|  |  |  |
| --- | --- | --- |
|  | **Universitatea *Transilvania* din Brașov**  **Facultatea de Inginerie Electrică și Știința Calculatoarelor**  **Departamentul Automatică și Tehnologia Informației** |  |

**Proiect   
Circuite logice programabile   
(I) Divizor de frecvență  
Driver de buton**

|  |  |
| --- | --- |
| Grabovenco Bogdan-Iulian  *Gr. 4LF412, AIA III* |  |

**BRAȘOV, 2024**

**Cuprins**

[1. Cerința 2](#_Toc167892763)

[1.1. Proiectare 2](#_Toc167892764)

[1.2. Divizor de frecvență 2](#_Toc167892765)

[1.3. Driver de buton 2](#_Toc167892766)

[1.4. Implementare 3](#_Toc167892767)

[1.5. Divizor de frecvență 3](#_Toc167892768)

[1.6. Driver de buton 4](#_Toc167892769)

[1.7. Testbench 6](#_Toc167892770)

[1.8. Rezultate 8](#_Toc167892771)

# Cerința

Analiza, proiectare și implementarea unui divizor de frecvență, precum și cea a unui driver de buton, cu aplicare într-o intersecție semaforizată.

## Proiectare

## Divizor de frecvență

Pentru a se putea folosi orice circuit digital care urmărește să intereacționeze cu oameni, trebuie mai întâi proiectat un divizor de frecvență. Încetinirea frecvenței cu care un circuit digital lucrează sau întârzierea tactului după care se face sincronizarea datelor circutului digital este necesară când se folosesc funcționalități temporale. Astfel pentru a ajunge să se folosească unitatea de timp de o secundă, unitate perceptibilă de către om, trebuie să întârziem tactul ceasului după care se face sincronizarea, spre exemplu, la o frecvență de 100 MHz sunt sincronizate datele la o periodă de 10 ns(nanosecunde), pentru a putea însă folosi doar o secundă se va proceda în felul următor:

La fiecare front crescător de ceas se va incrementa cu ajutorul unui circuit de counter valoarea unui contor până când contorul va atinge limita dată de frecvență, revenind la exemplul anterior, când contorul înregistrează 100 de milioane de incrementări, atunci contorul va fi numărat o secundă.

Pentru îndeplinirea celei de-a doua cerințe, însemnând atingerea unei frecvențe de 0.5 Hz, se va incrementa primul contor, iar când primul contor emite 2 semnale, atunci va da semnal și cel de-al doilea contor.

## Driver de buton

Driverul de buton este folosit pentru a trece tot circuitul într-o stare de service. Acest aspect a fost simulat folosind 3 tipuri de apăsări: scurtă, medie și lungă, fiecare având un anumit număr de ceasuri alocate care indică cât timp operatorul ține butonul apăsat. Driver de buton, dar mai exact butonul care este o componentă fizică, iar apasărea acestuia nu poate fi făcută în codiții ideale, însemnând datorita faptului că butonul este un element mecanic, în momentul apăsării, vor apărea inevitabil fluctuații sau spike-uri care pot produce funcționări eroante ale circuitului. Problema aceasta este rezolvată prin detectarea frontului și înregistrarea semnalului generat cu întârziere, de regulă, se folosesc 2 bistabile a căror ieșiri sune legate la o poartă ȘI, dar ieșirea celui de-al doilea bistabil trebuie negată, se folosește ieșirea sa negată. Astfel sunt evitate zonele unde se produc spike-uri mecanice.

## Implementare

Implementarea s-a făcut ajutorul ModelSim-Intel FPGA, iar secțiunea de cod a fost scrisă cu editorul de text Notepad++. În continuare vor fi atașate programele pentru fiecare modul. Atât divizorul de frecvență, cât și driverul de buton sunt descrise după următoarele figuri.

## Divizor de frecvență

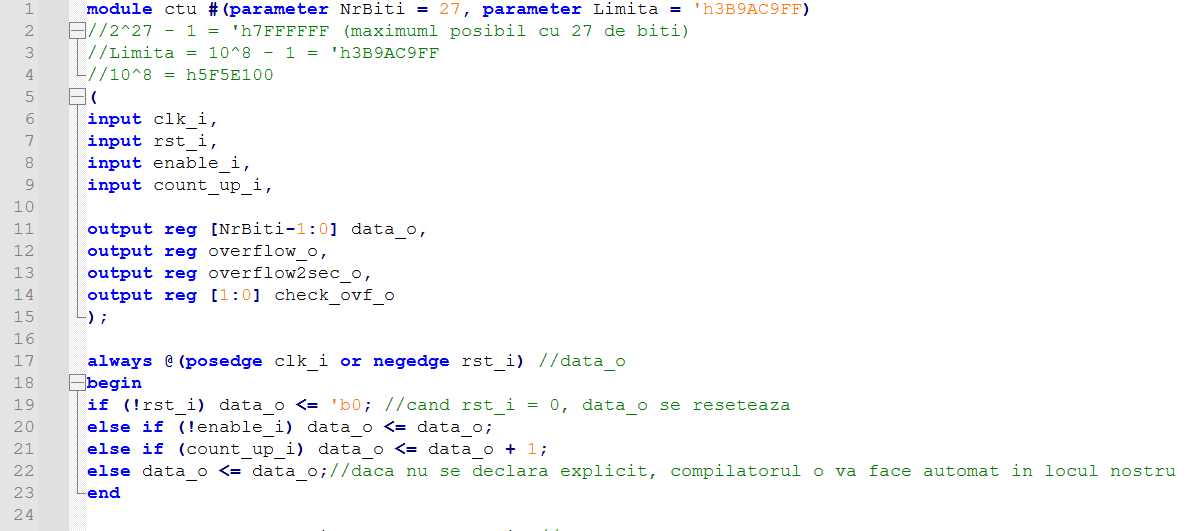


Fig 1.5.1  
Counter (I)

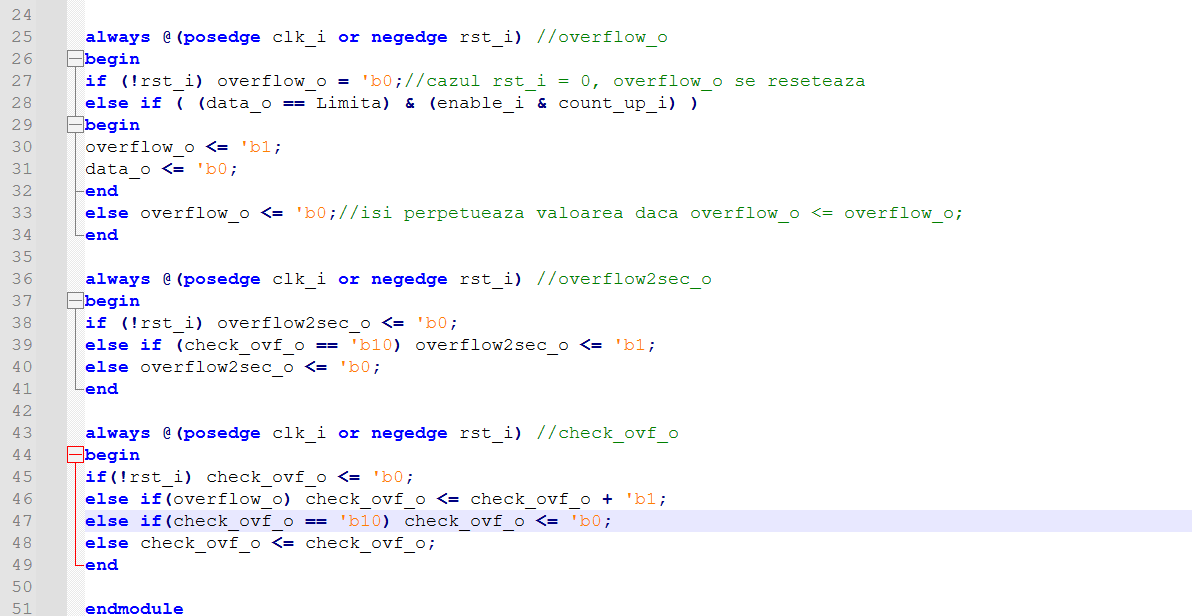


Fig 1.5.2  
Counter (II)

Scenariul de counter unde sunt inițializate semnalele de intrare.

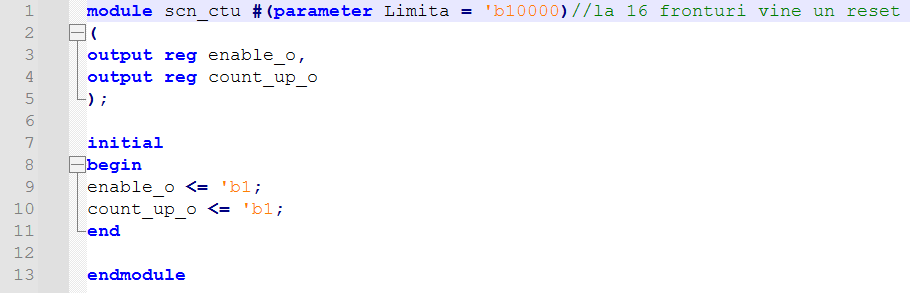


Fig 1.5.3  
Scenariul de counter

## Driver de buton

Pentru driverul de buton am folosit 4 bistabile de tip D legate în serie. Detectarea apăsării butonului se face cu ajutorul ieșirii negate ale celui de-al patrulea bistabil și ieșirea adevărată a celui de-al treilea bistabil, ieșirile celor două se aduc într-o poartă AND, astfel se evită spike-urile butonului, împiecând un comportament neașteptat. În figuri se va atașa în ordine modulele care formează driverul de buton.

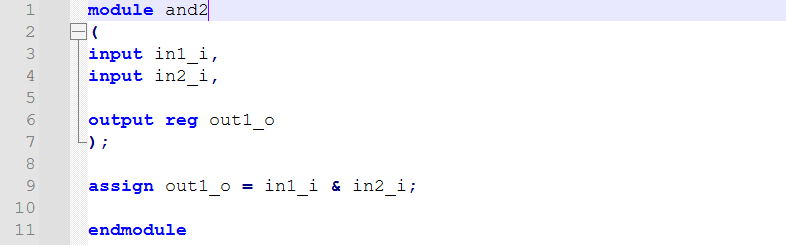


Fig 1.6.1  
Poarta AND

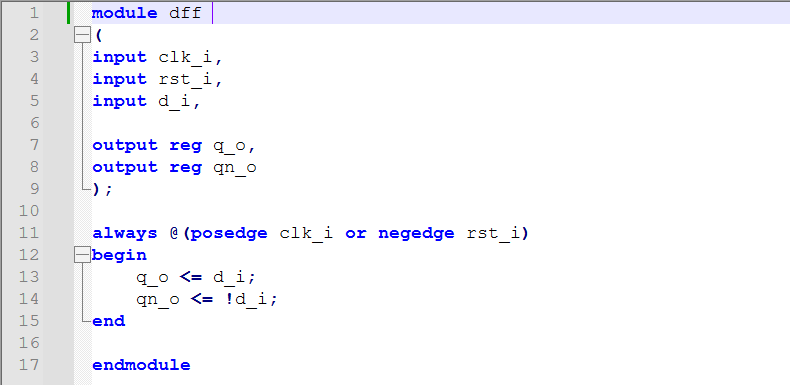


Fig 1.6.2  
Bistabilul D

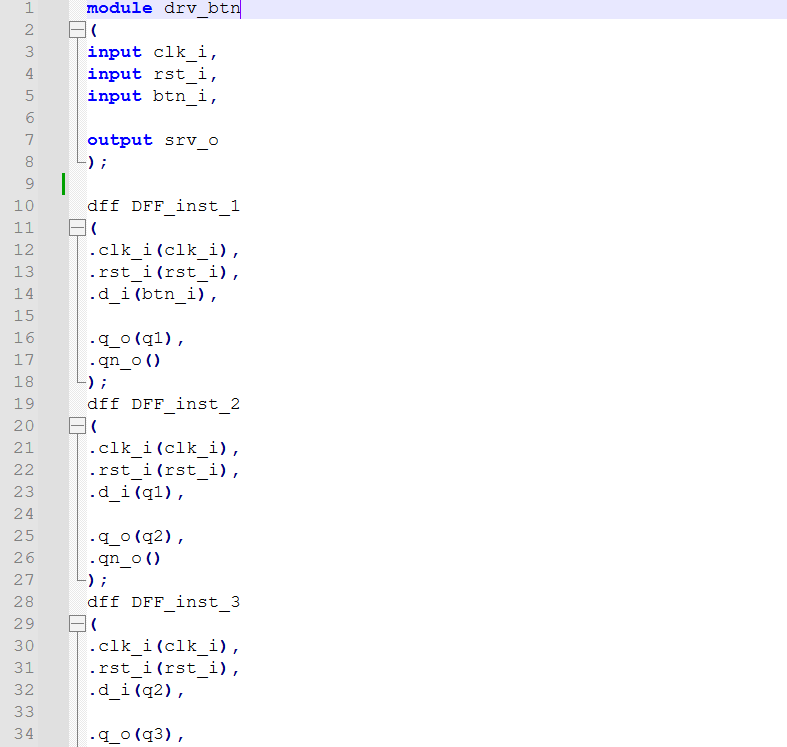


Fig 1.6.3  
Driver de buton (I)

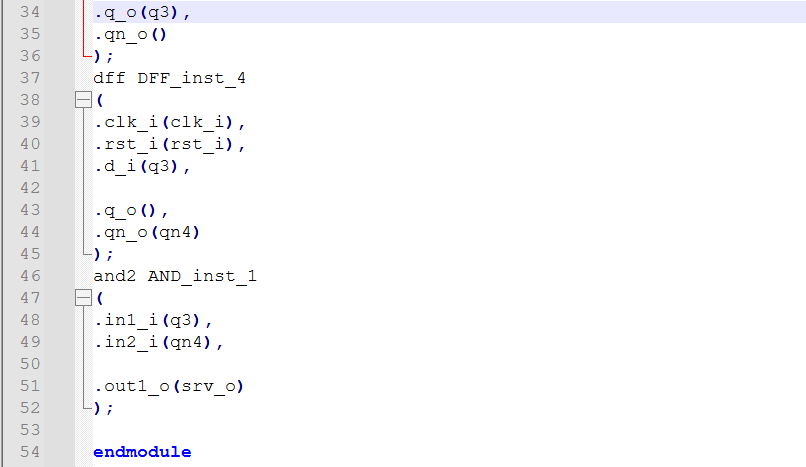


Fig 1.6.4  
Driver de buton (II)

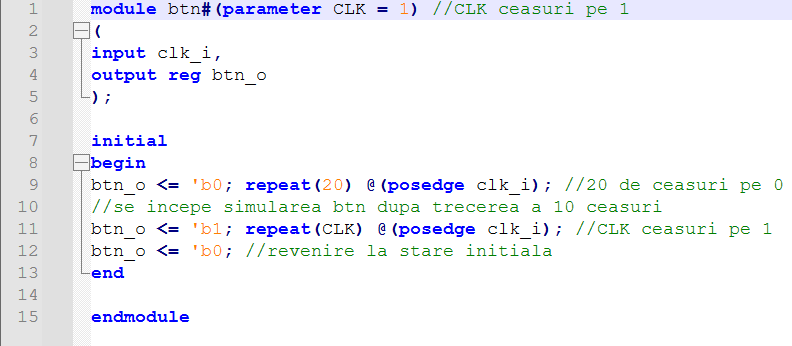


Fig 1.6.5  
Scenariul asociat driverului de buton

## Testbench

Testbench-ul este unealta cu care putem simula performanțele DUT-ului obținut. DUT este acronimul lui "device under test". Testbench-ul generează semnale de intrare, monitorizează răspunsurile, și verifică dacă DUT-ul se comportă conform specificațiilor. Acesta poate include stimulări, verificatori și module de verificare automată a rezultatelor, facilitând astfel identificarea și diagnosticarea problemelor în designul hardware. În figurile de mai jos se va introduce codul testbench-ului.

Fig 1.4.5  
Driverul de buton

Fig 1.4.7  
Scenariul cu service

Fig 1.4.6  
Scenariul fără service

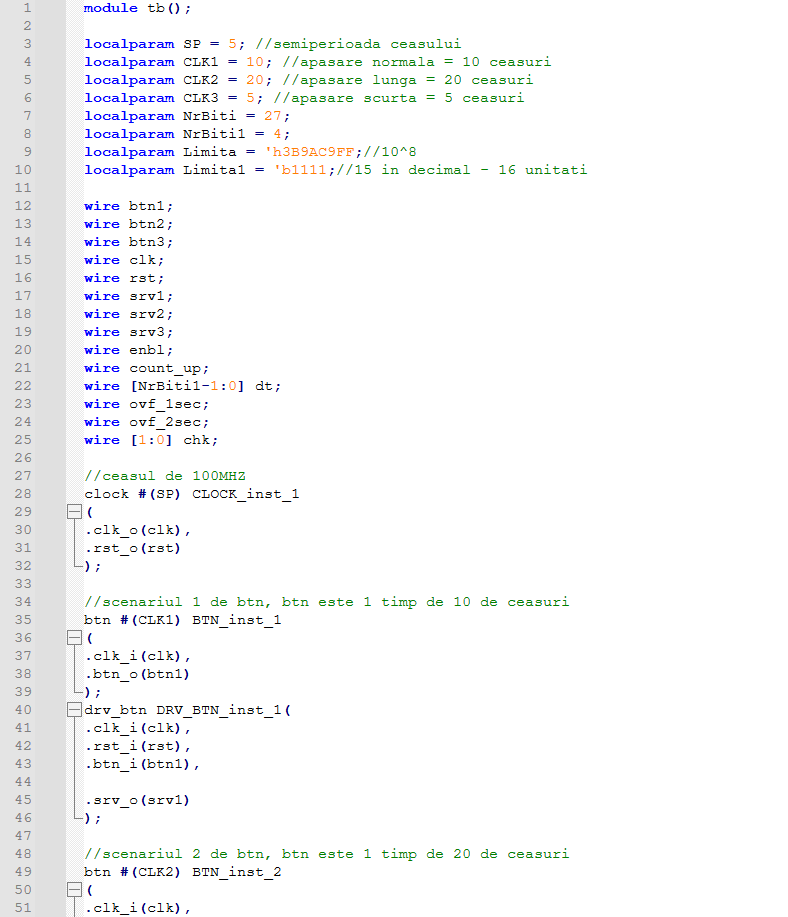


Fig 1.7.1  
Testbench (I)

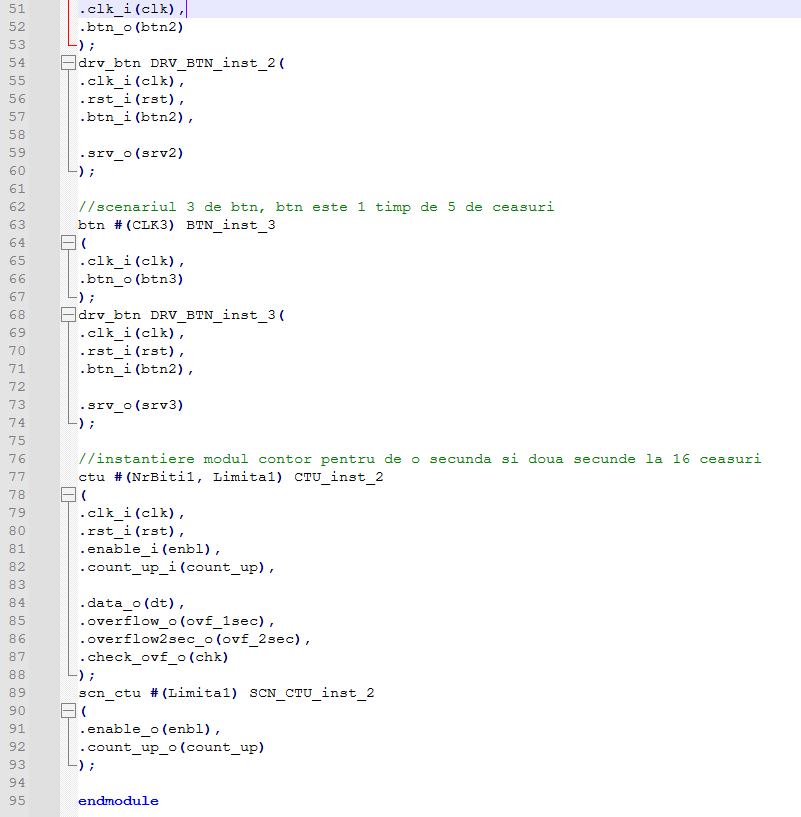


Fig 1.7.2  
Testbench (II)

## Rezultate

În această secțiune vor fi introduse simulările și se vor explica semnalele prezente în diagrama. O mențiune importantă este că pentru o vizibilitate mai mare s-a folosit o frecvență mult mai mică pentru a ilustra semnalul de o secundă și de 2 secunde mai clar. Tot în aceeași simulare este introdusă și partea pentru driverul de buton.

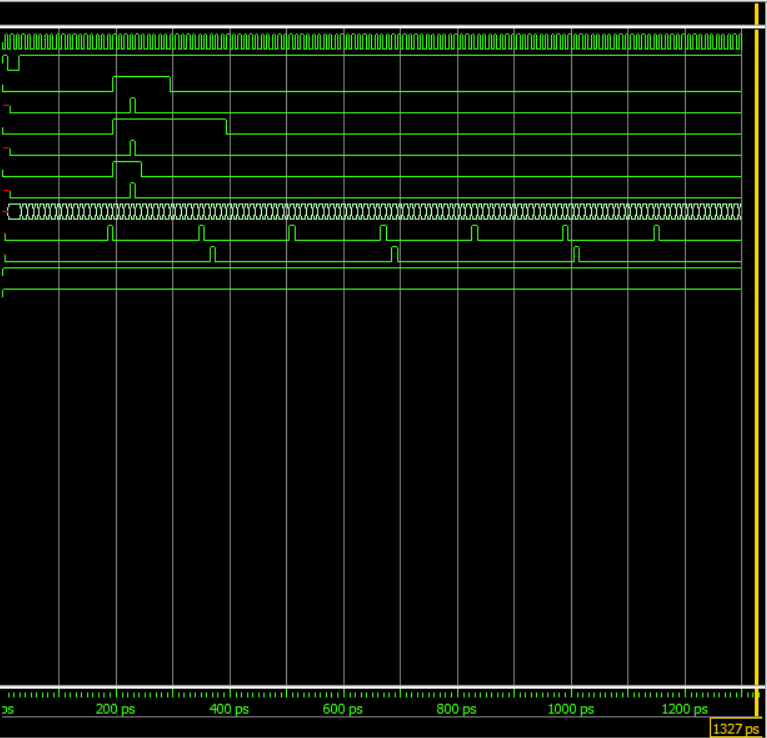


Fig 1.8.1  
Diagrama de semnale

Fig 1.5.1  
Diagrama de semnale

Semnalele vor fi trecute în odine, de sus în jos:

1. Semnalul de ceas/clock cu frecvența de 100MHz, s-a folosit o semiperioadă de 5 unități de timp, 5 ns, unitatea de timp default în Verilog este ns
2. Semnalul de reset, care trebuie configurat minim o dată pentru a atribui valori semnalelor care depind de el
3. Testul de buton unde apăsarea ține 10 ceasuri (apăsare medie)
4. Întârzierea semnalului de service – starea de service 1
5. Testul de buton unde apăsarea ține 20 ceasuri (apăsare lungă)
6. Întârzierea semnalului de service – starea de service 2
7. Testul de buton unde apăsarea ține 5 ceasuri(apăsare scurtă)
8. Întârzierea semnalului de service – starea de service 3
9. Semnalul contorizat care crește unitar cu fiecare front crescător de ceas
10. Semnalul de o secundă
11. Semnalul de 2 secunde
12. Semnalul de count-up, care permite incrementarea
13. Semnalul de enable, care permite utilizarea circuitului