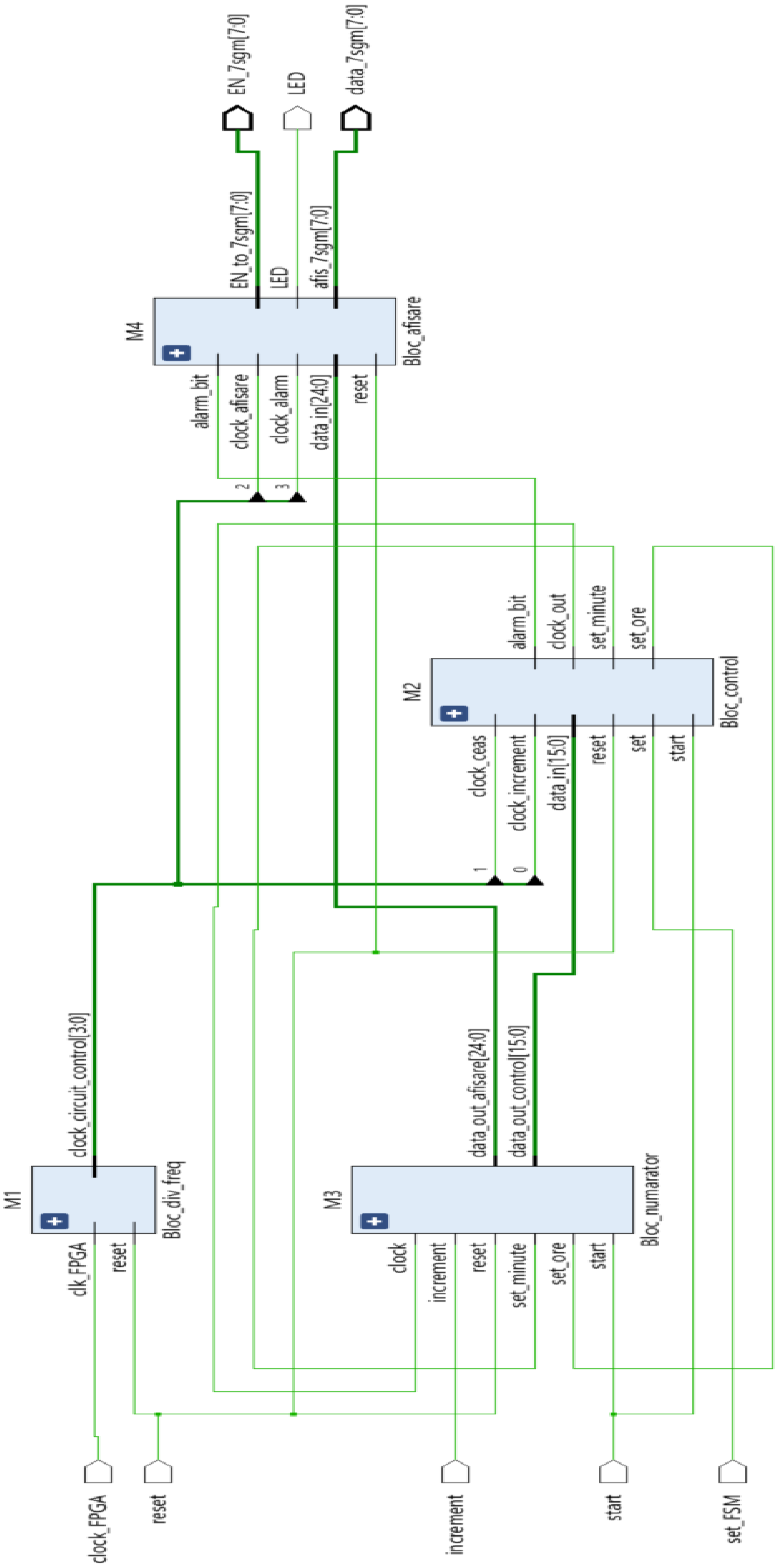
~ Project ~

Digital Numeric Clock

Project Details:

- Display type : AM – PM 12 : 00;
- Base frequency of quartz oscillator : 2 MHZ;
- Another functionalities :circuit to set the hour and minutes, alarm to ring at fixed hour–alarm will ring about 2 second with an 3Ghz frequency signal;

System block diagram :



System description:

When I put the system under voltage the clock will displaying 12: 00: 00 AM, this state will be system initial state. The user has possibility to bring the start button on 1 position and the system will start or to bring the start button on 0 and to push the set button to set the wanted hour.

If the start button was put in 1 position, the user don’t has control to set the clock, he has control only to reset the clock with reset button which is asynchronous button. If we push the reset button the clock will get back in the initial state.

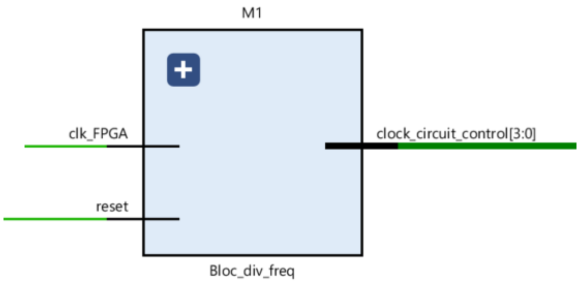
If we let the start button on 0 logic, we can set the wanted hour through pushing set button. This button goes to set de minutes, but if we want to increment the minutes is necessary to push the increment button until we get the wanted minutes.

To set the hour it’s necessary to push second time the set button, and the system will enter in the state wich sets the hour. To increment the hours will be push the set button until the wanted hour will appear. AM-PM configuration it can be modified when press the increment button and get an complete twelve cycle .

To start the system will bring the start button on 1 logic and setup mode will be deactivated.

Divider frequency block:

BLOCK small =>



Input signals:

- **clk_FPGA** : this input will be connected with FPGA clock wich has 100 MHz and then will be devided in one signal with 2 MHz frequency, etalon signal;

-reset : this is an asynchronous input. It's for reset the clock;

Output signals:

- clock_circuit_control[3:0] :

-bit 3 = clock alarm frequency (3 KHz) will be used like an PWM signal to control the alarm.

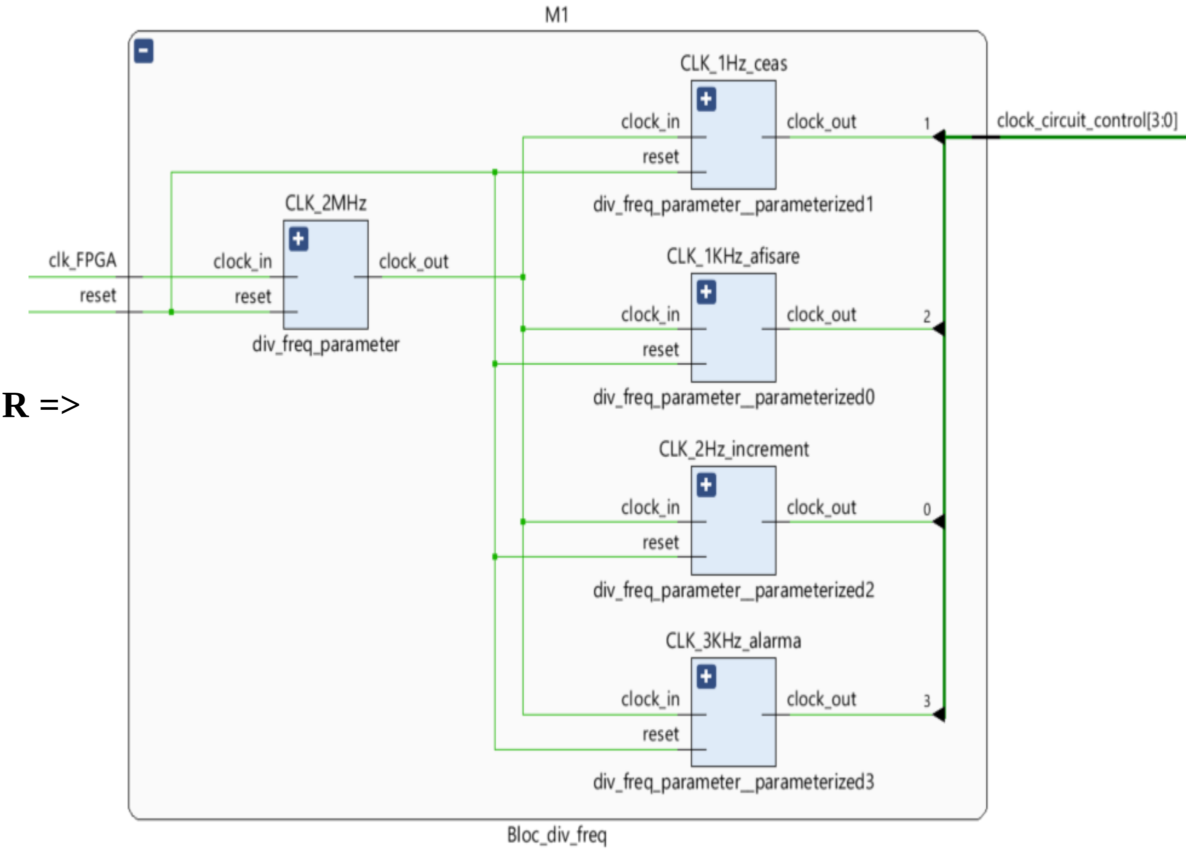
-bit 2 = multiplexed clock frequency (1 KHz) used to multiplexed display in display block.

-bit 1 = clock frequency (1 Hz) used in increment block in normal functionality.

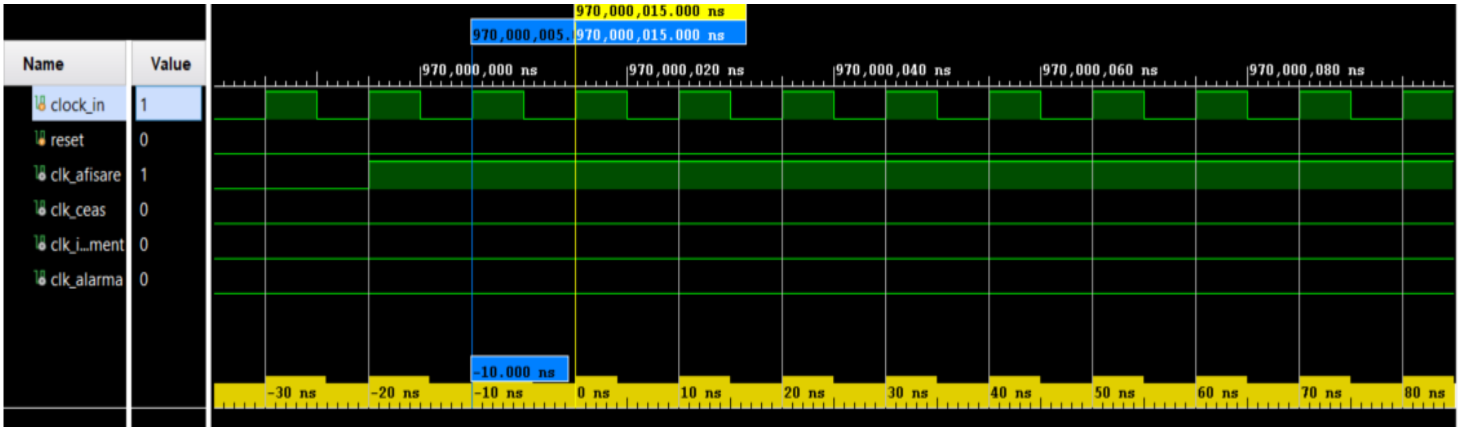
-bit 0 = increment frequency (2 Hz).

e;

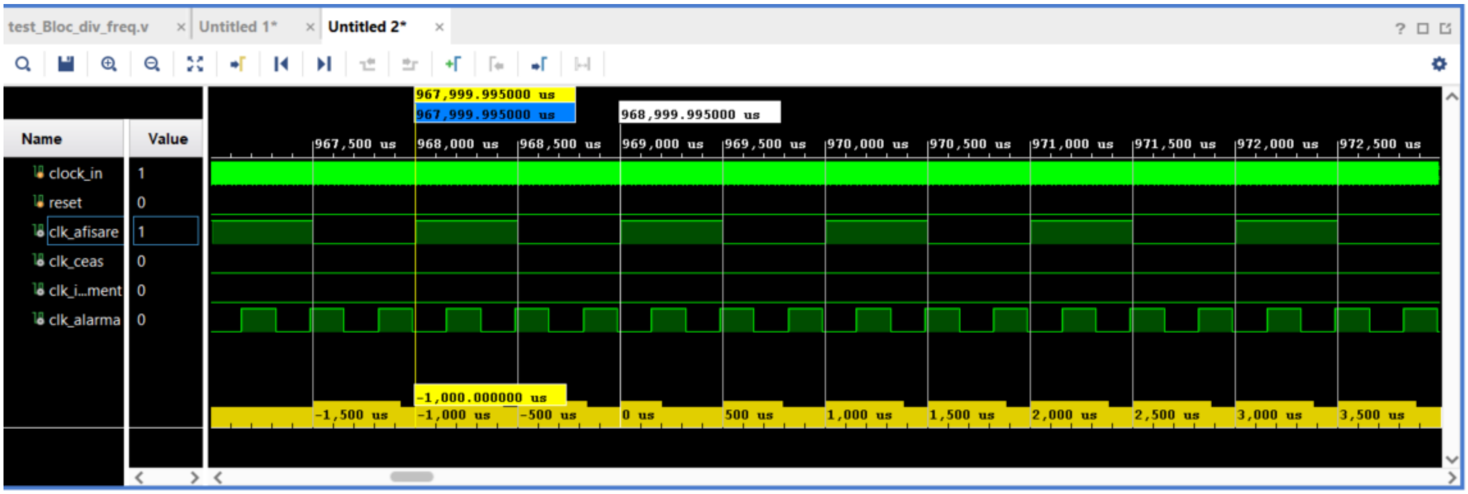
BLOCK LARGER =>



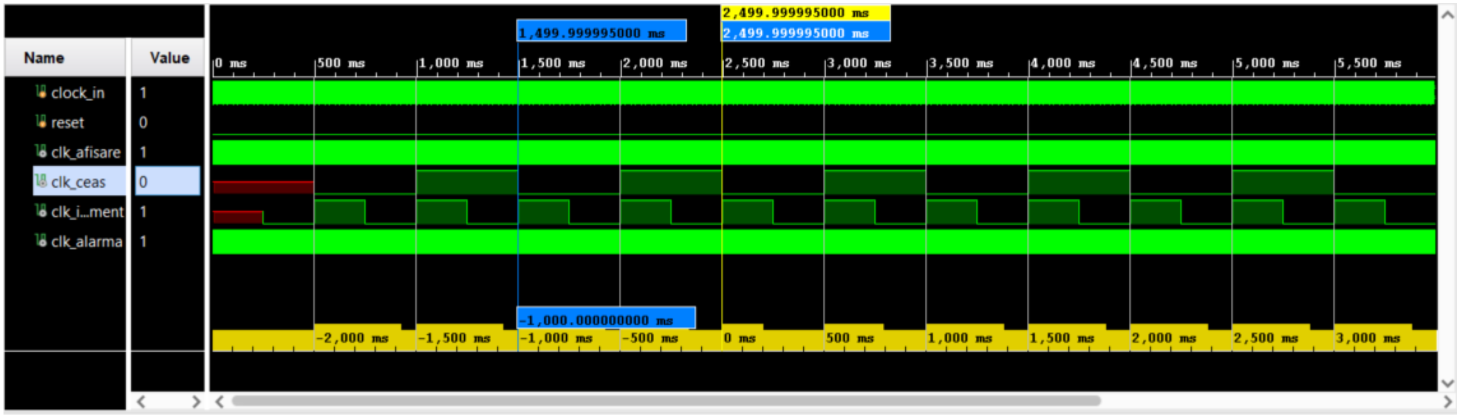
Sim results :



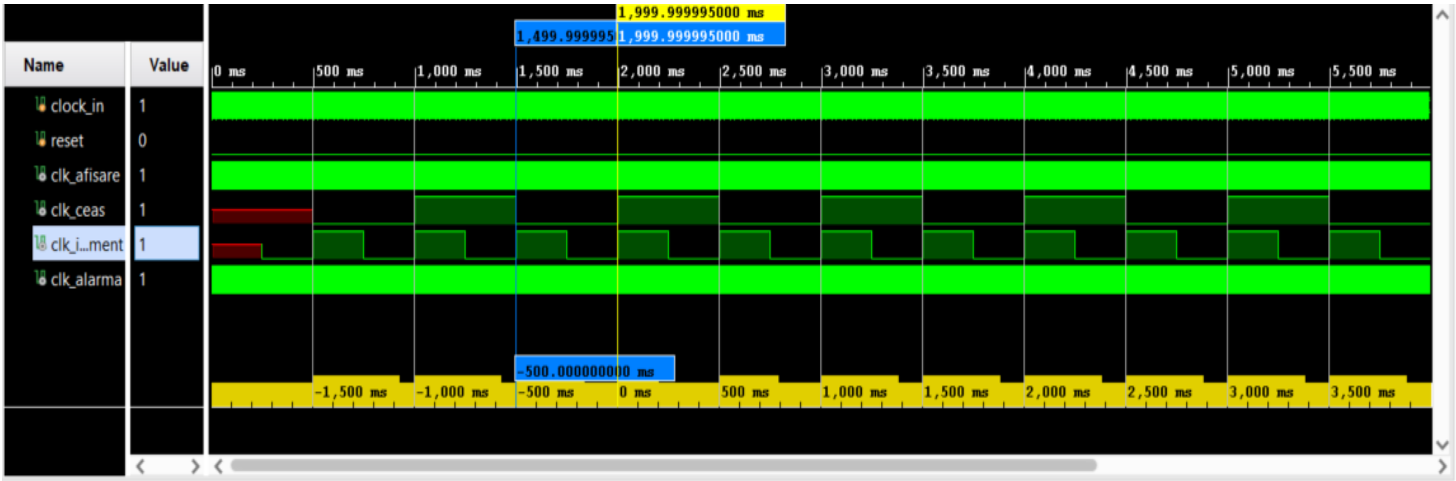
Input signal period : 10 ns => f = 100 MHz



Display signal period : 1 ms => f = 1KHz

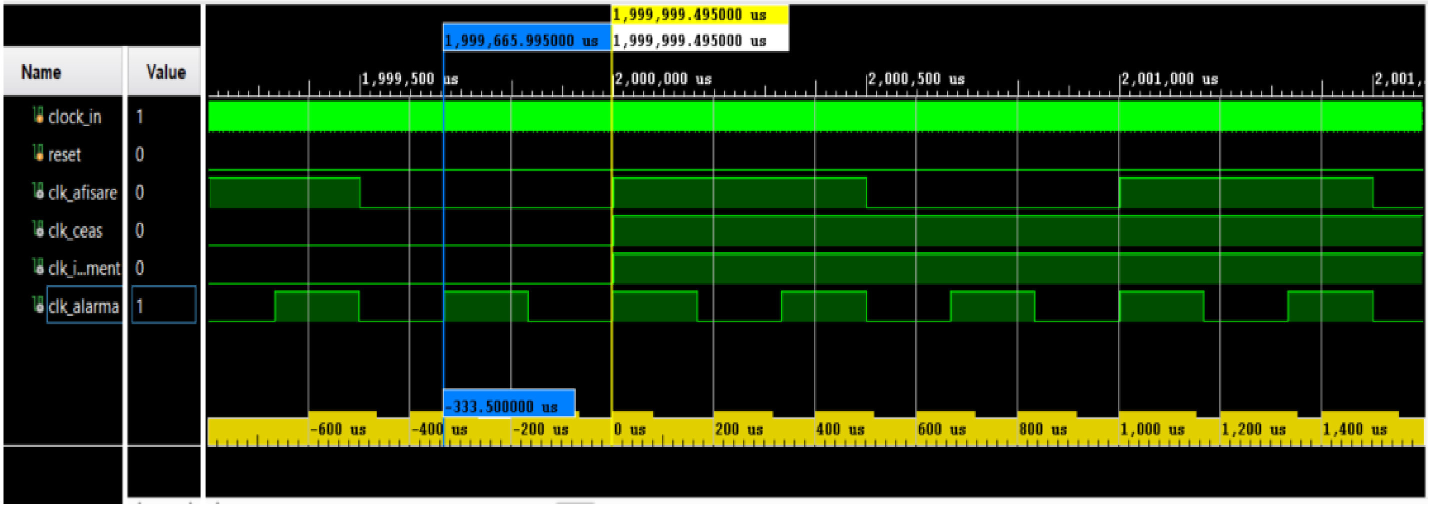


Incremental signal period : 1s => f = 1 Khz



Name	Value
clock_in	1
reset	0
clk_afisare	1
clk_ceas	1
clk_i...ment	1
clk_alarma	1

Incremental signal period: 500 ms => f = 2 Hz

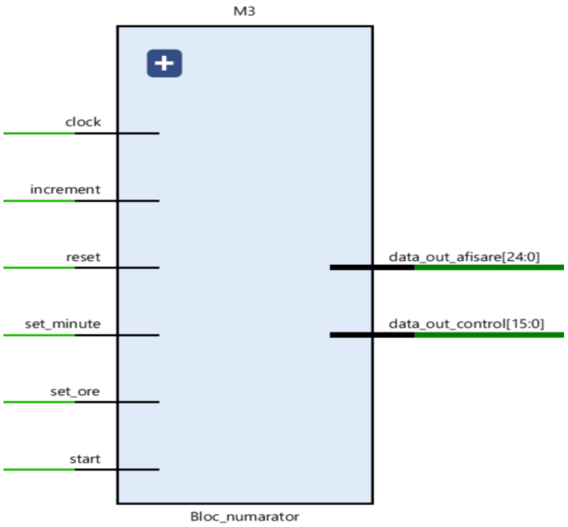


Name	Value
clock_in	1
reset	0
clk_afisare	0
clk_ceas	0
clk_i...ment	0
clk_alarma	1

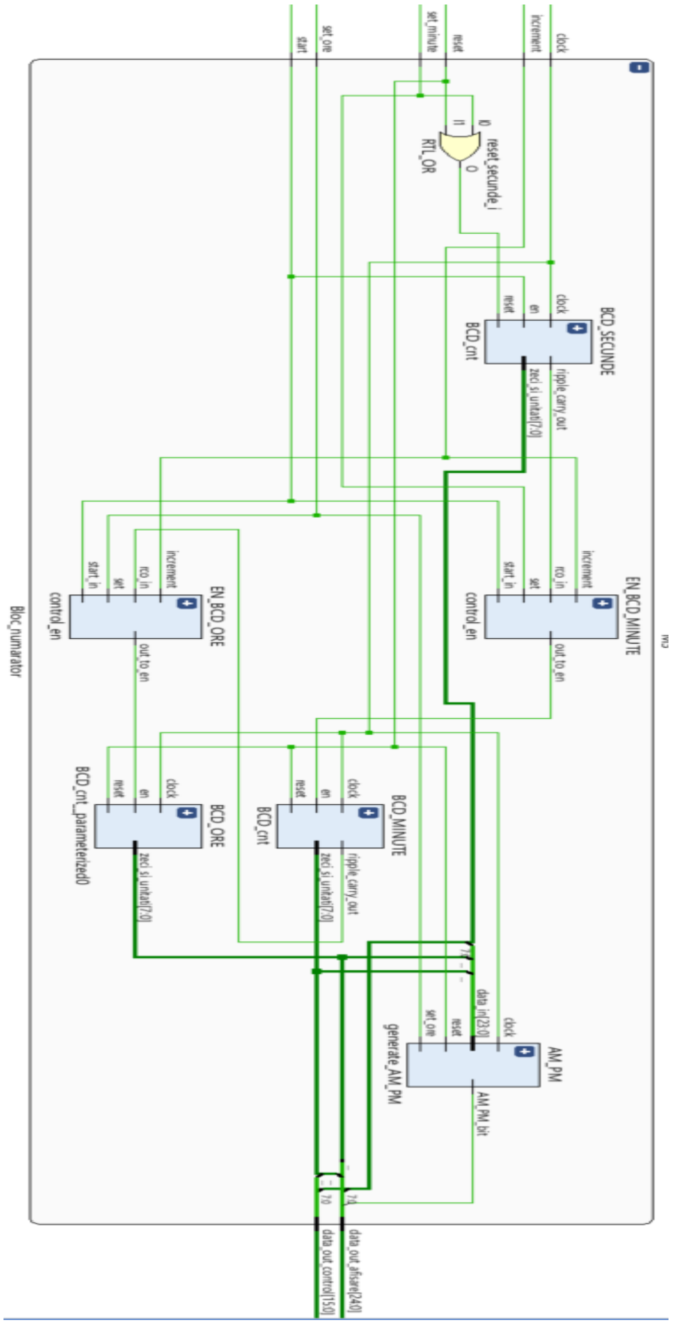
Display period signal: 333.5 us => f = 2,998 kHz

Counter Block:

Block small =>



Block larger =>



Input signals:

-clock : clock signal it's used for counter to increment counters inside block BCD_cnt.. The signal of this signal it's variable. This frequency it's used to the control block and are 2 2 values for this signal :: a) 1 Hz when clock it's in normal function;
b) 2 Hz when clock are in setup state.

-increment : this input will be externally connected and simultaneously connected to the two blocks(EN_BCD_ORE and EN_BCD_MINUTE). That functionality from that two blocks will accept to increment separately block BCD_cnt_ORE if : start = '0' and set_ore = '1'. Also I can increment EN_BCD_MINUTE block if: start = '0' and set_minute= '1'.
This pin has no effect in normal functionality state.

-reset : asynchronous input connected to the reset button. Press this button will get counters in this states: BCD_SECUNDE = '00', BCD_MINUTE = '00' , BCD_ORE = '12'.

-set_minute : input who comes from control block and it's used to permit increment of minutes with 2 HZ frequency when start pin = '0'.

-set_ore : input who comes from control block and it's used to permit increment of minutes with 2 HZ cand start = '0' logic.

-start : input who comes from an extern button wich can allow clock to get in functional state when the pin are'1' logic.

Output signals:

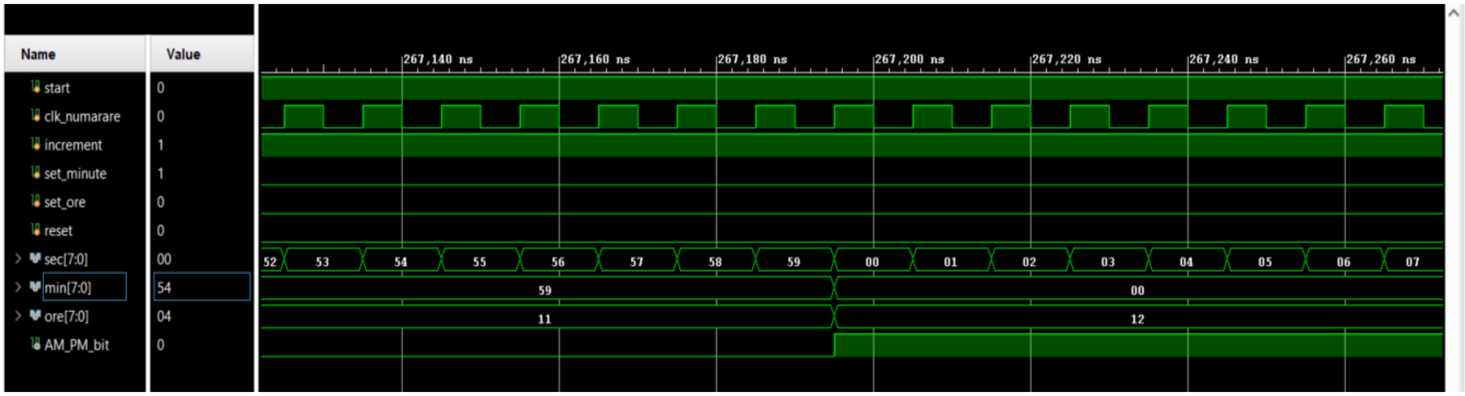
-data_out_afisare[24:0] : data bus who goes to display block. In this data bus the signal are grouped in this forms :

a) data_out_afisare[24] = AM_PM_BIT;
b)data_out_afisare[23:20] =hour tens;
c)data_out_afisare[19:16] =hour units;
d)data_out_afisare[15:12] = minutes tens;
e)data_out_afisare[11:8] =minutes units;
f) data_out_afisare[7:4] =seconds tens;
g) data_out_afisare[3:0] = seconds units;

-data_out_control[15:0] : data bus who goes to control block. In this data bus signal are grouped in this forms :

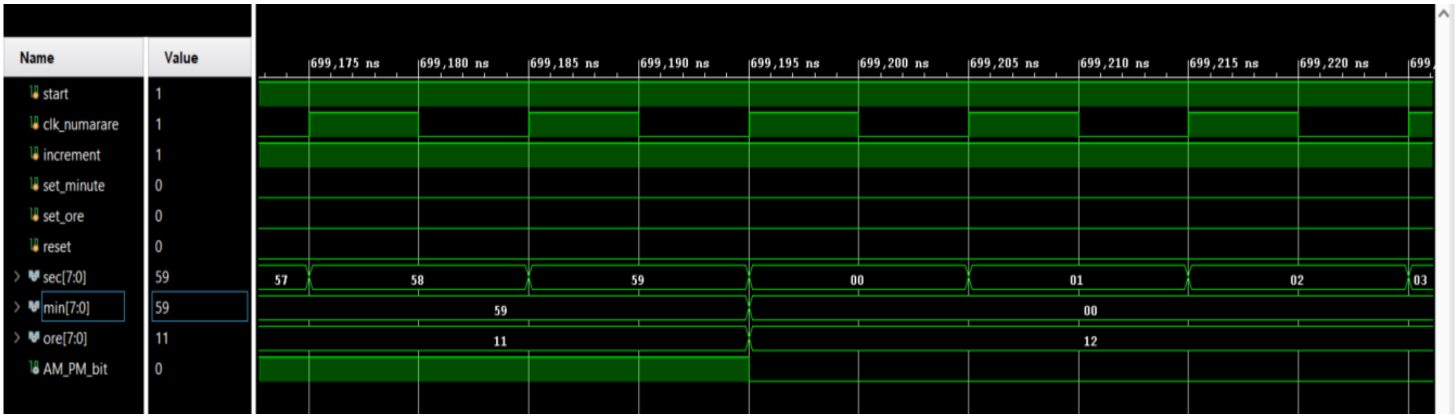
a)data_out_afisare[15:12] = minutes tens;
b)data_out_afisare[11:8] = minutes units;
c) data_out_afisare[7:4] = seconds tens;
d) data_out_afisare[3:0] = seconds units;

Sim result of counter block:



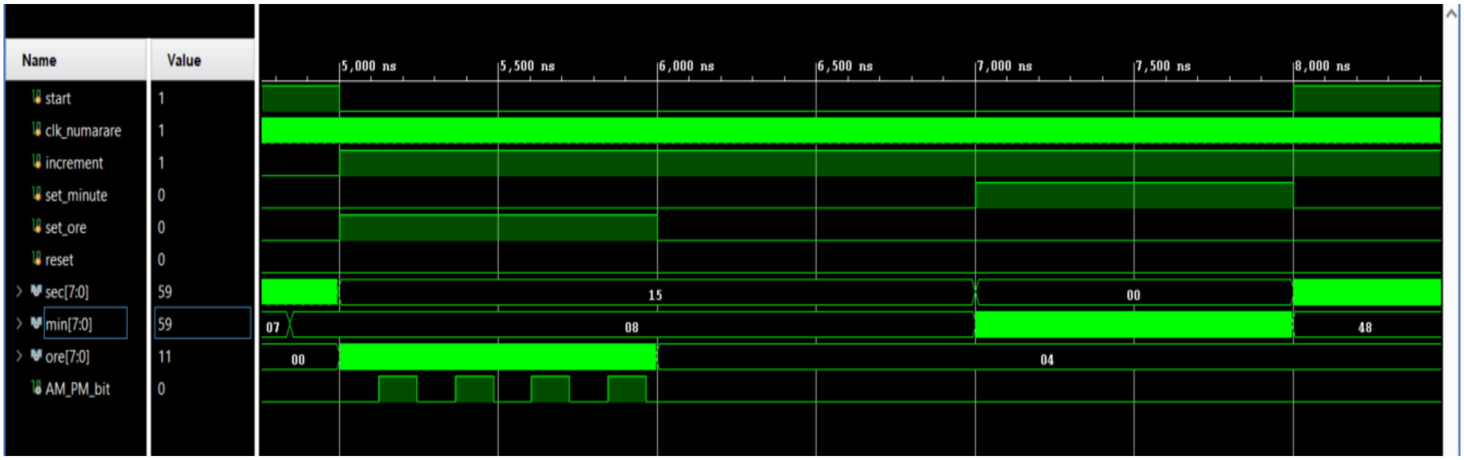
Name	Value
start	0
clk_numarare	0
increment	1
set_minute	1
set_ore	0
reset	0
sec[7:0]	00
min[7:0]	54
ore[7:0]	04
AM_PM_bit	0

Functionality of clock with AM to PM changes.



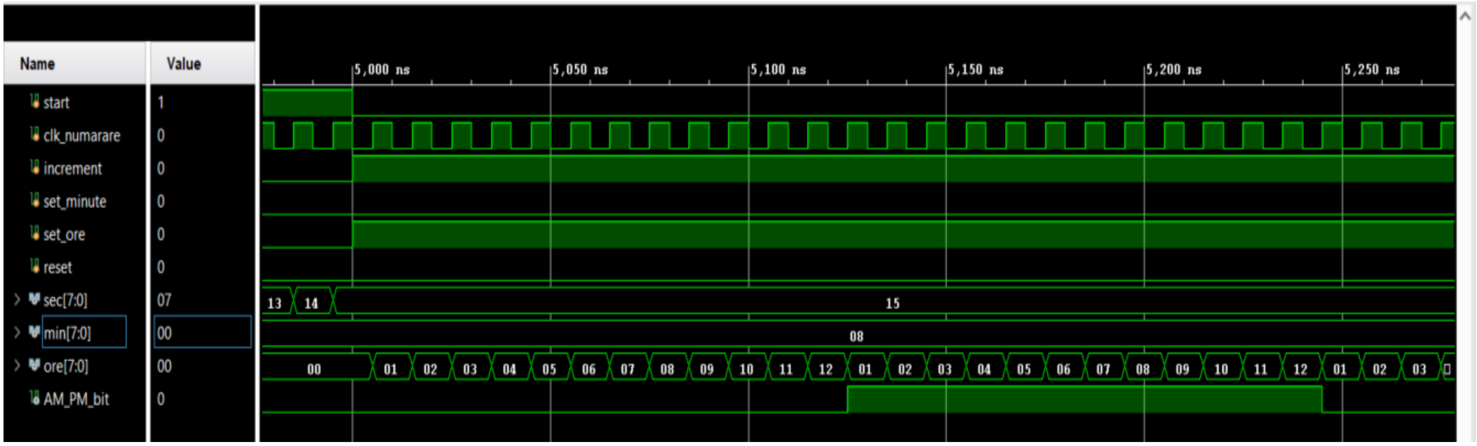
Name	Value
start	1
clk_numarare	1
increment	1
set_minute	0
set_ore	0
reset	0
sec[7:0]	59
min[7:0]	59
ore[7:0]	11
AM_PM_bit	0

Functionality of clock with PM to AM changes.

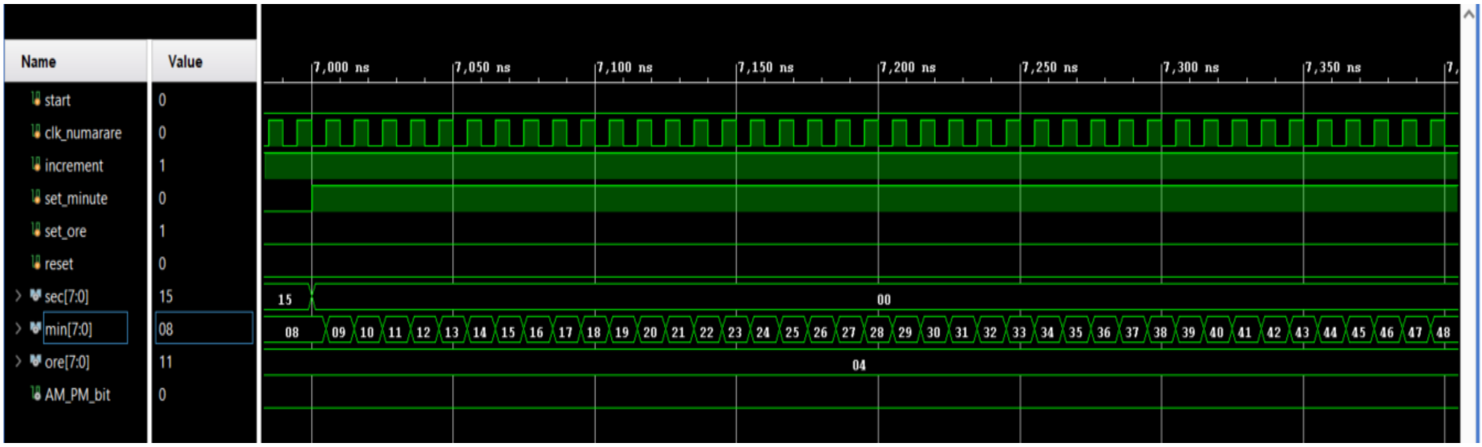


Name	Value
start	1
clk_numarare	1
increment	1
set_minute	0
set_ore	0
reset	0
sec[7:0]	59
min[7:0]	59
ore[7:0]	11
AM_PM_bit	0

Functionality of clock in setup mode => start = 0, increment = 1 , sets minutes / sets hours= 1.



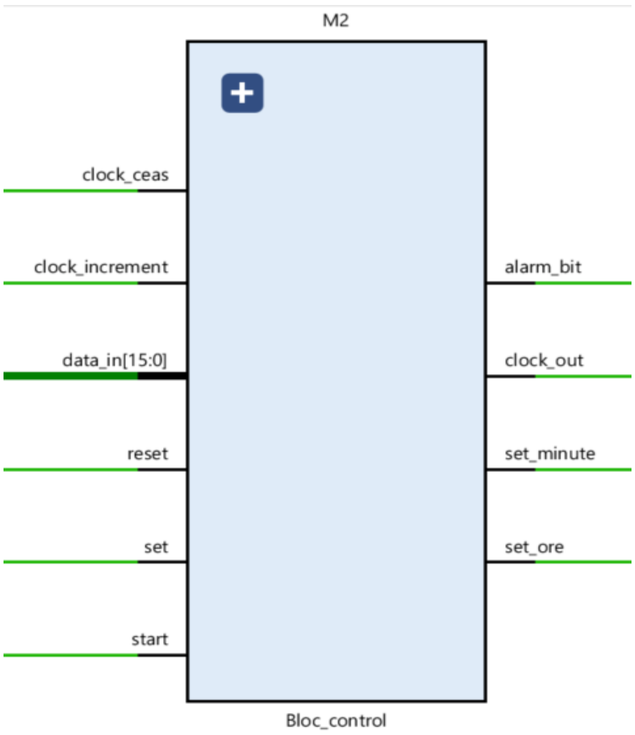
Functionality of clock in minutes setup state.



Functionality of clock in hours setup state.

Control Block:

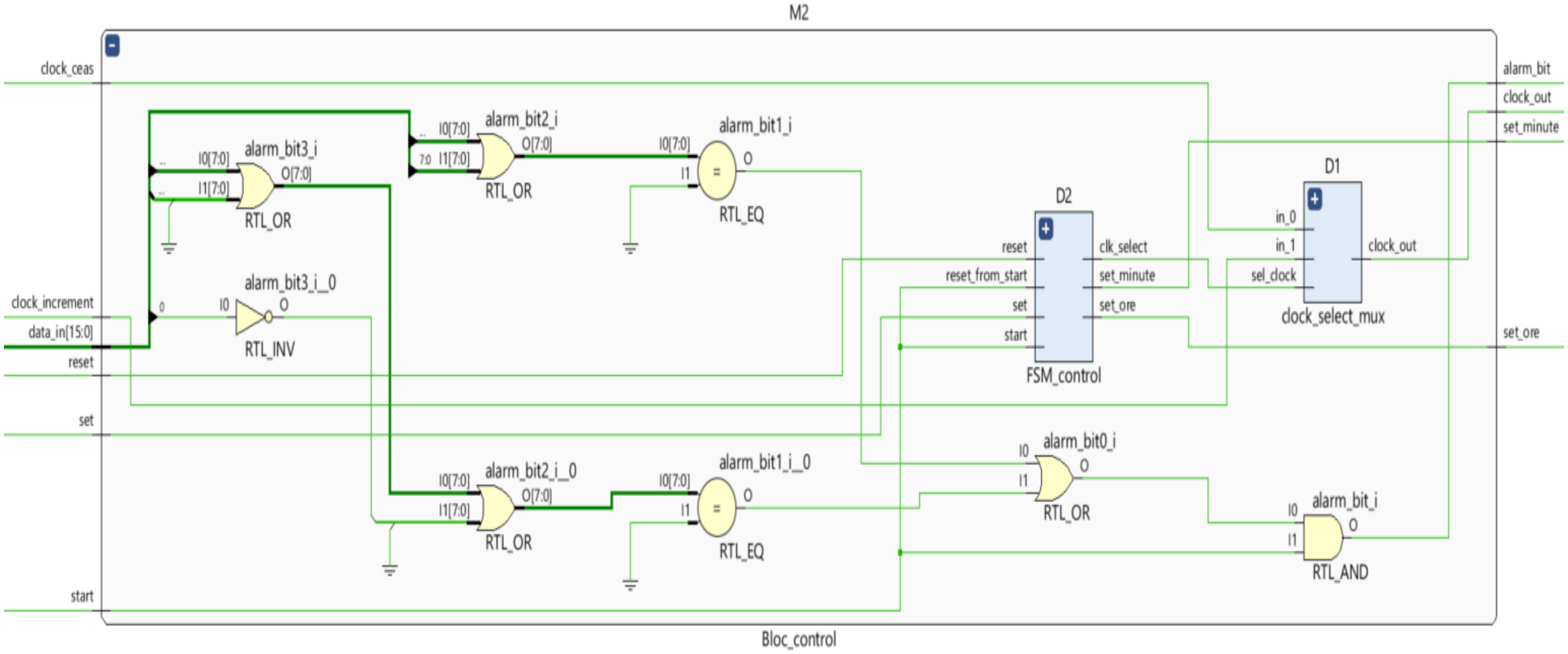
Block small =>



Block larger

| |

\ /



Inputs signals:

-clock ceas = In this input will be connect the output of .frequency divider block. Also , to this input will be present only 1 Hz clock, input who goes in the frequency multiplexor to increment counter block;

- clock_increment :to this input will be connect an clock with frequency 2 H. This input together with clock signal will be applied in the frequency multiplexor to increment counter block.

-data_in[15:0] : this input it's comes from counter block and contains information about minutes and seconds. This inputs will be used to generate the alarm signal.

- reset : asynchronous input to reset FSM.

- set :input to control the FSM. This input it's used to enter in setup state, to set the minutes and hours. To get out from this infinite loop it's necessary to push the start button.

- start : Input which connect to the input reset_from_start of FSM and it use to generate the alarm. When button it's bring from 0 logic to 1, the FSM will be reset. This button works like an reset button for FSM.

Output signals:

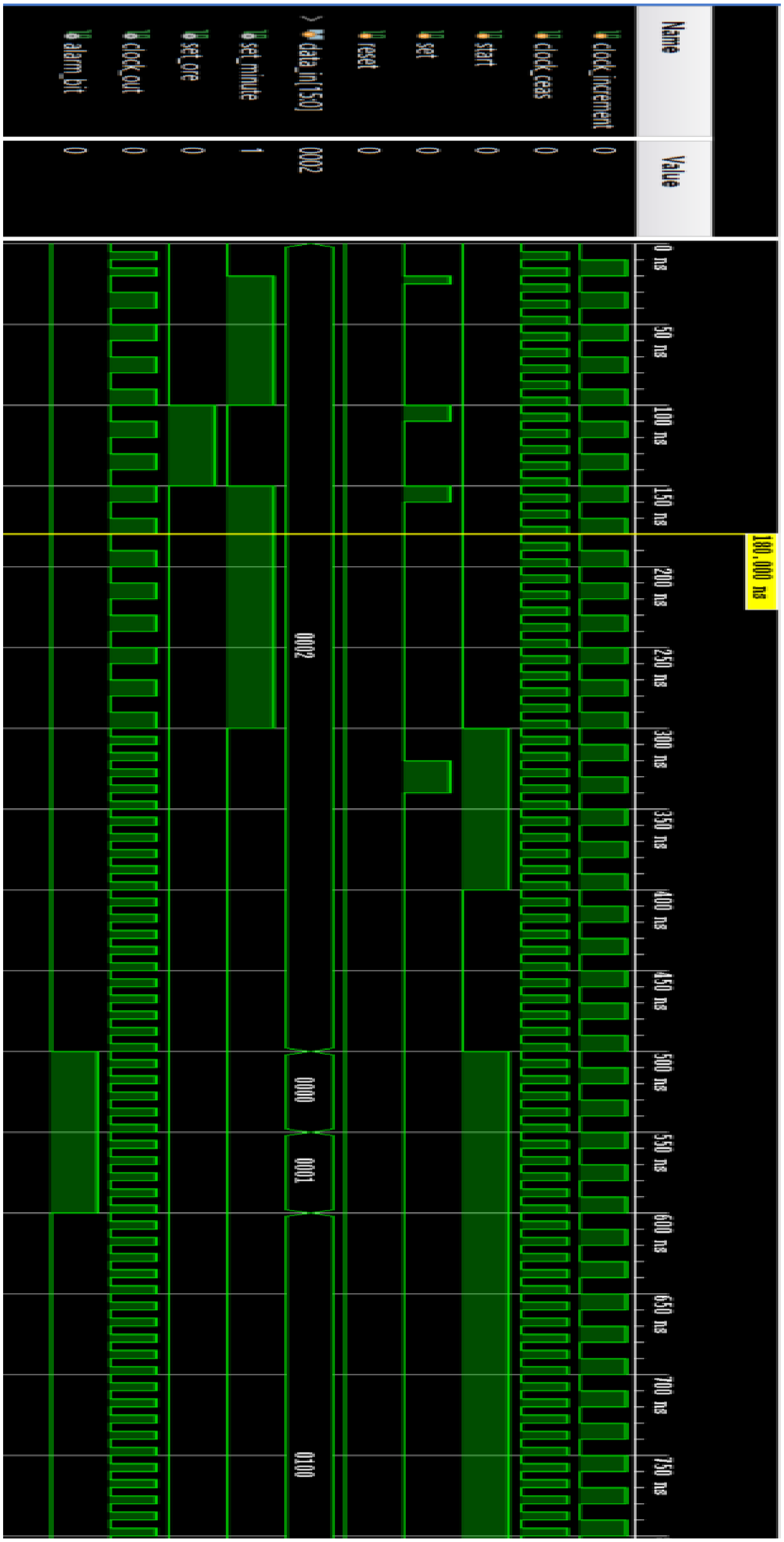
-alarm bit : this output will be active for 2 seconds at fixed hour and will be transmit to display block.

-clock_out : this output will be connect with counter block. This output will have(1Hz and 2 Hz). Output will be controlled by the present FSM.

- set_minute : this output will be active in control state to control minutes when start button it's on 0 logic.

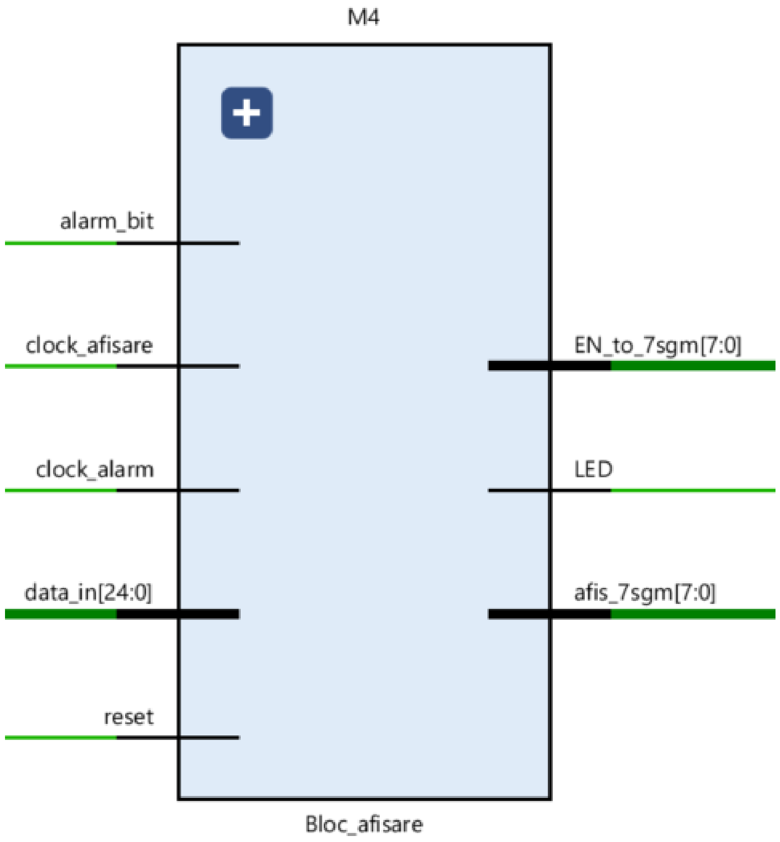
- set_ore : this output will be active in control state to control hours when start button it's on 0 logic.

Sim result for control block:

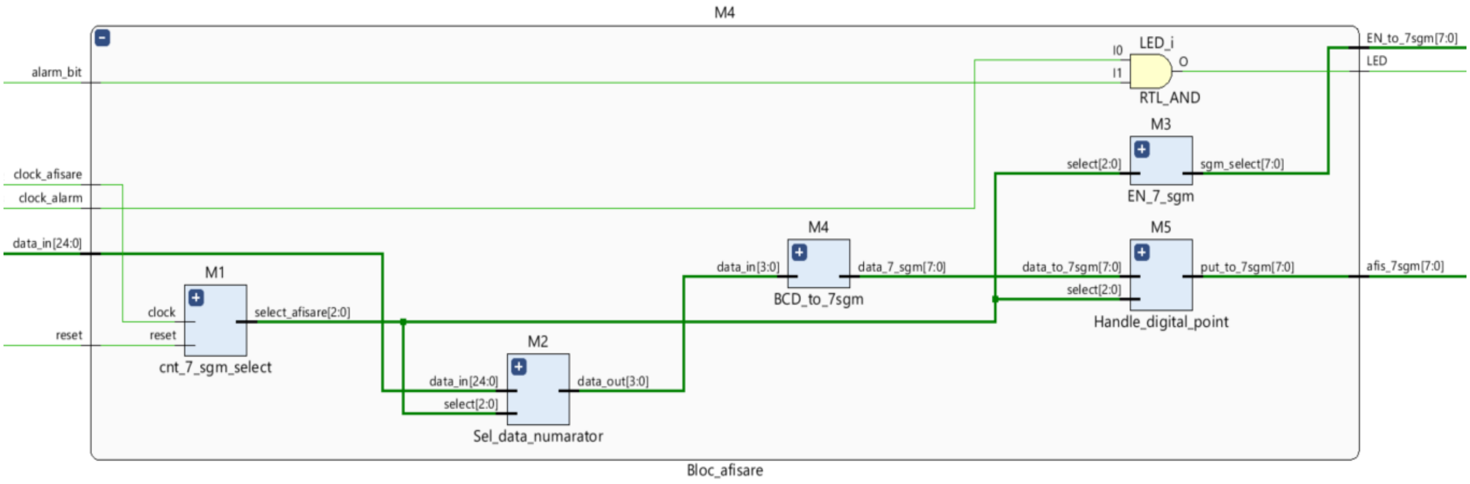


Blocul de afisare

Bloc comprimat =>



Blocul extins



Semnale de intrare :

-alarm bit : aceasta intrare vine de la blocul de control. Aceasta intrare este valida la ora fixa pe o perioada de de 2s. Impreuna cu aceasta intrare si clock_alarm se vor combina cu o poarta SI si se va afisa alarm ape LED.

-clock afisare : intrare de clock care comanda toate circuitele secventiale din interiorul blocului de afisare.

-**clock alarma :** intrarea de clock pentru generarea alarmei. Aceasta intrare este cu frecventa data in specificatii.

- data_in[24:0] : intrarea de date ce este conectata la blocul de numarare. Aceasta intrare va fi afisata pe afisaj.Aceasta intrare contine : a) data_out_afisare[24] = AM_PM_BIT;
b)data_out_afisare[23:20] =Ore_zeci;
c)data_out_afisare[19:16] = Ore_unitati;
d)data_out_afisare[15:12] = minute_zeci;
e)data_out_afisare[11:8] = minute_unitati;
f) data_out_afisare[7:4] = secunde_zeci;
g) data_out_afisare[3:0] = secunde_unitati;

-reset : intrare asincrona pentru resetarea circuitelor secventiale.

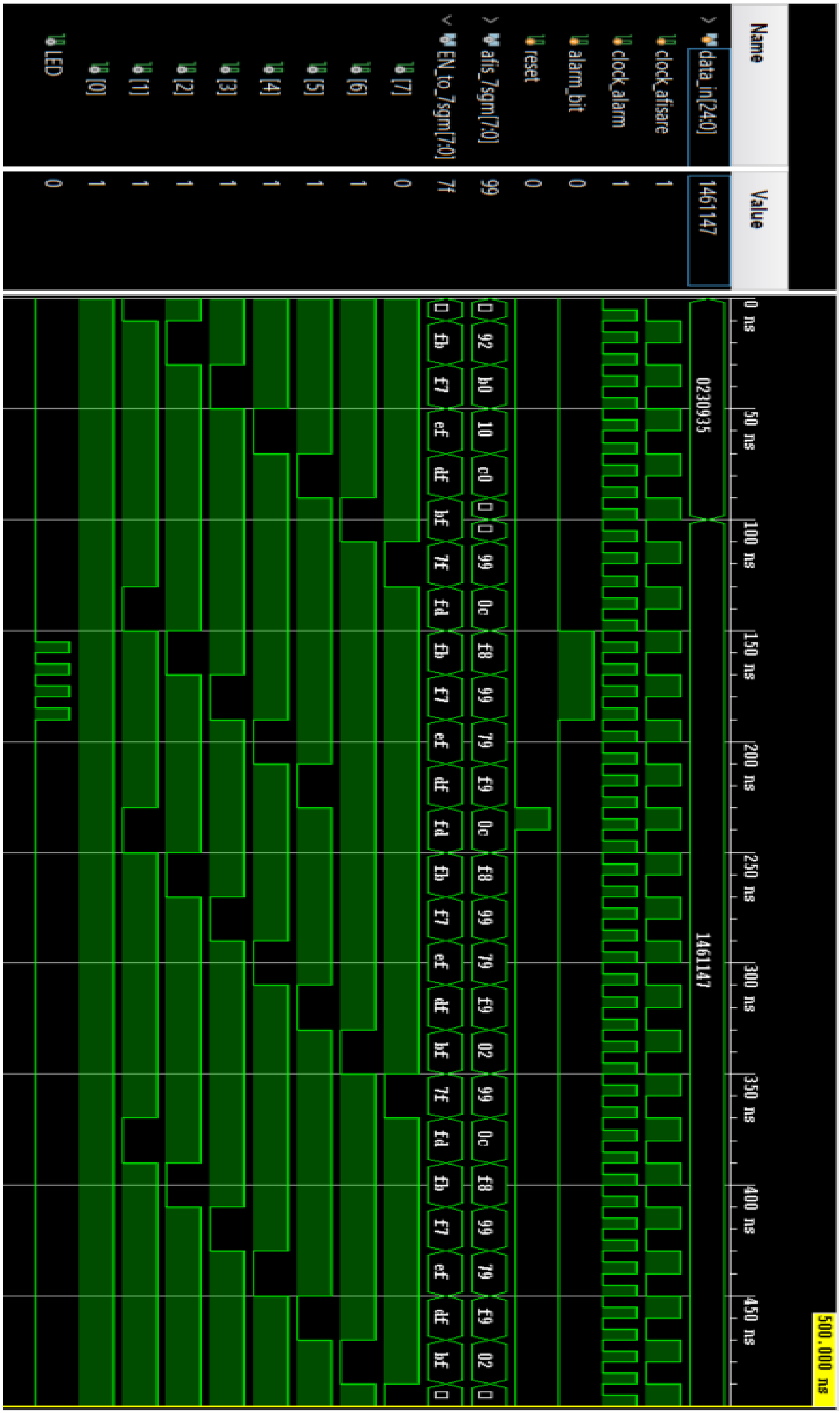
Semnale de iesire :

- En_to_7sgm[7:0] : aceasta iesire este folosita la afisarea multiplexata. Este utilizata la controlul fiecarui afisaj de 7 segmente care este in numar de 8 afisaje de tip 7 segmente.

- LED : Ledul este folosit pentru afisarea alarmei. Acesta este controlat PWM cu frecventa 3Khz.

-afis_7sgm[7:0] : iesire de date ce contine datele ce vor fi afisate pe afisajul 7 segmente.

Rezultate simulare pentru blocul de afisare :


Name
Value
data_in[24:0] 1461147
clock_afisare 1
clock_alarm 1
alarm_bit 0
reset 0
afis_7sgm[7:0] 99
EN_to_7sgm[7:0] 7f
[7] 0
[6] 1
[5] 1
[4] 1
[3] 1
[2] 1
[1] 1
[0] 1
LED 0
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
450 ns
500.000 ns
0230935
1461147
92 b0 10 c0 99 0c f8 99 79 f9 0c f8 99 79 f9 02 99 0c f8 99 79 f9 02
fb f7 ef df bf 7f fd fb f7 ef df fd fb f7 ef df bf 7f fd fb f7 ef df bf