Digitala Projekt - Snake
Grupp - 7

Erik Ljung, d01elj
Erik Simmons, d01es

25 maj 2005

Innehåll

1 Introduktion

Syftet med detta projekt är att introduceras i hur man jobbar från idè till färdig prototyp. Under projektes gång har idè kläckts, kravspecifikation har skrivits, plannering gjorts, utveckling, dokumentation och diverse felsökande.

Arbetet innefattar hårdvarukonstruktion, i allt från databladsstudier till lödning och virning av komponenter. Mjukvara har utvecklats parallet för att senare kunna testköras och anpassas till specifikt byggd prototyp.

Gruppmedlemmarna har goda kunskaper inom programmering, b.la lågnivå programmering i C och assembler. Därav valdes ett mer hårdvarubetonat projekt. Systemet byggdes runt en Motorola 68008 med diverse kringutrustning. Under projektes gång har den usprungliga designen utökats och extra komponenter har lagts till.

Målet var en fult fungerande prototyp av spelet Snake. En LCD-display plus en knappsats fungerar som användargränssnitt och samtliga enheter kontrolleras av Motorola-processorn.

2 Hårdvara

Det naturliga valet var ett system baserat på en Motorola 68008, vilket ledde till mer hårdvarukonstruktion. För ett fungerande system krävs minnesmoduler, en LCD, logik och en hel del analoga komponenter. Ett fullständigt kopplingschema återfinnes i Appendix A.

2.1 Processor - Motorola 68008

Motorolas 68008 är är en variant av deras klassiska 68000. Den har möjlighet att adressera upp till 1MB på sina 20 adressbitar. En 8 bitars bus transporterar data mellan processorn och pereferi enheterna. LCD:n som är en synkron modul kräver synkronierad dataöverföring vilket 68008 har ett speciellt tillstånd för, normalt arbetar processorn asynkront.

2.2 Klocka

En klocka konstruerades enligt ett givet exempel och fungerande direkt utan några större problem.

2.3 Minne - SRAM, EPROM och EEPROM

Designen använder sig av tre olika typers minne. Som programminne används ett brännbart EPROM (32k * 8). Här lagrar vi själva spelet och bootrutiner. Data som sparats i EPROM (bränns, engångsföretelse) ligger kvar även då spänningen över minnet slås ifrån. Det är ej möjligt att skriva till EPROM:et under exekvering.

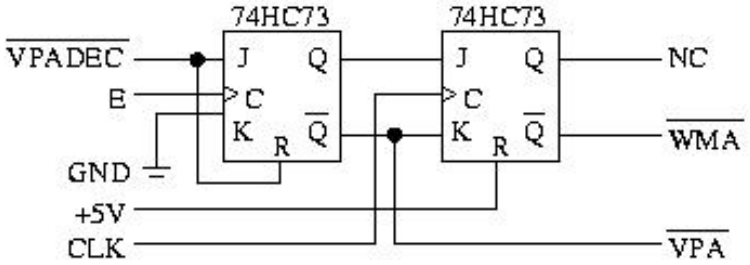
Vår applikation (Snake) kräver dynamisk data , dvs uppdateras under exekvering. Data är endast intressant under själv spelets gång och behöver då var

tillgänglig kvickt. Som RAM har vi därför valt en SRAM-modul (32k x 8) där samtlig information om den pågående spelomgången lagras.

Informationen som sparas i SRAM:et försvinner när strömmen slås ifrån vilket medför att en hårt förvärvad highscore kommer glömmas bort. Där för har ett tredje minne lagts till desigen, ett EEPROM. Fördelen med EEPROM:et är att data som lagrats komms ihåg när strömmen stängs av. EEPROM:et ganska långsamt men räcker bra för att skriva ner resultat efter fullföljd spelomgång.

2.4 LCD - Batron 128x64

LCD:n är en synkron enhet vilket medför att signalerna på databusen måste synkroniseras mellan CPU:n (Motrola 68008) och LCD:ns drivkretsar. För att försätta 68008 i synkront läge krävs två J-K vippor kopplade enligt figur 1. Från kopplingen tas styrsignalen WMA för att generera CS (Chip Select) synkront till LCD:n. Signalerna VPA och E används för att styra 68008:an till synkront läge.



Figur 1: LCD - Synkroniseringsnät

2.5 Logikretsar - PALCE22V10

För att styra rätt samtliga singnaler i desigen samt att se till att rätt chip aktiveras används 2st programmerbara logikkretsar. Samtliga enheter som är kopplade till bussarna (adress resp. data) har så kallade tri-state buffrar på sina in-/utgångar. Det innebär att förutom låg/hög kan porten även anta ett tredje värde (därav namnet tri-state), hög impedans. I det läget påverkar kretsen inte bussen vilket gör att signalerna till en annan krets inte förstörs. Logikkretsarna ser till att aktivera rätt chip och hålla de andra i hög impedans.

2.6 Knappar - Latch

Användarens gränssnitt till konstruktion är ett flertal knappar. Fyra stycken för att kontrollera spelet, en resetknapp och en pausknapp. De fyra kontrollknapparna är kopplade till en latch som låses av processorn vid läsning. Eventuella knappstudsar tas om hand av mjukvaran. Resetknappen är konstruerade med hjälp av en resetkrets (DS1233) vilken genererar en korrekt resetsignal till processorn och LCD:n. Pausknappen är kopplad via två J/K-vippor, en för att

generera ett avbrott och den andre för att låsa en lysdiod då mjukvaran är pausad.

2.7 LED:s - Övrig logik

Desigen är utsmyckad med lysdioder. En för strömförsörjning och en som triggas vid reset. I övrigt finns ett par J/K-vippor, vilka genererar synkroniserings signaler för dataöverföringen mellan processorn och LCD:n.

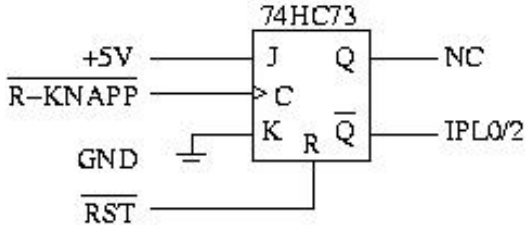
2.8 Adressering av komponenter

Processorn pratar med sin kringutrustning genom att en krets "enablas" av en styrsignal. Styrsignaler genereras av logikkretsarna vilka styrs av adressbussen. Beroende på vilken adress processorna läser/skriv "enablas" rätt krets. Under projektets inledning togs beslut om att lämna gott om plats mellan komponenterna för att eventuell senare kunna byta ut de mot större. Följande tabell visar hur adressrymden fördelats mellan de olika komponenterna.

Krets	*Adress*
EPROM	0x00000 - 0x03fff
SRAM	0x08000 - 0x0ffff
LCD1 - Instr	0x10000
LCD1 - Data	0x10001
LCD2 - Instr	0x20000
LCD2 - Data	0x20001
EEPROM	0x40000 - 0x41fff
Avbrottsreset	0x80000

2.9 Avbrottsgenerering

En låskrets i form av en J/K-vippa plockar bort knappstudsar och triggar ett avbrott på nivå 5 (IPL0/2 går låg) hos processorn (se figur 2). Efter avbrottet behandlats skriver processorn till en specifik adress. Logik generar då en resetsignal (RST) till vippan, som nollställs och återigen tillåter avbrott. Ett enkelt men effektivt sätt att försäkra sig om endast en avbrottssignal.



Figur 2: Avbrott - Kopplingsschema

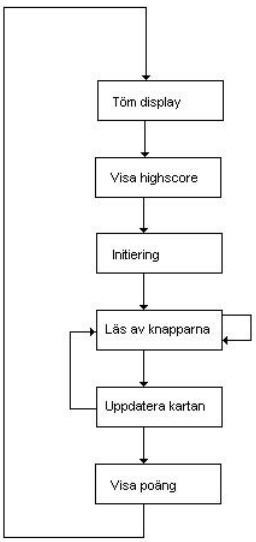
3 Mjukvara

Mjukvaran för ett inbyggt system skilljer sig från ett "vanligt" program på en PC med operativsystem. T.ex börjar processorn exekvera på en specifik adress som sätts av assemblerinstruktioner. Assemblerinstruktioner används för att sätta värden på register (programräknare, stackpekare m.m.).

Innan mjukvaran testades, m.h.a. utvecklingsverktyget (it-68), skrevs en version för PC. Den var snarlik den som sedan kördes på vårt system, vilket medförde enklare debuggningen av programmet.

För styrning av LCD:n respektive knapparna skrevs två separata drivrutiner, vilka inkluderades i huvudprogrammet. Drirutinerna är helt självstående och kunde utnyttjas till några små testprogram som skrevs under utvecklingsfasen.

Programmet består av en evig loop som inleds med att visa highscore, därefter startar spelet och fortsätter tills någon av maskarna träffar en vägg, sig själv eller den andre masken. Slutligen visas spelets vinnare och varje spelarnas respektive poäng skrivs ut. Programmets sekvens illustreras i figur 3.



Figur 3: Flödesschema - Program

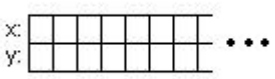
Den inre loopen läser endast knapptryckningar och exekveras med hög frekvens. Den mellersta loopen styr updateringen av LCD:n och själv spelets gång. Genom att öka eller minska frekvensen kan man reglera maskarnas hastighet och därmed spelets svårighetsgrad.

Vid ett avbrott (pausknappen trycks ner) sätts en flagga, vilken bryter vilkoret för den inre loopen och spelet försätts därmed i "paus". När ett nytt avbrott genereras (pausknappen trycks ner igen) ändras flaggans status och programmet fortsätter att exekvera.

3.1 Datastrukturer

Spelplanen lagras i en matris där varje element representerar en pixel på LCD:n. Varje element representerar olika "föremål" i spelet (vägg, mat, ...), därför valdes byte som elementstorlek. Eventuellt kunde två elements sparats i samma byte för att minska minnesanvändningen. Det fanns gott om SRAM att tillgå, så denna lösning kändes onödig.

Varje mask representeras av en $2 * N$ matris, där N är maskens maximala längd. Elementen i matrisen är x- respektive y-kordinaterna för maskens samtliga delar (en del per pixel). Figur 4 förtydligar maskens representation i minnet.



Figur 4: Datastruktur - Mask

4 Slutsats

Designen av mjukvara var ganska rakt fram och inga större problem uppstod. Större del av koden skrevs redan innan hårdvaran var färdig konstruerad. Testning av moment, ej specifika för den speciella desigen (t.ex spellogik), gjordes på en PC för att förenkla felsökandet.

Plannering och byggandet av hårdvara gick stort sett enligt planerna. De grundläggande kraven var ganska tidigt uppfyllda och utökningar så som avbrottshanterad pausfunktion, EEPROM för highscore och diverse utsmyckningar med lysdioder lades till.

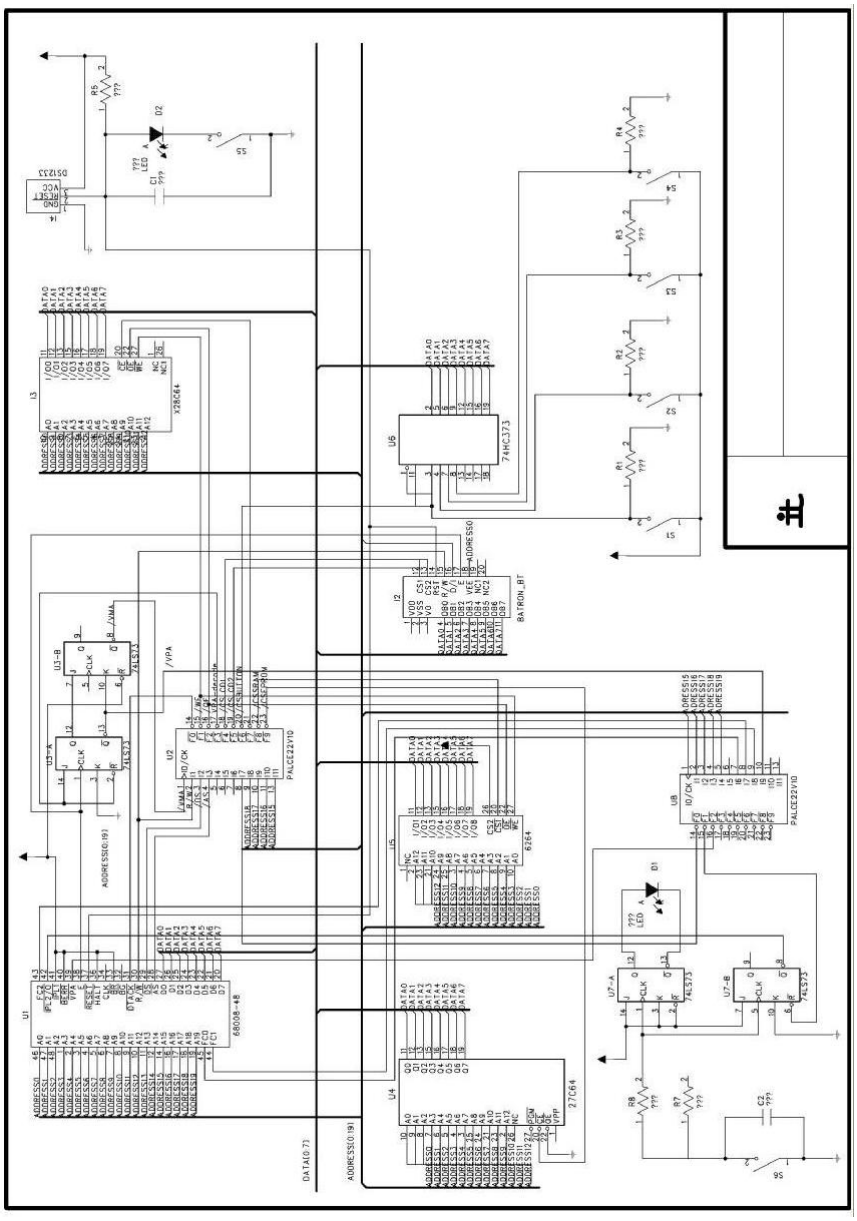
Under projektets gång har diverse mindre problem dykt upp. Problemen har lösts ganska omgående med hjälp av tankeverksamhet, datablad och ett antal diskussioner med handledaren.

Störst problem uppstod då utvecklingsvertyget plockades bort och en "riktig" Motorola 68008 kopplades in. Processorn fastnade i ett "wait-state" direkt efter en reset. Efter felsökning och konsulterande med handledaren konstruerades en ny resetknapp, uppbyggd kring en resetkrets. Den nya kopplingen löste

problemet och desigen fungerar självständigt, endast matningsspänning behövs tillföras.

A Kopplingsschema

Fullständigt kopplingschema över designen.



Figur 5: Kopplingsschema