Ciuciu Anca-Maria 333 AA

Tema 1 – Baggage Drop

Simularea aruncării automate a unui container cu provizii într-un câmp de luptă de către un elicopter către o echipă de pușcași marini a fost posibilă cu ajutorul a 4 module implementate in limbajul de descriere hardware Verilog. Astfel, cele patru module sunt: sensors_input, square_root, display_and_drop și baggage_drop.

1. Modulul sensors_input

Acest modul are rolul de a primi ca input 4 senzori pe 8 biți, reprezentând 4 înălțimi măsurate individual, pe care îi folosește pentru a calcula o înălțime medie aproximativă la care se află elicopterul în acel moment de timp.

Astfel, implementarea acestui modul a necesitat declararea celor două variabile de tip reg, *number_of_sensors* (pe 3 biți, deoarece avem un numar maxim de senzori egal cu 4) și *sum* (pe 12 biți, pentru a încadra suma celor 4 senzori pe 8 biți). Se pornește de la presupunerea că niciun senzor nu este nul, astfel că se reține suma acestora în variabila *sum*, pornind de la un număr de 4 senzori activi. În continuare, s-a verificat pe perechi de câte doi senzori, faptul că aceștia nu sunt nuli. În cazul în care unul dintre senzori este nul, se va scădea perechea respectivă din sumă, iar numarul de senzori activi va scădea cu 2 unități. În continuare, se verifică posibilitatea ca numărul final de senzori să nu fie egal cu 0 (chiar dacă acest caz nu este unul plauzibil), pentru a evita împărțirea la 0.

În continuare, s-a folosit ultimul și penultimul bit pentru rotunjirea mediei senzorilor după cum urmează: dacă se constată că sunt activi doar 2 senzori, înseamnă că, dacă numărul este impar, atunci se va rotunji în sus cu o unitate și viceversa. Totodată, dacă suma este compusă din toți cei 4 senzori, se interoghează penultimul bit, și în cazul în care acesta este 1, se va aproxima in sus cu o unitate, iar dacă este 0, media va rămâne neschimbată.

În final, odată ce se iese din blocul always, se va asigna outputului valorea mediei calculate.

2. Modulul square_root

Acest modul primește media înălțimilor de la modulul sensors_input și returnează rădăcina pătrată a acesteia pentru a putea calcula în continuare timpul limită.

Așadar, pentru acest proces, s-a folosit Algoritmul CORDIC, speculând faptul că, înmulțirea cu 2 echivalează cu deplasarea numărului în binar cu un bit la stânga, iar împărțirea la 2 echivalează cu rotirea bitului la dreapta. Prin urmare, rezultatul va fi obținut prin înmulțire, adăugare și testarea validității unei condiții.

Având în vedere faptul că inputul pentru acest modul este un număr pe 8 biți, se va defini o precizie de tip parameter ce va avea valoarea 8. Această precizie va contribui în continuare la formarea dimensiunii noului input ce va fi shiftat cu 16 biți la stânga, deci *INPUT_LEN = OUTPUT_LEN + PRECISION*. Totodată, rădăcina pătrată va fi salvată în variabila pe 16 biți, *out_reg,* tot în acest context, definindu-se alte variabile auxiliare precum *new_in*, reprezentat pe 24 de biți datorită shiftării ce va urma, respectiv un *iterator* pe 4 biți, pentru a putea face posibilă iterarea în cadrul instrucțiunii for.

În interiorul blocului always, se începe prin a shifta la stânga cu 16 biți valoarea dată spre input pentru a obține virgula în format binar. Totodată, baza va fi aleasă ca fiind o putere a lui doi care generează cel mai apropiat rezultat de numărul pentru care se calculează rădăcina pătrată. Având în vedere faptul că acest procedeu presupune aplicarea radicalului pe un număr “z” shiftat cu 16 biți, unde z poate lua valoarea maximă 255 (fiind declarat pe 8 biți), reiese că principiul CORDIC presupune obținerea virgulei prin prisma radicalului $\sqrt{z\,.\ 2^{2n}}$, unde n este precizia specificată în cerință. Mai exact:

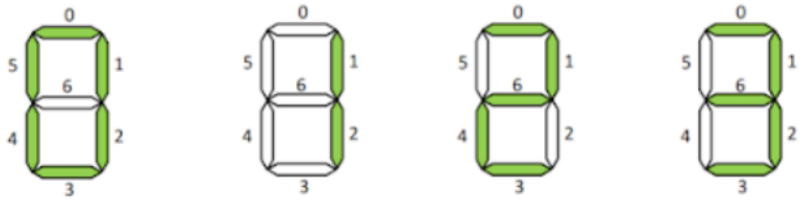
$$\sqrt{z\,.\ 2^{16}} = \sqrt{255\,.2^{16}} < 2^{12} < 4096$$

Astfel, se va alege puterea imediat mai mică, adică 11, *base = 2 ** 11*. Această bază va juca un rol esențial în formarea outputului în sensul că, la fiecare iterație, aceasta este adăugată la *out_reg*. Se verifică dacă nu cumva pătratul lui este mai mare decât inputul shiftat. În cazul în care este, se va scădea acea bază care inițial a fost adăugată. Iterația se termină cu împățirea bazei la 2, *base = base>>1;*. În final, la ieșirea din blocul always, *out* va lua valoarea lui *out_reg,* reprezentând rădăcina pătrată a lui *in*.

3. Modulul display_and_drop

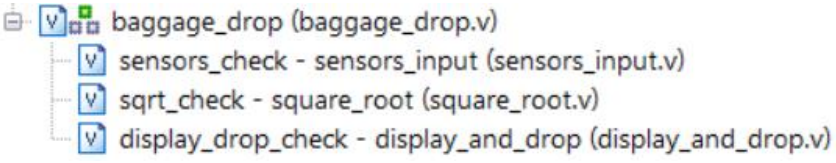
Acest modul va primi în continuare rezultatul de la modulul precedent, square_root ce a aplicat cu succes radicalul pe variabila *height*. Astfel, se interoghează timpul limită de coborâre a pachetului, *t_lim*, timpul curent posibil de coborăre a pachetului, *t_act*, (ce a fost calculat cu ajutorul formulei `t_act = sqrt(height) / 2 )` cât și posibilitatea lansării pachetului, *drop_en*, urmând să se afișeze unul dintre mesajele “CoLd”, “droP”, “Hot”.

Pentru acest modul s-au definit 4 variabile de tip reg pe 7 biți reprezentând fiecare literă din mesajul ce urmează să fie trimis pe display, cât și o variabilă intrinsec booleană de tip reg, pe 1 singur bit, *is_activated* ce are valoarea 1 în situația în care este confirmat faptul că acel container cu provizii poate fi aruncat. Pentru implementare acestui display s-a un Seven-segment display, fiecare segment aprinzându-se pe 1 logic, după cum urmează:



4. Modulul baggage_drop

Acest modul de natură “top”, are rolul de a instanția blocurile deja formate și enunțate mai sus pentru realizarea cu succes a misiunii. Așadar, va cuprinde toate elementele în ierarhie astfel:



Pentru acest modul s-au declarat variabilele *height*, de tip reg pe 8 biți, (ce va fi pasat modului sensors_input), *out*, pe 16 biți (ce va stoca rădăcina pătrată a lui height) și *t_act*, un reg de 16 biți (ce va fi calculat cu formula din cerință).

Prin urmare, după apelarea modulelor implementate, se va afișa informația corectă pentru a evita neutralizarea pachetului.