**Tasca 1. Implementeu el sistema dissenyat utilitzant el Quartus. Podeu utilitzar qualsevol dels 4 polsadors de la placa de suport com a start, stop i reset. Consulteu el manual de pràctiques per nombrar els PINs correctament.**

El codi que vem agafar a la hora de fer la practica, va ser el del Marc López, el vem triar a l’atzar.

Previs de la practica 3:

**-Previ BDCxtocounter:**

*--Declarem les llibreries*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

*--Declarem les entrades i sortides del bock*

entity BCDxtocounter is

Port(

clk,res,start,pausa: in std\_logic;

t1,t2 : out std\_logic\_vector(3 downto 0)

);

end BCDxtocounter;

*--Dissenyem l'arquitectura del block*

architecture arch\_BCDxtocounter of BCDxtocounter is

*--Definim els senyals nesaris*

signal clkcount: unsigned(13 downto 0):=(others=>'0');

signal count1: unsigned(3 downto 0):=(others=>'0');

signal count2: unsigned(3 downto 0):=(others=>'0');

*--Diem els estats que hi han i declarem* *amb quin s'inicia*

type estats is(s0,s1);

signal estat : estats:=s0;

begin

process(clk) is

begin

if rising\_edge(clk) then

case  estat is

*--Declarem el que passa en el primer estat*

when s0 =>

if res = 1 then

clkcount<=(others=>'0');

count1<=(others=>'0');

count2<=(others=>'0');

elsif start=1 then

estat<=s1;

end if;

*--Declarem el que passa en el segon estat*

when s1 =>

if res=1 then

estat<=s0;

count1<=(others=>'0');

count2<=(others=>'0');

clkcount<=(others=>'0');

elsif pausa=1 then

estat<=s0;

else

if clkcount=9999 then clkcount<=(others=>'0');

if count2=9 then

if count1=9 then res<=1;

else

count1<=count1+1;

count2<=0;

end if;

else

count2<=count2+1;

end if;

end if;

end if;

end case;

end if;

end process;

t1<=std\_logic\_vector(count1);

t2<=std\_logic\_vector(count2);

El codi resultant després de les modificacions va ser el següent:

*--Declarem les llibreries*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

*--Declarem les entrades i sortides del bock*

entity BCDxtocounter is

Port(

clk,res,start,pausa: in std\_logic;

t1,t2 : out std\_logic\_vector(3 downto 0)

);

end BCDxtocounter;

*--Dissenyem l'arquitectura del block*

architecture arch\_BCDxtocounter of BCDxtocounter is

--Definim els senyals nesaris

signal clkcount: unsigned(13 downto 0):=(others=>'0');

signal count1: unsigned(3 downto 0):=(others=>'0');

signal count2: unsigned(3 downto 0):=(others=>'0');

*--Diem els estats que hi han i declarem amb quin s'inicia*

type estats is(s0,s1);

signal estat : estats:=s0;

begin

process(clk) is

begin

if rising\_edge(clk) then

case estat is

*--Declarem el que passa en el primer estat*

when s0 =>

if res = '1' then

clkcount<=(others=>'0');

count1<=(others=>'0');

count2<=(others=>'0');

elsif start='1' then

estat<=s1;

end if;

*--Declarem el que passa en el segon estat*

when s1 =>

if res = '1' then

estat<=s0;

count1<=(others=>'0');

count2<=(others=>'0');

clkcount<=(others=>'0');

elsif pausa = '1' then

estat<=s0;

else

if clkcount=9999 then clkcount<=(others=>'0');

if count2=9 then

if count1=9 then

clkcount<=(others=>'0');

count1 <= (others=>'0');

count2 <= (others=>'0');

else

count1<=count1+1;

count2<=(others=>'0');

end if;

else

count2<=count2+1;

end if;

end if;

end if;

end case;

end if;

end process;

t1<=std\_logic\_vector(count1);

t2<=std\_logic\_vector(count2);

end arch\_BCDxtocounter;

**Codi del BCD2\_seg:**

*--Declarem llibreries*

*library ieee;*

*use ieee.std\_logic\_1164.all;*

*use ieee.numeric\_std.all;*

*--Creem el entity*

*entity BCD2\_seg is*

*port(*

*t1, t2: in std\_logic\_vector (3 downto 0);*

*--Tots els leds dels 2 segments*

*a1,b1,c1,d1,e1,f1,g1,a2,b2,c2,d2,e2,f2,g2: out std\_logic*

*);*

*end BCD2\_seg;*

*--Dissenyem l'arcitectura del bock*

*architecture arc\_BCD2\_seg of BCD2\_seg is*

*signal sortida1: std\_logic\_vector (6 downto 0);*

*signal sortida2: std\_logic\_vector (6 downto 0);*

*begin*

*--Mostrem el que fa el estat 1*

*with t1 select*

*sortida1 <= "1111110" when x"0",*

*"0110000" when x"1",*

*"1101101" when x"2",*

*"1111001" when x"3",*

*"0110011" when x"4",*

*"1011011" when x"5",*

*"1011111" when x"6",*

*"1110000" when x"7",*

*"1111111" when x"8",*

*"1111001" when x"9",*

*"-------" when others;*

*--Mostrem el que fa el estst 2*

*with s2 select*

*sortida2 <= "1111110" when x"0",*

*"0110000" when x"1",*

*"1101101" when x"2",*

*"1111001" when x"3",*

*"0110011" when x"4",*

*"1011011" when x"5",*

*"1011111" when x"6",*

*"1110000" when x"7",*

*"1111111" when x"8",*

*"1111001" when x"9",*

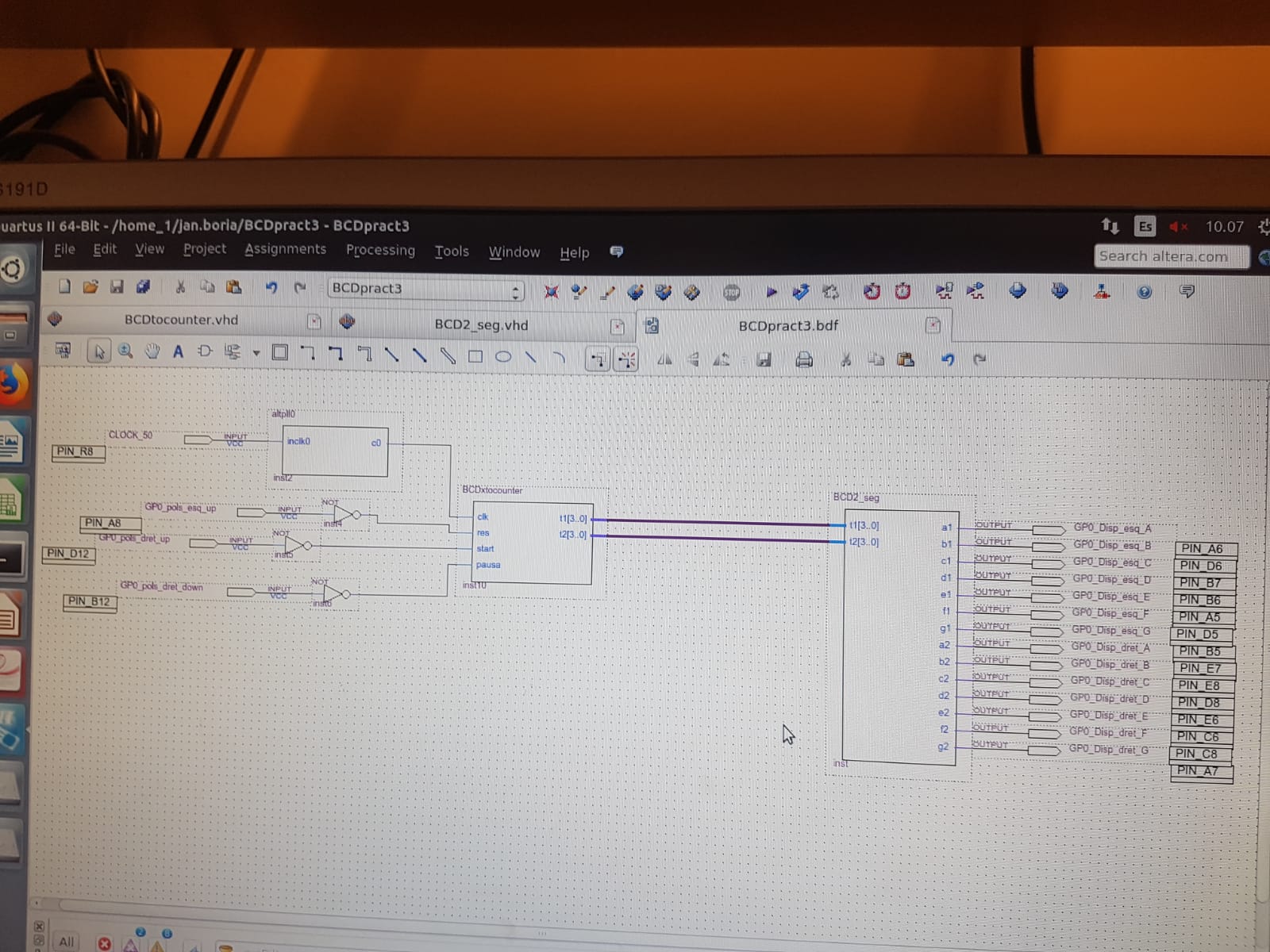
*"-------" when others;*

*(a1,b1,c1,d1,e1,f1,g1) <= sortida1;*

*(a2,b2,c2,d2,e2,f2,g2) <= sortida2;*

*end arch\_BCD2\_seg;*

No ha calgut posar el previ d'aquest mòdul ja que el previ i el que vam acabar fent servir a la pràctica era el mateix.



Aquest ha sigut el nostre diagrama final, on adjuntem tots els codis junts, declarem entrades i sortides, i afegim el codi 50 to 100, que necessitem per convertir freqüències.

**Aquí presentem el següents tests bench dels dos codis creats:**

**BCDxtoseg\_tb**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity BCD2toseg\_tb is

end BCD2toseg\_tb;

architecture arch\_BCD2\_seg of BCD2toseg\_tb is

component my\_BCD2to\_seg

port(

S1, S2: in std\_logic\_vector (3 downto 0);

a1, b1, c1, d1, e1, f1, g1, a2, b2, c2, d2, e2, f2, g2: out std\_logic);

end component;

for dut1: my\_BCD2to\_seg use entity work.BCD2toseg;

-- for dut2: my\_Hsync use entity work.H\_sync;

-- for dut3: my\_Vsync use entity work.V\_sync;

-- for dut4: my\_RGBgenerator use entity work.RGBgenerator;

signal sig1,sig2: std\_logic\_vector(3 downto 0);

signal sa1, sb1, sc1, sd1, se1, sf1, sg1,sa2, sb2, sc2, sd2, se2, sf2, sg2: std\_logic;

begin

dut1: my\_BCD2to\_seg port map(

S1=> sig1,

S2=> sig2 ,

a1=>sa1,

b1=>sb1,

c1=>sc1,

d1=>sd1,

e1=>se1,

f1=>sf1,

g1=>sg1,

a2=>sa2,

b2=>sb2,

c2=>sc2,

d2=>sd2,

e2=>se2,

f2=>sf2,

g2=>sg2);

sig1<= "0000","0001" after 100 ns, "0010" after 200 ns,"0100" after 300 ns;"----" after 400 ns;

sig2<="1001","0101" after 100 ns, "0101" after 200 ns,"1000" after 300 ns;"----" after 400 ns;

end arch\_BCD2\_seg;

**BCDxtocounter\_tb**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity BCDxtocounter\_tb is

end BCDxtocounter\_tb;

architecture arch\_BCDxtocounter of BCDxtocounter\_tb is

component my\_BCDxtocounter

port(clk : in std\_logic;

res:in std\_logic:='0';

start:in std\_logic:='1';

pausa: in std\_logic:='0';

d1,d2 : out std\_logic\_vector(3 downto 0));

end component;

-- component my\_Hsync

-- port ( H : in std\_logic\_vector (9 downto 0);

-- H\_sync : out std\_logic );

-- end component;

for dut1: my\_BCDxtocounter use entity work.BCDxtocounter;

-- for dut2: my\_Hsync use entity work.H\_sync;

-- for dut3: my\_Vsync use entity work.V\_sync;

-- for dut4: my\_RGBgenerator use entity work.RGBgenerator;

signal sclk,sRst,sStp,sPaus: std\_logic;

signal sd1,sd2 : std\_logic\_vector(3 downto 0);

end arch\_BCDtoxcounter;

Begin

dut1: my\_BCDxtocounter port map(

clk=> sclk,

d1=>sd1,

d2=>sd2);

-- dut2: my\_Hsync port map(

-- H=>sH

-- );

-- dut3: my\_Vsync port map(

-- V=> sV

-- );

-- dut4: my\_RGBgenerator port map(

-- H=>sH,

-- V=>sV,

-- R=>sR,

-- G=>sG,

-- B=>sB);

clock\_process: process

begin

sclk <= '0';

wait for 5 ns;

for i in 1 to 4000000 loop

sclk <= not sclk;

wait for 5 ns;

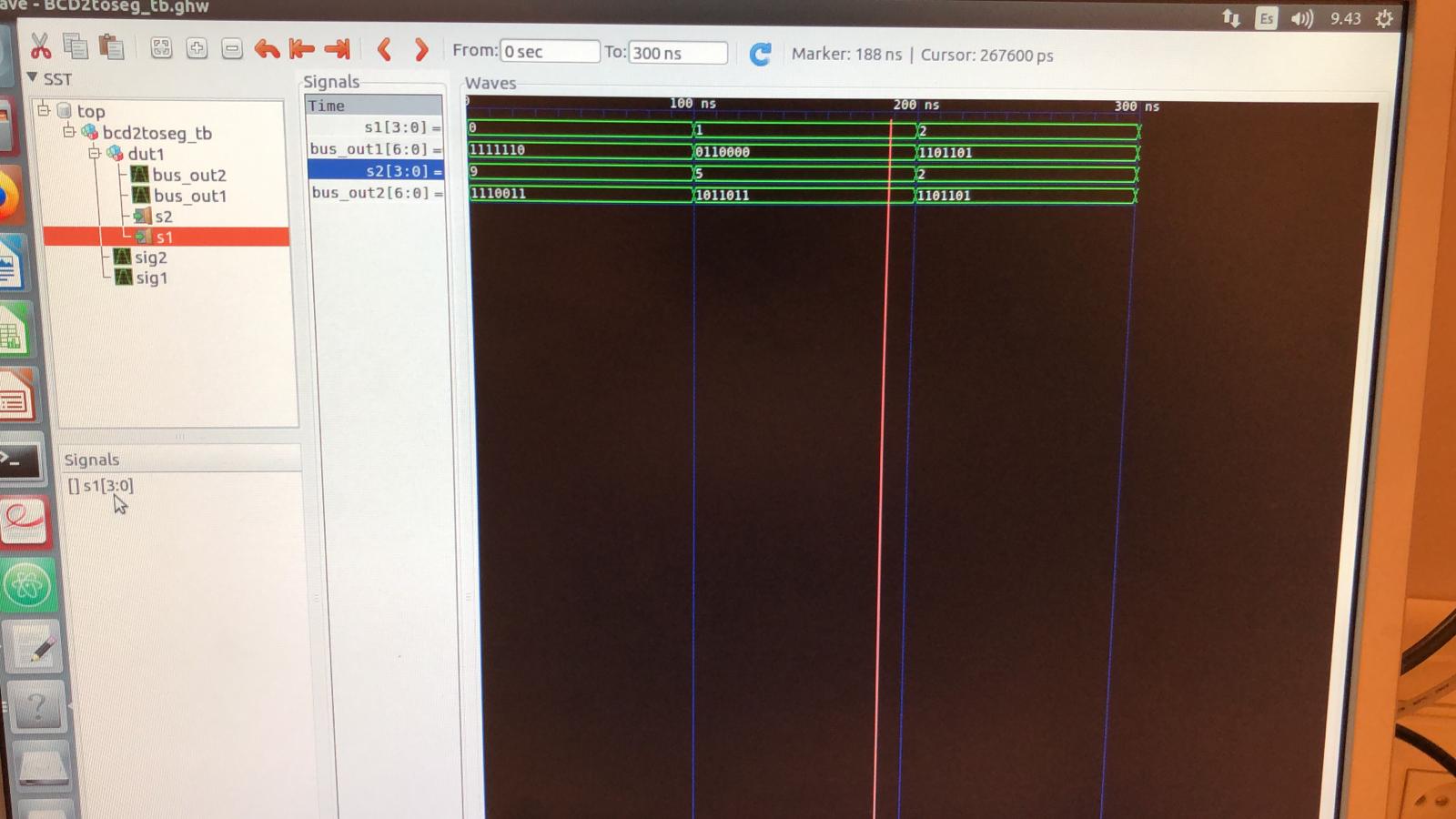
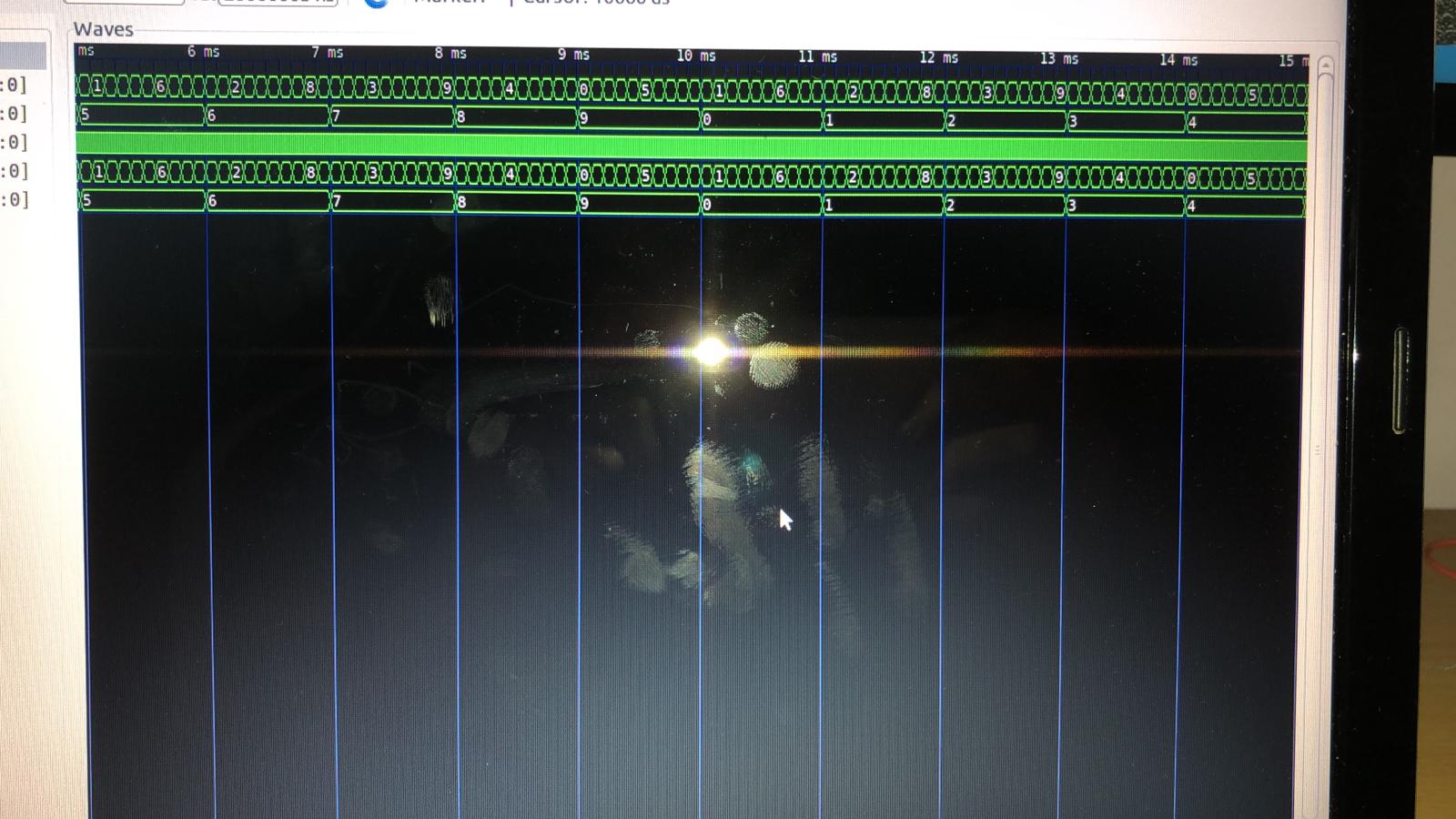
end loop;

wait;

end process clock\_process;

end arch;

Simulacions amb GtkWave:



**Conclusions:**

El primer error que hi havia era que no vaig posar el "end arch\_BCDxtocounter;" cosa que feia que no es pogués fer res, però no em va caldre ni simular-lo per a adonar-nos del que havíem fet malament; tal com vam posar el codi al Quartus ho vam afegir.

També vam reparar en què el codi a l'hora d'arribar a "9 : 9", és resetejava, però no era la manera correcta de fer-ho, llavors, simplement vam afegir que un cop arribes a "9 : 9", tornes a l'estat 0.

Un cop vam sintetitzar tots i cadascun dels components els vam afegir al top level entity; tot seguit vam afegir el bloc "50 to 100Hz" a l'entrada del clock i ja per a acabar, vam afegir els inputs i els outputs i vam fer les connexions entre blocs necessàries.

Quan vam compilar i enviar-ho al FPGA, els números del display no es movien i es quedaven tota l'estona a "0 : 0".

UN cop vam veure això vam revisar el BCDxtocounter, ja que sabíem de sobres que l'error havia de ser d'allà perquè el mòdul del previ tres simplement passava el que rebia del BCDxtocounter al display.

Quan ens vam posar a revisar on estava l'error, vam veure que era una cosa molt simple i que el Quartus no petava a l'hora de compilar perquè realment no era estrictament necessari que afectés al mal desenvolupament del codi.

El que passava és que el codi contava els clocks, però no els sumava al comptador pel qual aquest no podia incrementar i com que totes les altres funcions depenien d'aquest comptador, doncs no s'executava res.