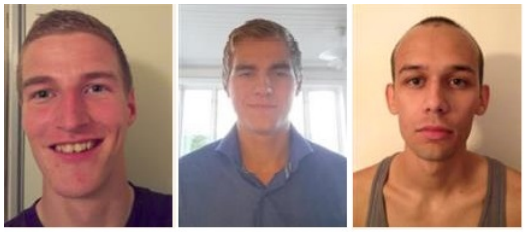
30010 - Programmeringsprojekt

Reflexball

Gruppe 3

Martin Boye Brunsgaard, s144012(1)
Tore Gederaas Kanstad, s144021(2)
Peter Asbjørn Leer Bysted, s144045(3)



1 2 3

Alle medlemmer har været tilstede under øvelserne, og deltaget i udarbejdelse af journalerne. Ydermere har arbejdet været fordelt ligeligt over gruppemedlemerne, og løst i fællesskab. Rapporten er blevet udarbejdet og gennemlæst i kollektiv.

Tecnical University of Denmark DTU
National Space Institute

30010 - Programming Project
24.06.2015

Abstract

This report covers the reflexbal game, which is a mandatory part of the B.Sc. EE course 30010 Programming Project.
The report documents the entire course of the exercises in which the digital logic behind a simple vending machine was designed using VHDL. The project was split into three sub-assignments: The first assignment was to drive a seven-segment display with hexadecimal numbers, the second was to display two different 2-digit decimal numbers simultaneously and the last was to implement a CPU using a data path controlled by a FSM. The 3 assignements were combined into one circuit and implemented on a Basys2 Spartan FPGA board.

Contents

1 Introduktion

Målet med dette projekt er at designe og implementere et program. Programmet skal skrives i C og det skal implementeres på en Zilog Z8 encore microprocessor vha. ZDS II - Z8Encore! 4.9.3 værktøjer. Programmet skal dokumenteres vha. flowcharts, grafer og beskrivelser af de enkelte funktioner.

Programmet skal være et spil, Reflexball. Spilleren styrer en striker, som skal bruges til at reflektere en bold, således den kan bevæge sig rundt på banen. Hvis bolden rammer en af kanterne, skal bolden ligeledes også reflekteres. Hvis spilleren ikke rammer bolden ryger bolden ud af banen, og spilleren fratrækkes et liv. Såfremt spilleren ikke har flere liv tilbage, afsluttes spillet. Desuden indføres der nogle bokse i spillet, som spilleren skal ødelægge. Når spilleren har ødelagt alle disse bokse går spilleren videre til næste bane, eller vinder såfremt han er på sidste bane. Den grafiske flade bliver implementeret ved at skrive til en terminal. Ydermere får brugeren fremvist informationer fra spillet på LED'erne på boardet.

2 Teori

Vi vil i dette afsnit gennemgå den basale teori bag binære tal og slutteligt indføre læseren i de forskellige formater, deriblandt fixed-point format, og hvorfor det er interessant at bruge denne repræsentation i vores projekt.

2.1 Binære tal

Et binært tal er et tal der kan udtrykkes i det binære talsystem/base-2, hvor grundtallet er 2. Da det er meget let at implementere i digital logik, er det et system der bruges internt i computere verden over.
Et binært tal består af bits, som svarer til et ciffer. Et bit kan have en af to tilstande: logisk højt eller logisk lavt. Dette medfører da hvis vi har n bits har vi 2^n forskellige tilstande. Disse forskellige tilstande kan fortolkes på forskellige måder, og vi vil i de næste afsnit gennemgå nogle af de forskellige representationer.

2.2 Unsigned repræsentation

I det binære talsystem er grundtallet vanligvis 2(det kunne potentielt også være -2). Det betyder således at i en n-bit streng, vil bittet yderst til højre være vægtet med 2^0, det næste med 2^1 op til 2^n gående mod venstre. Tallet 5(base-10) kan da skrives som i ligning 1

$$5_{10} = 101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \quad (1)$$

Med de indførte definitioner har vi kun mulighed for at repræsentere positive heltal. Vi ønsker også at kunne skrive kommatal og negative tal.

2.3 Fixed point kommatal

Kommatal kan indføres på en simpel måde, ved blot at vægte i omvendt retning når man går mod højre, således at bittet til højre for kommaet har vægtningen 2^{-1}, bittet 2 til højre for kommat vægtningen 2^{-2} osv. Hvis man har en n-bit streng med b tal til højre for kommaet, har man da muligheden for at skrive tal mellem 0 og $\frac{2^n-1}{2^b}$ [1, s. 4]

Tallet 13.625 kan f.eks skrives som

$$13.625_{10} = 1101.101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-3} \quad (2)$$

2.4 Repræsentation af negative tal

Hvis vi ønsker at repræsentere negative tal, gøres det oftest på 3 forskellige måder: Signed magnitude, 1's kompliment eller 2's kompliment. Vi vil her gennemgå signed magnitude og 2's komplement.

2.4.1 Signed magnitude

En af måderne at repræsentere fortegnet på bit-strengen, er ved at lade det mest signifikante bit(MSB: længst til venstre) bestemme fortegnet, hvor 0 indikerer et positivt tal og 1 indikerer et negativt tal. F.eks. kan tallet -37 i signed magnitude repræsentation skrives således:

$$-37_{10} = 1100101_2 \quad (3)$$

Signed magnitude repræsentation, har dog den ulempe, at man spilder et bit, f.eks. hvis man har en 4-bit streng gælder der at $1000 = 0000$, så istedet for at have 2^4 tilstande har man blot $2^4 - 1$.

2.4.2 2's komplement

En anden måde at repræsentere negative tal, kan gøres vha. 2's komplement. 2's komplement findes ved at invertere et unsigned tal og derefter lægge 1 til. 2's komplement har to fordele: Der er kun et 0, og subtraktion kan gøres

på samme måde som addition, så hvis vi ønsker at subtrahere 3 fra 5, skal vi blot finde 2's kompliment af 3 og lægge det til 5, som i 4

$$5 - 3 = 5 + (-3) \tag{4}$$

Disse fordele gør 2's komplement et af de mest udbredte metoder til at repræsentere negative tal i digitale systemer.

2.5 Fixed point vs floating

SHIT HERE

3 Design af Reflexball

I udarbejdelsen af dette program havde vi nogle forskellige tekniske krav og mål, som vi ønskede at designe programmet efter.

3.1 Tekniske mål

Vi lavede en liste af krav til programmets design som vi i så høj grad som muligt ønskede at overholde.

1. Vi ønsker en veldefineret struktur. Vi vil derfor undgå globale variable i så høj grad som muligt, derfor skal vi lave funktioner som tager pegere til strukturer eller variable som inputs, frem for at tilgå globale variable. Få undtagelser findes dog til dette, f.eks. i bibloteket der tilgår timeren.

2. Vi ville udvikle nogle biblioteker der var uafhængige af hinanden, således at vores grafik i mindst mulig grad kommunikerede med vores bibliotek indeholdende spillets back-end(Refball.c). Denne kommunikation skulle såfremt foregå igennem main-metoden, således man let kan få et overblik ved at se på main-metoden.

3.2 Krav til spillet

3.2.1 Krav til strikeren

Der blev stillet nogle krav til strikeren:

1. Strikeren skal maskimalt fylde 10% af skærmen på x-aksen.

2. Strikeren skal være delt ind i 5 forskellige områder. Disse 5 områder skal reflektere bolden på forskellig vis afhængig af indgangsvinklen og hvilken del af strikeren den rammer.

3. Brugeren skal kunne styre strikeren, vha. knapperne på boardet.

3.2.2 Krav til spillet

1. Spillet er et arkanoid spil, bestående af 3 levels. Banerne skal være i stigende sværhedsgrad. Dette gøres ved at boksene gøres stærkere, således de skal rammes flere gange for at ødelægges, og også tilføje flere kasser.

2. Spilleren har 3 liv til at starte med, og får et liv for hver bane han vinder.

3. Hvis spilleren ikke har flere liv tilbage, afsluttes spillet og der vises game over på skærmen. Brugeren har mulighed for at gå tilbage til menuen og starte forfra igen.

4. Når banen begynder, eller hvis spilleren mister et liv, placeres bolden over strikeren, og spilleren kan frit bevæge strikeren, hvor bolden følger efter. Hvis spilleren trykker på den givne knap, affyres bolden i en opadgående lodret linje.

5. Spillerens liv og tiden skal skrives på LED-skærmen når spillet er igang

6. Når spilleren får en power-up, vinder en bane, vinder spillet eller dør skal der rulles en tekst over LED-skærmene. Alt afhænigt af situationen, skal livene og tiden igen vises på skærmen efter teksten er rullet over.

7. Spilleren skal have mulighed for at få en power-up, således spilleren ødelægger kasserne uafhængigt af deres styrke, og ikke ændrer retning efterfølgende. Denne power-up skal kun være aktiv i et vist antal sekunder, og forsvinder hvis bolden ryger ud af banen.

3.3 U

Kildeliste

[1] Randy Yates, *Fixed-Point Arithmetic: An Introduction*, Digital Signal Labs 23. August 2007