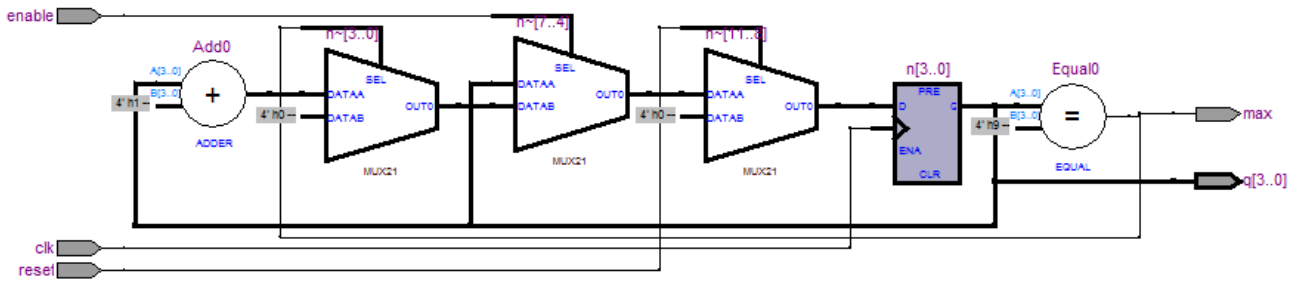
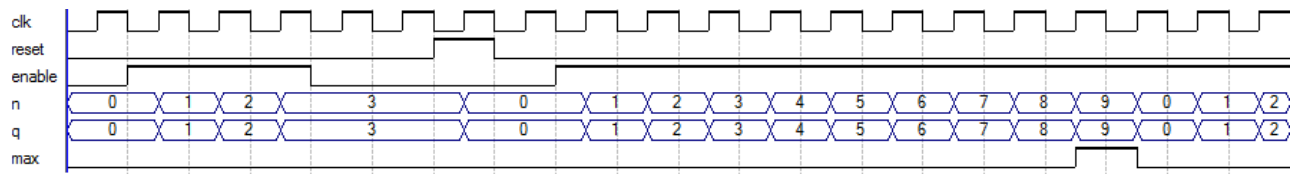
DISEÑO DE UN CRONÓMETRO PARA EXPERIMENTOS DE CINEMÁTICA

Contador BCD

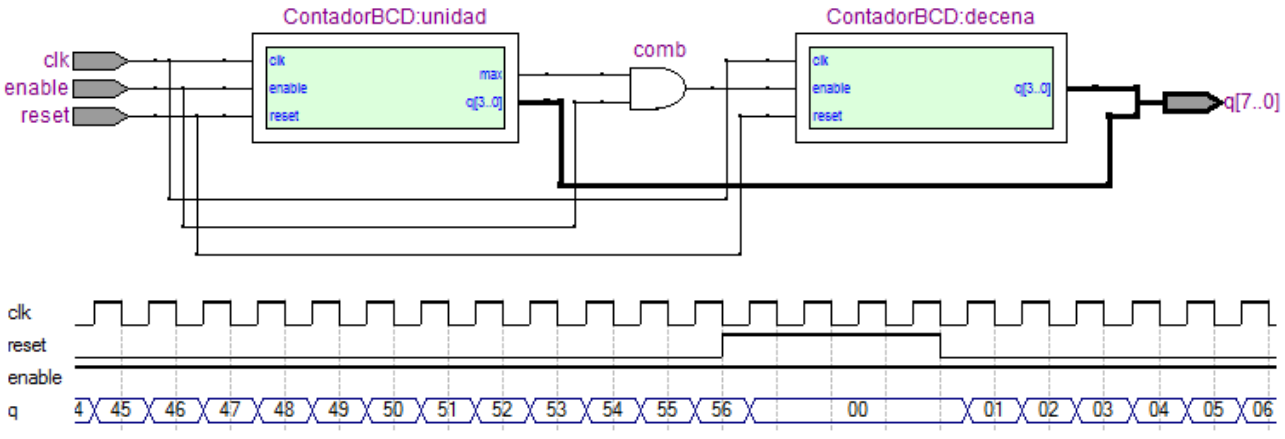
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ContadorBCD is
    port
    (   clk       : in std_logic;
        reset     : in std_logic;
        enable    : in std_logic;
        max       : out std_logic;
        q         : out std_logic_vector(3 downto 0)
    );
end entity;

architecture rtl of ContadorBCD is
    signal  n    : std_logic_vector(3 downto 0);
begin

    process (clk)
    begin
        if (rising_edge(clk)) then
            if reset = '1' then
                n <= "0000";
            elsif enable = '1' then
                if n = "1001" then
                    n <= "0000";
                else
                    n <= n + 1;
                end if;
            end if;
        end if;
    end process;

    q <= n;
    max <= '1' when n = "1001" else '0';

end rtl;
```





Contador BCD de dos dígitos

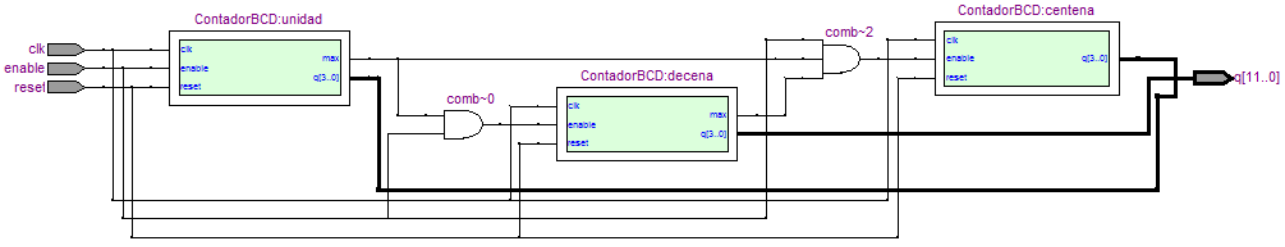
```
entity Contador2digitosBCD is
    port
    (   clk       : in std_logic;
        reset     : in std_logic;
        enable    : in std_logic;
        q         : out std_logic_vector(7 downto 0)
    );
end entity;

architecture rtl of Contador2digitosBCD is
    signal  hab, x : std_logic;
    component ContadorBCD
        port
        (   clk       : in std_logic;
            reset     : in std_logic;
            enable    : in std_logic;
            max       : out std_logic;
            q         : out std_logic_vector(3 downto 0)
        );
    end component;
begin

    unidad: ContadorBCD
    port map
    (   clk => clk,
        reset => reset,
        enable => enable,
        max => hab,
        q => q(3 downto 0)
    );

    decena: ContadorBCD
    port map
    (
        clk => clk,
        reset => reset,
        enable => hab and enable,
        max => x,
        q => q(7 downto 4)
    );

end rtl;
```



Contador BCD de tres dígitos

```
architecture rtl of Contador3digitosBCD is
    signal  hab1, hab2: std_logic;
    component ContadorBCD
        port
        (
            clk       : in std_logic;
            reset     : in std_logic;
            enable    : in std_logic;
            max       : out std_logic;
            q         : out std_logic_vector(3 
        );
    end component;
begin
    unidad: ContadorBCD
    port map
    (   clk => clk,
        reset => reset,
        enable => enable,
        max => hab1,
        q => q(3 downto 0)
    );
    decena: ContadorBCD
    port map
    (   clk => clk,
        reset => reset,
        enable => hab1 and enable,
        max => hab2,
        q => q(7 downto 4)
    );
    centena: ContadorBCD
    port map
    (   clk => clk,
        reset => reset,
        enable => hab1 and hab2 and enable,
        q => q(11 downto 8)
    );

end rtl;
```

```
entity LatchSR is
    port
    (
        set       : in std_logic;
        reset     : in std_logic;
        q         : out std_logic
    );

end entity;

architecture rtl of LatchSR is
begin

    process (set, reset)
    begin
        if set = '0' then
            q <= '1';
        elsif reset = '0' then
            q <= '0';
        end if;
    end process;

end rtl;
```

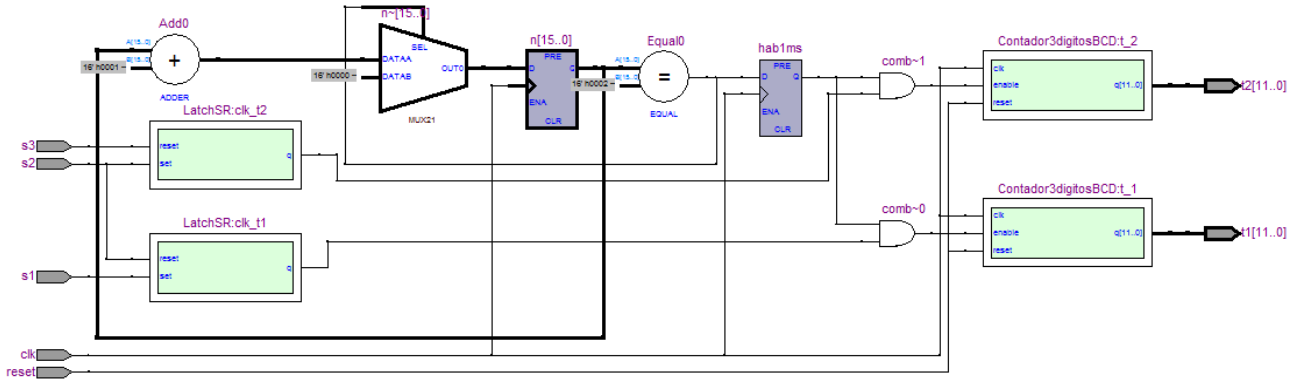


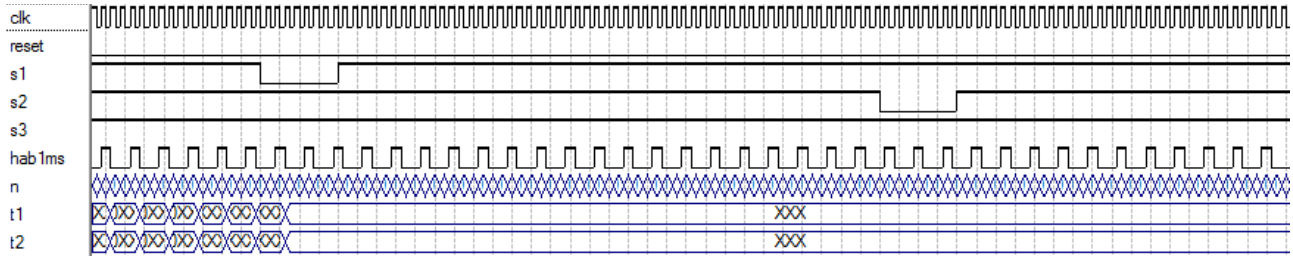
Cronómetro con Latches

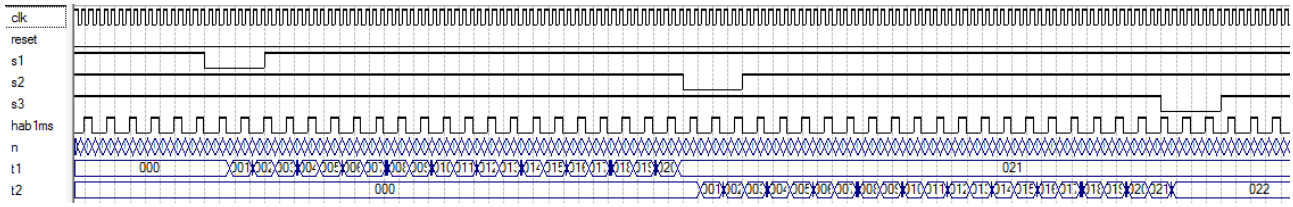
```
entity Aceleracion is
port(
    clk,
    reset       : in std_logic;
    s1, s2, s3  : in std_logic;
    t1, t2      : out std_logic_vector(11 downto 0)
    );
end entity;

architecture rtl of Aceleracion is
    signal  hab1ms, habT1, habT2: std_logic;
    component Contador3digitosBCD
        port(
        clk       : in std_logic;
        reset     : in std_logic;
        enable    : in std_logic;
        q         : out std_logic_vector(11 downto 0)
        );
    end component;

    component LatchSR is
        port
        (   set       : in std_logic;
            reset     : in std_logic;
            q         : out std_logic
        );
    end component;
begin

    -- Habilitador de 1KHz (T = 1ms)
    -- a partir de un reloj de 50MHz
    process(clk)
        variable n: integer range 0 to 50000 - 1;
    begin
        if rising_edge(clk) then
            --if n = 50000 - 1 then -- síntesis
            if n = 3 - 1 then -- simulación
                n := 0;
                hab1ms <= '1';
            else
                n := n + 1;
                hab1ms <= '0';
            end if;
        end if;
    end process;

    clk_t1: LatchSR port map
    (set => s1, reset => s2, q => habT1);

    t_1: Contador3digitosBCD
    port map (  clk => clk, reset => reset,
                enable => habT1 and hab1ms,
                q => t1);

    clk_t2: LatchSR port map(s2, s3, habT2);

    t_2: Contador3digitosBCD
    port map
    (   clk => clk,
        reset => reset,
        enable => habT2 and hab1ms,
        q => t2);
end rtl;
```

```
architecture fsm of Aceleracion is
    signal  hab1ms, habT1, habT2: std_logic;
    component Contador3digitosBCD
        port
        (   clk       : in std_logic;
            reset     : in std_logic;
            enable    : in std_logic;
            q         : out std_logic_vector(11 downto 0));
    end component;

    type ESTADOS is (ESPERA, ENT1, ENT2, MUESTRA);
    signal estado: ESTADOS;
begin
    process(clk)
        variable n: integer range 0 to 50000 - 1;
    begin
        if rising_edge(clk) then
            --if n = 50000 - 1 then -- síntesis
            if n = 3 - 1 then -- simulación
                n := 0;
                hab1ms <= '1';
            else
                n := n + 1;
                hab1ms <= '0';
            end if;
        end if;
    end process;

    t_1: Contador3digitosBCD
    port map (clk => clk, reset => reset,
        enable => habT1 and hab1ms, q => t1);

    t_2: Contador3digitosBCD
    port map (clk => clk, reset => reset,
        enable => habT2 and hab1ms, q => t2);

    process (clk, reset)
    begin
        if reset = '1' then
            estado <= ESPERA;
        elsif (rising_edge(clk)) then
            case estado is
                when ESPERA =>  if s1 = '0' then
                                    estado <= ENT1;
                                end if;
                when ENT1 =>    if s2 = '0' then
                                    ESTADO <= ENT2;
                                end if;
                when ENT2 =>    if s3 = '0' then
                                    ESTADO <= MUESTRA;
                                end if;
                when MUESTRA => ESTADO <= ESPERA;
            end case;
        end if;
    end process;

    habT1 <= '1' when estado = ENT1 else '0';
    habT2 <= '1' when estado = ENT2 else '0';

end fsm;
```

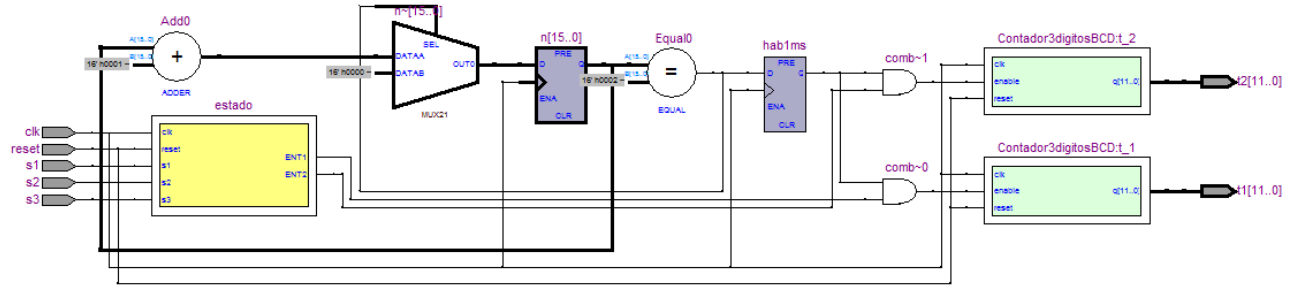
Con Latches

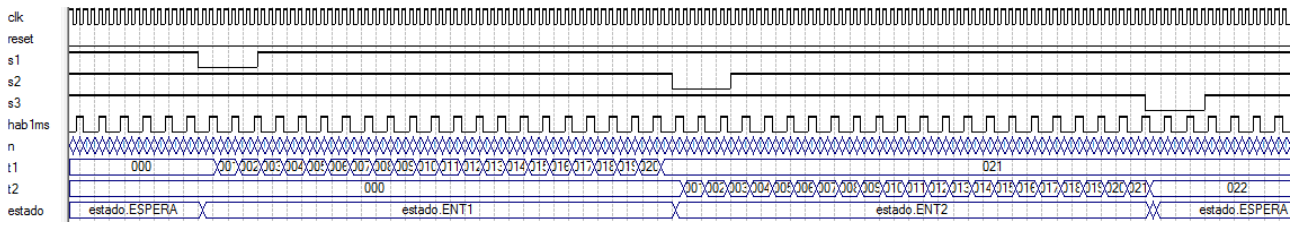






Con FSM





Programa en C para Arduino Uno

```
unsigned long a, b;
int bUno = LOW;
int bDos = LOW;
int bTres = LOW;

int uno = 7;
int dos = 4;
int tres = 2;
int reset = 8;

void setup(){
  Serial.begin(9600);
}

void loop()
{
    if (digitalRead(uno) == HIGH)
    {
      bUno = LOW;
      bDos = HIGH;
      bTres = HIGH;
    }
    if (digitalRead(uno) == LOW and bUno == LOW)
    {
      bUno = HIGH;
      bDos = LOW;
      bTres = LOW;
      a = millis();
    }
    if (digitalRead(dos) == LOW and bDos == LOW)
    {
      bDos = HIGH;
      b = millis();
      Serial.print("T1 ");
      Serial.println(b - a);
    }
    if (digitalRead(tres) == LOW and bTres == LOW)
    {
      bTres = HIGH;
      bUno = LOW;
      a = millis();
      Serial.print("T2 ");
      Serial.println(a - b);
    }
```