigo VHDL:

```
library ieee;
use ieee.std_logic_1164.all;

entity shiftreg is

    port(
        clk         : in  std_logic;
        clr_bar     : in  std_logic;
        load        : in  std_logic;
        en          : in  std_logic;
        qout        : out std_logic_vector ( 7 downto 0)
    );

end entity;

architecture move of shiftreg is


 -- type state_type is (set, shift, reset);

 -- signal state   : state_type;

begin

    process (clk, clr_bar, load)
        variable ok:std_logic_vector(7 downto 0);
    begin
    if (clr_bar='0') then
          ok:="00000000";--state<=reset;
       elsif (rising_edge(clk)) then
          if (load = '1') then
             ok := "10000000";--state<=set;
          elsif (en= '1') then
                for i in 0 to 6 loop
                      ok(i):=ok(i+1);
                end loop;--state<=shift;
                ok(7) := '0';
          end if;
       end if;
    qout<=ok;
    end process;


end architecture;
```

RTL:

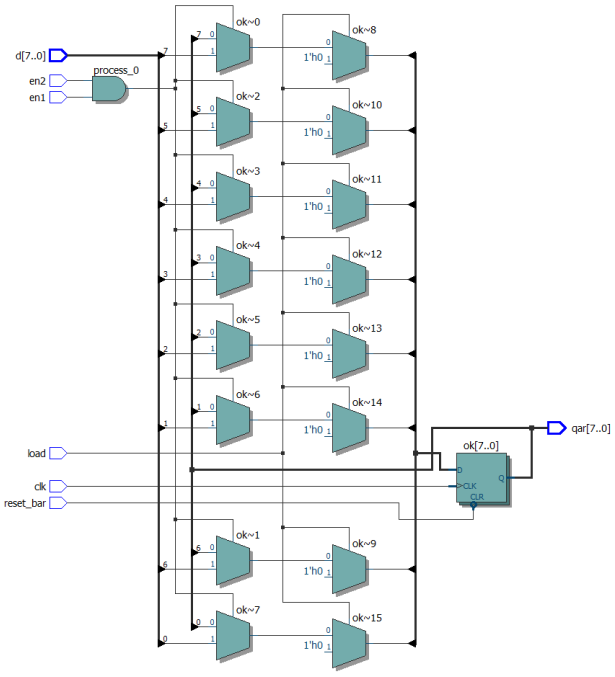


Test Bench:

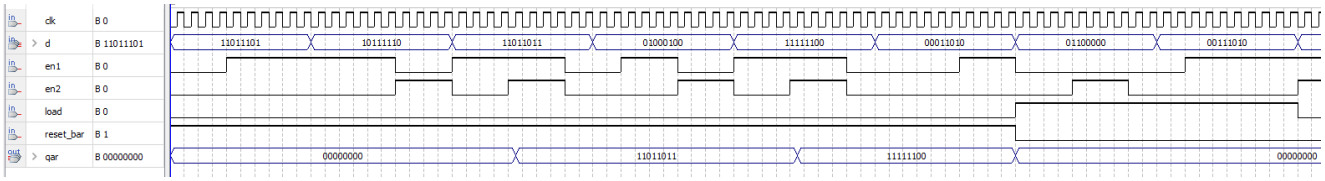


Modulo: approx_reg

```
library ieee;
use ieee.std_logic_1164.all;
entity approx_reg is

   port(
      reset_bar,load,en2,en1,clk : in std_logic;
      d:in std_logic_vector(7 downto 0);
      qar : out std_logic_vector(7 downto 0)
      );
end entity;

architecture registro of approx_reg is


-- type state_type is (set, shift, reset);

-- signal state   : state_type;

begin

   process (clk, reset_bar, load)
         variable ok:std_logic_vector(7 downto 0);
   begin

   if (reset_bar='0') then
         ok:="00000000";--state<=reset;
      elsif (rising_edge(clk)) then
         if (load = '1') then
            ok := "00000000";--state<=set;
            elsif (en1='1' AND en2='1') then
                  ok:=d;
         end if;
      end if;
   qar<=ok;
   end process;


end architecture;
```



Test Bench:

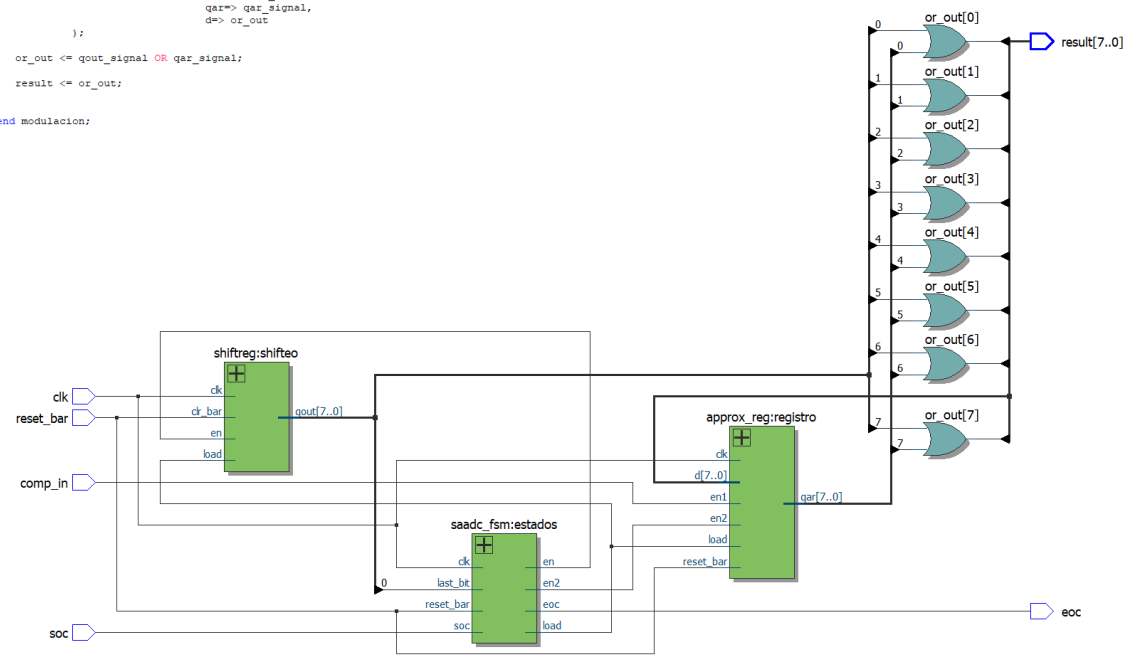


Modulo: SAR

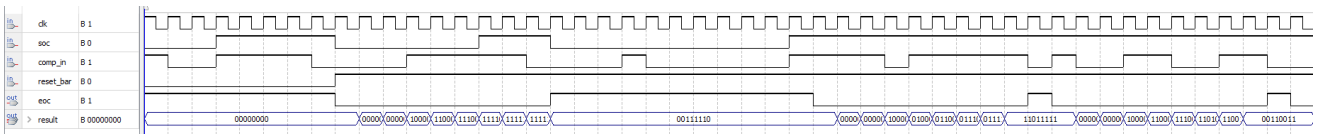
Código VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
entity SAR is

    port(
        reset_bar,soc,comp_in,clk : in std_logic;
        eoc:out std_logic;
        result : out std_logic_vector(7 downto 0)
        );
end entity;

architecture modulacion of SAR is
    signal load_signal      : std_logic;
    signal en_signal        : std_logic;
    signal en2_signal       : std_logic;
    signal qout_signal      : std_logic_vector (7 downto 0);
    signal qar_signal       : std_logic_vector (7 downto 0);
    signal or_out           : std_logic_vector (7 downto 0);

begin
    estados: entity work.saadc_fsm port map(  clk => clk,
                                reset_bar => reset_bar,
                                soc => soc,
                                last_bit => qout_signal(0),
                                load => load_signal,
                                en => en_signal,
                                en2 => en2_signal,
                                eoc => eoc
            );

    shifteo: entity work.shiftreg port map(   clk => clk,
                                clr_bar => reset_bar,
                                load => load_signal,
                                en => en_signal,
                                qout=> qout_signal
            );

    registro: entity work.approx_reg port map(clk => clk,
                                reset_bar => reset_bar,
                                load => load_signal,
                                en1 => comp_in,
                                en2 => en2_signal,
                                qar=> qar_signal,
                                d=> or_out
            );

    or_out <= qout_signal OR qar_signal;

    result <= or_out;

end modulacion;
```

RTL:



Test Bench:

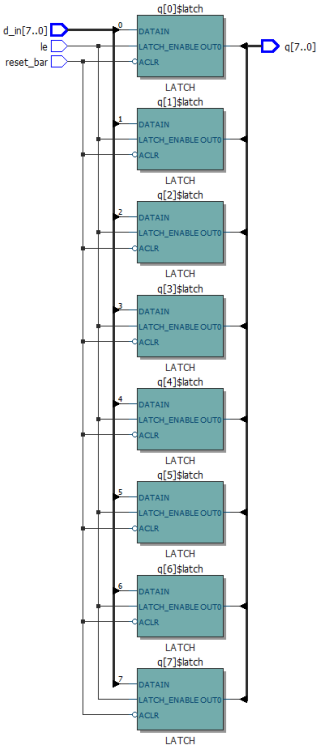


Modulo: latch

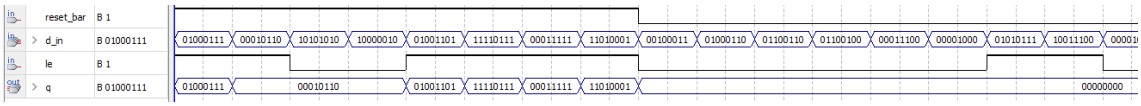
Codigo VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
entity xlatch is

   port(
      reset_bar,le: in std_logic;
      d_in  : in std_logic_vector(7 downto 0);
      q : out std_logic_vector(7 downto 0)
      );
end entity;

architecture save of xlatch is
begin
   process(reset_bar,le)
      begin
         if (reset_bar='0') THEN
            q <= "00000000";
         elsif (le = '1') then
            q <= d_in;
         end if;
   end process;
end save;
```

RTL:



Test Bench:

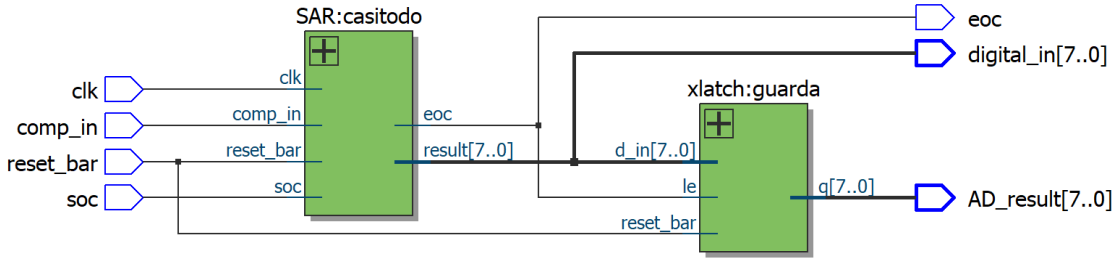


Modul: AD_SAR

Código VHDL:

```
library ieee;
use ieee.std_logic_1164.all;

entity AD_SAR is

   port(
      reset_bar,soc,comp_in,clk : in std_logic;
      eoc:out std_logic;
      AD_result,digital_in : out std_logic_vector(7 downto 0)
      );
end entity;

architecture todo of AD_SAR is

   signal eoc_signal          : std_logic;
   signal result_signal       : std_logic_vector (7 downto 0);

begin
   casitodo:   entity work.SAR port map(  clk => clk,
                                          reset_bar => reset_bar,
                                          soc => soc,
                                          eoc => eoc_signal,
                                          comp_in => comp_in,
                                          result=> result_signal
            );

   guarda: entity work.xlatch port map(   reset_bar => reset_bar,
                                          d_in => result_signal,
                                          le => eoc_signal,
                                          q => AD_result
           );

   eoc<=eoc_signal;

   digital_in<=result_signal;

end todo;
```

RTL:



Test Bench: