INF3430 - Laboppgave 2

Magnus Andersen

1 Subprogrammer

Når det gjelder a); det ser ut som den første metoden har størst LUT-forbruk. Usikker på hvordan jeg ser hvilken av de som har størst forsinkelse.

2 seg7decoder

Implementerte sannhetstabellen som en boolsk funksjon ved å lese ut 1-ere og forenkle ved hjelp av Karnaugh-diagram (usikker på om dette ble for "overkill", men ville friske opp dette).

3 seg7ctrl

Jeg lagde en modul *clkdiv*[1] som tar som input mclk (i vårt tilfelle 50MHz) og gir som output et tilsvarende signal men med kraftig redusert frekvens, dvs. 400Hz. Tanken her var å bruke denne reduserte klokken i sensitivitetslista til en process i *seg7ctrl_rtl*, hvorpå nevnte process for hver "iterasjon" aktiverer et nytt display (implementert ved hjelp av en teller som teller fra og med 1 til og med 4). Slik vil hvert display få en oppdateringsfrekvens på 100Hz. Denne frekvensen er forholdsvis arbitrær, men det virket som om konsensusen etter litt googling er at lysstimuli forblir på netthinnen i ca. $\frac{1}{30}s$ etter at lyskilden er opphørt, og jeg endte da på 100Hz for å være på den sikre siden. I praksis ga i alle fall ikke denne frekvensen verken flimring eller overflyt.

Videre sendes nåværende display-tall og desimaltegn inn i en instansiert *seg7decoder*-komponent, som da nettopp dekoder tallet+desimaltegnet slik at det kan vises i form av de sju ledlampene på nåværende display.

4 regs

Satte opp modulen *regs* med knappene som brukes som input. Tallet representert av *SW(3 downto 0)* blir matet inn i registeret representert ved *SW(7 downto 6)* gitt at *load* er høy. For å få til dette lagde jeg en modul *regctrl* som jeg koblet til den tidligere *seg7ctrl*-modulen i en toppnivåmodul *regs*. Sistnevnte er rent strukturell (kort sagt: slik som oppgaven var spesifisert). Koden er temmelig selvforklarende.

5 clock

La sjekkingen av start/stop/reset-knappetrykk i *mclk*-processen for å få kjappere reaksjon; om disse sjekkene hadde ligget i *rclk*-processen måtte man holde inne knappene i worst-case ett sekund, som ble masete. Måten jeg telte opp til ett sekund på var å lage en process som var sensitiv til *mclk*, og invertere et annet klokkesignal når *mclk* hadde gått i et halvt sekund ($25 \cdot 10^6$perioder = 0.5s).

I en annen process som var sensitiv til dette reduserte klokkesignalet lagde jeg tellere for hvert display, slik at hvis teller for disp0 når 9 øker teller for disp1 med 1, når teller for disp1 når 9 øker teller for disp2 med 1, osv. Dette ble kanskje litt en smule rotete, men det fungerte. Signaler som ble oppdatert til tellerverdiene for hvert display ble så port mappet inn i seg7ctrl.

[1] *http://en.wikipedia.org/wiki/Frequency_divider*