PSN-Proiect

***WHACK-A-MOLE GAME***

**Student: Indrumator:**

**Coldea Cosmin Dan Fleger**

**CUPRINS:**

**1. Premisa Proiectului**

**2. Functionarea de baza**

* **MPG**
* **Divizorul de frecventa**
* **SSD**
* **FSM**
* **Componenta principala**

**3. Dezvoltari ulterioare**

**Premisa proiectului**

Scopul proiectului este de a simula un joc de tip Whack-a-Mole folosind o placa de dezvoltare FPGA ce dispune de minim 2 SSD (Seven-Segment-Display), 16 led-uri, 16 switch-uri, un buton de start si un buton de reset.

Whack-a-Mole este un joc popular de tip arcade in care jucatorul trebuie sa marcheze cat mai multe puncte prin lovirea unor cartite ce apar si la intamplare si dispar dupa un interval scurt de timp cu ajutorul unui ciocan. Conceptul jocului este unul simplu, fiind astfel recunoscut ca unul dintre cele mai captivante si cunoscute jocuri arcade.

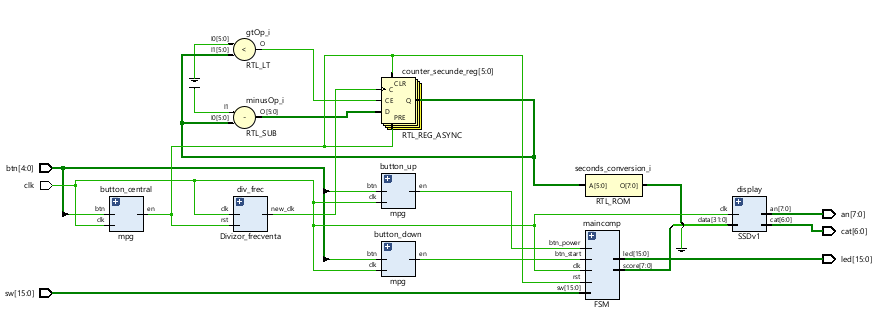
Pentru a reprezenta cat mai ideal un model de joc Whack-a-Mole pe o placa FPGA vom folosi:

* 16 led-uri ce reprezinta cartitele.
* 16 switch-uri, astfel incat fiecarei cartite (led) sa ii corespunda switch-ul aferent.
* Un buton de power on/off
* Un buton de start care odata apasat porneste jocul
* Un buton de reset cu rolul de a reporni jocul de indata ce este apasat. Astfel va fi resetat timer-ul, scorul, pozitia cartitei.
* Seven-Segment Display pentru a afisa scorul, timpul ramas (descrescator, de la 59 la 0 secunde).

Modul de functionare:

Dupa ce jucatorul apasa butonul de start, primul led se va aprinde, iar timer-ul va porni, acesta avand 60 de secunde la dispozitie sa inscrie cat mai multe puncte. Pentru a inscrie un punct, jucatorul trebuie sa apese switch-ul de sub led-ul aprins, iar led-ul se va stinge, aprinzandu-se altul in locul sau. Orice alt switch nu are nici un efect.

**Functionarea de baza**

****

In figura de mai sus este reprezentat schematicul proiectului.

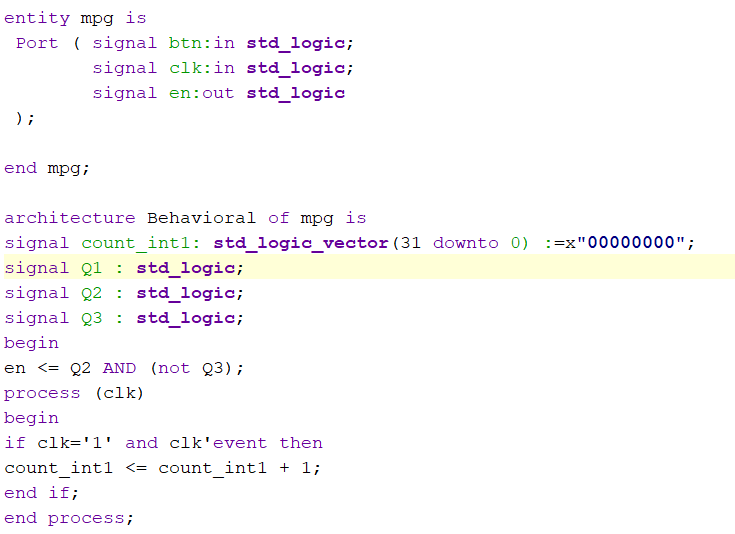
Dupa cum se poate observa, pentru realizarea proiectului avem nevoie de anumite componente de baza, fiecare avand un rol special in functionarea acestuia.

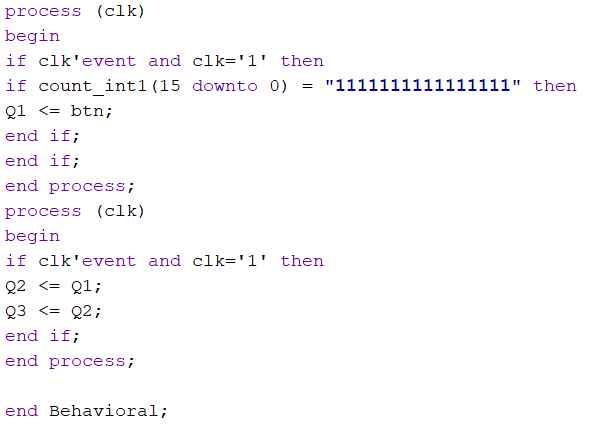
***MPG:***

O componenta importanta este debouncer-ul sau generatorul de monoimpuls care asigura functionarea corecta a butoanelor de start, reset, power on/off.

Are 2 semnale de intrare, un buton si ceasul si ca semnal de iesire are enable.

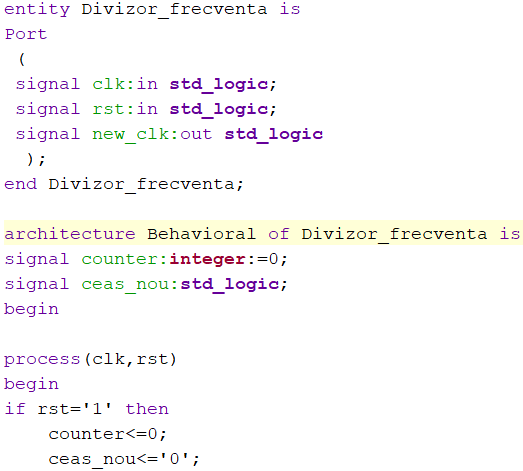
MPG-ul folosește un contor de tip std\_logic\_vector pentru a se asigura ca avem o singura apasare de buton.

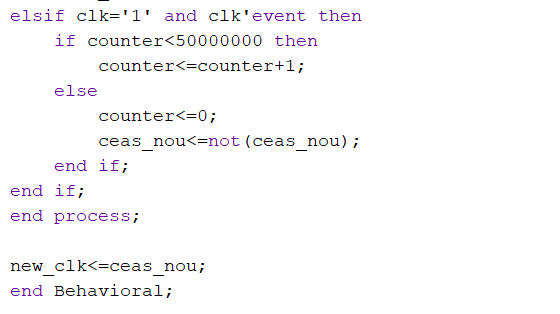
****

****

**Divizorul de Frecventa:**

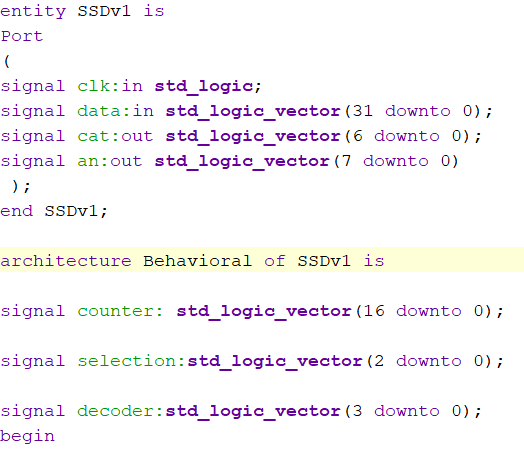
Acesta este folosit pentru a crea un ceas de o frecventa mai mica, folosind un ceas deja existent, rapid. Dispune de asemenea de un semnal de reset asincron. Deoarece proiectul a fost realizat pentru a fi testat pe o placa Nexys4 cu frecventa de 100MHz (10 ns), pentru timer vom avea nevoie de un ceas cu un ciclu de 1Hz (o secunda) astfel incat pe SSD sa se afiseze ceasul actualizat la fiecare secunda.

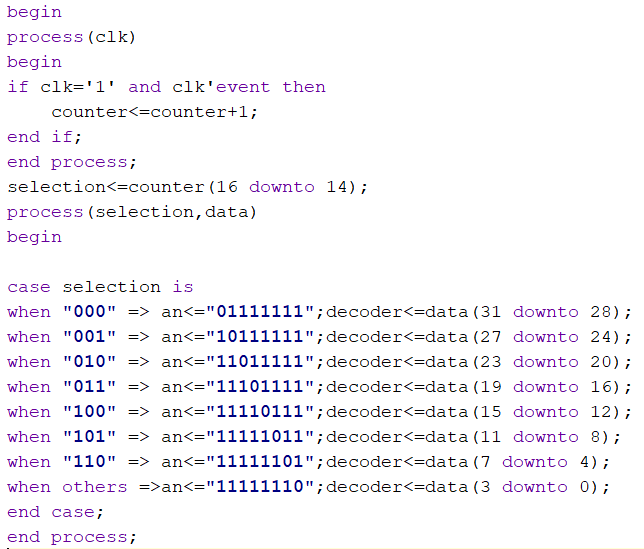
****

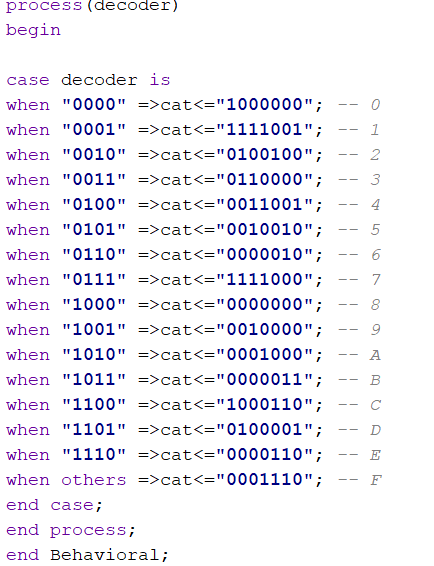
****

**SSD**

Componenta SSD este folosita pentru a afisa pe Display informatiile dorite precum timer si scor.

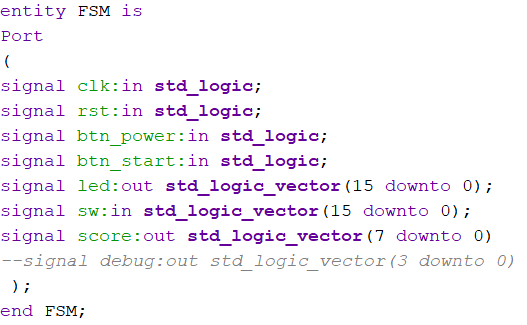
****

****

****

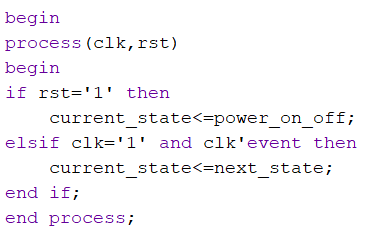
**FSM**

FSM sau Finite State Machine este componenta ce se ocupa de actualizarea starilor. Ca semnale de intrare are un ceas, un reset asincron, switch-urile ce corespund fiecarei cartite, un buton de power on/off si unul de start, iar ca semnale de iesire va avea un scor ce se actualizeaza pe masura ce sunt apasate switch-urile corecte si led-urile ce se vor aprinde in functie de stare.

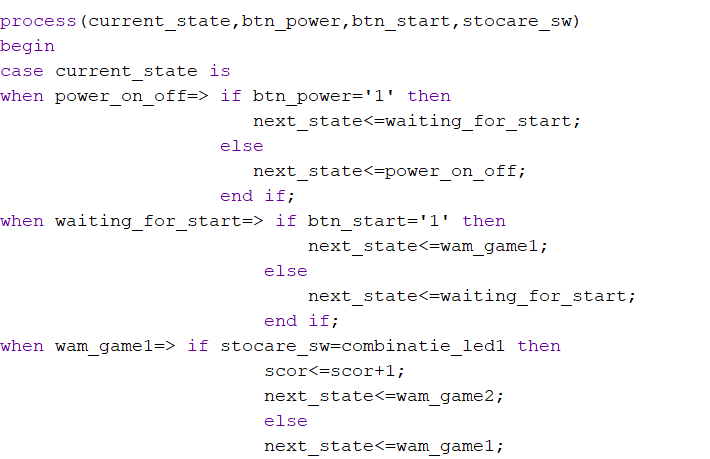


In cadrul arhitecturii sale vom avea 5 procese

Primul proces verifica daca a avut loc un eveniment de ceas, caz in care starea curenta se actualizeaza, iar daca resetul asincron a fost activat se revine in starea de power on/off



Al doilea proces se va ocupa in principiu de actualizarea starilor, astfel va verifica daca switch-ul corespunzator va fi afisat, caz in care se trece in starea urmatoare si se actualizeaza scorul.



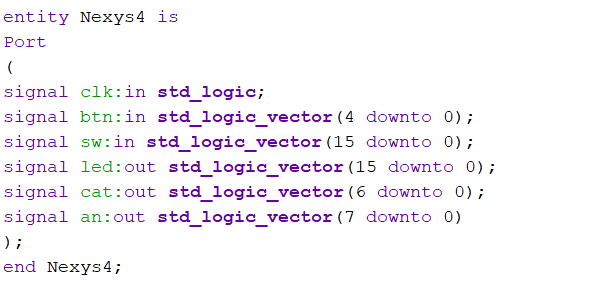
Urmatorul proces va codifica scorul pentru a aparea in format zecimal pe afisor.

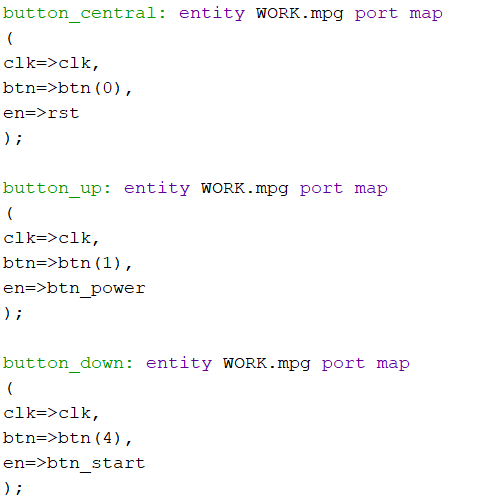
Al patrulea proces actualizeaza vectorul stocare\_sw in functie de starea in care se afla.

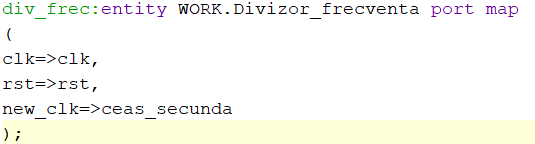
Iar in final, ultimul proces actualizeaza led-urile in functie de stare si afiseaza scorul corespunzator.

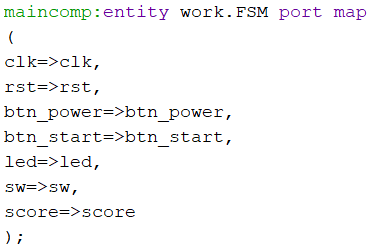
**Modulul principal**

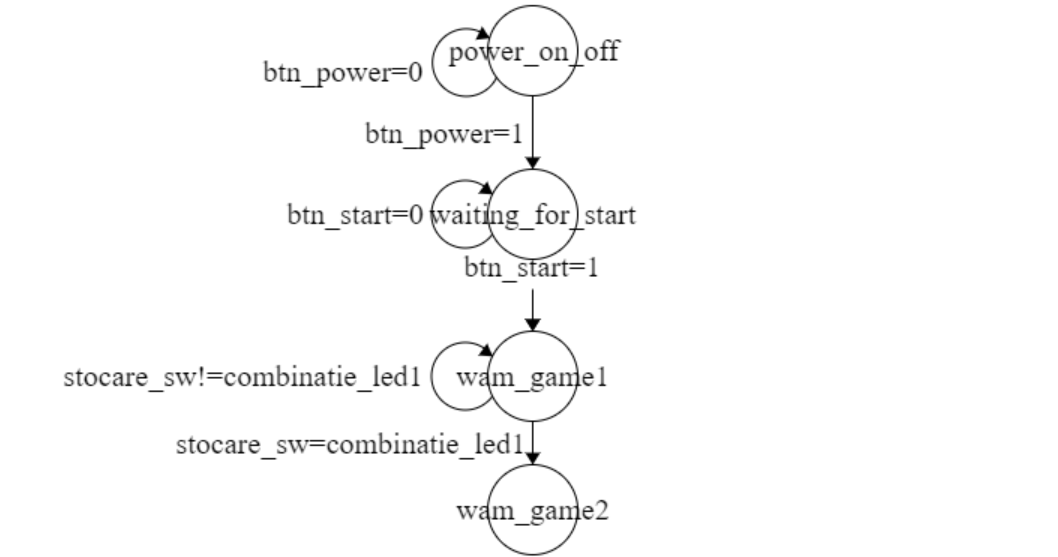
Este denumit Nexys4 iar acesta conține portmapping-ul fiecareia dintre componente si este cutia neagra principală a intregului proiect, avand intrarile si iesirile placii, determinate si de fisierul de constrangeri. Tot aici este si afisat ceasul.



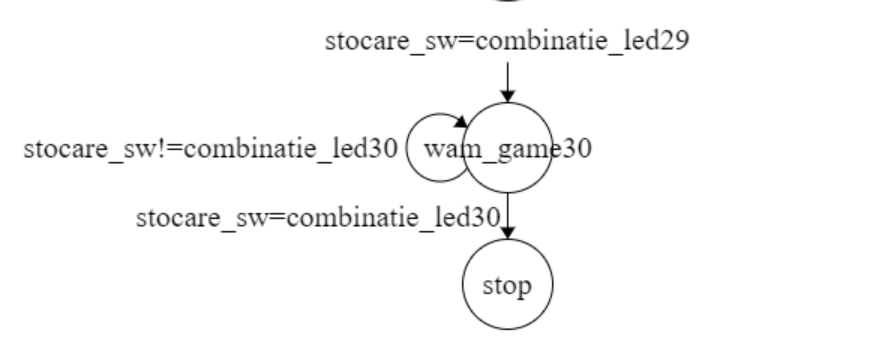






Vom avea o stare de power\_on\_off, o stare de waiting \_for\_start, 30 de stari pentru diferitele combinatii (jucatorul poate avea scor maxim 30) si ultima stare va fi de stop. Tranzitia de la o combinatie la alta este similara, starea actualizandu-se daca switch-ul corespunzator este apasat, altfel nu se va schimba. 

Doar in cazul ultimei combinatii, tranzitia va avea loc catre starea de stop



**Dezvoltari ulterioare**

Un pas urmator in dezvoltarea proiectului ar fi legarea timer-ului descrescator astfel incat dupa ce acesta ajunge la 0, scorul sa fie memorat pana la o urmatoarea apasare a butonului start.