TFE4105 Digdat Løsningsforslag lab 4, 2008

Løsningsforslag forarbeide

Oppgave 3

DEC mangler. Dette signalet forteller telleren i kontrolldelen at den skal minke tallet med 1. Det er kontrolldelens måte å holde orden på hvor ”langt” den har kommet i multiplikasjonen.

Oppgave 4

DEC må legges til i flere

Boksen merket ”?” blir utført når multiplikatoren er ”midt i” utregningen (fordi da er ZERO=0). Samtidig er MTOR_LSB=1, som betyr at resultatet fra ADDER (i figur 4-3) *skal* lastes inn i DO.
Disse utgangssignalene skal være høye (1):

- LOAD_DO, fordi resultatet fra ADDER skal lastes inn i DO
- LSL, fordi MAND skal venstreskiftes før neste siffer behandles
- LSR, fordi MTOR skal høyreskiftes før neste siffer behandles

Oppgave 5

```
-- "entity" definerer tilstandsmaskinens grensesnitt i form av innganger og utganger
entity fsm is
    port (clk      : in  STD_LOGIC; -- 1 bit, system clock
          reset    : in  STD_LOGIC; -- 1 bit, system reset, active high
          mtor_lsb : in  STD_LOGIC; -- 1 bit, lsb from multiplicator register
          start_fsm: in  STD_LOGIC; -- 1 bit, signal to start the fsm, active high
          zero     : in  STD_LOGIC; -- 1 bit, zero detection from down counter
-- **** HER MANGLER ET UTGANGSSIGNAL SOM MAA LEGGES TIL ****
          dec      : out STD_LOGIC;
          lsl      : out STD_LOGIC; -- 1 bit, shift left command to multiplicand register
          lsr      : out STD_LOGIC; -- 1 bit, shift right command to multiplicator register
          twoc     : out STD_LOGIC; -- 1 bit, two's complement command to adder
          load_do  : out STD_LOGIC; -- 1 bit, load command to output register
          reset_do : out STD_LOGIC; -- 1 bit, reset command to output register
          load_dcnt: out STD_LOGIC  -- 1 bit, load command to down counter
         );
end fsm;

-- "architecture" beskriver innholdet i en entity
architecture fsm_arch of fsm is

-- "constant" gir verdier til navn, i dette tilfellet navngis de to tilstandene IDLE og OPERATE
constant IDLE   : STD_LOGIC := '0';
constant OPERATE: STD_LOGIC := '1';

-- "signal" definerer signaler og registere i designet. I dette tilfellet trenger vi et 1-bit register for tilstanden
signal state: STD_LOGIC; -- 1 bit state register

-- Etter "begin" beskrives logikken i tilstandsmakinenen
begin

-- Foelgende prosessdeklarasjon forteller at for hver endring i signalet "clk" trigges prosessen
-- Denne prosessen styrer overgangen mellom tilstandene
process (clk) is begin
    if (clk'event and clk = '1') then                                       -- dersom vi har en positiv klokkeflanke...
        if (reset = '1') then
            state <= IDLE;                                                  -- dersom reset er 1, gaa til IDLE
        elsif ((state = IDLE) and (start_fsm = '1')) then
            state <= OPERATE;                                               -- osv...
        elsif ((state = OPERATE) and (zero = '0')) then
            state <= OPERATE;
        elsif ((state = OPERATE) and (zero = '1')) then
            state <= IDLE;
        else
            state <= IDLE;
        end if;
    end if;
end process;

-- Denne prosessen setter tilstandsmaskinens kontrollsignaler avhengig av tilstanden og inngangssignalene
process (state, start_fsm, zero, mtor_lsb) is begin

-- Hver if-linje som innleder en blokk med kontrollsignaler tilsvarer forgreningsflyten i tilstandsdiagrammet
-- Hver blokk med kontrollsignaler tilsvarer en blokk i tilstandsdiagrammet
-- I tillegg kommer den siste blokken i denne prosessen, som for ryddighets skyld setter alle kontrollsignaler
-- til null dersom ingen av de andre tilfellene skulle inntreffe
-- En av blokkene med kontrollsignaler maa endres fullstendig slik det er beskrevet lenger ned
-- I tillegg maa et ukjent kontrollsignal gis verdier I ALLE blokkene
```

```
    if ((state = IDLE) and (start_fsm = '1')) then
        lsl       <= '0';
        lsr       <= '0';
        twoc      <= '0';
        load_do   <= '0';
        reset_do  <= '1';
        load_dcnt <= '1';
        dec       <= '0';
    elsif ((state = OPERATE) and (zero = '0') and (mtor_lsb = '0')) then
        lsl       <= '1';
        lsr       <= '1';
        twoc      <= '0';
        load_do   <= '0';
        reset_do  <= '0';
        load_dcnt <= '0';
        dec       <= '1';
    elsif ((state = OPERATE) and (zero = '0') and (mtor_lsb = '1')) then

-- I FØLGENDE BLOKK SKAL KONTROLLSIGNALENE ENDRES (alle er foreløpig bare satt til 0):
-- *** BLOKK FOR ENDRING - START ***

        lsl       <= '1'; -- UKJENT
        lsr       <= '1'; -- UKJENT
        twoc      <= '0'; -- UKJENT
        load_do   <= '1'; -- UKJENT
        reset_do  <= '0'; -- UKJENT
        load_dcnt <= '0'; -- UKJENT
        dec       <= '1';
-- *** BLOKK FOR ENDRING - SLUTT ***

    elsif ((state = OPERATE) and (zero = '1') and (mtor_lsb = '0')) then
        lsl       <= '0'; -- don't care
        lsr       <= '0'; -- don't care
        twoc      <= '0'; -- don't care
        load_do   <= '0';
        reset_do  <= '0';
        load_dcnt <= '0'; -- don't care
        dec       <= '0';
    elsif ((state = OPERATE) and (zero = '1') and (mtor_lsb = '1')) then
        lsl       <= '0'; -- don't care
        lsr       <= '0'; -- don't care
        twoc      <= '1';
        load_do   <= '1';
        reset_do  <= '0';
        load_dcnt <= '0'; -- don't care
        dec       <= '0';
    else
        lsl       <= '0';
        lsr       <= '0';
        twoc      <= '0';
        load_do   <= '0';
        reset_do  <= '0';
        load_dcnt <= '0';
        dec       <= '0';
    end if;

end process;
end fsm_arch;
```

Oppgave 6

	Inngangs-signaler			Utgangssignaler							Involverte registre	
CLK PERIODE	MTOR_LSB	START_FSM	ZERO	LSL	LSR	TWOC	LOAD_DO	RESET_DO	LOAD_DCNT	DEC	MTOR (HEX)	DCNT
1	0	1	0	0	0	0	0	1	1	0	B3AC	15
2	0	0	0	1	1	0	0	0	0	1	B3AC	15
3	0	0	0	1	1	0	0	0	0	1	59D6	14
4	1	0	0	1	1	0	1	0	0	1	2CEB	13
5	1	0	0	1	1	0	1	0	0	1	1675	12
6	0	0	0	1	1	0	0	0	0	1	0B3A	11
7	1	0	0	1	1	0	1	0	0	1	059D	10
8	0	0	0	1	1	0	0	0	0	1	02CE	9
9	1	0	0	1	1	0	1	0	0	1	0167	8
10	1	0	0	1	1	0	1	0	0	1	00B3	7
11	1	0	0	1	1	0	1	0	0	1	0059	6
12	0	0	0	1	1	0	0	0	0	1	002C	5
13	0	0	0	1	1	0	0	0	0	1	0016	4
14	1	0	0	1	1	0	1	0	0	1	000B	3
15	1	0	0	1	1	0	1	0	0	1	0005	2
16	0	0	0	1	1	0	0	0	0	1	0002	1
17	1	0	1	0	0	1	1	0	0	0	0001	0
18	1	0	1	0	0	0	0	0	0	0	0001	0

Oppgave 7

Her kan det lages en kort eller lang liste som i alle fall bør inneholde multiplikasjon av to positive tall, av et negativt og et positivt tall, av et positivt og et negativt tall og to negative tall.

Oppgave 8

Vi tar for oss det negative tallet 0x87 = 0b10000111 = -121
Dette kan vi se på som -128+4+2+1 = -57
Altså vi kan se på det mest signifikante bitet som negativt, med samme størrelse som om bitet var positivt.

0b10000000 betraktet som positivt er 128.
0b10000000 betraktet som negativt er -128.

0b10000001 betraktet som negativt er -128 +1 = -127¨
0b10000111 betraktet som negativt er -128 +4+2+1 = -121
Se notat om negative tall på it's:learning for detaljer.

Når multiplikatoren er negativ vil det mest signifikante bitet ha størrelsen 32768, og være negativ. I stedet for å multiplisere multiplikanden med dette negative tallet, multipliseres først multiplikanden med 32768 (høyreskift 15 ganger), og så gjøres det negativt. Alle de andre bitene kan sees på som positive. Multiplikasjonen kan derfor foregå som om tallet er positivt helt til fortegnsbitet skal vurderes.

Dersom multiplikanden er negativ vil den alltid betraktes som negativ (selv om den venstreskiftes), på grunn av sign-extension. Derfor vil multiplikasjonen bestå av en sum av negative tall.

Løsningsforslag forarbeide

Fase 1

For å få multiplikatoren ut av IDLE-tilstanden må vi ha en positiv flanke på signalet DVI (og klokken må gå og reset_n må være høy). Merk at det strent tatt ikke er nødvendig at ENMUL er høy, siden denne bare bestemmer om resultatet fra multiplikasjonen skal legges ut på result_a utgangen.

Fase 6

Hvis ABS og FILT er deaktivert vil multiplikasjonen starte straks DVI går høy og resultatet blir korrekt. Hvis ABS er aktiver (men ikke FILT), så starter også multiplikasjonen straks, men resultatet blir som om A-inngangen er positiv, selv om den er negativ. Hvis FILT er aktivert, vil det først gå en rekke klokkepulser mens denne beregner factor. Først når dette er gjort og den har satt ut sin DVO, vil multiplikatoren starter. Resultatet ut av multiplikasjonen er uviss, siden det er factor som multipliseres med B-inngangen.

Fase 7

Hvis man ikke snakker i mikrofonen så vil bakgrunnsstøyen bli sterkere og sterkere. Hvor fort dette skjer avhenger av reduksjonshastigheten. Er denne veldig rask går det momentant, er den minimal, går det veldig langsomt. Hvis man først snakker høyt og så snakker lavere og lavere så vil lyden i hodetelefonene likevel bli omtrent like sterk (noe avhengig av valgt reduksjonshastighet).

Med lydfilen vil man med ubehandlet lyd høre veldig stor variasjon i volumet. Med behandlet lyd vil denne variasjonen bli mye mindre.

Fase 8

Nedenfor vises et eksempel på behandlet lyd.