Lab3 INF3430/INF4431

malina@student.matnat.uio.no
wonhol@student.matnat.uio.no

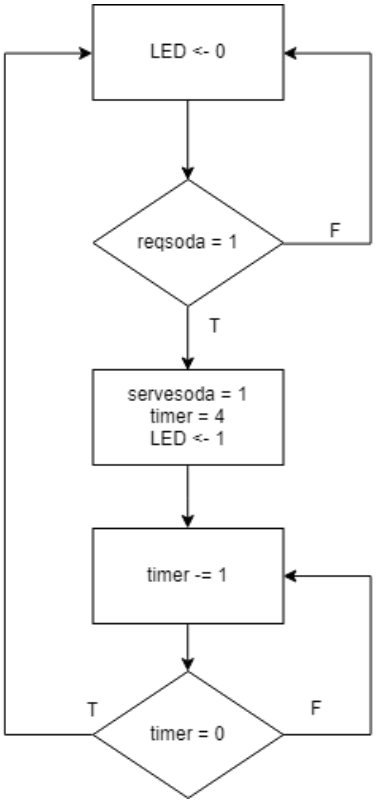
Exercise 1:

Hensikt med oppgaven er og bli kjent med Moore og Mealy FSM, kunne skille mellom disse og skissere opp ASM flytskjema ut i fra tekstinstruksjon og gitte entiteter.

a) Moore Machine:

Output er avhengig kun av nåværende tilstand. Hvis input endres, har det ikke innvirkning på output. Trenger flere tilstander for å løse et gitt problem, krever større plass. Reagerer på endringer en klokkeperiode senere. Synkron output og state-generering. Etter sigende mindre krevende, generelt, å designe.

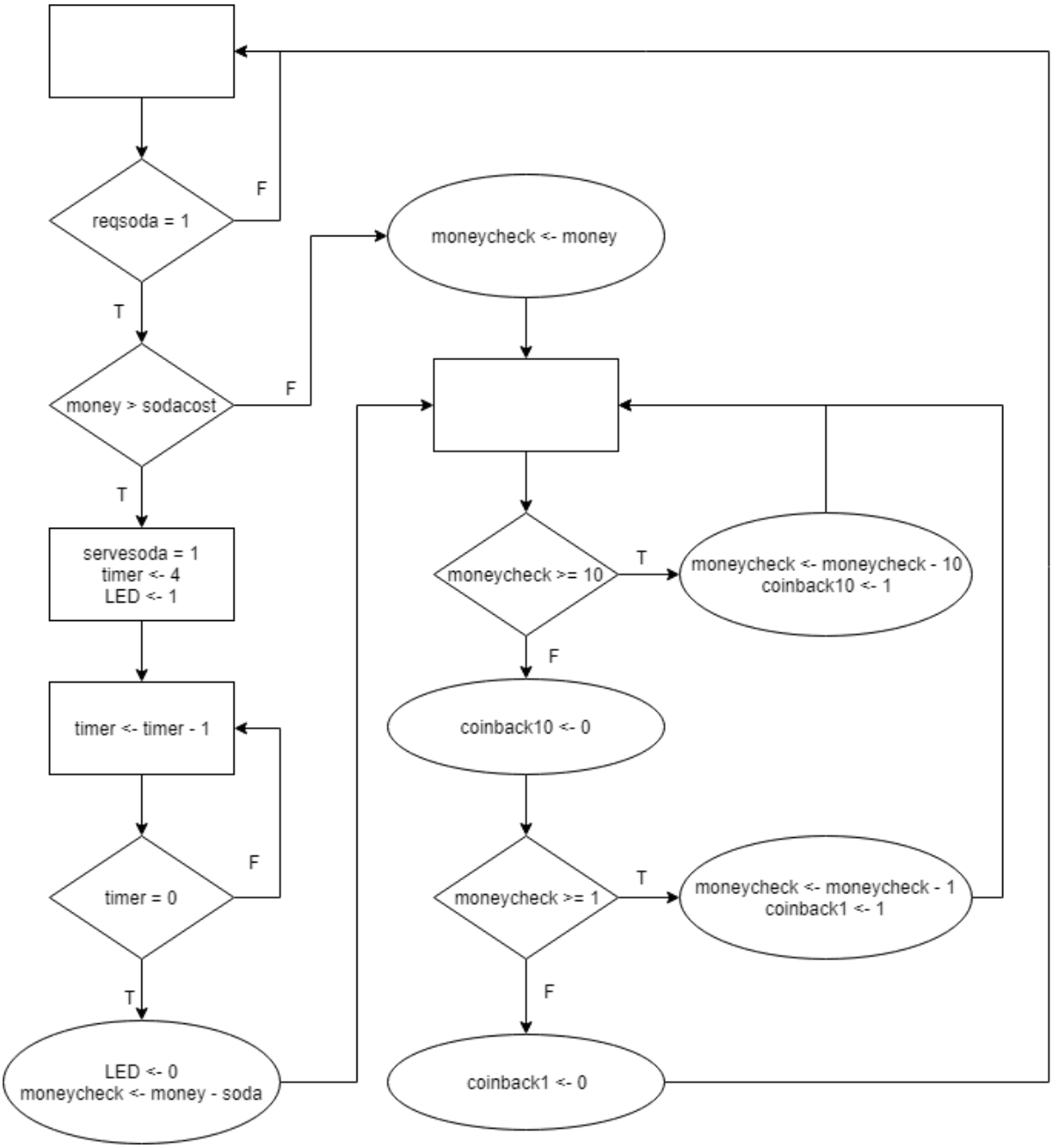
Utfordringen her lå i å initiere timeren på riktig sted, for å få logikken til å gå opp. Da metoden har mindre fleksibilitet, siden den kun er avhengig av nåværende tilstand, var det ikke med en gang opplagt hvordan vi skulle bruke timeren.



b) Mealy Machine:

Output avhenger av både nåværende tilstand og input til denne. Mao. endrer output seg, hvis input endres. På denne måten krever metoden færre tilstander og mindre plass. Reagerer på endringer i inneværende klokkeperiode. Asynkron output. Mer krevende å designe.

Utfordringen i denne oppgaven var å få coinback(1/10) inn på riktig sted. Løsningen er løst basert på ASM til p_ctrl gitt i exercise 3, med med en egen coinback-del. Timeren var lettere å implementere i denne oppgaven, enn i oppgave a).



Konklusjon:

Å kunne skrive gode ASM er nyttig for å få oversikt over en problemstilling og i det videre arbeidet med denne. Å ha god intuisjon for hvilke problemer som bør løses med henholdsvis Moore eller Mealy trenes opp med erfaring.

Exercise 2:

Hensikten med oppgaven er å kunne konstruere koden til en posisjonsmåler ut i fra et ASM av typen Moore.

Bruker motor_ent og motor_beh fra lab2 og instantierer + henter de inn i tb sammen med pos_meas. Stimulerer motor_cw og motor_ccw. Pos sendes ut.

Viktig å dobbel-floppe interne signaler for a,b,sync_rst, da disse er asynkrone. Bruk av teller for å endre posisjonen.
ASM diagrammet var greit og tydelig å lese.

På oppfordring fra labassistenter, er oppgaven ikke kjørt i Vivado, da filene uansett må testes i exercise 5 og kjøres ut på brett.

Exercise 3:

Hensikten med oppgaven er å konstruere koden til en regulator ut i fra et ASM av typen Mealy. Regulatoren er en proporsjonalregulator (P- reg) med tilbakekobling. Kombinatorisk og synkroniserende prosess sørger sammen for å implementere en avviksberegning som tilbakekobles.

ASM greit og tydelig å lese også her.

Syr det hele sammen med pos_meas og motor filene. Sp stimuleres i tb og motor_cw og motor_ccw sendes ut.

Konklusjon Excercise 2 og 3:

Å skrive kode ut i fra et ASM letter arbeidet med programmeringen betraktelig, derfor er det definitivt lurt å skaffe seg oversikt over problemer vha. dette. Nyttig å lære og dobbel floppe, for å synkronisere signaler. Ser at vi har færre tilstander å

forholde oss til i en Mealy FSM, slik definisjonsforskjellene på Moore og Mealy understreker.

Exercise 4:

Hensikten med oppgaven er å tvinge motoren med/mot klokka vha. force_cw/ccw ut i fra oppgitt sannhetstabell og blokkdiagram.

Legger til rst_div og mclk_div i p_ctrl og synkroniserer disse.

Ser ut i fra sannhetstabellen gir ut samme verdier når force_cw/ccw er henholdsvis '1' og '1', '0' og '0'. Multiplexeren kan derfor implementeres i en prosess vha. en if,else if,else løkke.

Setter klokkeperioder i tb, en for mclk og en for mclk_div. Husker på å initiere alle resetter i stimulidel. Motor og pos_ctrl instantieres som overordnede komponeneter og mappes. Force_cw/ccw stimuleres og går til output.

Opplever problemer med å få riktig respons fra motor i tb, men løser det med å sette generic til 50 ns.

Konklusjon:

Lettere å få oversikt på program med sannhetstabell og blokkdiagram enn bare med tekstuell instruksjon. Veldig nyttig øvelse å sette sammen biter til stadig større komponenter, da det gir god trening på å forstå interne/eksterne signaler. Innføring av egen klokke/reset i p_ctrl gir trening i å synkronisere.

Exercise 5:

Hensikt er å få dette slottet av komponenter over på brett for å kjøre motoren, samtidig som setpunkt og posisjon oppdateres og vises på display. Force_cw/ccw,sync_rst settes opp på pinne for manuell styring.

Constraints fil inneholder timing constraints for eksternt og internt klokkesignal.

Komponent CRU tar inn en global,asynkron reset og referanseklokke og viderefører interne signaler: synkronisert rst og rst_div + masterklokke og mclk_div. Mclk_div realiseres vha. en teller, faseskift 128 ganger så fort som masterklokke.

Opplever problemer i vivado med å få motoren og display til å samstemme. Dette skyldes latch introdusert i kombinatorisk logikk i tilstandmaskinen. Dette problemet ble ikke fanget i simuleringsfasen, men i syntesefase. Problemet ble oppdaget av synteseverktøyet og det ble logget. Problemet ble løst ved å introdusere aktiveringssignal for utgang og standardverdi for aktiveringssignaler.

Konklusjon:

Det er viktig å unngå sperre ved innføring av kombinatorisk logikk. Da det ikke var synlig problem i simuleringen, tok det lang tid å feilsøke.