Mazilu Mihnea Mihai 333AA

Modulul baggage_drop

Baggage_drop este modulul principal in care folosim toate celelalte 3 module (sensors_input, square_root, disp_and_drop). In baggage doar am initializat 3 wire-uri de care aveam nevoie pentru a folosi functiile si am realizat formula „t = sqrt(height) / 2” dupa ce am apelat functia square_root prin comanda assign „t_act = out / 2;”

Modulul sensors_input

Am declarant 2 variabile de tip reg (o, s) care ma vor ajuta in cadrul always. Varibila o este echivalentul lui height iar variabila s este un reg de care am nevoie pentru a calcula suma senzorilor.

Am implementat 3 if-uri care imi verifica dupa cerinta temei daca macar unul dintre senzorii 1 sau 3 este 0. In else-ul acestui if am implementat aceeasi conditie pentru senzorii 2 si 4, iar in else-ul acestui if am implementat suma celor 4 senzori. Ceea ce merita mentionat este modul in care am abordat problema numerola divizibile sau cu 2 sau 4 (cazul in care am numere cu 0.5). Problema apare deoarece verilog rontunjeste un numar cu 0.5 la numarul mai mic spre exemplu 139.5 la 139, dar 139.5 trebuie sa fie 140 in tema noastra.

Ma voi uita in cazul impartirii la 2 la ultimul bit al numarul si voi verifica daca acesta este 0 sau 1. In cazul in care este 0, inseamna ca numarul este divizibil cu 2 deci il voi lasa asa, dar in cazul in care bitul este 1 (fapt ce semnifica numar impar), voi incrementa dupa impartire cu 1.

In cazul impartirii la 4 voi aplica acelasi proces, dar voi tine cont de data aceasta de ultimii 2 biti. Rezultatul este divizibil cu 4 doar in cazul 00, deci acesta nu ne intereseaza. In cazul 01 numarul va da cu .25 , numar care ar trebui rotunjit la mai mic deci este ok. Astfel daca ultimii doi biti ai numarul sunt 10 sau 11 atunci numarul il voi incrementa dupa impartire cu 1.

La sfarsitul always-ului am dat „assign height = o;” deoarece eu am calculat outputul in variabila de tip reg o, dar outputul trebuie sa fie in height.

Modulul square_root

Acesta este un modul putin mai complex. Algoritmul folosit este cel luat din resursele puse la dispozitie, link-ul fiind acesta Square root Algorithms. Algoritmul se numeste digit by digit calculation si functioneaza pe shiftare si scaderi repetate.

Am declarat mai multe variabile:

- **i** este pentru while (programul este sintetizabil deoarece while-ul are numar fix de repetari si anume 16)
- **num** este varibila cu care voi lucra in locul in-ului (deci in num voi pune in)
- **res** este variabila in care va fi memorat rezultatul final
- **bit_** este o variabila auxiliara de care am nevoie pentru realizarea calcului (bit_ incepe la cea mai mare putere a lui 4 mai mica decat numarul meu)

Mazilu Mihnea Mihai 333AA

In cadrul always-ului am initializat i cu 0 pentru a putea fi folosit in while, am pus in pe primii 8 biti ai lui num dupa care am shiftat cu 24 pentru a-l pune pe 32 de biti si a putea face calculele intre res si num. Res este pe 32 de biti deoarece rezultatul nostru final va trebui sa fie pe 16 biti. La fiecare iteratie a algoritmului am facut shifatrea lui res la dreapta cu 1 bit, asa ca 16 biti finali + 16 biti pierduti in cadrul algoritmului, rezulta ca avem nevoie de 32 de biti initiali. Bit_ este initializat cu 1 si shiftat la stanga cu 30 pentru ca asta este cea mai mare putere a lui 4 mai mica decat orice numar pe 32 de biti.

Primul while realizeaza acest bit_ prin shiftarea repetata cu 2 a sa (impartire la 4) pana cand acesta este mai mic decat numarul meu).

Al doilea while este algortimul digit by digit luat si transcris in verilog din C, din codul de pe wikipedia.

La final i-am atribuit valoarea rezultata (res) out-ului meu (out). Din aplicarea algoritmului pe mai multe numere (de mana) am realizat ca in out trebuie sa pun bitii [19:4] din res.

Modulul disp_and_drop

In acest modul am avut nevoie doar de o singura variabila auxiliara de tip reg (number) deoarece am facut toate celelalte out-uri (seven_seg-urile si drop_activated) de tip reg. Number este o varibila necesara la case-ul pe care l-am implementat in cadrul algoritmului.

Am imlementat 3 if-uri, unul pentru fiecare caz (Hot, Cold, Drop) cu ajutorul conditiilor aferente, iar pentru fiecare am dat o valoare lui drop_activated (depinde de caz el putand fi 0 sau 1) si o valorare de la 0 la 2 lui number, valoare ce imi indica cazul.

Urmeaza un case in care am cele 3 cazuri, fiecare afisand un mesaj.