
02321 Hardware/Software Programming

3 ugers projekt

Udarbejdet af:

Gruppe 10+14



s093485 - Christensen, Anders Jan



s093478 - Hansen, Mathias



s000536 - Vandall Zimsen, Jakob



s072657 - Tang Khoa Nguyen, Nikolai



s010164 - Mitri Emil Kjær Rebeiz,
Sami

Timeregnskab

Dato	Deltager	Design	Impl.	Test	Dok.	Andet	I alt	Kommentarer
1-3-11	Mathias	2	4				6	Færdiggørelse af wiring af rat som input device.
1-3-11	Anders	2	4				6	Block-diagram + memory
1-3-11	Nikolai	2	3				5	VGA
1-3-11	Jakob	2	4				6	VGA
1-3-11	Sami	2	3				5	
1-4-11	Mathias		6		1	1	8	Interface mellem rat og fpga færdigt
1-4-11	Anders						0	Syg
1-4-11	Nikolai		6				6	VGA
1-4-11	Sami						0	Syg
1-4-11	Jakob		8				8	Vga og Rat
1-5-11	Mathias	2	3		2	2	9	Memory
1-5-11	Anders	3	6				9	Memory + vga
1-5-11	Nikolai	2	2		3		7	Portmapping + Dokumentation
1-5-11	Jakob	3	6				9	Vga
1-5-11	Sami	1	2		4	2	9	Memory og rapport
1-6-11	Mathias	4	4				8	memory sammen med
				Gruppe 10+14				lc3 cpu
1-6-11	Anders	s093485 - 4	s072657 - 4	s093478 -	s010164 -	s000536	8	memory sammen

Akkumuleret timeregnskab

Mathias	Anders	Nikolai	Jakob	Sami	Total
119	100	81	114	63	477

Rollefordeling**Analyse/Design**

Lorem ipsum: ?

LC3 Implementering

VGA samt ROMs til VGA: Primært udarbejdet af Jakob & Anders

Hukommelse: Primært udarbejdet af Anders & Jakob

Rat I/O: Primært udarbejdet af Mathias

UART: Primært udarbejdet af Anders & Jakob

Sprites: Primært udarbejdet af Anders & Jakob & Mathias

Spil implementering

Spillogik(Instruktioner til LC3): Primært udarbejdet af Mathias

Hardware(Signaler og objekter): Primært udarbejdet af Jakob & Anders

Test

Lorem ipsum: ?

Dokumentation

Lorem ipsum: ?

Indholdsfortegnelse

1 Opgave formulering.	1
2 Design	4
2.1 Design valg	5
2.1.1 Wrapper	5
2.1.2 Padding	6
2.1.3 Tri State Buffer	6
2.1.4 Chip Selector	6
2.2 Ram	6
2.3 VGA	7
2.4 Rat	7
2.5 DIO4 I/O	7
2.6 UART.	8
2.7 Grafik.	8
2.7.1 Tiles	8
2.7.2 Sprites	8
2.8 LC3 Instruktioner	9
2.9 Highscore database.	9
3 Diskussion.	11
3.1 Videreudvikling	12
4 Konklusion	13
4.1 ET UDKAST!!!.	13
Kildehenvisninger.	13
Bilag	A-1

Figurer

1.1 Simpelt mock-up over systemet	2
1.2 Videreudviklet mock-up over systemet	2
2.1 Design system	5
2.2 Screenshot af highscore applikationen	9

Kapitel 1

Opgave formulering

I dette 3 ugers projekt skal der sammensættes en komplet computer, baseret på von Neumann arkitekturen udfra en udleveret implementeret LC3 CPU, computeren implementeres på et Virtex-II Pro FPGA board.

LC3 Cpu'en er leveret til os som en black box, hvortil vi skal tilslutte de andre nødvendige komponenter for at have en komplet kørende computer; memory, vga samt et rat påsat en A/D-converter som I/O til LC3'en. Et eksternt stykke hardware i form af en skærm er også nødvendig.

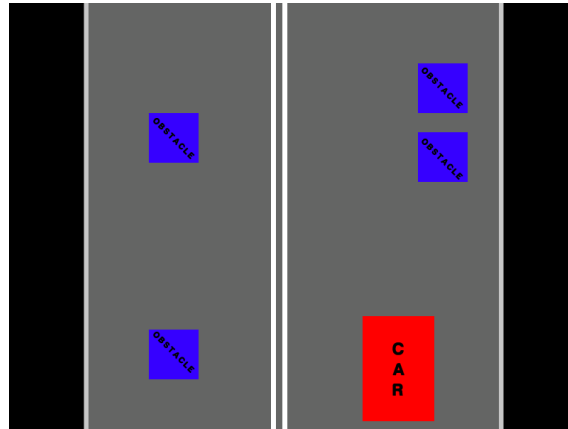
Såfremt det bliver nødvendigt, kan der være tale om at dele af spillet implementeres på en almindelig PC og med data sendt over seriel forbindelse til LC3 computeren. Derefter udvikles et spil baseret på den implementerede LC3 computer. Det er valgt at udvikle et bil spil, hvor bilen skal styres af et fysisk rat eller tastatur. Målet med spillet er at undgå forhindringer i form af andre biler, kasser og lignende som indsættes på kørebanen tilfældigt til forskellige tilfældigt tidspunkt.

På figur 1.1 på den følgende side ses et simpelt mock-up over hvordan spillet kunne se ud, hvor der er indtegnet en simpel baggrund, en bil man styrer samt forhindringer man skal undgå.

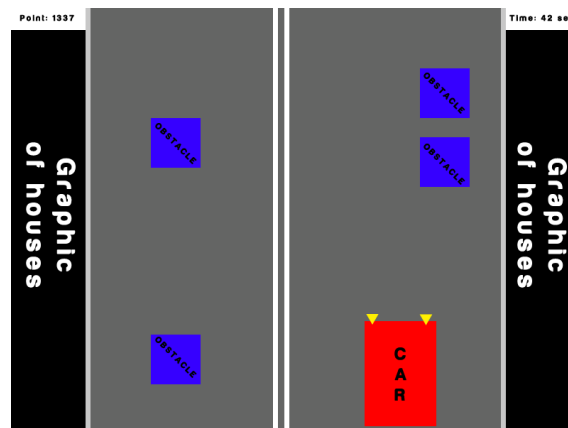
Afhængigt af tiden op til aflevering, påtænkes der at udvide projektet med en eller flere ekstra tilføjelser. Disse tilføjelser kunne f.eks. være at indsætte flere forskellige forhindringer på vejen, benytte forskellige baggrunde, have flere forskellige baner, lave bedre grafik eller at implementere lyd.

På figur 1.2 på næste side ses et tænkt videreudviklet mock-up over spillet, hvor points og tid også er vist.

Vores mål og krav er, at implementere et minimum af funktionalitet, som beskrevet i den opgave case som ligger til grund for projektet, inden for den givne tidshorisont. Ikraft af



Figur 1.1: Simpelt mock-up over systemet



Figur 1.2: Videreudviklet mock-up over systemet

vores roller som udviklere, stiler vi naturligvis efter at lave det bedst kørende simulator spil på den bedst kørende LC3 computer. Tiden er dog en afgørende faktor for, hvor meget vi når at implementere ud over vores minimum krav. Vi påtænker at programmere hardwaren i VHDL og spillogik i C.

Desuden er det ønskeligt, at bruge viden fra andre af semesterets kurser til projektet, herunder f.eks. implementering af en database til at gemme spiller navne og highscores i spillet, med henvisning til kurset 02344 OOAD og databaser.

Kapitel 2

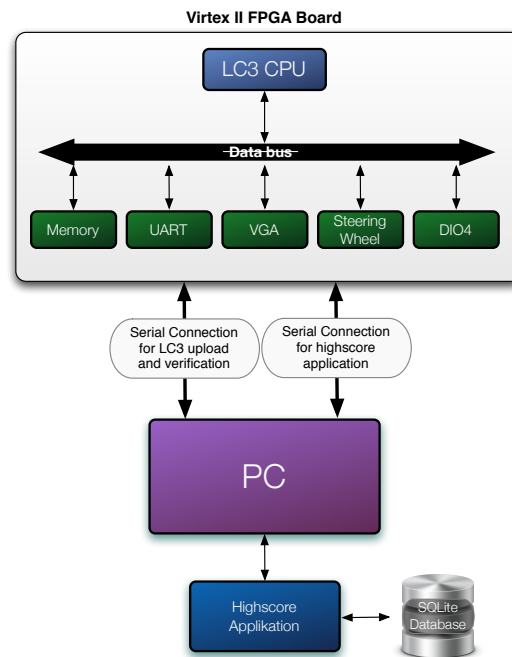
Design

2.1 Design valg.	5
2.1.1 Wrapper	5
2.1.2 Padding	6
2.1.3 Tri State Buffer	6
2.1.4 Chip Selector	6
2.2 Ram	6
2.3 VGA	7
2.4 Rat.	7
2.5 DIO4 I/O	7
2.6 UART	8
2.7 Grafik	8
2.7.1 Tiles.	8
2.7.2 Sprites.	8
2.8 LC3 Instruktioner	9
2.9 Highscore database	9

Vores system som overordnet er beskrevet på figur 2.1 på modstående side, er opbygget af et FPGA board der implementerer en computer med en black box LC3 CPU. Til computeren er der af eksterne I/O komponenter tilknyttet en skærm og et I/O board¹, til Digilent boardet er der tilsluttet et rat som essentielt er en variabel modstand. Desuden benyttes der to-vejs serial kommunikation til en PC til upload af bruger-programmer til LC3 computeren samt verificering af disse.

Spillogikken er skrevet på PC i C kode, derefter skal indholdet overføres til FPGA board vha. en serial forbindelse. Indholdet af spillet gemmes i block ram på FPGA. En VGA skærm viser output og er opdateret løbende, for at man kan se og følge med i hvad der

¹Digilent DIO4™I/O board



Figur 2.1: Design system

sker og dermed kan styre spillet. Spillet kan styres direkte fra FPGA board ved hjælp af en styrings enhed, denne er et rat der er blevet implementeret.

LC-3 system design består af forskellige komponenter som er sat sammen. De forskellige komponenter er CPU, rat I/O, Memory (RAM og ROM), UART, VGA og DIO-4 board.

Til dette projekt har det været nødvendigt at tilkoble en VGA skærm og et rat via en A/D converter.

Design valg

2.1

2.1.1 Wrapper

Til LC3 systemet valgtes det at bruge wrappers til de vhdl moduler der både har et input og et output, dette gjorde det nemmere at samle padding og tri state buffers. Ved at lave en wrapper kunne der sendes signaler videre ned i LC3 systemet fra bussen.

2.1.2 Padding

Padding er en metode hvorved længden af en bitstreng ændres. Da det var nødvendigt, at ændre bus adresse længden i f.eks. UART'en fra en længde på 16 bit ned til 8 bit, da det er den længde UART forventer. Dette gøres ved, at når UART wrapperen modtager et

signal sender den kun de 8 mindst betydende bit videre ned til UART'en at arbejde med, og når et signal modtages fra UART'en padder der med 0'er foran, så der bliver sendt et signal tilbage til bussen der er konstrueret således '00000000' samt UART signalet. Derved ender vi med et signal på 16 bit igen, som er den længde bussen arbejder med.

2.1.3 Tri State Buffer

En tri state buffer fungerer således at der bliver sendt et enable signal og et andet signal der angiver om der må skrives eller læses fra et element, og derudover sørger for at hvis det ikke er det element der skal bruge signalet, kun vil modtage Z'er der i et signal er en tom værdi. På denne måde er det muligt at styre, hvilke elementer der bliver skrevet og læst fra på en given clock cycle.

2.1.4 Chip Selector

Chip selector gør det muligt at lave en kontrol af hvilke adresser der bliver skrevet til og igennem et I/O adresse register, er det blevet bestemt hvilke værdier der har med hvilke signaler at gøre. Ved at tjekke på denne adresse kan man derved bestemme om et signal til tri state bufferen skal være 1 eller 0.

Ram

2.2

LC3 complete system RAM implementering. Til projektet valgtes det at bruge single-port synchronous RAM. Der blev valgt single-port RAM på baggrund af at der i første omgang ikke var brug for at skrive til eller læse fra 2 adresser samtidig, som dual-port RAM ville have givet mulighed for. Synchronous tillader at bruge block RAM på FPGA boardet, og ikke CLB RAM², der er forbeholdt til logik og asynchronous RAM. Til dette projekt var der ingen grund til at bruge asynchronous, da der ikke er behov fra parallelle udregninger i samme clock cycle i dette projekt. Implementeringen af RAM skete ved at følge bogen og derefter justere længden af adressen så det passede til den størrelse af det designede program der blev kørt på FPGA'en. Ved at reducere størrelsen på RAM sørgede dette for at syntetiseringen af LC3 systemet ikke tog unødvendigt langt tid. Der blev dog løbende ændret på adresse længden af hukommelsen, da C koden blev længere, og krævede en større hukommelse.

Til design af RAM blev der brugt følgende design valg: Der blev lagt en wrapper omkring RAM for at gøre det nemmere at benytte padding. Derudover er der benyttet en tri state buffer til at afgøre om der skulle læses eller skrives til RAM.

² – Indsæt footnote her –

VGA

2.3

I starten af projektet blev der udarbejdet et VGA sync modul, på grund af fysisk plads mangel måtte dette modificeres til at benytte XSGA på Virtex-II PRO motherboardet. Dette blev gjort ved at tage enhed 12.1 vga_sync fra lab bogen³ og modificere den om til de timings som motherboardet bruger⁴. Efter test af sync enhed blev 640 x 480 @ 60 Hz valgt som standard indstilling for projektet.

Rat

2.4

Det er valgt at benytte et rat som input. Rattet er et gammelt playstation/pc rat som er skilt ad for at bruge rattets egenskaber og komponenter. Når rattet drejes påvirkes en variabel modstand som giver et analogt signal om rattets position, signalet læses af en 12 bit A/D converter⁵ og kan på den måde fortolkes af LC3 processoren, når rattet er drejet helt til venstre er outputtet fra A/D converteren 0 mens at outputtet er 4096 når rattet er drejet helt til højre, A/D converteren er forbundet til et interface board fra Digilent⁶

Der er desuden implementeret andre dele af rattets input og output funktionaliteter; en knap på rattet er forbundet til FPGA boardet igennem samme interface board, knappen er dog ikke i brug i det endelige projekt⁷. Endvidere er to vibratorer fra rattet også styret af FPGA boardet således at det kan gives feedback når man kører ind i en forhindring, styringen af disse er udviklet i samarbejde med nogle diplom elektro studerende⁸.

DIO4 I/O

2.5

I projektet blev DIO4 I/O brugt til debugging og single stepping, hovedsageligt til de første test for at undersøge om RAM og CPU kommunikerede korrekt sammen, ved at bruge det udleverede test program, hvor programmet blev stoppet, og derefter kunne single steppes igennem kode linjerne. Dette var dog nemt, at implementere da det ikke var nødvendigt at lave ændringer på koden fra bogen. Inputs er implementeret i den grad at knapperne og switches kan benyttes i c-koden, mens outputs 7 segment displayet og led ikke kan.

³FPGA PROTOTYPING BY VHDL EXAMPLES

⁴http://www.digilentinc.com/Data/Products/XUPV2P/XUPV2P_User_Guide.pdf table 2-6 page 37

⁵En analog til digital converter konverterer et analogt signal til et digitalt signal således at FPGA boardet kan læse fra ikke-digitale komponenter.

⁶Digilent FX2 MIB

⁷Der henvises til afsnittet Diskussion vedrørende videreudvikling af projektet

⁸Lasse Møller, s093440 og Emil Møller, s08310 hhv. 3. semester diplom elektro studerende og 5. semester diplom elektro studerende.

UART

2.6

UART er en forkortelse for Universal Asynchronous Receiver and Transmitter. UART blev designet ud fra eksemplet i bogen, UART'en var nødvendig for at kunne uploade C kode på LC3 systemet. Implementeringen af denne var kompliceret, men nødvendig da den blev benyttet til test, om kommunikation mellem RAM og den black box CPU der var stillet til rådighed. Til UART'en var design valget at pakke den ind i wrapper for at gøre padding nemmere. Der blev benyttet en chip selector til at bestemme hvilke af vores I/O signaler der skulle benyttes.

Grafik

2.7

Til projektet efter minimum implementeringen var færdiggjort, blev grafikken til spillet udvidet.

2.7.1 Tiles

Tiles blev lavet ved hjælp af værktøjer⁹, udarbejdet i forbindelse med projektet. De genererede hex arrays blev lagt ind i en Tile ROM. Værdierne 0 eller 1 arrayet bruges til at bestemme farven på hver pixel. I anden ROM er forgrunds og baggrundsfarve defineret for hver tile, disse farver er af 8 bits længde da der zero paddes.

2.7.2 Sprites

Sprites blev tegnet i Adobe Photoshop og gemt i png format, derefter blev de konverteret til et hex array ved hjælp af online værktøjer¹⁰, udviklet i forbindelse med projektet. Disse Hex arrays blev lagt ind i en VGA_ROM som tegnede rammen for objektet, derefter blev der ud fra png filen lave en inverted udgave som blev gemt i en anden ROM som udgjorte masken til objektet. Til hver af disse ROM der tildelt 1 farve hvor de værdier der er sat til 1 fik denne farve. Ved hjælp af denne metode kunne rammen og masken få 2 forskellige farver og danne et objekt med 24 bit farver.

LC3 Instruktioner

2.8

Instruktionerne til LC3 processoren er skrevet i C. Udgangspunktet i programmet er main løkken. Der er implementeret en variabel `game_state` som definerer hvilken game state spillet er i, afhængig af spillets game state foretages forskellige handlinger i main løkken.

⁹http://matisen.dk/dtu/02321/tile_maker.php

¹⁰<http://matisen.dk/dtu/02321>

Der er implementeret en `sleep(...)` metode som ved hjælp af `VGA_REFRESH_TICK` udfører en simpel forsinkelses handling.

```

45 void sleep(unsigned int ticks, unsigned int multiplier)
46 {
47     unsigned short i, m;
48     short data;
49
50     for (m = 0; m < multiplier; m++)
51     {
52         for (i = 0; i < ticks; i++)
53         {
54             do
55             {
56                 data = io_read(VGA_REFRESH_TICK);
57             }
58             while (data != 1);
59             io_write(VGA_REFRESH_TICK, 0);
60         }
61     }
62 }

```

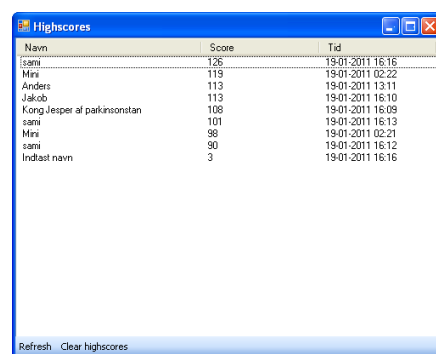
Da LC3 cpu'en ikke understøtter `long` er der istedet tilføjet et `multiplier` argument til metoden som blot venter det angivne antal `ticks` ganget med `multiplier` ved hjælp af et for loop.

Highscore database

2.9

Som proof-of-concept er der udviklet en applikation til PC'en som kommunikerer med LC3 Computeren over en serial forbindelse for at lagre highscores i en database, applikationen er skrevet i C#.

Applikationen opretter en tråd hvis formål er at poole for data på serial forbindelsen, af den årsag er det heller ikke muligt at køre andre programmer samtidig som bruger serial forbindelsen. Når en linje modtages med scoren oprettes et "Game Over" vindue hvor det er muligt at skrive sig på highscore listen, spillet er imens på pause indtil karakteren 's' (for start) sendes tilbage over serial forbindelsen af highscore applikationen når highscoren er gemt hvorefter spillet nulstiller og starter forfra.



Navn	Score	Tid
sani	126	19-01-2011 16:16
Mini	119	19-01-2011 02:22
Anders	113	19-01-2011 13:11
Jakob	113	19-01-2011 16:10
Kong Jesper af parkinsonstan	108	19-01-2011 16:09
sani	101	19-01-2011 16:13
Mini	98	19-01-2011 02:21
sani	90	19-01-2011 16:12
Indtast navn	3	19-01-2011 16:16

Refresh Clear highscores

Applikationen benytter en database til at lagre highscores, det er valgt at benytte en `sqlite`¹¹ database da denne danner baggrund for en simpel og hurtig database gemt i en enkelt fil i modsætning til alternativer som f.eks. MySQL som benytter kræver en database server.

Figur 2.2: Screenshot af highscore applikationen

¹¹For yderligere information om SQLite databaser se <http://sqlite.org>

En simpel tabel er oprettet til at indeholde highscore listen, bemærk at `submitted_at` angiver tiden for oprettelsen af scoren i unix tid¹² formatet.

```
1 CREATE TABLE highscore(  
2   name varchar(200),  
3   score int,  
4   submitted_at int  
5);
```

¹²Unix tid er angivet i antal sekunder siden d. 1. Januar 1970

Kapitel 3

Diskussion

3.1 Videreudvikling 12

Igennem 3 uger projektet er der blevet arbejdet med at designe og implementere et 2 D bilspil. Der er blevet lagt en stor arbejdsindsats fra gruppen,, blandt andet ved at arbejde i weekenderne, og derfor har det været muligt at nå minimumsimpliciteringen tidligt i forløbet. Dette har gjort at der er blevet tilføjet flere features løbende. Spiller objektet er gået fra at være en rød kasse, der kunne bevæge sig til højre og venstre på skærmen til at være en tegnet sprite, der forestiller en racerbil med 2 forskellige 24 bit farver. Objekterne spilleren skal undgå, er ligesom spiller objektet, gået fra at være en boks til at være en sprite i 2 forskellige 24 bit farver. De modstandere der blevet tegnet er en guld nissan, en lilla ATV, og en army grøn tank. I stedet for at styre spillerobjektet med knapper på DIO4 I/O boardet, er det muligt at styre spillerobjektet med et modificeret PS2/PC rat der er tilsluttet via en A/C converter. Dette var ikke så svært som forventet, og giver meget til spil oplevelsen at man kører en bil. Der er også blevet udviklet en highscore applikation, der kører på en lokal pc, der via serial kablet kan sende spillerens score til en database hvor der gemmes navn og score, dette trækker på viden fra kurset 02344 OOAD og Databaser.

Der var en del udfordringer under vejs, nogle af de mest tidskrævende var at få UART og RAM til at virke efter hensigten, da disse var nødvendige for at kunne uploade noget til FPGA boardet, dette blev dog gjort nemmere ved de udleverede testprogrammer, så fokus ikke var på udviklingen af brugbare testprogrammer. Derfor var det nødvendigt at bruge nogle design valg, der gjorde det muligt at sende og modtage fra disse to VHDL moduler til bussen. Da dette var på implementeret kunnet udviklingen af spillet og dets logik begynde. Da det grundlæggende i LC3 computeren var på plads, begyndtes arbejdet med spillogik og anden styring fra C, dette gjorde arbejdet nemmere.

Videreudvikling

3.1

Hardware Videreudvikling Der er latches i projektet, hvilke kan give fejl og gøre det meget svært at rette disse, så til videreudvikling af hardwaren var dette noget, der ville prioriteret. Der kunne gøres noget ved opbygningen af ROMs, da disse er implementeret og fungerer, men ikke er lavet efter en entydigt plan. Tilslutte lydkortet, og lave simpel lyd i spillet.

Spil Videreudvikling Der er opnået produkt der kører og fungere som et bilspil, hvilke var målet med projektet. Derfor er der stadig mulighed for udvidelser af spillet. Der kunne være bedre grafik, i og med, de 24 bit farver ikke bliver udnyttet til fulde som projektet er nu. Flere baner med forskellige grafik på vejen var også en oplagt mulighed for en udvidelse til spillet. En større database med flere data gemt, ud over point, f.eks tid, statistik over hændelser i spillet, hvis det var samme spiller der spillede. Power ups blev også diskuteret, da der allerede er implementeret 1 knap på rattet der ikke bliver benyttet til noget. Hvilke ville kræve mere detaljeret brugerflade, med både score, power ups, bane og liv på skærmen. En anden udvidelse kunne være at have 2 spillere, enten på samme computer eller over nettet. Ved udvidelse med flere baner, lave en save/load funktion til spillet, så en spiller kan vende tilbage på et senere tidspunkt.

Kapitel 4

Konklusion

4.1 ET UDKAST!!!	13
----------------------------	----

ET UDKAST!!!

4.1

Vi har med dette projekt fået en bedre forståelse for sammenspillet mellem hardware og software, hvordan de nødvendige komponenter i LC3'en bruges i samspil med et stykke software - her vores simulator spil. Vi har udviklet os fremadrettet mht. at forstå hvad der egentligt sker på komponent niveau når man bruger en computer, der kører noget software. Vi føler os bedre til at kunne abstrahere på forskellige niveauer mht. at "dykke ned" på/i komponent niveau og få et sådant til at fungere, i sammenspil med et stykke software (her vores simulator spil) og bevæge os op og ned mellem de forskellige abstraktions lag iht. low-level/high-level udvikling. Det har været meget interessant at lave/kode de respektive komponenter(så som memory, VGA.. etc) vi har brugt og derefter få dem til at "snakke" sammen med vores spil. Det har været et meget interessant projekt at gå i krig med, det har givet en meget bedre forståelse for hvad der egentligt sker "inden i" en computer, når man som bruger "bare" sidder foran skærmen og bruger den til div. ting. Vores viden er blevet bredere efter dette projekt. Dog er tidsfaktoren en afgørende faktor for udviklingen af produktet kontra ønskede opnåede mål, vi forestiller os at man ude i erhvervslivet har mere tid til udviklingen, men vi er generelt tilfredse med det vi har nået på den givne tid.. :)

Bilag

I/O Registers

.1

Address	Description
xFE00	Stdin Status Register
xFE02	Stdin Data Register
xFE04	Stdout Status Register
xFE06	Stdout Data Register
xFE0A	Switches Data Register
xFE0E	Buttons Data Register
xFE12	7SegDisplay Data Register
xFE16	Leds Data Register
xFE18	Steering Wheel Status Register
xFE1A	Steering Wheel Data Register
xFE1C	Car VGA X position
xFE1E	Obstacle 1 VGA X position
xFE20	Obstacle 1 VGA Y position
xFE22	Obstacle 2 VGA X position
xFE24	Obstacle 2 VGA Y position
xFE26	Obstacle 3 VGA X position
xFE28	Obstacle 3 VGA Y position
xFE2A	VGA Refresh Tick register
xFE2C	Vibrator register
xFE30	VGA Road movement data register

Kildetekst

.2

"Spil manual": Efter udviklingen af en masse VHDL til hardware implementeringen og for vores vedkommende, dertil svarende C-kode, skal man i Xilinx miljøet compile sit projekt for at få lavet en bit fil som man skal uploade til FPGA boardet. Når dette er gjort, er LC3 computeren kørt i stilling/gjort klar til at kunne køre selve spillet ("softwaren"). Efter endt kompilering, hvilket principielt er en form for fejlfinding (idet der tjekkes om alle signaler og komponenter er korrekt sat sammen), skal man compile C-koden og derefter er man klar til at køre/spille spillet. Som beskrevet tidligere i rapporten, bruger vi et rat som input til at kommunikere med spillet og styre bilen. For at få en kort gennemgang af hvordan man bruger spillet, tjek venligst "Bruger Manual"afsnittet.

Indholdsfortegnelse

Figurer
