**SISTEM INTELIGENT PENTRU MONITORIZARE UMPLERE PARCARE**

**CUPRINS**

1. **SPECIFICAȚIE**
2. **DESCRIERE SCHEMA BLOC**
3. **PROIECTARE ȘI IMPLEMENTARE**
4. **SIMULAREA ÎN VHDL**
5. **SEMNIFICAȚIA INTRĂRILOR / IEȘIRILOR**
6. **UTILIZARE ȘI REZULTATE**
7. **POSIBILITĂȚI DE DEZVOLTARE ULTERIOARĂ**

**SPECIFICAȚIE**

SISTEM INTELIGENT UMPLERE PARCARE

**Cerința proiectului este următoarea:**

Proiectați un sistem inteligent pentru monitorizare umplere parcare organizată ca un dreptunghi, cu un număr par de locuri. Sistemul dispune de un afișaj și controlează două intrări / ieșiri. Fiecare intrare / ieșire are 2 senzori optici astfel poziționați încât să permită detecția sensului de mers a mașinii (prin ordinea de activare) cât și prezența acesteia și nu a corpurilor mai mici (prin activarea simultană).

Inițial sistemul nostru este într-o stare inactivă. Odată cu activarea butonului de Reset, el ajunge într-o stare cunoscută și de aici în colo începe funcționarea, acestă făcându-și treaba. Acest buton de Reset are rolul de inițiere, de pornire al sistemului.

**DESCRIERE SCHEMA BLOC**

comparator

comparator

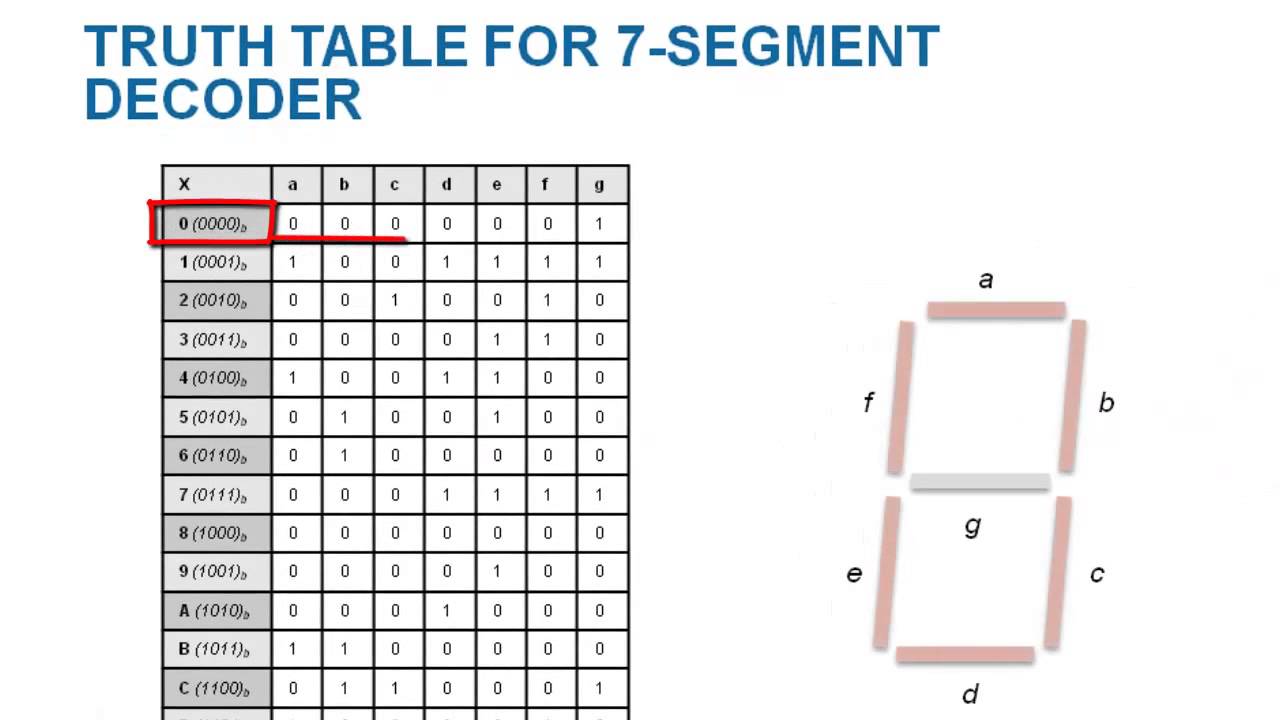
Numărător special reversibil

BCD 7 segmente

În schița de mai sus putem vedea că fiecare comparator are câte 2 intrări. Eu le-am numit intrare1, ieșire1, intrare2, ieșire2. Fiecare dintre cele 2 intrări ale comparatorului reprezintă senzorii optici pe care îi are una dintre intrările parcării noastre. În funcție de ordinea de activare a acestor senzori putem să aflăm sensul de mers al mașinii (intră sau iasă din parcare). Mai jos o să vedem exact si secvențele corespunzătoare intrării sau ieșirii. De exemplu, pentru o intrare, avem ordinea 00 ceea ce înseamnă că nu se întâmplă nimic. Dacă după 00, vine secvența de 10 înseamnă ca este posibil ca o mașina să dorească să intre in parcare. Dar ca să fim siguri că este o mașină și nu este vreun obiect, trebuie ca după 10 să vina secvența 11, în acest fel fiind siguri ca obiectul este o mașină.

Numărătorul l-am numit special deoarece nu este un numărător oarecare. Este un numărător creat de mine astfel încât să poată face mai multe chestii, cum ar fi: incrementare / decrementare cu 1 și incrementare / decrementare cu 2. Am avut nevoie de creearea unui astfel de numărător deoarece se presupune că parcarea noastră nu respectă regulile de circulație ceea ce înseamnă că pot intra, respectiv ieși 2 mașini din parcare în același timp, iar fără acest numărător incrementarea si decrementarea cu 2 a numărului de locuri din parcare era aproape imposibil.

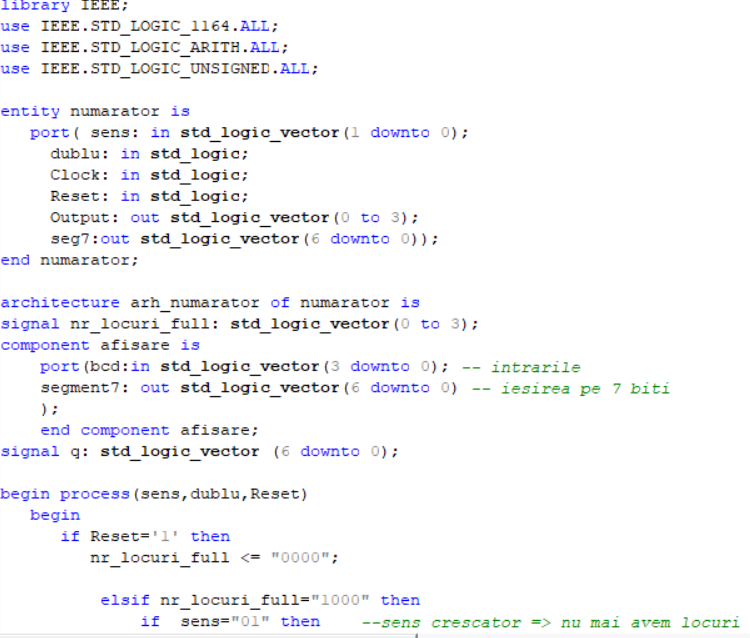
Cea de-a treia componentă a schiței noastre este un afișor BCD 7 segmente. Acesta primește rezultatul de la numărător (numărul de locuri ocupate) și îl afișează. De precizat este faptul că afișorul nostru funcționează pe 0 logic. Ca să avem o idee despre cum arată un BCD 7 segmente, o să atașez mai jos o poză.

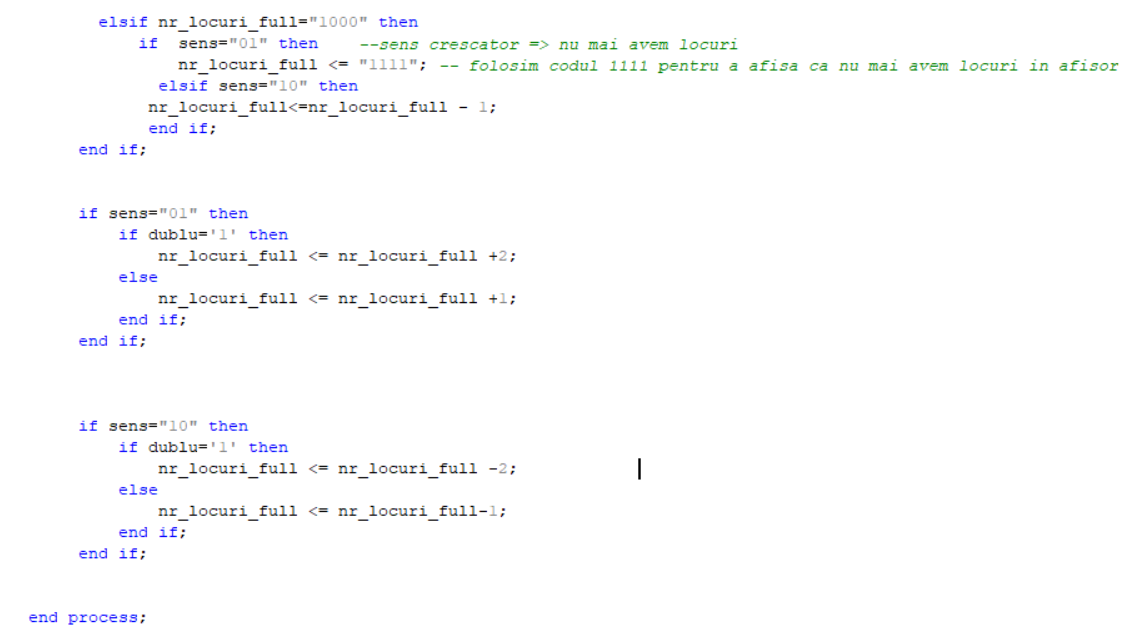


**PROIECTARE ȘI IMPLEMENTARE**

**PROIECTARE COMPONENTE**

**COMPONENTA NUMĂRĂTOR**





Numărătorul nostru are 4 intrări de tipul in și 2 intrări de tipul out. Intrare sens este un vector de tipul std logic care îmi determină ordinea de activare a senzorilor (00,01,10,11) și din acest motiv am avut nevoie de un vector pe 2 biți. Intrarea dublu este 1 doar în momentul în care au intrat sau au ieșit două mașini în același timp, iar numărătorul va trebui să incrementeze / decrementeze cu 2. Când este activ Reset-ul, numărătorul ajunge în starea inițială pe 0000. Output si seg7 sunt ieșirile numărătorului cu ajutorul cărora vom face legătura cu BCD 7 segmente. Componenta afișare este componenta BCD

Să începem prin a studia funcționarea numărătorului nostru. Din declararea procesului, observăm că acesta este sensibil la semnalul sens, dublu, Reset. Făcându-l pe Reset 1, aducem numărătorul în stare inițială, 0000, iar de aici sistemul își face singur treaba. Semnalul nr\_locuri\_full este un semnal pe 4 biți care ne determină numărul de locuri din parcarea noastră.

Ținând cont că trebuie să avem un număr par de locuri, am verificat acest lucru. Dacă numărul de locuri ocupate este 1000(8), iar sensul este crescător, adică o mașină dorește să intre în parcare, afișăm un cod de eroare pe BCD, adică 1111. Dacă avem 8 în numărător iar sensul este descrescător(10) atunci este evident că mașina poate ieși din parcare.

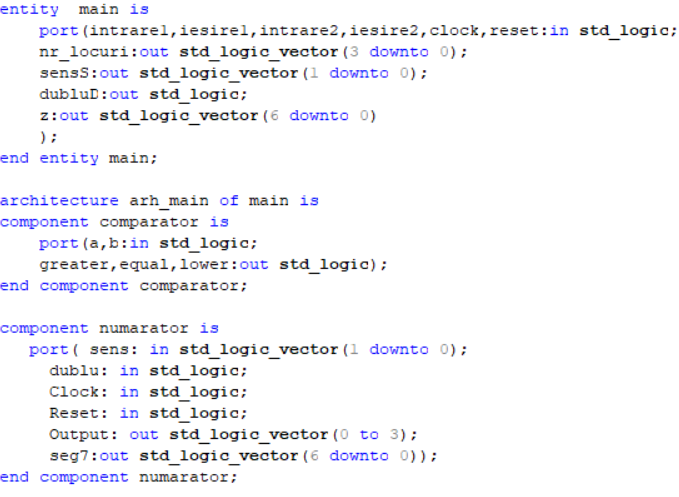
După aceasta verificăm cele 2 cazuri speciale. Dacă sensul este crescător, iar dublu este 1 înseamnă că 2 mașini doresc să intre deodată(câte una pe fiecare intrare) ceea ce înseamnă că semnalul nr\_locuri\_full trebuie incrementat cu 2.Dar dacă semnalul dublu este 0 și sensul este crescător înseamnă că numărătorul trebuie incrementat doar cu 1, nu cu 2 ceea ce înseamnă că doar o mașină a intrat în parcare. Același lucru îl verificăm și dacă sensul este descrescător, iar semnalul dublu este 1. La final, Output primește nr\_locuri\_full și se face legătura către afișorul BCD 7 segmente.

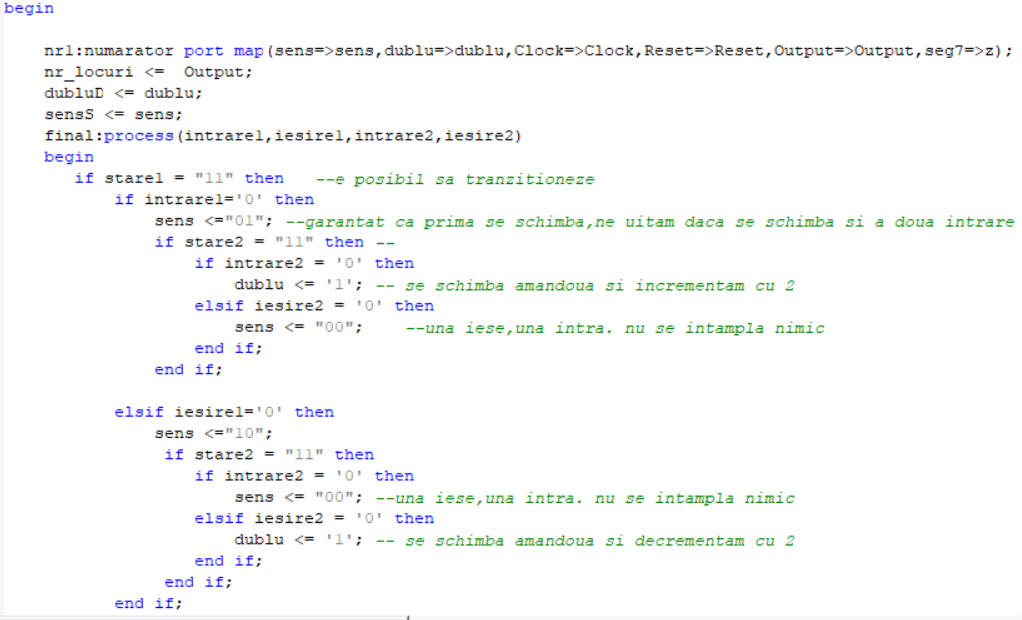
**COMPONENTA AFIȘOR**

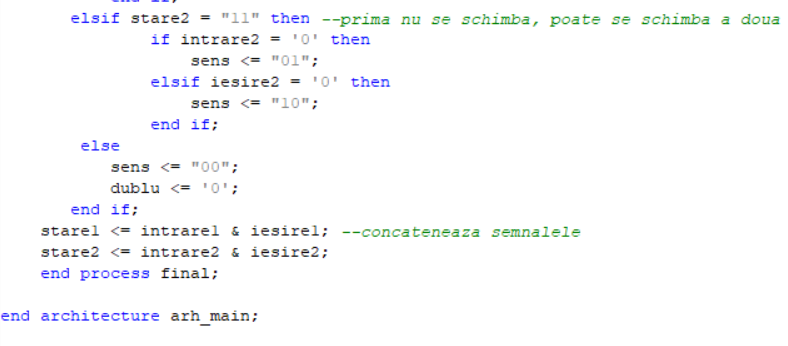


Acesta este un afișor BCD 7 segmente. Are ca intrare numărul de locuri ocupate, acesta fiind pe 4 biți și ne returneaza segement7, acesta fiind pe 7 biți. Fiecare bit din segment 7 este pentru fiecare liniuță din afișor. Acest BCD este sensibil doar la intrarea bcd, adică numărul de locuri ocupate din parcare. În momentul în care numărul de locuri ocupate se schimbă, procesul se reîncepe iar afișorul nostru ne va arăta alte lucruri. Pentru fiecare număr de la 0 la 9 este făcută configurația afișorului în cadrul unui case. Lucrul foarte important este faptul că afișorul este activ pe 0.

**COMPONENTA MAIN**



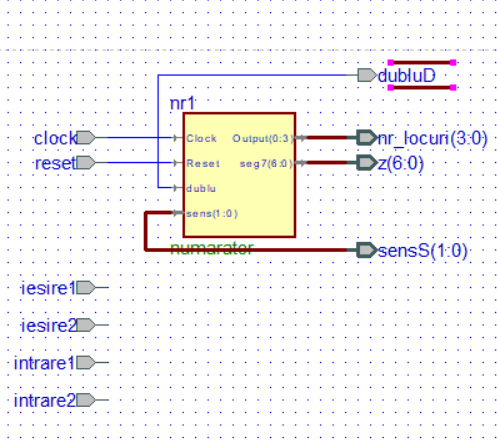




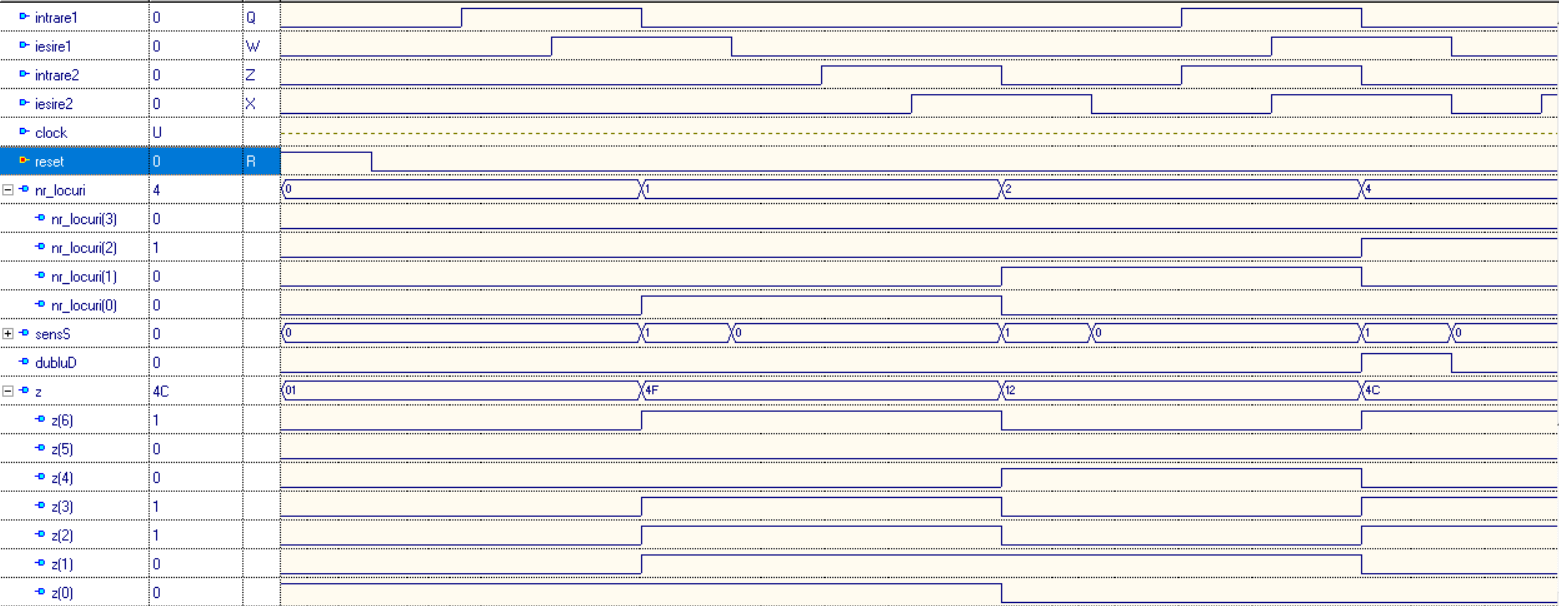
În cadrul componentei main, se leagă componentele iar proiectul prinde contur. În interiorul componentei main avem un proces în interiorul căruia se află mai multe instructiuni de tipul if. Am încercat pe cât posibil sa le indentez, iar acestea să nu fie indendate. Procesul este sensibil la semnalele intrare1, iesire1, intrare2, iesire2 care reprezintă cei 4 senzori de la cele 2 intrări ale parcării. În momentul în care unul dintre aceștia 4 își modifică valoarea, procesul se reia ca să verifice ce schimbări s-au întamplat. Pe noi ne interesează doar momentul în care sensul trece din starea 11 în 01. Adică dacă mașina intră în parcare. Aici am rezolvat acel caz în care e posibil ca un obiect mai mic să activeze unul dintre senzori, iar sistemul nostrum să-l perceapă ca fiind o mașină. În acest prim if, acest lucru îl verificăm, dacă avem o mașină care dorește sa intre în parcare sau avem 2 mașini care doresc să intre deodată în parcare sau dacă o mașină intră, iar una iasă, nu facem nimic deoarece numărul de locuri ocupate e același. +1-1=0.

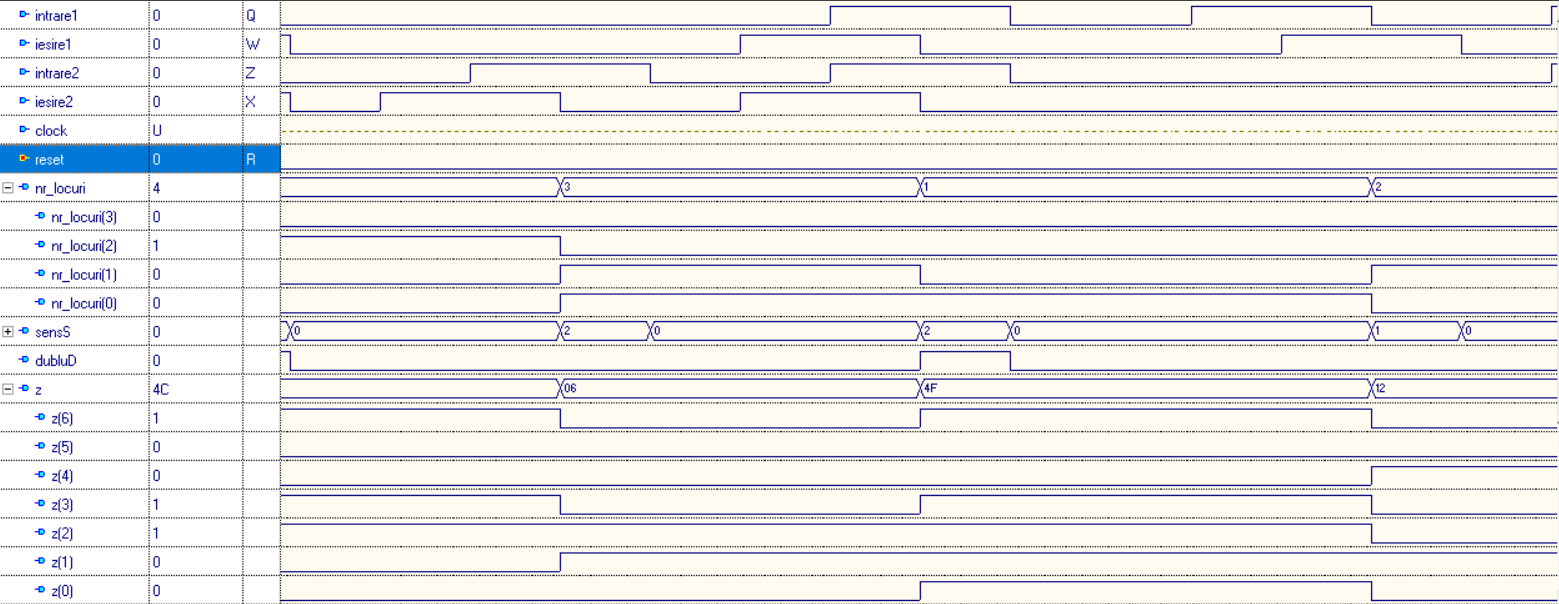
Primul if verifică lucrurile acestea pentru ambele intrări ale parcării. În cel de-al doilea if verificăm dacă suntem în cazul în care 2 mașini vor să iasă din parcare. Semnalul stare1 este pentru cei doi senzori de la intrarea numărul 1, iar stare2 este pentru cei doi senzori de la intrarea numărul 2. Cel de-al doilea if verifică acest lucru tot pentru ambele intrări ale parcării, iar cel de-al treilea if verifică doar pentru intrarea a 2-a.

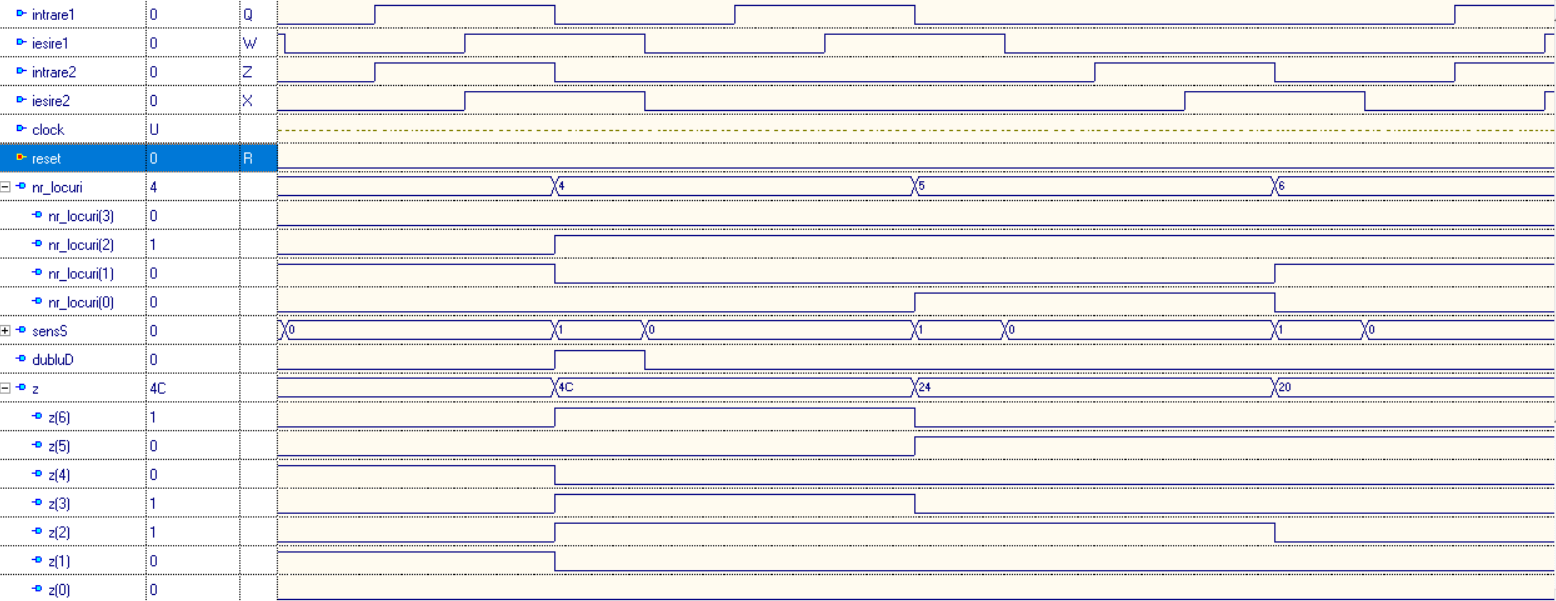
Dacă nu suntem în nici unul din aceste cazuri înseamnă că sensul nostrum este 00, adică nu este active nici un sensor iar valoarea dublu este 0 deoarece nu este nevoie să incrementăm sau decrementăm cu 2 numărul de locuri ocupate. În final, stare1 și stare2 concatenează semnalele celor 2 senzori optici ai fiecărei intrări.

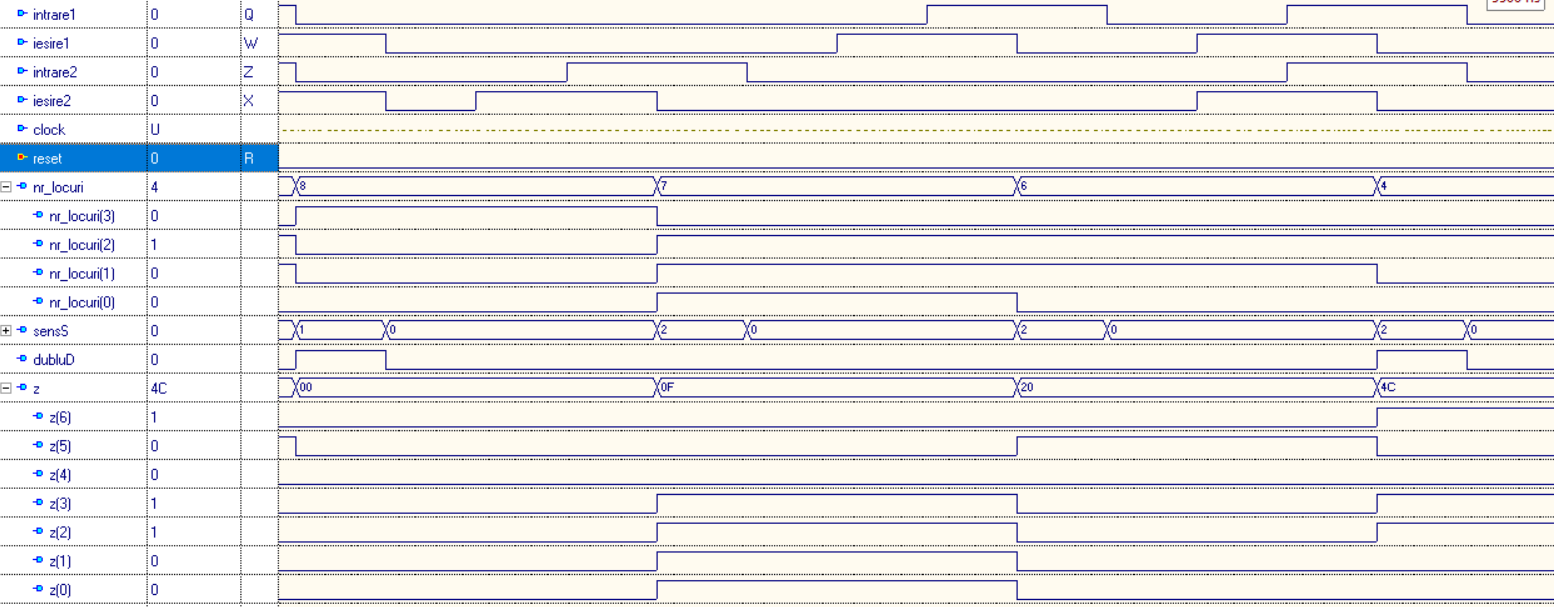


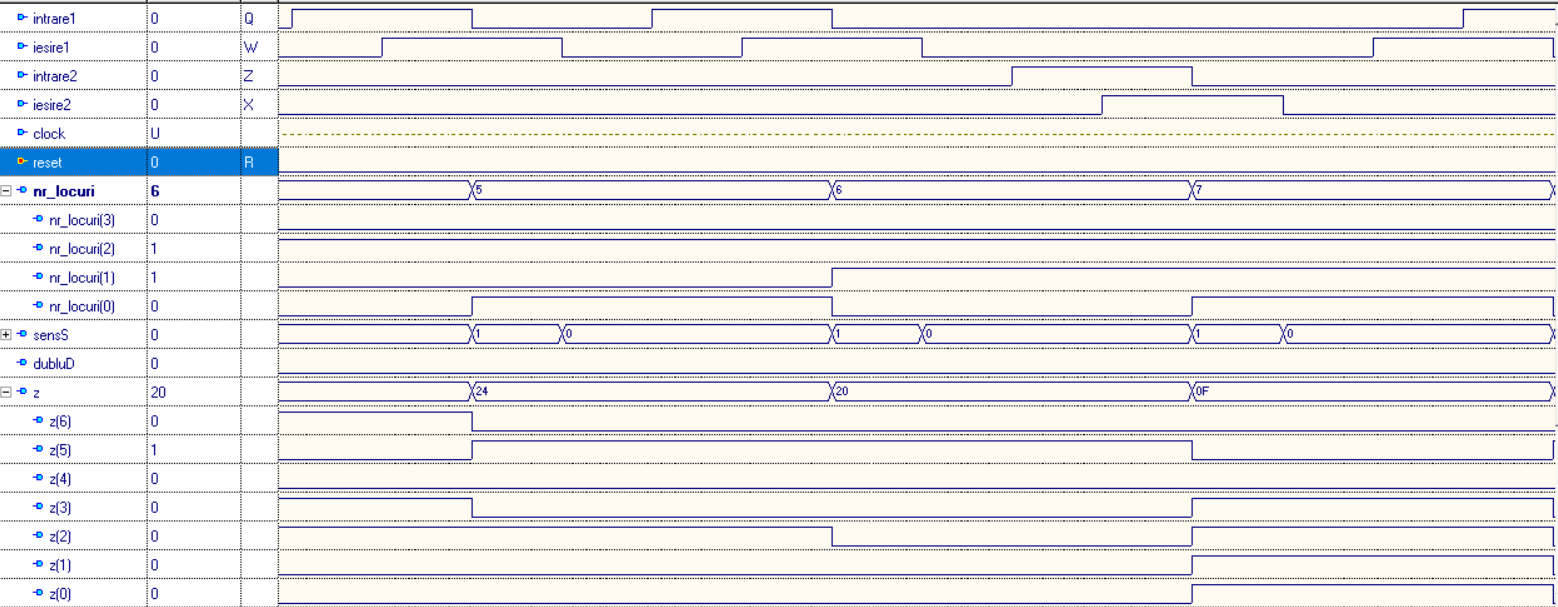
**SIMULAREA ÎN VHDL**







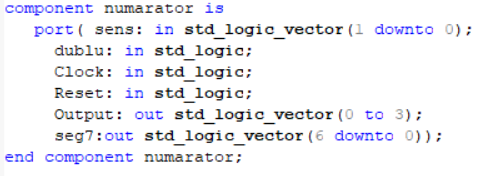




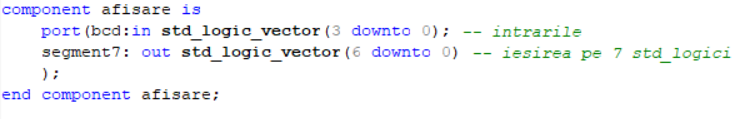
În această simulare am încercat să introducem toate cazurile posibile: să intre o mașină de la intrarea 1, să intre o mașină de la intrarea 2, să intre 2 mașini în același timp de la ambele intrări, să iasă 2 mașini de la ambele intrări, să iasă o mașină de la intrarea 1, respectiv să iasă o

mașină de la intrarea 2. În funcție de modul în care alternează semnalele primite de variabilele intrare1, iesire1, intrare2, iesire2 putem vedea cum variază numărul de locuri ocupate(nr\_locuri) dar și cum pe ieșirea z diferă valorile datorita afișorului BCD 7 segmente.

**COMPONENTELE UTILIZATE**

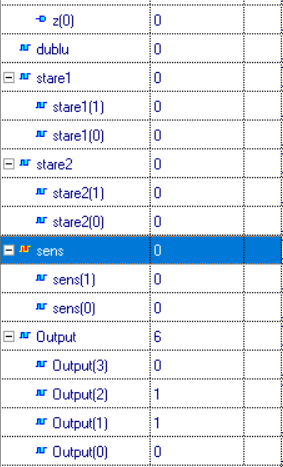
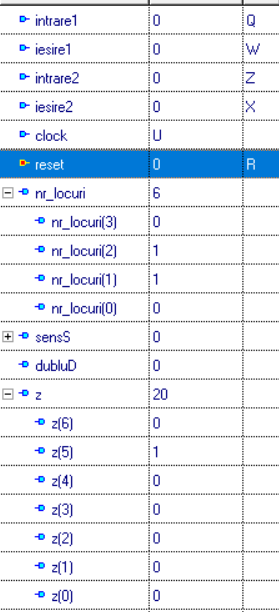


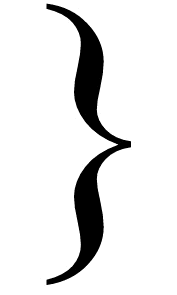
Componenta numărător ne ajută să ținem evidența numărului de locuri ocupate din interiorul parcării noastre. În parcarea noastră, numărul maxim de mașini care pot intra este de 8.



Cu ajutorul componentei afișare, facem legătura dintre numărător și afișor. Componenta afisare ne afișează pe un BCD 7 segmente numărul de locuri din parcare.

**SEMNIFICAȚIA INTRĂRILOR / IEȘIRILOR**



Intrare 1

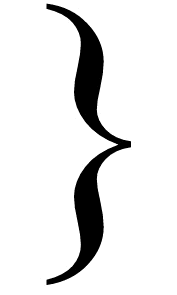
Iesire 1 Senzorii optici de la cele 2 intrări

Intrare 2

Iesire 2

Reset Butonul cu ajutorul căruia numărătorul ajunge în starea 0000

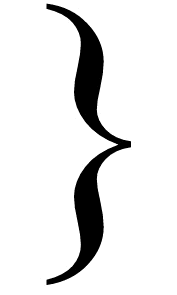
Dublu Este 1 în momentul în care ies sau intră 2 mașini deodată în parcare

sensS Semnale interne cu ajutorul cărora am verificat în simulator

dubluD evoluția lui sens și a lui dublu

z (pe 7 biți) Ieșirea afișorului BCD 7 segmente. Ne indică care liniuțe

din cadrul afișorului sunt aprinse si care sunt stinse

stare 1 Ne indică starea de la fiecare intrare, adică care care senzori sunt activi

stare 2

sens Ne arată care este sensul deplasării mașinii. 10 crescător, mașina intră, 01 descrescător, mașina iasă

**UTILIZARE ȘI REZULTATE**

Am ales această metodă deoarece mi s-a părut cea mai ușoară de implemenat și de înțeles. Proiectul se poate ușor explica oricărui utilizator, fie că e specialist sau nu. Sistemul de monitorizare umplere parcare este gândit cât mai clar și mai eficient. Am încercat să folosesc variabile cât mai sugestive pentru a fi foarte clar ce fac ele, în acest fel simularea fiind ușor de urmărit. Odată ce a fost adus în starea inițială prin activarea butonului de Reset, sistemul nostru funcționează singur, fără probleme, așa că nu este nevoie de a fi supravegheat.

Resursele software pentru realizarea acestui proiect sunt minime. Avem nevoie de un PC pe care să ruleze fără probleme limbajul în care am scris proiectul, adică VHDL și cam atât. Timpul în care PC-ul meu executa codul pentru proiect a fost de maxim 0.8 secunde, adică nesemnificativ.

**POSIBILITĂȚI DE DEZVOLTARE ULTERIOARĂ**

Una dintre componentele care ar trebui îmbunătățite la acest sistem de monitorizare umplere parcare ar fi numărătorul. Acest numărător va trebui să poată număra mult mai mult, până la numere cu cel puțin 3 cifre. Acest lucru nu este deloc greu de făcut. O altă imbunătățire în viitor ar fi ca sistemul nostru să aibă atașat un automat de bilete. În momentul în care un șofer dorește să intre in parcare trebuie să-și cumpere un bilet. Acest automat de bilete ar fi util deoarece ar rezolva problema prezenței corpurilor mici care doresc să intre în parcare. Din moment ce o persoană cumpără un bilet de la automatul de bilete este clar că deține o mașină și va intra cu ea în parcare, netrebuind să fie făcută acea verificare la intrare, cea cu corpurile mici.

O altă îmbunătățire ar fi ca sistemul nostru să afișeze atât numărul de locuri ocupate, cât și numărul de locuri libere din parcare și să țină evidența numărului de mașini care au vizitat parcarea pe parcursul unei zi, acest lucru fiind foarte folositor proprietarului.