Digitala Projekt (EDI021) - Tunnel 68K

d01mbr@student.lth.se

2007-12-04

Sammanfattning

The goal of this project was to build a simple arcade game using a Motorola MC68008 cpu coupled with a number of peripheral devices such as a pixel based display and a few buttons. The game consists of a small space ship in a tunnel with constantly moving and shrinking walls, and the object of the game is to move the space ship sideways so as to avoid colliding with either wall. First the hardware was designed, then assembled and finally tested. When the hardware had passed all tests sufficiently, the software was developed and tested until the project goals could be considered fulfilled.

Innehåll

1 Kravspecifikation

Projektets slutresultat ska bli ett mycket enkelt spel där ett rymdskepp flyger i en tunnel och försöker undvika väggar som rör sig i sidled och samtidigt sakta krymper.

De komponenter som utgör gränssnittet mot användaren är en pixelbaserad display och tre knappar. Rymdskepp och tunnel ska visas på displayen och två av knapparna ska användas för att styra skeppet åt vänster eller höger. Den tredje knappen ska starta om spelet.

2 Övergripande arkitekturval

Valet av arkitektur för lösningen stod mellan enchipsdatorn AVR ATMega16 eller processorn Motorola MC68008. Jag valde den senare eftersom jag då skulle bli tvungen att bygga fler saker från grunden, och därmed lära mig mer om hur ett sådant system är uppbyggt. Eftersom jag valde MC68008, skulle genomföra projektet ensam och var mer intresserad av hårdvarudesignen än att programmera avancerade applikationer till systemet så kom jag och min handledare överens om att kraven på mjukvaran skulle vara förhållandevis låga.

3 Komponenter

3.1 Processor

Jag valde Motorola MC68008 som processor eftersom, som tidigare nämnt, det skulle kräva mer grundläggande arbete med att designa och bygga hårdvaran, vilket var min främsta anledning att gå kursen. 68008 är en asynkron 16-bitars processor med 8-bitars databuss och 20 bitars adressbuss. Den kan således adressera upp till 1 MB minne.

3.2 Klocka

Som klocka användes oscillatorn EXO-30G som i grundfrekvens jobbar i 16 MHz. Eftersom 68008 som mest kan jobba i 10 MHz kopplades klockan så att grundfrekvensen halverades och hela systemet jobbade i 8 MHz.

3.3 ROM

Som minne för lagring av programmet valdes komponenten 27C64Q, ett EPROM på 8 KB som mer än väl kom att rymma projektets mjukvarudel.

3.4 RAM

Arbetsminnet som valdes var 6264, en SRAM-krets med 8 KB minne. Även detta utrymme kom att räcka gott åt sitt syfte, bland annat exekveringsstack.

3.5 Programmerbar logik

En PAL22V10-krets användes som programmerbar logik till bl.a. chip select-signaler och liknande. Med 11 insignaler och 9 utsignaler räckte det nätt och jämnt med en enda sådan till mitt projekt.

3.6 Display och kontrollkort

Displayen var en Batron 128x64 (eller kompatibel) med två identiska kontrollkort av typen KS0108B. Storleken på den monokroma displayen kan utläsas ur namnet, och vardera kontrollkort hanterade således en area på 64x64 pixlar.

3.7 Synkroniseringskomponent

En 74HC73 användes för att synkronisera processor och display.

3.8 Knappar med tillhörande latch

För att ta emot indata användes tre vanliga tryckknappar av okänt fabrikat tillsammans med en latch av modell 74HC365.

4 Teori

Att använda en Motorola MC68008 istället för en enchipsdator innebar som tidigare nämnts en del merarbete, men jag lärde mig mycket av att göra detta arbete själv. Framför allt innebar det att processorn var tvungen att kopplas till externa minnen med hjälp av en 20-bitars adressbuss och en 8-bitars databuss. Konstruktionen krävde ett minne vars tillstånd bevarades även när strömtillförsel bröts men som endast behövde vara läsbart (i mitt fall ett EPROM) och ett minne där tillstånd inte behövde bevaras vid strömförlust men som behövde vara både läs- och skrivbart (i mitt fall en SRAM-krets). På EPROM:et lagrades programkoden och SRAM-kretsen användes som arbetsminne vid exekvering. Via samma adress- och databussar och diverse andra specialkanaler på processorn anslöts de andra periferienheterna PAL (den programmerbara logiken), knapplatch, display och synkroniseringskomponent för nämnda display. Exakt hur anslutningarna dragits kan utläsas ur systemets kopplingsschema i appendix.

Eftersom alla dessa komponenter kopplades till samma bussar och kanaler var det viktigt att de alla var tri-state-kapabla, dvs hade möjlighet att för varje kanal anta förutom VCC och GND ett högimpedivt läge i vilket de inte påverkade kanalens tillstånd över huvud taget. PAL-kretsen användes bl.a. för att generera chip select-signaler till de olika komponenterna, och PAL:ens insignaler för att avgöra vilken komponent som skulle aktiveras skickades bl.a. över de högre värda adressbitarna, vilket kan utläsas ur PAL-koden i appendix.

Två av knapparna kopplades i ett resistor- och kondensatornät till latchen, vilken i sin tur kopplades till databussen. Vid aktivering (chip select) gick latchen

från att ha haft utkanalerna i högimpedivt läge till att släppa igenom knapparnas värde för avläsning från databussen. Dessa knappar skulle användas för att styra rymdskeppet åt vänster eller höger. Den tredje knappen kopplades till reset-funktionen på processor och display för enkel omstart av systemet.

Displayen kopplades in på databussen och diverse andra kanaler enligt kopplingsschema, och synkroniseringskomponenten kopplades in till både display och processor. Systemets klocka kopplades också till både processor och synkroniseringskomponent.

Systemet kopplades upp på ett kopplingsbräde och VCC och GND kopplades till för detta syfte i förhand utdragna ledningar som var strategiskt placerade för att enkelt kunna försörja utplacerade komponenter.

5 Metod och genomförande

Det allra första steget på vägen till ett fungerande slutresultat var att designa hårdvaran, dvs rita ett komplett kopplingsschema för hela konstruktionen. Det krävde en uppfriskning av elektronikkunskaperna, mycket läsande i de i konstruktionen ingående komponenternas datablad och att en hel del konceptuella poletter skulle trilla ner. Schemat designades i Powerlogic, och efter ett flertal versioner kom jag fram till något som jag hade fullt förtroende för att det skulle fungera. När schemat godkänts av handledaren fick jag tillgång till de komponenter och verktyg som skulle behövas och kunde börja bygga systemet. Komponenternas VCC och GND lödades fast, men alla andra kanaler skulle dras med sladdar som virades på komponentbenen. Jag hade aldrig virat innan, men efter några minuter med virpistolen i hand var det riktigt smidigt och enkelt. När alla på kopplingsschemat utmärkta kanaler var virade var det så dags att testa hårdvaran. Systemet kopplades till en emulator (it-68) som i sin tur kopplades till en PC. Via terminalanslutning över serieport kunde man ge kommandon till emulatorn och därigenom inspektera funktionaliteten hos minnena och de andra komponenterna. Jag hade glömt att koppla den andra chip select-kanalen till RAM-minnet och gjort några småfel i PAL-koden, men i övrigt hade jag konstruerat systemet korrekt. Det var mycket glädjande att upptäcka att jag inte skulle behöva göra några stora ändringar i konstruktionen. Ett annat problem dök upp när jag inte lyckades få displayen att fungera. Efter många försök och assistans från handledare konstaterades till slut att något troligtvis inte stod rätt till med exemplaret jag fått, för när jag fick låna en annan grupps display fungerade systemet felfritt. När allt fungerade till belåtenhet var det så dags att börja utveckla mjukvara till systemet. Denna skrevs i C (med ett fåtal tillhörande stycken assembler för bl.a. bootstrap), kompilerades, formatterades till Motorola S2-format och laddades via terminalgränssnittet över till emulatorn. Sedan var det bara att exekvera programmet och leta upp eventuella buggar i koden tills slutresultatet var uppnått.

6 Resultat och slutsatser

Slutresultatet måste anses lyckat, eftersom man kan styra sitt rymdskepp och väggarna rör sig och krymper. Jag försökte inledningsvis att göra det mer grafiskt detaljerat, men det visade sig att systemets prestanda då inte räckte för att leverera så många bilder per sekund som vore önskvärt för en trevlig spelupplevelse. Slutresultatet blev kanske simplare och inte lika spektakulärt som jag föreställde mig i designstadiet, men för mig är det vägen dit som räknas. Jag har lärt mig mycket om grundstenarna i ett enkelt datorsystem och fyllt i förståelseluckor som irriterat mig innan jag gick kursen. Jag har också lärt mig mer om olika sätt att felsöka både hårdvara och mjukvara som beter sig felaktigt. På det hela taget är jag mycket nöjd med kursen och rekommenderar den varmt åt andra som överväger att gå den!

A Programmerbar logik

```
Title Tunnel68K
Pattern Pal
Revision 0.0
Author Bruce
Date 20-11-07
device 22v10

A13     1
A14   2
A15   3
A16   4
A17   5
RW    6
DS    7
AS    8
VMA   9
CSROM 14
CSRAM         15
CSKEY         16
CSDISP1       17
CSDISP2       18
RD   19
WR   20
DTACK         21
CSDISP        22

start

CSROM /= /A17*/A16*/A15*/A14*/A13*/AS;
```

```
CSRAM /= /A17*/A16*/A15*/A14*A13*/AS;
CSKEY /= /A17*/A16*/A15*A14*/A13*/AS;
CSDISP1 /= CSDISP*/A13*/VMA;
CSDISP2 /= CSDISP*A13*/VMA;
RD /= /DS*RW;
WR /= /DS*/RW;
DTACK /= /CSROM+/CSRAM+/CSKEY;
CSDISP = /A17*/A16*A15*/A14*/AS;

end
```

B Kopplingsschema