

**Facultatea de Automatica și Calculatoare**

**Alarmă**

Îndrumător: Student:Ilieș Oana-Elena

ing. Fleger Dan Grupa:30215

28.05.2022

**Cuprins**

1.Rezumat

2.Introducere

2.1. Obiectivul proiectului

2.2.FPGA

3.Fundamentare teoretică

3.1.VHDL-limbaj de descriere hardware

3.2.Componente utlizate

3.2.1 Seven Segment Display

3.2.2 Divizor de frecvență

3.2.3 Generator de monoimpuls

3.3.Placa Nexys-4

4.Proiectare și implementare

5.Rezultate experimentale

6.Concluzii

Bibliografie

**1.Rezumat**

Proiectul prezinta sinteza de circuit a unei alarme cu timer crescator si descrecator avand ca scop principal afisarea orei folosind afisoarele digitale cu 7 segmente „seven segment display”. Pentru implementarea hardware a acestui proiect se folosește placa Nexys4, Artrix-7 FPGA. De îndată ce FPGA este pornit, ceasul pornește, iar acesta și clock-ul propriu al plăcii genereaza semnale de sincronizare pentru afișarea orei, odata ce butonul de start-stop este activat. Ceasul digital afiseaza in mod implicit timpul de functionare, iar functia de alarma poate fi setata utilizand time set-ul care consta in resursele externe ale placii.

**2.Introducere**

2.1 Obiectivul proiectului

Obiectivul principal al acestei sinteze de circuit este funcția de alarmă și afișarea digitală a orei folosind afișoarele cu 7 segmente de pe FPGA. Proiectul este implementat sub forma unui meniu de selectie a functionalitații prin utlizarea butoanelor, fiecărei selecții îi se asociaza propriul buton de activare a stării dorite. Meniul dispune de 3 selectii:timer crescator, timer descrescator și functie de alarmă. Ceasul digital este conceput sub un format de 24 de ore care fișează ora sub formă de ore:minute:secunde.Funcția de alarmă dispune de un mod de selecție a timpului dorit folosind swich-urile plăcii, iar în momentul în care timpul selectat de utilizator este echivalent cu timpul real afișat pe placă se activeză alarma, “zgomotul” produs in acel moment consta in aprinderea unui led.

Proiectul a fost implementat cu ajutorul codului VHDL, care a apărut în anul 1987 și se află în forma actuală încă din anul 1993. Este utilizat în concepția asistată pe calculator (CAD=Computer Aided Design), a circuitelor integrare (de exemplu ASIC) sau, în cazul nostru, pentru configurarea FPGA-urilor

2.2.FPGA

Progresele în tehnologia digitală au fost uimitoare de-a lungul anilor și au dat naștere designului sistemelor digitale, care continuă să servească drept o mare sursă de ajutor și confort pentru omenire în multe feluri. În zilele noastre, numeroase aplicații în electronică și alte tehnologii folosesc tehnici digitale pentru a efectua operațiuni care cândva au fost efectuate prin metode analogice. Sistemele digitale ofera versatilitate și superioritate față de metodele analogice datorita faptului că nu sunt afectate de fluctuații false ale tensiunii, au o mai mare precizie și acuratețe și pot stoca miliarde de biți de informații într-un spațiu relativ mic. Mai mulți autori au făcut diferite cercetări pentru a minimiza cerințele de putere în fabricarea unui cip integrat și pentru a avea flexibilitate și performanță ridicate. Dezvoltarea plăcii FPGA este o parte a cercetării.

Field-programmable gate array (FPGA) este un circuit integrat proiectat pentru a fi configurat de către un client sau un proiectant după fabricație - de unde și termenul programabil în câmp. Configurația FPGA este în general specificată folosind un limbaj de descriere hardware (HDL), similar cu cel utilizat pentru un circuit integrat specific aplicației (ASIC).

.

**3.Fundamentare teoretică**

**3.1 VHDL-limbaj de descriere hardware**

VHDL nu este un limbaj de programare, ci un limbaj de descriere a sistemelor electronice hardware pornind de la structura lor modularăşi de la interconexiunile dintre acestea. El a fost definit şi integrat în rândul instrumentelor de CAD (Computer-Aided Design) din domeniul electronicii, pentru a introduce o metodologie riguroasă de proiectare în ciclul de dezvoltare al sistemelor hardware. VHDL a devenit un limbaj industrial standardizat, utilizat pentru descrierea hardware de la nivelul abstract până la nivelul concret. VHDL a fost rapid asimilat ca un mediu universal de comunicaţie în proiectare. Toţi producătorii de staţii de lucru şi de software CAE (Computer-Aided Engineering) îşi standardizează produsele pentru a avea intrări şi ieşiri standard VHDL. Aceste produse includ software pentru simulare, sintezăşi trasare de cablaj imprimat.

În VHDL, orice componentă (în sensul software) este descrisă sub două aspecte:

-interfața cu lumea exterioară, descrisă într-o secțiune numită *entitate*

-implementarea în sine, descrisă într-o secțiune numită *arhitectură*

În VHDL există trei clase de obiecte: constantele, variabileleşi semnalele. Constantele au o valoare fixă definită o dată pentru totdeauna, cel mai târziu după o fază de iniţializare numităelaborare. Variabilele au o valoare modificabilă prin atribuire. Constantele şi variabilele sunt obiecte ce pot fi întâlnite şi în limbajele de programare. Semnalele sunt specifice limbajelor de descriere hardware. Ele modelează informaţia care tranzitează prin fire, magistrale sau între componentele hardware.

**3.2.Componente utlizate**

3.2.1 Seven Segment Display

În schema de codificare Binary Coded Decimal (BCD), fiecare dintre numerele zecimale (0-9) sunt reprezentate de modelul său binar echivalent care este în general de 4 biți.Întrucât, afișajul cu șapte segmente este un dispozitiv electronic care constă in 7 diode emițătoare de lumină, led-uri, dispuse într-un model definit (catod comun sau tip anod comun)acesta este folosit pentru a afișa numere hexazecimale in BCD prin activarea corespunzatoare a catozilor.

|  |  |
| --- | --- |
|  |  |

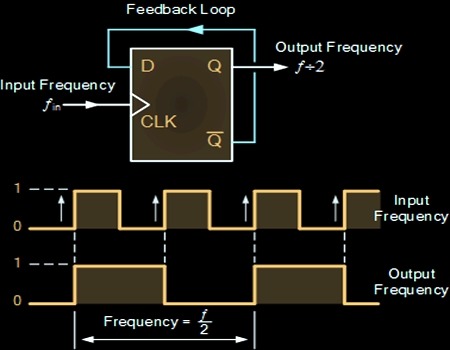
-după cum se observă din imagine, afișoarele au cele 7 leduri reprezentând segmentele, si un simbol **dp** reprezentând punctul decimal

-fiecare are un anod activ pe 0 (funcționează numai atunci când sunt setate pe 0 logic)

|  |  |
| --- | --- |
| Afisorul pe 7 segmente  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity displ7seg is  Port ( Clk : in STD\_LOGIC;  Rst : in STD\_LOGIC;  Data : in STD\_LOGIC\_VECTOR (31 downto 0);  An : out STD\_LOGIC\_VECTOR (7 downto 0);  Seg : out STD\_LOGIC\_VECTOR (7 downto 0));  end displ7seg;  architecture Behavioral of displ7seg is  constant CNT\_100HZ : integer := 2\*\*20;  signal Num : integer range 0 to CNT\_100HZ - 1 := 0;  signal NumV : STD\_LOGIC\_VECTOR (19 downto 0) := (others => '0');  signal LedSel : STD\_LOGIC\_VECTOR (2 downto 0) := (others => '0');  signal Hex : STD\_LOGIC\_VECTOR (3 downto 0) := (others => '0');  begin  divclk: process (Clk)  begin  if (Clk'event and Clk = '1') then  if (Rst = '1') then  Num <= 0;  elsif (Num = CNT\_100HZ - 1) then  Num <= 0;  else  Num <= Num + 1;  end if;  end if;  end process;  NumV <= CONV\_STD\_LOGIC\_VECTOR (Num, 20);  LedSel <= NumV (19 downto 17);  An <= "11111110" when LedSel = "000" else  "11111101" when LedSel = "001" else  "11111011" when LedSel = "010" else  "11110111" when LedSel = "011" else  "11101111" when LedSel = "100" else  "11011111" when LedSel = "101" else  "10111111" when LedSel = "110" else  "01111111" when LedSel = "111" else  "11111111"; | Hex <= Data (3 downto 0) when LedSel = "000" else  Data (7 downto 4) when LedSel = "001" else  Data (11 downto 8) when LedSel = "010" else  Data (15 downto 12) when LedSel = "011" else  Data (19 downto 16) when LedSel = "100" else  Data (23 downto 20) when LedSel = "101" else  Data (27 downto 24) when LedSel = "110" else  Data (31 downto 28) when LedSel = "111" else  X"0";  Seg <= "11111001" when Hex = "0001" else -- 1  "10100100" when Hex = "0010" else -- 2  "10110000" when Hex = "0011" else -- 3  "10011001" when Hex = "0100" else -- 4  "10010010" when Hex = "0101" else -- 5  "10000010" when Hex = "0110" else -- 6  "11111000" when Hex = "0111" else -- 7  "10000000" when Hex = "1000" else -- 8  "10010000" when Hex = "1001" else -- 9  "10001000" when Hex = "1010" else -- A  "10000011" when Hex = "1011" else -- b  "11000110" when Hex = "1100" else -- C  "10100001" when Hex = "1101" else -- d  "10000110" when Hex = "1110" else -- E  "10001110" when Hex = "1111" else -- F  "11000000"; -- 0  end Behavioral; |

3.2.2 Divizor de frecvență

Divizorul de frecvență este o componentă simplă al cărui obiectiv este reducerea frecvenței de intrare. Componenta este implementată prin utilizarea factorului de scalare și a unui contor ce incetineste ceasul placii, in cazul de fata, la o secunda. Factorul scalar pentru divizor de frecventa este gasit prin impartirea frecventei de intra pe 100mHz/1Hz, deoare placa utilizata functioneaza pe 100 mHz. Dar pentru a determina valoarea necesara oferita acestuia se face imparitrea la doi, deoarele semnalul de ceas este comutat din frontul ascendent pe frontul descendent la fiecare jumatate de secventa. Deci valoarea primita de semnalul de ceas divizat este: 50000000.Pentru o mai buna functionare se utilizeaza un divizor de frecventa cu functia de reset ce ulterior primeste butonul de reset. Astfel putem vedea cu ochiul liber schimbarile efectuate pe afisoarele folosite. Ca si intrare avem un clock, iar ca si iesire avem noul clock, cel divizat.



3.2.3 Generator de monoimpuls

Acesta fixeaza continuu nivelul logic al butonului si stocheaza intotdeauna ultimele doua niveluri logice ale acestuia. In momentul in care butonul este apasat se genereaza pe langa semnalul de ceas mici impulusri ce impiedica buna functionare a butonelor. Practic acesta reprezinta un stabilizator, o practica de optimizare pentru rezultatul oferit de buton.

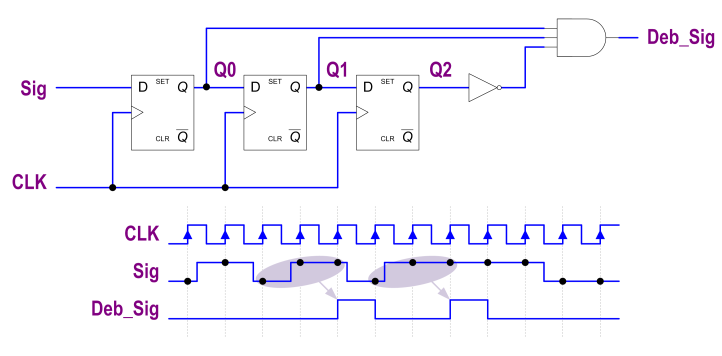
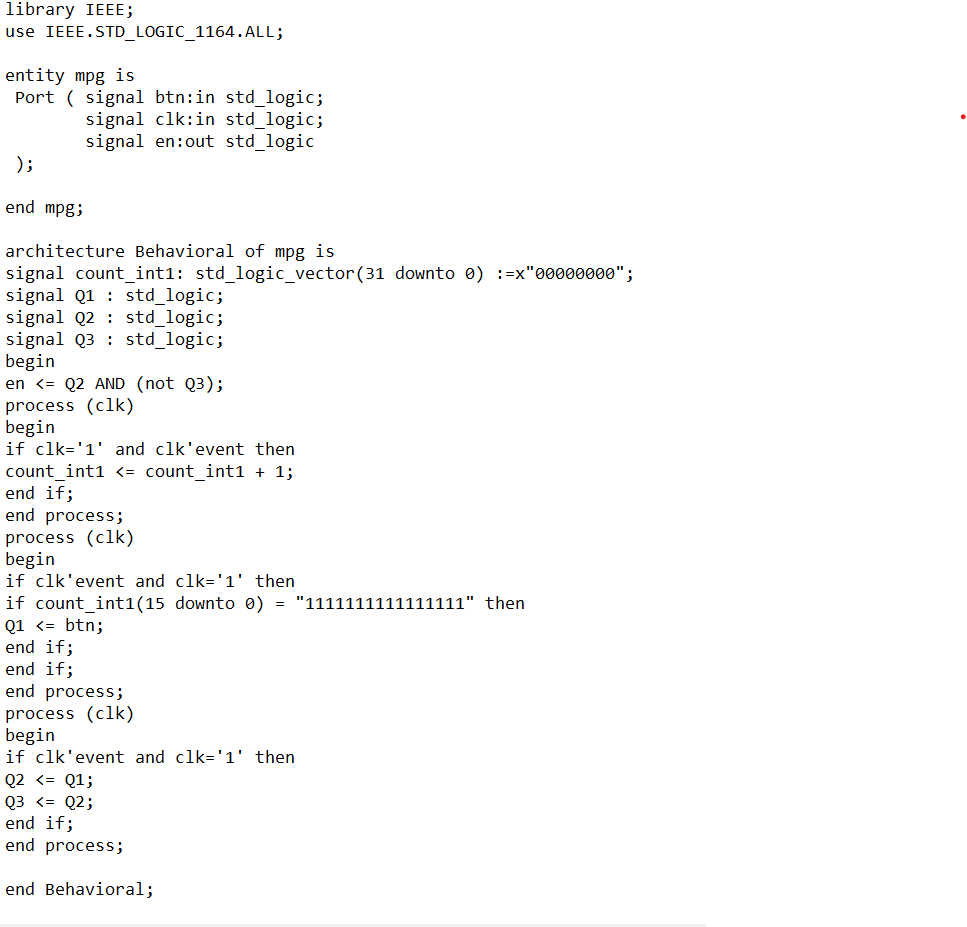


Figura de mai sus descrie rezultatul ieșirii de debouncing, atunci când butonul este apăsat

Pentru implementarea acestui integrat este necesara utilizarea a 3 bistabile D si a unei porti SI.

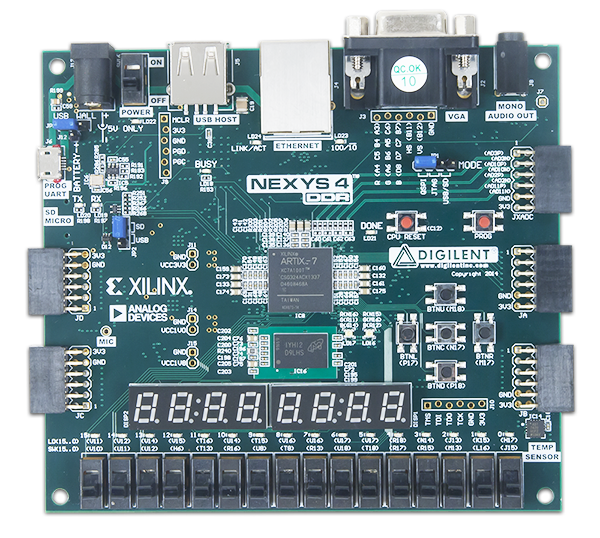


Codul VHDL de mai sus descrie metoda pentru a evita deblocarea butonului de apăsare. În loc să atribuiți datele de intrare ale butonului de apăsare la ieșire, aici 3 semnale au fost atribuite Q1, Q2 și Q3 care sunt responsabile pentru atribuirea datelor de ieșire.

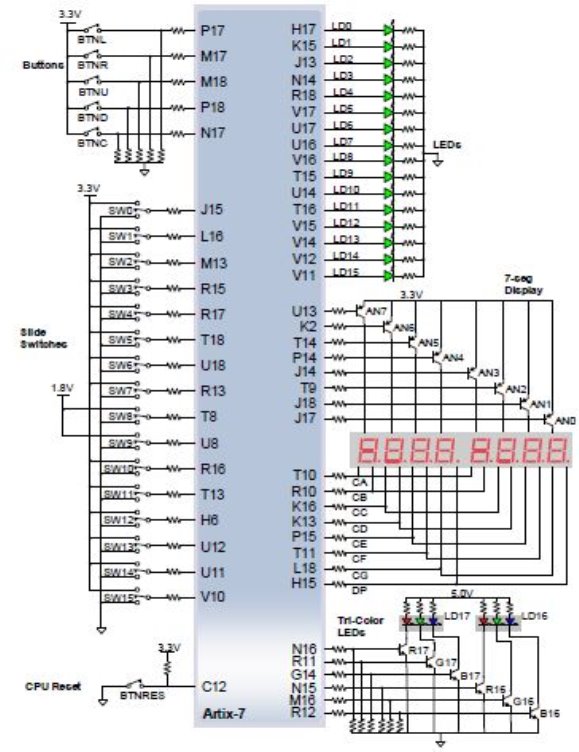
.

**3.3 Placa Nexys 4**

Placa Nexys4 DDR este o platformă de dezvoltare a circuitelor digitale completă, gata de utilizat, bazată pe cel mai recent Artix-7 Field Programmable Gate Array (FPGA) de la Xilinx. Cu memoriile externe de mare capacitate, colecția de USB și diverse alte porturi, găzduiește modele care variază de la circuite combinaționale introductive până la cele mai puternice procesoare încorporate, ceea ce reprezintă un avantaj suplimentar. Placa poate fi utilizată pentru o gamă largă de modele datorită mai multor periferice încorporate, cum ar fi accelerometrul, senzorul de temperatură, microfonul digital MEM, amplificatorul difuzorului și dispozitivele I/O.



Placa Nexys4 DDR încorporează un oscilator de 100 MHz. Ceasul informativ al FPGA poate conduce MMCM-uri sau PLL-uri pentru a crea cronometre de diferite frecvențe și cu conexiuni de etapă cunoscute care ar putea fi necesare pe tot parcursul unui proiect. Placa încorporează 16 swich-uri, care creează surse constante de informații ridicate sau scăzute, bazându-se pe poziția lor, 5 butoane, 8 anozi , 8 catozi pentru afisarea pe SSD-uri, 2 LED-uri tricolore și 16 LED-uri individuale. Butoanele și swich-urile sunt asociate cu FPGA prin intermediul rezistențelor de aranjare pentru a preveni daunele cauzate de scurtcircuite accidentale (s-ar putea produce un scurtcircuit dacă un stick FPGA desemnat la un buton sau un swich a fost caracterizat întâmplător ca un randament. Cele 16 LED-uri individuale de înaltă eficiență sunt conectate prin anod la FPGA prin intermediul rezistențelor de 330 ohm, astfel încât se vor porni atunci când se aplică o tensiune logică înaltă la pinul lor I/O.



**4.Proiectare și Implementare**

Pentru implementarea acestui proiect am ales ca si metoda de lucru utilizarea FSM-ului(finite state machine) pentru o functionare mai sigura si mai clara in ceea ce priveste legarea starilor si a componentelor utilizate.

Am optat pentru un FSM cu 3 procese, mai exact un process de actualiare a starii curente, un proces de determinare a starii urmatoare si un proces de determinare a iesirilor.De asemenea au fost necesare si alte procese spefice functuionalitatii dorite.

Am integrat in aceasta sinteza si procesul divizorului de frecventa, care incetineste functionalitatea placii la o secunda.

Entitate:

*-signal clk:in std\_logic*

*-signal rst:in std\_logic*

*-signal sw:in std\_logic\_vector(15 downto 0)*

*-signal btn\_on\_off:in std\_logic*

*-signal btn\_directie:in std\_logic*

*-signal btn\_alarma:in std\_logic*

*-signal date:out std\_logic\_vector(23 downto 0)*

*-signal led:out std\_logic\_vector(3 downto 0)*

Semnalul de reset preia ulterior butonul destinat acestei functionaliati, N17(butonul din mijloc).

**Declarare stari**: *type stari is (meniu,ceas\_on\_off,directie,alarma);*

*1.Meniu*

-reprezinta o stare initiala, de asteptare, care se modifica prin apasarea unui buton in functie de starea urmatoare dorita.In in urma apasari butonului “btn\_on\_off” starea urmatoare va fi “ceas\_on”, dar acest buton are functionaliate multipla, deoarece activeaza sau dezactiveaza ora afisata in orice stare s-ar afla fsm-ul. Directia de functionare a ceasului poate fi schimbata prin apasarea butonului “btn\_directie”, buton de count down, care modifica functionalitatea din timer crescator in timer descrescator si realizeaza trecerea in starea de “directie”, iar butonul “btn\_alarma” realizeaza trecerea in starea de “alarma”.

*2.Ceas\_on*

-in aceasta stare ora este afisata sub format ore:minure:secunde, pentru fiecare dintre acestea au fost declarate in arhitectura semnale intermediare care ulterior vor fi preluate de catre semnalul de iesire “date”.Secundele si minutele au fost codificate pe 6 biti, iar orele pe 5 biti.Pentru o afisare optime pe SSD-uri s-au realizat 3 procese care cuprinde toate cazurile, secunde(0-59), minute(0-59) si ore(0-23) si prin care s-a realizat o convertie in “date” a acestora.

Exemple pentru afisarea secundei “1”:

“when "000001"=>date(7 downto 4)<="0000";date(3 downto 0)<="0001";”

-in momentul in care secundele depasesc valoarea 59 minutele sunt incrementate, iar secundele preiau valoara 0 si sunt in continuare incrementate, analog in cazul minutelor, iar cand ora depaseste 23:59:59 ceasul preia valoarea 00:00:00 intrand astfel intr-un ciclu infinit care se reseteaza odata la 24 de ore.

*3.Directie*

-starea de count down, decrementare, cu semnal intermediar “dir”

-in momentul in care secundele au valoare mai mica decat 0, minutele se decrementeaza si preiau valoarea 59, analog minute, iar in momentul in care ceasul preia valoarea 00:00:00 ora va devini 23:59:59, ciclu infinit cu logica inversa ceas\_on.

*4.Alarma*

-in acest caz se utilizeaza ca semnale intermediare “ore1” si “minute1”, codificate pe 5, respective 4 biti.Valoarea acestora se selecteaza prin activarea swich-urilor de pe placa, ore1 preia sw(15 downto 11), iar minute 1 sw(10 downto 5).In momentul in care semnalul minute1=minute si semnalul ore1=ore se declaseaza alarma.

**Entitate top:nexys4**

-este bazata pe resursele placii, preluate din fisierul de constrangere

*signal clk:in std\_logic;*

*signal btn:in std\_logic\_vector(4 downto 0)*

*signal sw:in std\_logic\_vector(15 downto 0)*

*signal led:out std\_logic\_vector(15 downto 0)*

*signal cat:out std\_logic\_vector(7 downto 0)*

*signal an:out std\_logic\_vector(7 downto 0)*

-acestor semnale le sunt asociate porturile de intrare si iesire din entitatile utilizate(MPG, displ7seg si FSM)

-exemplu:buton de reset

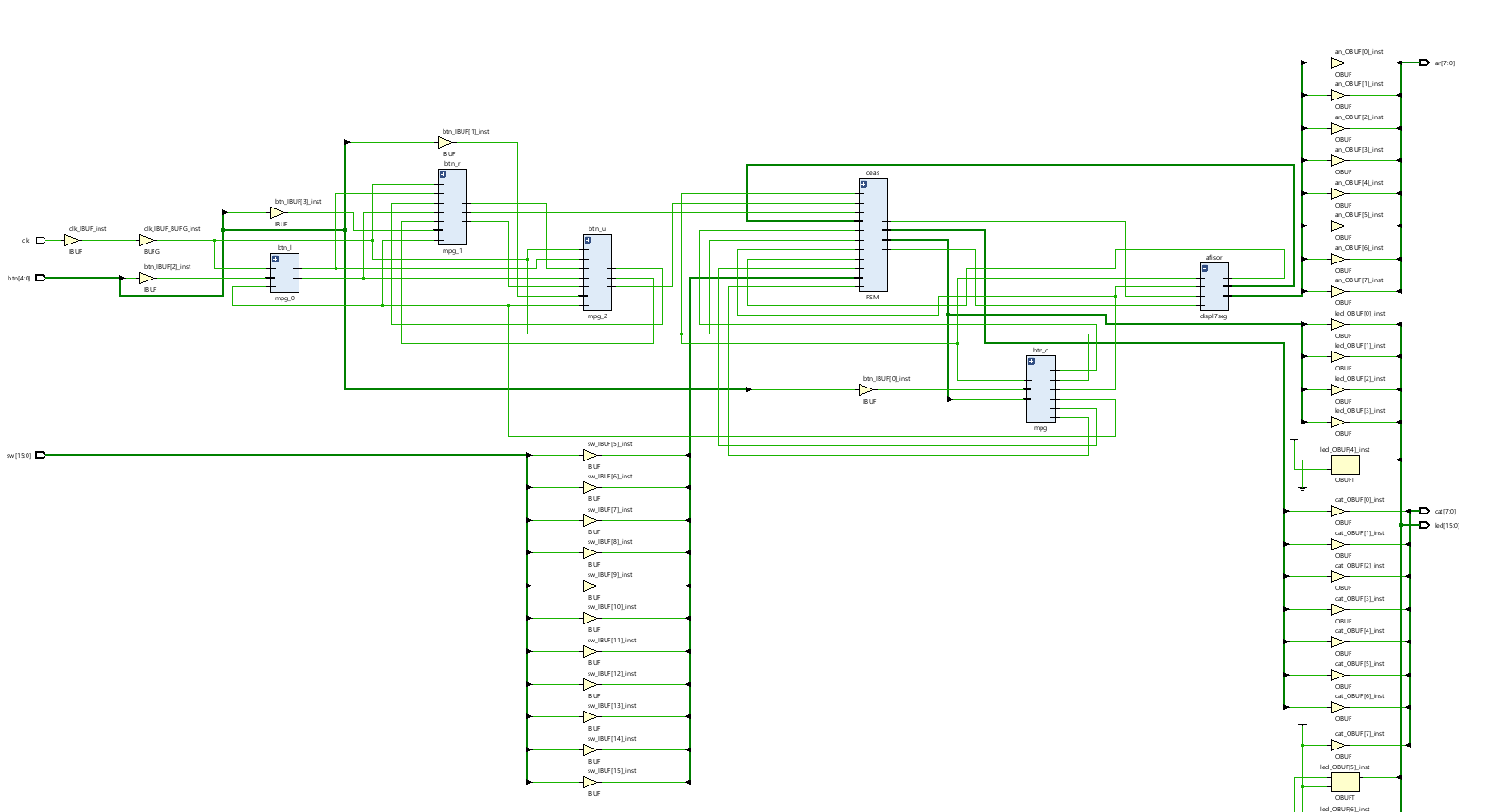
*btn\_c:entity WORK.mpg port map*

*(btn=>btn(0),*

*clk=>clk,*

*en=>btn\_rst);*

Schema generata in Vivado in sectiunea “Schematic”



**5.Rezultate experimentale**

*1.Functia de ceas digital(incrementare)*

-activare prin PIN M18(buton 1/up)

-semnalare prin J13(led 13)

Prima stare selectata este reprezentata de timer-ul crescator, cum bine se poate observa in momentul in care secundele depasesc valoarea de 59 minutele se incrementeaza, iar secundele iau valoarea 0, urmand ca apoi odata la o secunda sa se incrementeze iar valoarea secundelor.

|  |  |
| --- | --- |
| C:\Users\Oana\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot 2022-05-28 125607.png | C:\Users\Oana\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot 2022-05-28 125529.png |

*2.Decrementare*

-activare prin P17(buton 2/left)

-semnalare prin K15(led 14)

Cea de-a doua stare este reprezentata de timer-ul descrescator care descrementeaza valoare afisata pe SSD-uri odata la o secunda.

|  |  |
| --- | --- |
|  | C:\Users\Oana\AppData\Local\Microsoft\Windows\INetCache\Content.Word\dec2.png |

*3.Alarma*

-activare prin M17(buton 3/right)

-semnalare prin H17(led 15)

Cum am mai precizat in momentul in care timpul selectat de pe swich-uri este egal cu cel afisat pe placa se declanjeaza alarma.Este necesara si activarea decrementarii.

|  |  |
| --- | --- |
|  |  |

**6.Concluzii**

# *Posibilități de dezvoltare ulterioară*

Proiectul dat poate fi dezvoltat prin adăugarea de noi funcționalități ca:

-inregistrarea orei dorite pentru functia de alarma pe o memorie ROM

-utilizarea unei componente precum difuzorul parametric, care semnaleaza activarea alarmei in momentul de echivalenta a timpului selectat cu cel real afisat pe placa

-afisarea datei, swich de comutare intre ora si data

-dezvoltarea componentei manuale prin adăugarea unui buton Count up

-utilizarea unui senzor de proximitate, shimbarea functionalitatii din reminder in alerta prin utilizarea unui buton, afisarea pe placa a distantei dintre obiectul sesizat si senzor, in momentul in care distanta e minima, adica 0, se declanseaza alarma sub forma unui joc de leduri

Bibliografie:

<https://www.nandland.com/vhdl/tutorials/index.html>

<https://vhdlguru.blogspot.com/p/example-codes.html>

<https://digilent.com/reference/programmable-logic/nexys-4/start>

1. Ahmed MA, Rani DE, Sattar SA. FPGA based high speed memory BIST controller for embedded applications. Indian Journal of Science and Technology. 2015 Dec; 8(3):1–8. DOI: 10.17485/ijst/2015/v8i33/76080.

2. Kumar MS, Inthiyaz S, Mounica J, Bhavani M, Adidela CN, Endreddy B. FPGA implementation by using XBee Transceiver. Indian Journal of Science and Technology. 2016 May; 9(1):1–6. DOI: 10.17485/ijst/2016/v9i17/93032.

3. Navamani JD, Vijayakumar K, Lavanya A. FPGA-based digitally controlled Isolated Full-Bridge DC-DC Converter with Voltage Doubler (IFBVD). Indian Journal of Science and Technology. 2016 Apr; 9(16):1–7. DOI: 10.17485/ijst/2016/v9i16/76672.