Hw #11 { Relojes en tiempo real }
Lenguajes de descripción de hardware

Hernández Castellanos César Uriel

1 Sección uno

Los relojes de tiempo real son circuitos que proporcionan la hora exacta en un circuito digital, (RTC por sus siglas en inglés "Real Time Clock"). El nombre se utiliza para diferenciarlos de las señales de reloj usadas en circuitos secuenciales. Los relojes de tiempo real pueden ser alimentados con baterías e independientemente de la fuente de alimentación principal de los circuitos, lo que garantiza que conserven su información, aunque falle la alimentación general. Los RTC son usados en todas las aplicaciones que requieran conocer la hora exacta en que sucede un evento

Recordar el contador con clock enable y carga de datos.

1.1 VHDL

```
process(clock)
begin
if (reset = '1') then
 cuenta <= (others => '0');
elsif (rising_edge(clock)) then
if (clock_en = '1') then
 if (load_en = '1') then
  cuenta <= entrada;
  else
  cuenta <= cuenta + 1;
end if;
end if;
end if;
end process;
```

1.2 Verilog

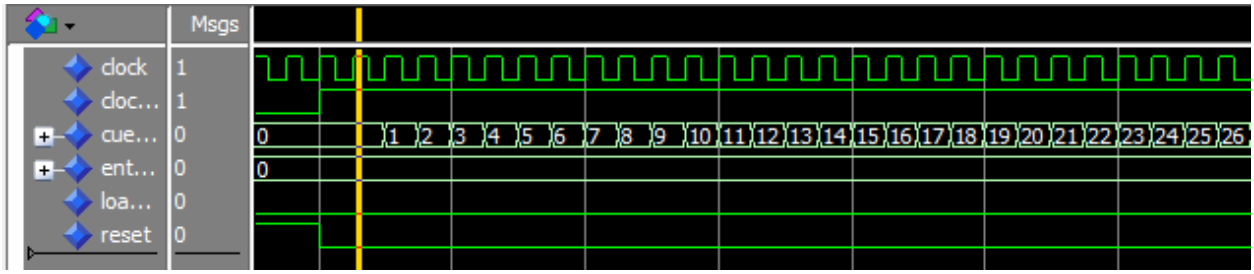
```
always @(posedge clock)
if (reset)
 cuenta <= 0;
else if (clock_en)
 if (load_en)
  cuenta <= entrada;
 else
  cuenta <= cuenta + 1;
```

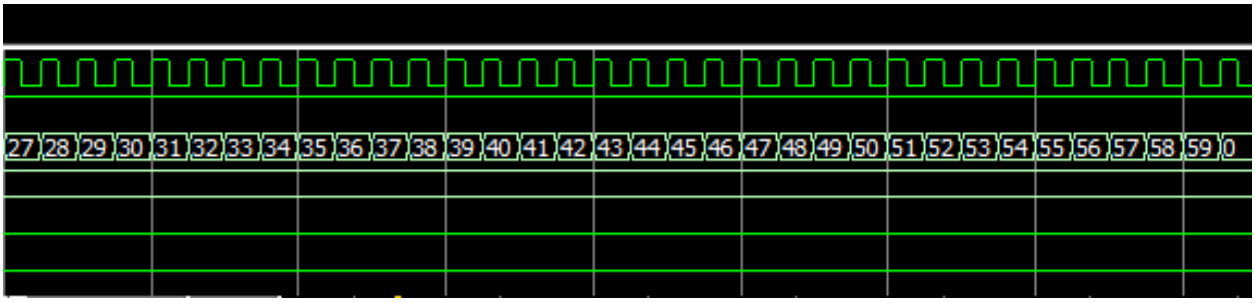
Modificar el código anterior en el lenguaje que seleccionen, para que exista un límite en el conteo, es decir, normalmente un contador de 6 bits contará de 0 a 63, pero si se quisieran contar segundos deberá contar sólo de 0 a 59. Simularlo. ¿Por qué es deseable que en esta aplicación el reset sea síncrono?

```
process(clock)
begin
if (reset = '1') then
 tmp <= (others => '0');
 cuenta<=tmp;
elsif (rising_edge(clock)) then
 if (clock_en = '1') then
  if (load_en = '1') then
   tmp <= entrada;
   cuenta<=tmp;
  else
    if(conv_integer(tmp) > N-1) then
         tmp <= (others => '0');
         cuenta<=tmp;
        else
         tmp <= tmp + 1;
         cuenta<=tmp;
        end if;
  end if;end if;end if;end process;
```

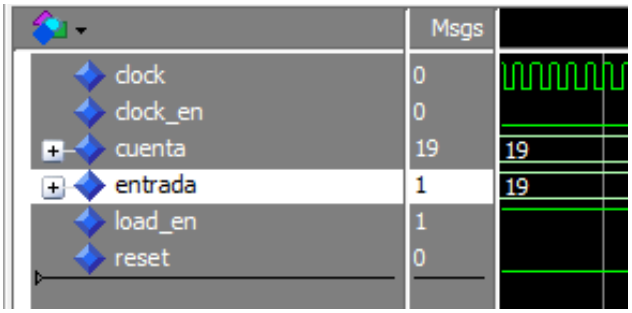
1.3 Simulación

En los 3 primeros ciclos de reloj es posible observar que nuestro contador se resetea a 0, posteriormente se inicia la cuenta hasta llegar al número 59, donde se reinicia la cuenta.





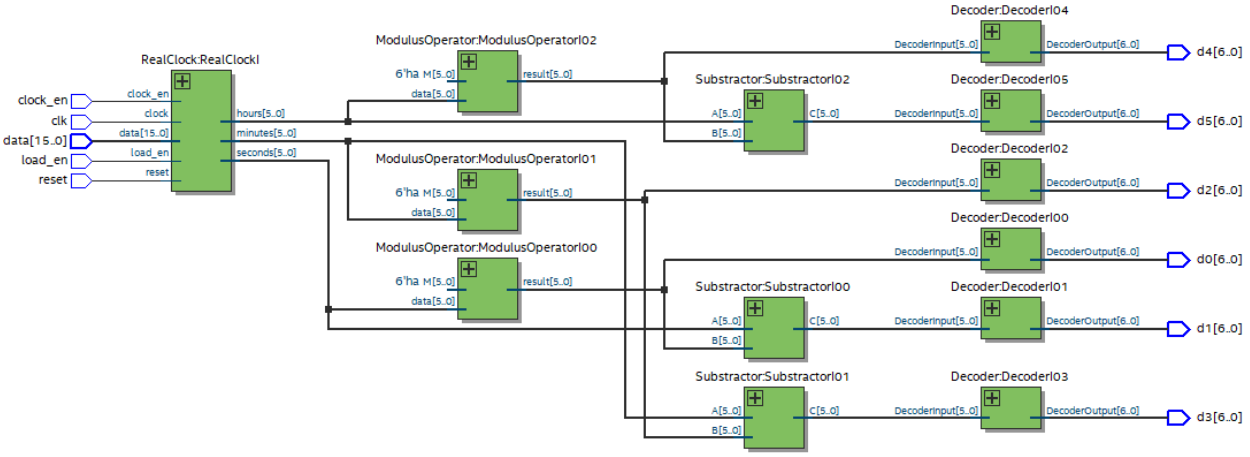
Finalmente tenemos la carga de un dato.



¿Por qué es deseable que en esta aplicación el reset sea síncrono?
El reset debe ser sincrono, porque pienso que los contadores pueden ser usados para mediciones precisas, por lo que un retraso en la cuenta podría llevar a una mala medición.

2 Sección dos

Describir un reloj que muestre la hora del día usando tres contadores con límite como el del inciso 1: 0 a 23 horas, 0 a 59 minutos y 0 a 59 segundos. Simularlo. Asignar 6 displays de siete segmentos para mostrar la hora del día. Usar 16 interruptores para indicar cuatro dígitos (sw0 a sw15) para ajustar la hora y los minutos (poner el reloj a tiempo), al ajustar, los segundos mostrarán cero.



En la figura anterior es posible apreciar la descripción del circuito que muestra un reloj real, para esto se implementó un módulo que nos entrega los segundos, minutos y horas de un reloj.
Además se realiza un proceso de decodificación, para poder visualizarlo en los display, ya que el módulo que nos entrega los segundos,minutos y horas, nos los da en un bus de 5 bits, el cual representa el número de los segundos,minutos u horas.
Para poder dividir esto en dos números, se implementan dos módulos más (un módulo 10 y un restador), esto con la finalidad de obtenerlas unidades y decenas por separado, para posteriormente decodificarlos.
A continuación se muestran los códigos implementados:

2.1 Módulo "RealClock"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY RealClock IS
GENERIC (
        HoursLimit   : INTEGER := 24;
        MinutesLimit : INTEGER := 59;
        SecondsLimit : INTEGER := 59
);
PORT (
        clock,reset,clock_en,load_en : IN  STD_LOGIC;
        hours,minutes,seconds: INOUT STD_LOGIC_VECTOR(5 DOWNTO 0);
        data         : IN  STD_LOGIC_VECTOR(15 DOWNTO 0)
);
END RealClock;

ARCHITECTURE RCArchitecture OF RealClock IS
SIGNAL hourSi,minuteSi,secondSi: STD_LOGIC_VECTOR(5 DOWNTO 0);

begin
process(clk)
begin

 if (reset = '1') then
        hours   <= (others => '0');
        minutes <= (others => '0');
        seconds <= (others => '0');
 elsif (rising_edge(clk)) then
        if (clock_en = '1') then
                if (load_en = '1') then
                        seconds <=  CONV_STD_LOGIC_VECTOR(
                                    CONV_INTEGER(UNSIGNED(
                                    "00" & data(3 DOWNTO 0)))*10, seconds'length )
                                    + ("00" & data(7 DOWNTO 4));
                        minutes <=  CONV_STD_LOGIC_VECTOR(
                                    CONV_INTEGER(UNSIGNED(
                                    "00" & data(11 DOWNTO 8)))*10, seconds'length)
                                    + ("00" & data(15 DOWNTO 12));
                else

                        --Seconds
                        if(conv_integer(seconds) > SecondsLimit-1) then
                                seconds <= (others => '0');
```

```
                        minutes <= minutes + 1;
                    else
                        seconds <= seconds + 1;
                    end if;
                    --

                    -- Minutes
                    if(conv_integer(minutes) > MinutesLimit-1) then
                        minutes <= (others => '0');
                        hours <= hours + 1;
                    end if;

                    if(conv_integer(hours) > HoursLimit-1) then
                        hours   <= (others => '0');
                        minutes <= (others => '0');
                        seconds <= (others => '0');
                    end if;

            end if;
        end if;
 end if;
end process;
END RCArchitecture;
```

2.2 Módulo "Modulus Operator"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY ModulusOperator IS
          GENERIC (
                N    : INTEGER := 6
        );
        PORT (
                M : IN  STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                data    : IN  STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                result : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
        );

END ModulusOperator;

ARCHITECTURE MOArchitecture OF ModulusOperator IS
BEGIN
        result<= CONV_STD_LOGIC_VECTOR(
        CONV_INTEGER(UNSIGNED(data)) MOD CONV_INTEGER(UNSIGNED(M)),
        result'length);
END MOArchitecture;
```

2.3 Módulo "Substractor"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY Substractor IS
          GENERIC (
                N    : INTEGER := 6
        );
        PORT (
                A,B    : IN  STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                C    : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
        );
END Substractor;

ARCHITECTURE SubstractorArchitecture OF Substractor IS
BEGIN
        C<= A-B;
END SubstractorArchitecture;
```

2.4 Módulo "Decoder"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.NUMERIC_BIT.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;


ENTITY Decoder IS

        GENERIC (
                InputSize: INTEGER := 6;
                OutputSize: INTEGER := 7;
                NumberOfSymbols: INTEGER := 10

        );

        PORT (
                DecoderInput : IN  STD_LOGIC_VECTOR(InputSize -1 DOWNTO 0);
                DecoderOutput: OUT STD_LOGIC_VECTOR(OutputSize-1 DOWNTO 0)
        );

END Decoder;

ARCHITECTURE DecoderArchitecture OF Decoder IS

TYPE myArray IS ARRAY(0 to NumberOfSymbols-1) OF STD_LOGIC_VECTOR (OutputSize-1 downto 0);

CONSTANT Symbols : myArray := ("0000001","1111001","0010010","0000110","1001100",
"0100100","0100000","0001110","0000000","0001100");

BEGIN
        DecoderOutput <=
        Symbols(CONV_INTEGER(DecoderInput)) WHEN DecoderInput < 10
        ELSE "1111110";
END DecoderArchitecture;
```

2.5 Módulo "Hw11"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;



ENTITY Hw11 IS

        PORT (
                clk,reset,clock_en,load_en : IN  STD_LOGIC;
                data        : IN  STD_LOGIC_VECTOR(15 DOWNTO 0);
                d0,d1 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0); -- seconds
                d2,d3 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0); -- minutes
                d4,d5 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0) -- hours
        );

END Hw11;

ARCHITECTURE Hw1Architecture OF Hw11 IS

SIGNAL hourSi,minuteSi,secondSi: STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL secondsUnits,minutesUnits,hoursUnits: STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL secondsTens,minutesTens,hoursTens: STD_LOGIC_VECTOR(5 DOWNTO 0);

COMPONENT RealClock IS
        PORT (
                clock,reset,clock_en,load_en : IN  STD_LOGIC;
                hours,minutes,seconds: OUT STD_LOGIC_VECTOR(5 DOWNTO 0);
                data        : IN  STD_LOGIC_VECTOR(15 DOWNTO 0)
        );
END COMPONENT;

COMPONENT ModulusOperator IS
        PORT (
                M      : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
                data   : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
                result : OUT STD_LOGIC_VECTOR(5 DOWNTO 0)
        );
END COMPONENT;

COMPONENT Substractor IS
        PORT (
                A                   : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
                B   : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
```

```
                C : OUT STD_LOGIC_VECTOR(5 DOWNTO 0)
        );
END COMPONENT;

COMPONENT Decoder IS
        PORT (
                DecoderInput : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
                DecoderOutput: OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
        );
END COMPONENT;

BEGIN

RealClockI : RealClock
        PORT MAP(
                clock => clk,
                reset => reset,
                clock_en => clock_en,
                load_en => load_en,
                hours => hourSi,
                minutes => minuteSi,
                seconds => secondSi,
                data => data
        );

ModulusOperatorI00 : ModulusOperator
        PORT MAP(
                M=> "001010",
                data=> secondSi,
                result => secondsUnits
        );


ModulusOperatorI01 : ModulusOperator
        PORT MAP(
                M=> "001010",
                data=> minuteSi,
                result => minutesUnits
        );


ModulusOperatorI02 : ModulusOperator
        PORT MAP(
                M=> "001010",
                data=> hourSi,
                result => hoursUnits
```

```
        );

SubstractorI00 : Substractor
        PORT MAP(
                A=> secondSi,
                B=> secondsUnits,
                C=> secondsTens
        );

SubstractorI01 : Substractor
        PORT MAP(
                A=> minuteSi,
                B=> minutesUnits,
                C => minutesTens
        );

SubstractorI02 : Substractor
        PORT MAP(
                A=> hourSi,
                B=> hoursUnits,
                C => hoursTens
        );

DecoderI00 : Decoder
        PORT MAP(
                DecoderInput=>secondsUnits,
                DecoderOutput=>d0
        );

DecoderI01 : Decoder
        PORT MAP(
                DecoderInput=>secondsTens,
                DecoderOutput=>d1
        );

DecoderI02 : Decoder
        PORT MAP(
                DecoderInput=>minutesUnits,
                DecoderOutput=>d2
        );

DecoderI03 : Decoder
        PORT MAP(
                DecoderInput=>minutesTens,
                DecoderOutput=>d3
        );
```

```
DecoderI04 : Decoder
        PORT MAP(
                DecoderInput=>hoursUnits,
                DecoderOutput=>d4
        );

DecoderI05 : Decoder
        PORT MAP(
                DecoderInput=>hoursTens,
                DecoderOutput=>d5
        );
END Hw1Architecture;
```

2.6 Simulación

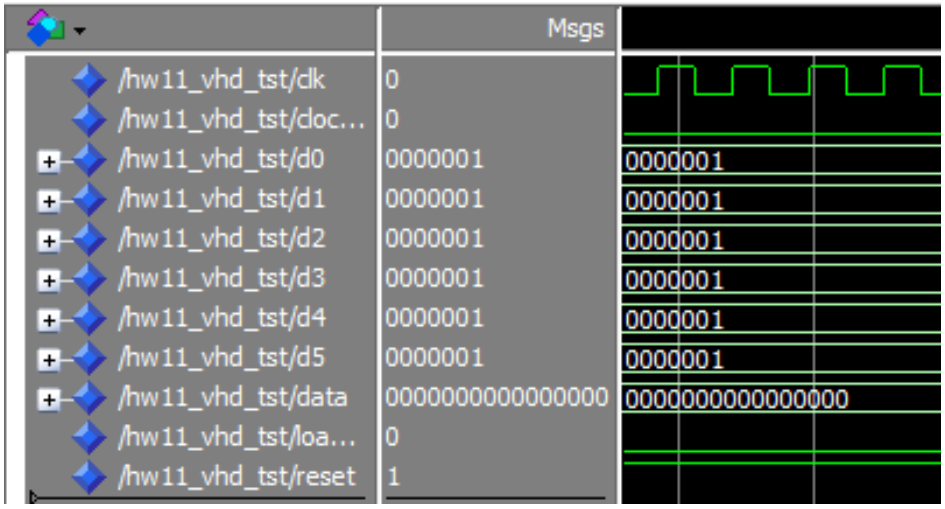


Figure 1: *Reseteo del circuito*

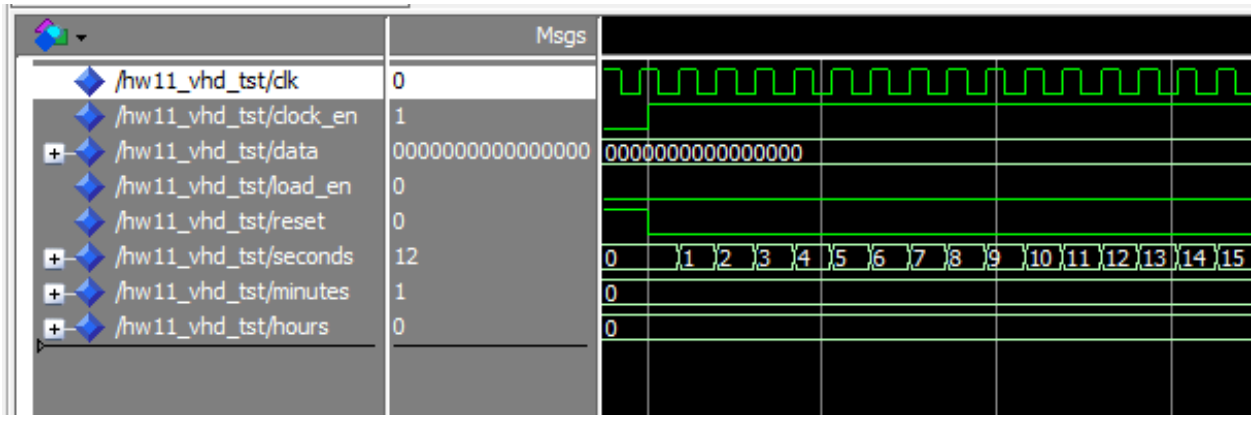


Figure 2: *Inicio del conteo en segundos*

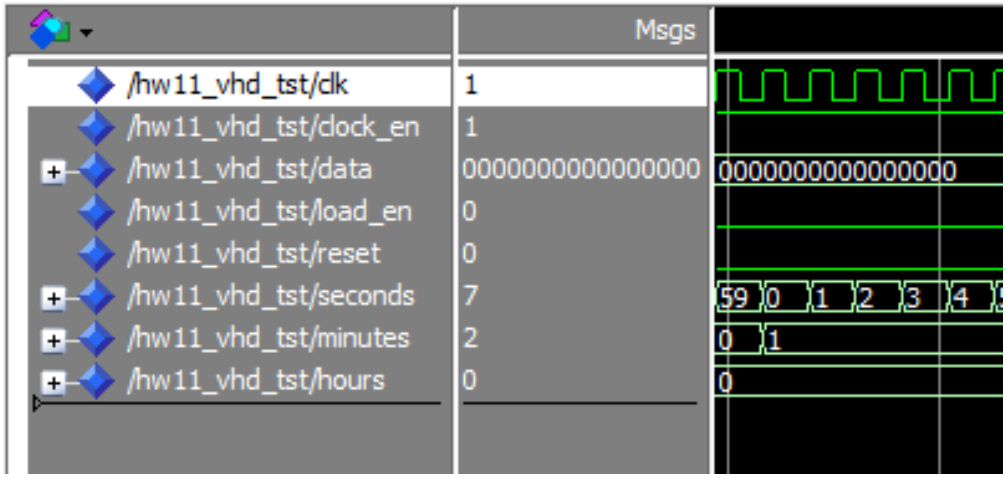


Figure 3: *Transición a minutos*

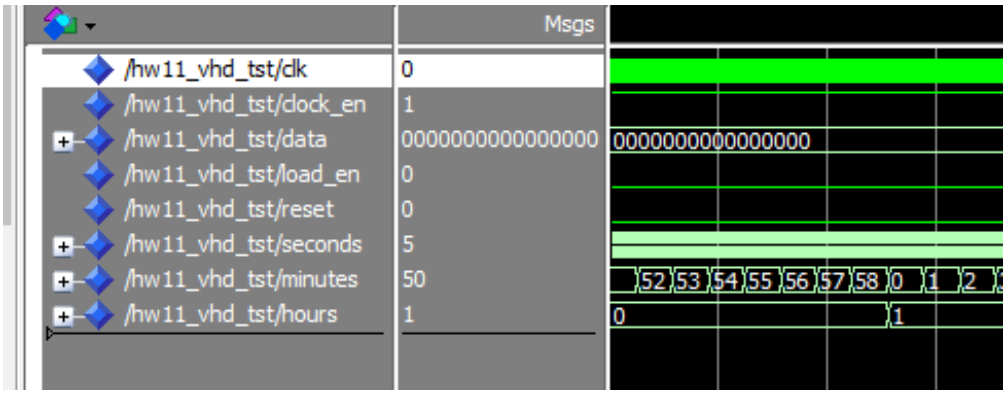


Figure 4: *Transición a horas*

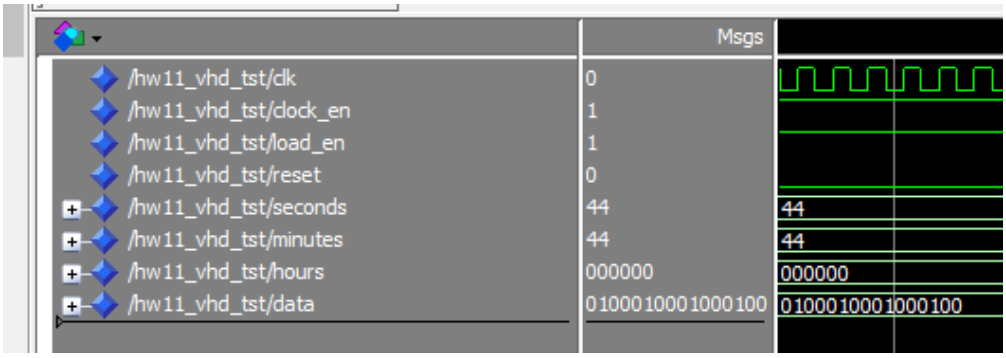


Figure 5: *Carga de datos*

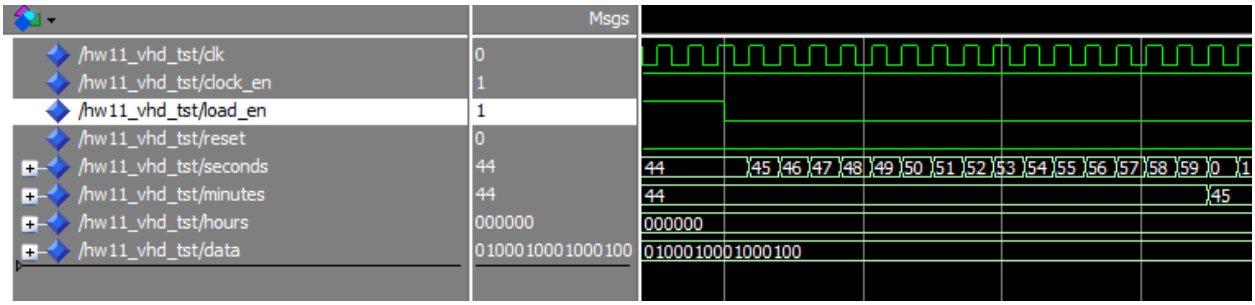


Figure 6: *Seguimiento de la cuenta después de la carga*