**Proiect Verilog - Joc Snake pe FPGA**

**Voda Mihai Alexandru**

**Grupa 421D**

**Introducere**

Scopul principal al jocului este de a controla sarpele pe ecran cu ajutorul tastelor (**↑,←,↓,→**) , evitarea colisiunii cu marginea neagra a ecranului, si acumularea de puncte prin culesul merelor de pe ecran.

- Merele se genereaza random pe ecran , iar in momentul colisiunii cu un mar dimensiunea sarpelui creste cu 1;

-In momentul colisiunii cu marginea neagra a ecranului , ecranul se face rosu, iar pentru a incepe un **joc nou** se apasa tasta **SPACE**;

-Sarpele nu moare cand se loveste de propriul corp (optional);

-**Scorul** (contorul merelor culese) se poate observa pe afisajul cu 7 segmente al placutei DE1, scorul maxim este 30;

-La atingerea scorului maxim ecranul se face verde, pentru a reseta scorul si jocul se apasa tasta **SPACE**;

-Jocul incepe in momentul in care este apasata una dintre tastele (**↑,←,↓,→**);

-Jocul a fost realizat in asa fel incat nu este posibila schimbarea sensului doar pe orizontala sau doar pe verticala (ca jocul snake clasic).

**Modulul TOP**

**clk**

**sda**

**scl**

**v\_sync**

**h\_sync**

**red**

**green**

**blue**

**points\_7seg**

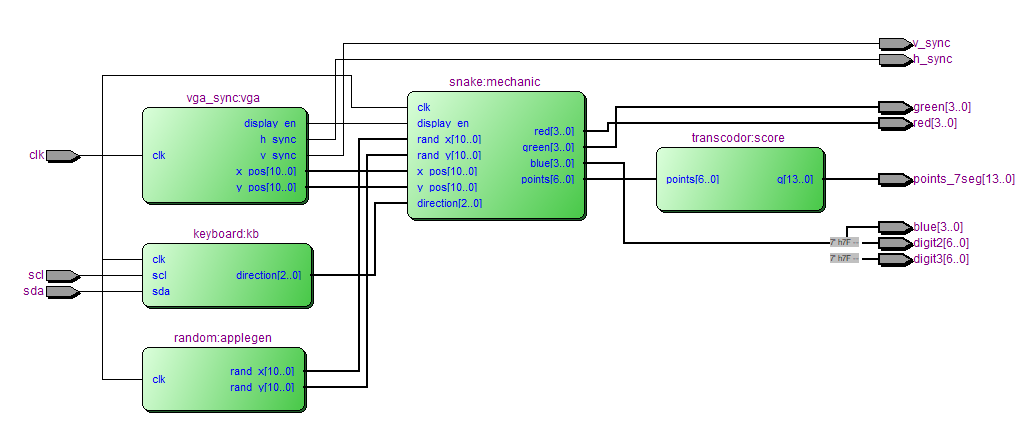
**3**

**3**

**3**

**14**

**Schema Interconectarii Blocurilor**



**Inputs:**

* clk (ceas placa DE1 de 50MHz)
* scl (ceas tastatura ~17kHz)
* sda (date primite de la tastatura pe negedge de scl)

**Outputs:**

* v\_sync (sincronizare verticala pentru monitor)
* h\_sync (sincronizare orizontala pentru monitor)
* red (4 biti pentru culoarea rosie)
* green (4 biti pentru culoarea verde)
* blue (4 biti pentru culoarea albastra)
* points\_7seg (bitii semnificativi pe HEX1, restul pe HEX0)
* digit2 (mereu HIGH stinge segmentul HEX2)
* digit3 (mereu HIGH stinge segmentul HEX3)

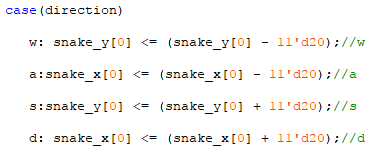
**Module:**

* vga\_sync (face sincronizarea,transmite zona activa, pozitiile x si y)
* keyboard(citeste datele de la tastatura si le transmite sub forma de stari)
* random(genereaza aleator doua pozitii x si y)
* snake(contine mecanismul jocului)
* transcodor(afisarea scorului pe afisajul cu 7 segmente)

**Modulul Snake**

Acesta contine logica efectiva a jocului. Pentru memorarea sarpelui am generat 2 memorii ram cu 32 de adrese, snake\_x si snake\_y corespunzatoare pozitiilor x si y. Pe adresa 0 se va afla mereu capul sarpelui iar pe adresele 1-31 se va afla corpul sarpelui.

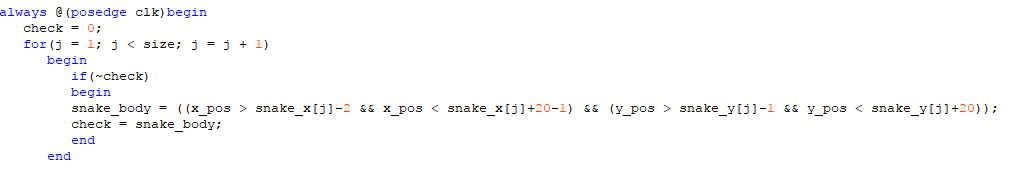
Jocul porneste din starea de reset in care capul sarpelui (snake\_x[0] si snake\_y[0]) se genereaza pe mijlocul ecranului. **Acesta se actualizeaza la o frecventa redusa de cea a ceasului de 50Mhz al VGA-ului pentru a putea observa cu ochiul liber schimbarile pe ecran.** Cat timp resetul este activ (nu a fost apasata nicio sageata de la tastatura) se initalizeaza cu ajutorul unei structuri repetitive adresele 1-31 pentru corpul sarpelui in afara rezolutiei monitorului (este important sa reinitializez mereu toate adresele in afara rezolutiei ori de cate ori a fost apasat resetul ca sa nu ramana pe display). Odata apasata o sageata de la tastatura jocul iese din starea de reset. Fiecare sageata corespunde unei alte stari **↑-starea ‘w’ , ←-starea’ a’, ↓-starea ‘s’, → -starea ‘d’.**



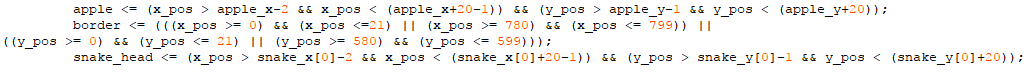
In functie de stare capul sarpelui isi schimba pozitia cu 20 pixeli (pe x sau pe y). Cat timp resetul nu este activ cu ajutorul unei structuri repetitive snake\_x[0] devine snake\_x[1], snake\_x[1] devine snake\_x[2] si tot asa pana la size, care se incrementeaza la fiecare mar mancat. (la fel si pentru y). Aceasta procedura este importanta pentru a crea iluzia de miscare.

Obs! Am folosit o structura repetitiva (for) deoarece toate interschimbarile dintre adresele pozitiilor trebuiesc realizate in paralel in momentul in care s-a activat ceasul redus (update).

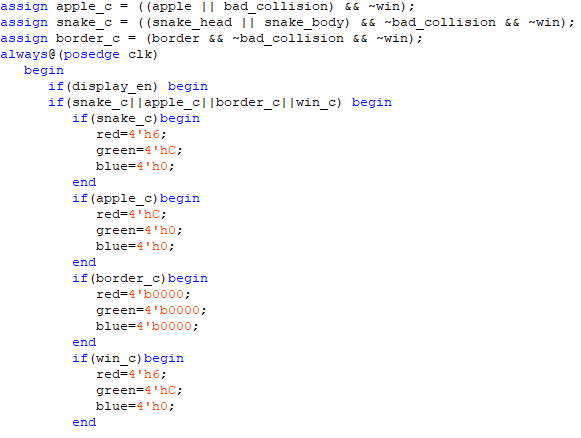
La fiecare posedge al ceasului de pe VGA este actualizata marginea de pe display, marul, capul de sarpe si fundalul. Pentru a actualiza corpul sarpelui am folosit o structura repetitiva mai speciala in functie de valoarea registrului size. Folosesc un registru check pentru a verifica ce segment al corpului trebuie colorat. Toate verificarile se fac in paralel pe acelasi posedge.



Si pentru restul de conditii pentru a colora pe display:



In momentul colisiunii cu un mar , adica daca se indeplinesc simultan conditiile pentru snake\_head si apple, size precum si scorul care urmeaza sa fie trimis la transcodor se incrementate cu 1. Tot pe acelasi principiu in momentul colisiunii capului cu marginea ecranului flagul bad\_collision se activeaza , iar in momentul acumularii a 30 de puncte flagul de win se activeaza. Toate aceste conditii au ca rezultat actualizarea culorilor transmise prin VGA catre monitor.



De exemplu cat timp este activ snake\_head sau snake\_body se va colora in verde.

**Modulul Random**

Se genereaza pe frontul pozitivc al ceasului de 50Mhz cu ajutorul a doua countere doua pozitii x si y care vor fi folosite in modulul snake pentru generarea marului in mod aleator.

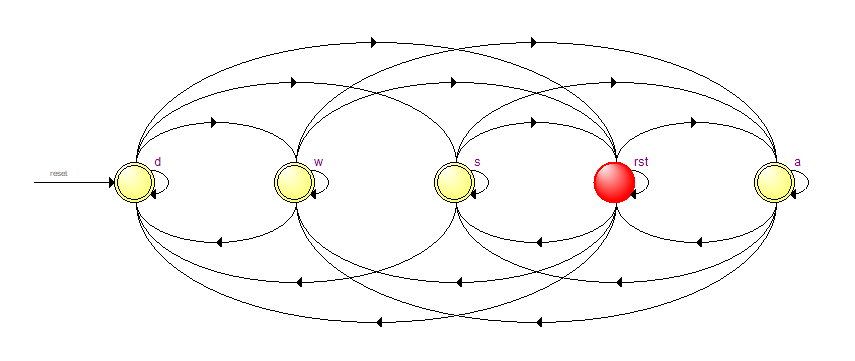
**Modulul Transcodor**

Se afiseaza pe afisajul cu 7 segmente scorul adica numarul de mere culese transmis din modulul snake . Afisajele 3 si 2 sunt mereu HIGH ca sa ramana stinse iar cei mai semnificativi 7 biti sunt folositi pentru afisajul 1 si restul pentru afisajul 0 pentru a genera cu usurinta numerele .

**Modulul Keyboard**

Acest modul este resposabil cu transmiterea datelor primite de la tastatura catre modulul snake sub forma de stari (w,a,s,d). Datele se valideaza doar atunci cand un pachet complet de 11 biti a fost primit. Pentru stocarea datelor trimise am folosit un registru de shiftare la dreapta.

Automat pentru starile trimise catre modulul snake.



**De exemplu daca automatul este in starea ‘d’ nu va putea niciodata trece in starea ‘a’ deci nu se va putea schimba sensul sarpelui doar pe orizontala sau doar pe verticala.**

**Modulul VGA\_sync**

Este modulul in care se sincronizeaza datele trimise prin vga cu monitorul, se genereaza pozitiile pixelilor de pe display x\_pos si y\_pos care sunt folosite mai departe in modulul snake pentru a genera culorile.