Designskiss

Grupp: 50

Redaktör: Martin Söderén

Version 0.1

Status

Granskad	-	-
Godkänd	-	-

PROJEKTIDENTITET

VT, 2016, Grupp 50
Linköpings Tekniska Högskola, IDA

Gruppdeltagare

Namn	**Ansvar**	**Telefon**	**E-post**
Martin Söderén	Senior Hardware design engineer	070 816 32 41	marso329@student.liu.se
Oskar Joelsson	Junior Hardware design engineer	076 185 17 17	oskjo581@student.liu.se
Jesper Jakobsson	Hardware design intern	070 673 25 10	jesja947@student.liu.se

Hemsida: https://gitlab.ida.liu.se/oskjo581/tsea83

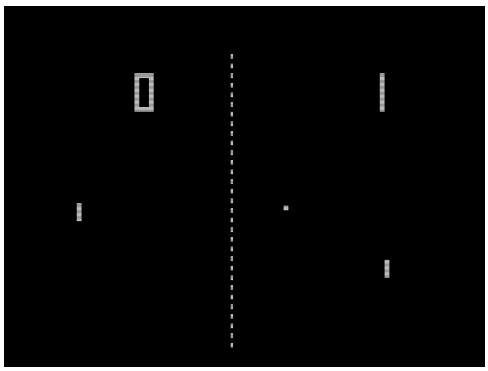
Kund: LIU
Kontaktperson hos kund: -
Kursansvarig: Anders Nilsson
Handledare: Carl Ingemarsson

Dokumenthistorik

Version	**Datum**	**Utförda förändringar**	**Utförda av**	**Granskad**
0.1	2016-02-16	Första utkast	Grupp 50	Carl Ingemarsson
0.2	2016-04-04	Andra utkast	Grupp 50	

1 Inledning

Pong är ett av de första arkadspelen. Det påminner om tennis då två spelare kontrollerar var sin stapel och med hjälp av dem skjuter de en boll fram och tillbaka. Målet med spelet är att man ska skjuta bollen bakom motståndarens stapel, vilket leder till att spelaren får poäng.



Figur 1 – Pong.

2 Analys

Det finns inga speciella problem som uppkommer i detta spel. Det är mycket jämförelser hela tiden för att hålla reda på vart bollen är. Den svåraste är troligtvis att beräkna den resulterande hastigheten(riktningen) efter en träff och det är inte speciellt svårt.

Det borde inte finnas några begränsningar i hårdvaran som kan sätta käppar i hjulet. Vi använder inte speciellt mycket minne så båda PM och GM rymms definitivt i varsit 2Kb block-ram. FPGA:n hinner definitivt med att generera bilden vi vill. CPU:n kommer vara en generell dator av Olle-Roos modell.

2.1 Spelimplementation

Algorithm 1 PONG

```
procedure PONG
    while True do
        if winner then
            LP=0;
            RP=0;
        end if
        Clear screen;
        Read both pots;
        Draw both players accoring to pot value;
        Draw ball at right player ;
        BSx=- speed pot;
        BSy=0;
        Draw points accoring to LP and RP;
        reset=false;
        winner=false;
        while not reset or not winner do
            P1=Player 1 pot;
            P2=Player 2 pot;
            SP= speed pot;
            reset=Reset button;
            Calculate balls new position;
            Draw the old ball black;
            Draw the new ball white;
            if ball=Right player then
                BSx=-BSx
                BSy=correct value
            end if
            if ball=left player then
                BSx=-BSx
                BSy=correct value
            end if
            if ball<Right player then
                LP=LP+1
                reset=true;
            end if
            if ball>left player then
                RP=RP+1
                reset=true;
            end if
            if RP=5 or LP=5 then
                winner=true;
            end if
        end while
    end while
end procedure
```

2.2 Opkoder

Processorn kommer stödja direkt addressering, omedelbar operand, indirekt adressering och indexerad addresering. Följande Opkoder ska finnas tillgängliga för programmeraren:

Instruktion	Betydelse	adresseringsmetoder	Påverkar flaggor
LOAD GRx,M,ADR	GRx:=PM(A)	00,01,10,11	-
STORE GRx,M,ADR	PM(A):=GRx	00,10,11	-
ADD GRx,M,ADR	GRx:=GRx + PM(A)	00,01,10,11	Z,N,O,C
SUB GRx,M,ADR	GRx:=GRx - PM(A)	00,01,10,11	Z.N,O,C
AND GRx,M,ADR	GRx:=GRx and PM(A)	00,01,10,11	Z.N
LSR GRx,M,Y	GRx skiftas logiskt Y steg	00	Z,N,C
BRA ADR	PC:=PC+1+ADR	00	-
BNE ADR	PC:=PC+1+ADR	00	(Z=0)
HALT	STOPP	00	-
STOREV GRx,M,ADR	GM(A):=GRx	00,10,11	-
CMP GRx,M,ADR	GRx - PM(A)	00,01,10,11	Z.N,O,C
BGE ADR	PC:=PC+1+ADR	00	(N)
BEQ ADR	PC:=PC+1+ADR	00	(Z)
IN GRx	GRx:=IN	00	-
OUT GRx	OUT:=GRx	00	-

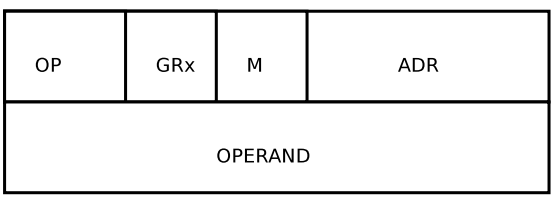
Tabell 1 – Opkoder

Följande opkoder använder formatet enligt figur 3:

- LOAD
- STORE
- ADD
- SUB
- AND
- STOREV
- CMP

Med andra ord de opkoder som arbetar mot PM eller GM. De övriga opkoderna använder formatet enligt figur 2

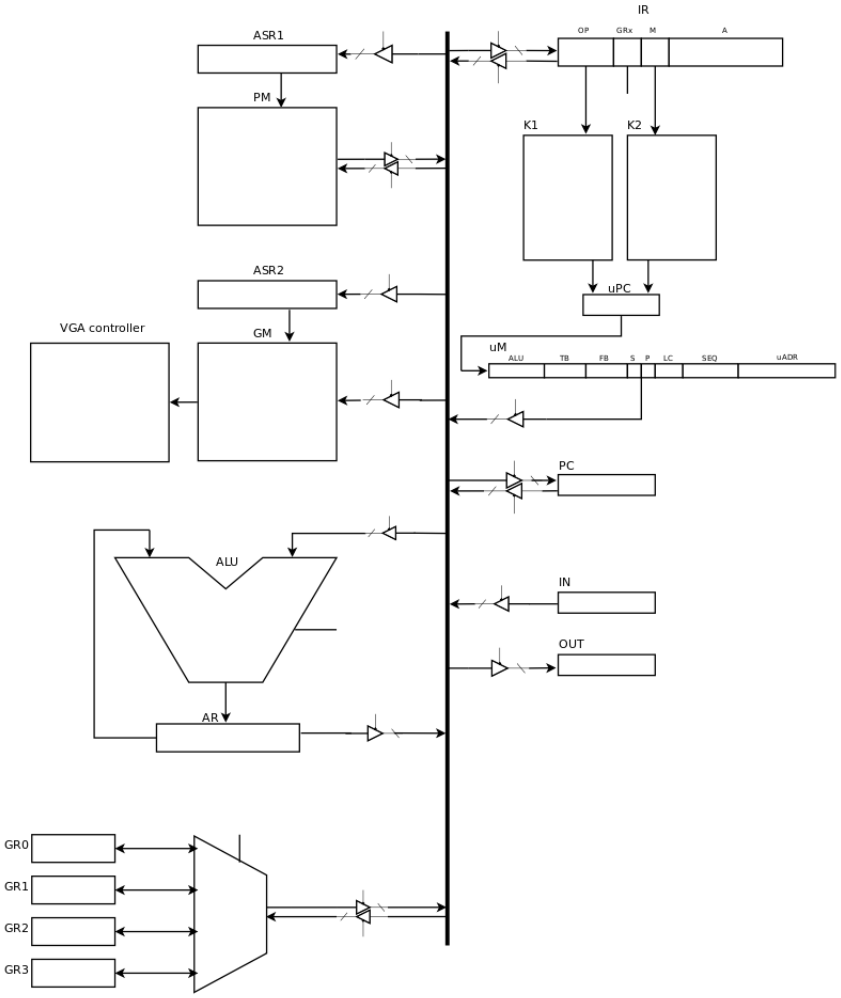
Figur 2 – Opkod.



Figur 3 – Opkod.

3 Design

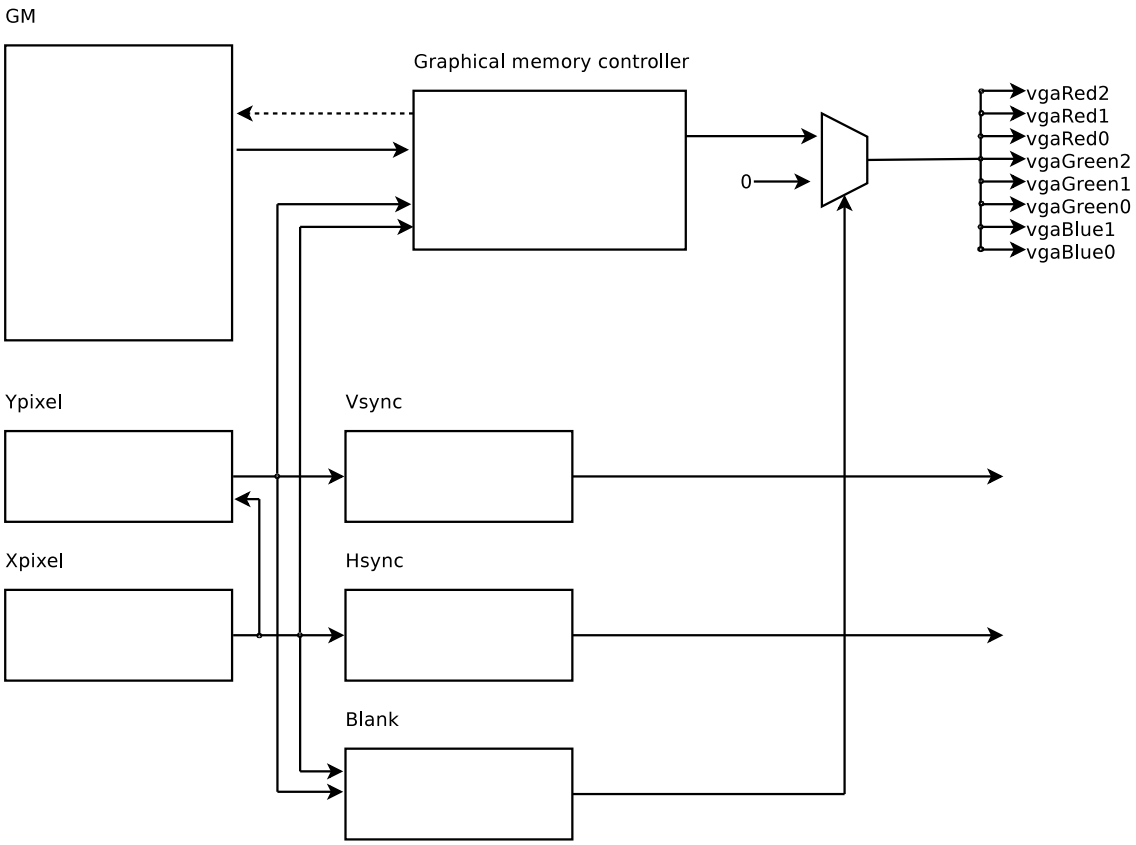
3.1 CPU



Figur 4 – Översiktlig design

3.2 Grafik

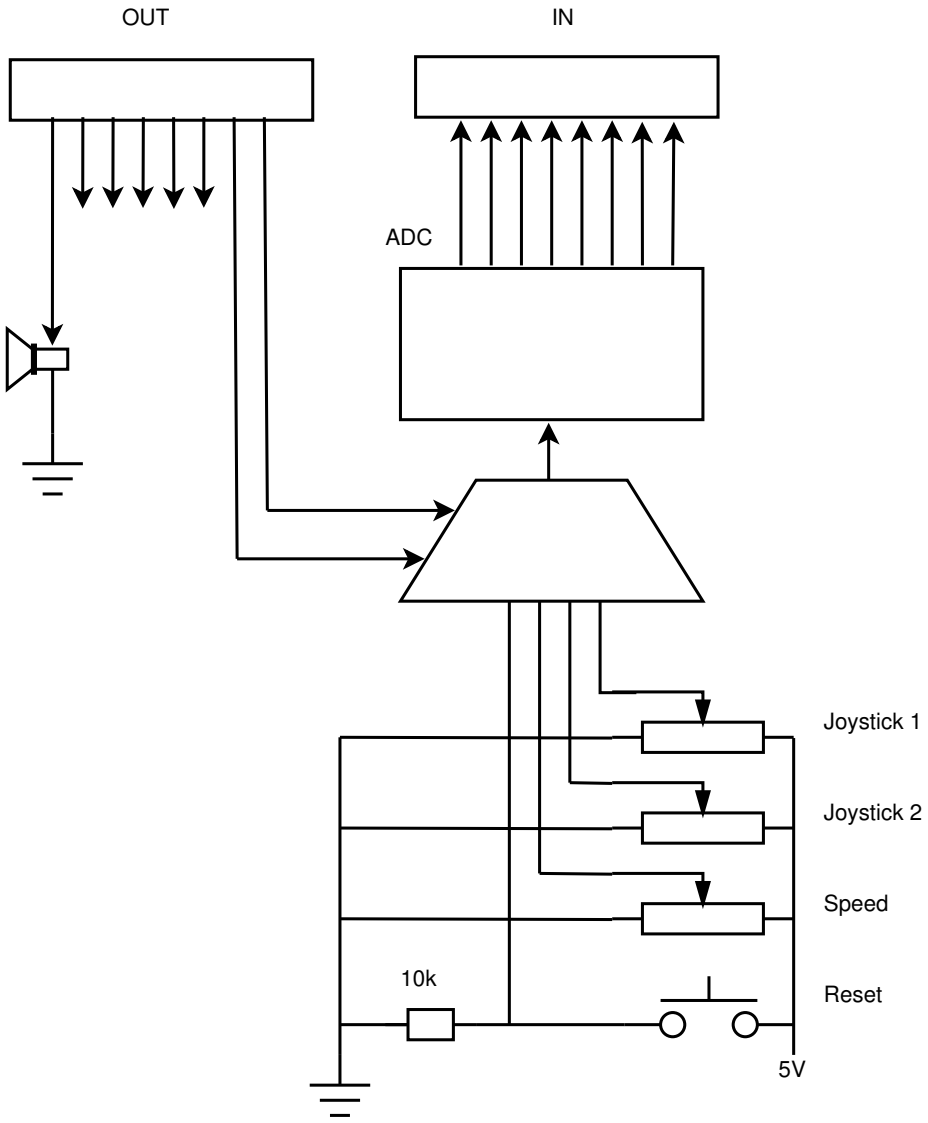
Grafikminnet kommer ett eget minnet med 256 ord à 16 bitar. För att utnyttja grafikminnet maximalt kommer den faktiska upplösningen vara 72x54. Detta leder till att vi behöver 3888 bitar för en bild och vi har 4096 tillgängliga i minnet. För att addressera grafikminnet så har vi en speciell instruktions STOREV som fungerar precis som STORE fast den skriver till grafikminnet.



Figur 5 – Översiktlig design

3.3 I/O enheter

Processorn kommer ha en 8 bitars ingång och en 8 bitars utgång. Instruktionen IN GRx gör att man läser in de 8 bitarna till de 8 minst signifikanta bitarna på register GRx. OUT GRx skriver till utgången. Dessa två portar gör det möjligt att ha en extern 8 bitars ADC och en analog mux för att läsa av flera potentiometers som kommer användas som styrning av staplarna i PONG samt styra hastigheten av bollen. Detta möjliggör även för enklare ljud samt en resetknapp.



Figur 6 – IO design

3.4 Minnen

Datorn kommer ha två minnet, dels ett delat programinne och dataminne och sedan ett grafikminne. Primärminnet kommer vara 16 bitar brett och ha 256 ord. Grafikminnet kommer vara likadant.

3.5 Programmering

Vi kommer göra en assembler(troligtvis i Python) som från en textfil direkt genererar VHDL kod som vi direkt kan kopiera in i koden för PM.

4 Milstolpe

Halvvägs genom projektet ska datorn kunna köra enklare program. Till exempel bubblesort från lab 1 borde kunna köras då vi använder samma instruktioner.