Fredrik Kloster, fredrizk
Obligatorisk oppgave 2

Rapport: Mini CPU

Introduksjon

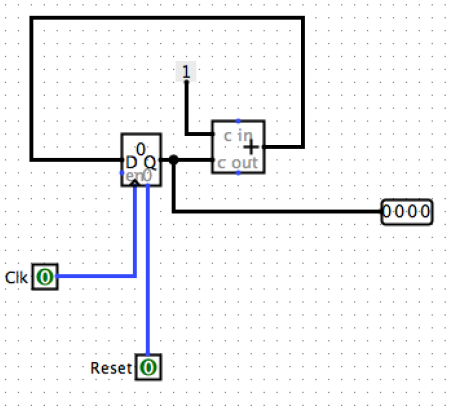
Oppgaven er å implementere en 8-bit mini CPU designet med Harvard Arkitektur. Denne arkitekturen bruker et separat programmene og dataminne. Disse to RAM komponentene og et 8-bit dataregister er de viktigste komponentene vi skal bruke, som skal kobles sammen ved hjelp av en instruksjonsbuss. I tillegg skal en selvdesignet ”Counter” bli laget ved hjelp av en 4 bits adder. Denne skal stoppe å telle på 16. Counter’en skal blir brukt til å iterere gjennom instruksjonene i programminnet.
I min løsning har jeg valgt å lage enda en krets som jeg har kalt for ”Instruction Decoder” som tolker instruksjonene.

Implementasjon

Counter

I uke 9 var en av oppgavene å lage en Counter lik den vi skal lage i denne oppgaven. Jeg brukte derfor min besvarelse fra denne oppgaven.

I min krets er den bygget opp av 2 delkretser. Den første heter ”Counter”.

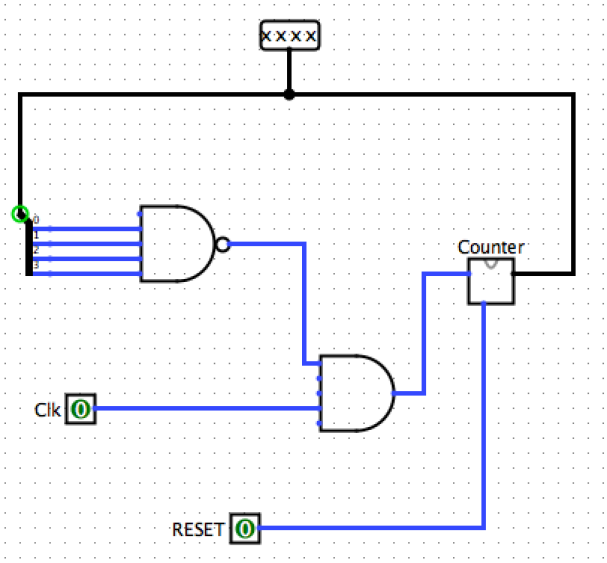


Figur 1 - Counter

Den er laget av et 4-bits register, og en 4-bits adder. Begge disse komponentene er tatt fra LogiSim. For hvert klokkeslag skal registret øke verdien sin med 1. Utgangen fra registret er koblet inn i adder-en, hvor nåværende verdi blir plusset med konstanten 1. Utgangssignalet blir også koblet til en ”output” til senere bruk. Reset er for å sette registeret til 0.

Denne kretsen blir tatt i bruk i instruksjonspekeren, som også kontrollerer at counter-en ikke skal telle over 16.

Fredrik Kloster, fredrizk
Obligatorisk oppgave 2

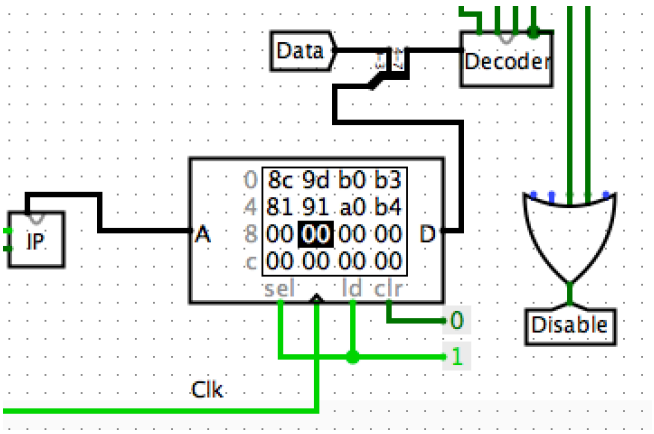


Figur 2 - Instruksjonspeker

Output-en fra Counter-en er også outputen til denne kretsen, men går også gjennom en nand-port. Dersom alle bitene er 1, går blir signalet satt til 0, og klokkesignalet vil alltid være 0. Dette resulterer i at counteren stopper å telle.

Programminne

Programminnet er en 4 bits RAM-brikke. Instruksjonspekeren er koblet inn i adresseporten. En instruksjon er del inn i to deler, databiten og instruksjonsbiten. Derfor blir utgangssignalet splittet inn i 2, hvor LSB går til dekoderen, mens MSB går til data.



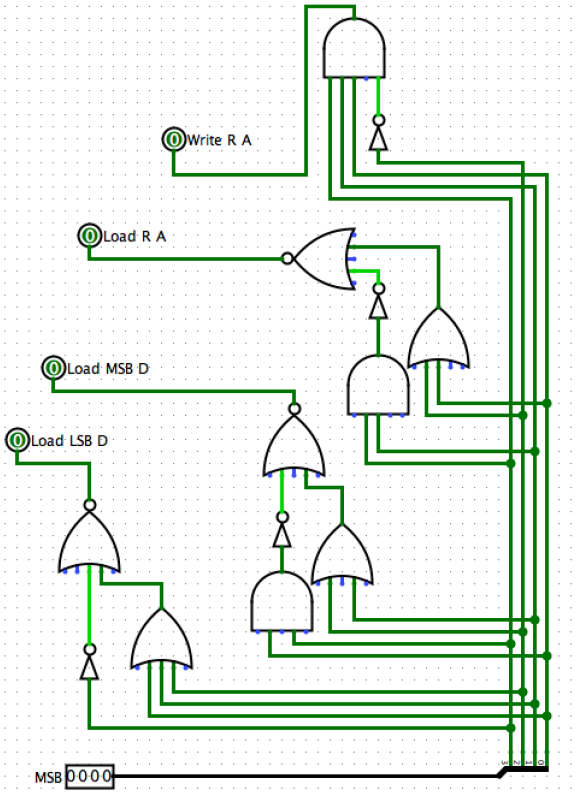
Figur 3 - Programminnet

Fredrik Kloster, fredrizk
Obligatorisk oppgave 2

Instruksjonsdekoder

For å tolke instruksjonen valgte jeg å designe en enkel dekoder. Det skal være totalt 4 ulike operasjoner, og har derfor et signal per operasjon. Det er kun et av signalene som kan være høy av gangen. De forskjellige operasjonene er:

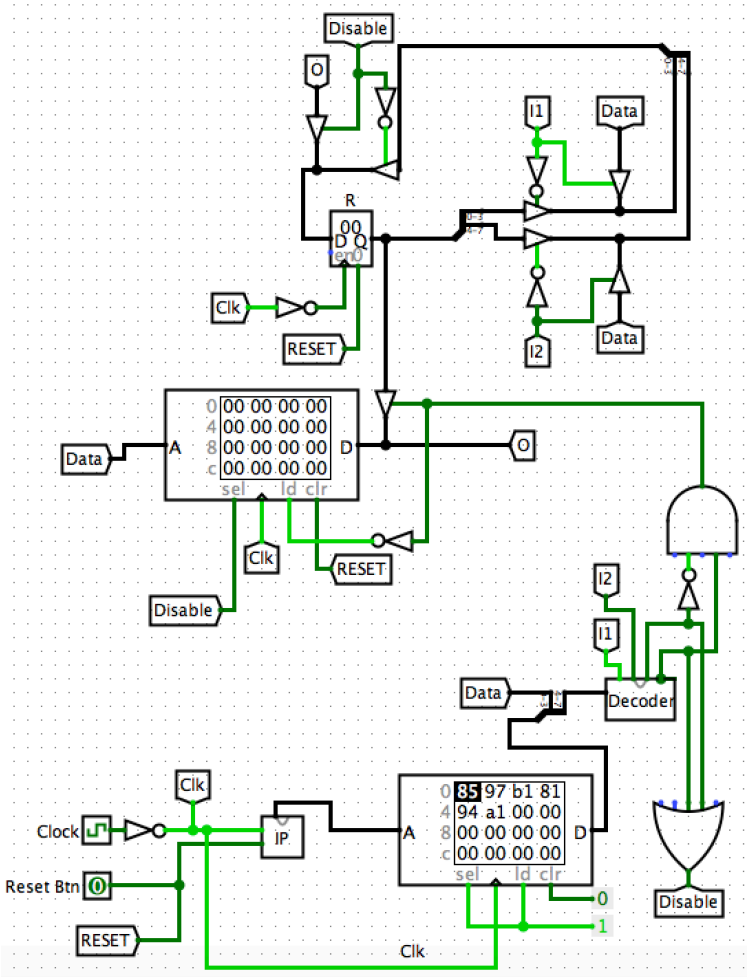
- 1000 (8) er å sette de 4 LSB til R lik data.
- 1001 (9) er å sette de 4 MSB til R til data.
- 1010 (A) sette R lik verdien til en adresse.
- 1011 (B) skrive verdien til R til en adresse.



Figur 4 - Instruksjonsdekoder

Fredrik Kloster, fredrizk
Obligatorisk oppgave 2

Dataminnet og registeret



Figur 5 - Mini CPU

På bildet over det det bilde av hele designet . Dataminnet har kun en utgang, som også fungerer som inngang. Dersom ”ld” er 1, er D en output, og en input om D er 0. Sel bestemmer om det skal være mulig å interagere med komponenten.

Siden to av operasjonene ikke bruker dataminnet, blir sel satt til 1 når dataminnet skal brukes. På dekoderen representerer de to utgangene til venstre operasjonene som kun angår R. Derfor forblir ”Disable” signalet 0. Ved inputen til R velges da outputen fra operasjon 1 / 2.

De to utgangene til høyre hos dekoderen representerer operasjonene til dataminnet. Dersom en av disse signalene er 1, blir ”Disable” satt til 1. Dataminnet vil da bli slått på, og inputen til registeret vil da få signaler fra dataminnet. Utgangssignalet helt til høyre til dekoder bestemmer om dataminnet skal skrive en verdi til R. Når den er på, vil dataminnet sende ut et signal som går til porten O. Siden ”Disable” er på, vil signalet fra O overskrive R. Når data skal bli skrevet på minnet, bestemmer ”Data” hvilken adresse som skal bli overskrevet, og henter verdien fra R og legger den verdien på riktig plass.

Fredrik Kloster, fredrizk
Obligatorisk oppgave 2

Når R sine LSB eller MSB skal endres (instruksjon 1 eller 2), blir R sin verdi sendt til stasjonen plasser øverst til høyre på bildet. Bufferne bestemmer om om det er de 4 første eller 4 siste bitene som skal endres på. Dersom l1 er 1 blir LSB endret-, og om l2 er 1 blir MSB endret til Data-verdien.

Sidenotater

Kretsen fungerer som den skal, bortsett fra om kretsen blir stoppet mens klokkesignalet er på 1, resetter pekeren, og kjører prosessen om igjen. Da vil CPU-en hoppe over den første instruksjonen. For å få en sikker kjøring, reset simulasjonen og last inn programmet i minnet på nytt, eller se at klokken begynner på lav.

”Reset btn” vil resette både dataminnet og registeret, og sette Counteren til 0.