Versjon 1.1/20.10.2015

INF3430/4431 Høsten 2015

Laboppgave 4

System on Chip (SoC) konstruksjon

Innledning.

Hovedmålet med denne laboppgaven er at dere skal lære å lage et såkalt ”System on Chip” (SoC) hvor det skal legges inn en prosessor i FPGA kretsen. Dere skal også lære å benytte ekstern SRAM på testkortet. For hver deloppgave skal det følges samme designflyt som i tidligere laboppgaver.

Rapport.

En kortfattet rapport som oppsummerer hva som er gjort, med problemer/utfordringer. Tegn gjerne figurer med f.eks. blokkdiagrammer.

For hver deloppgave skal dere levere:
VHDL-kildefil(er)
VHDL-testbenk der dette er relevant
Do-fil(er) for kompilering/simulering i Modelsim.
Skjermbilde fra simuleringene der det er relevant
Place & Route report/Static Timing report der man har gjennomført Place og route.

Det komplette systemet i deloppgave 4 skal demonstreres for labveilder.

Alle innleverte VHDL-filer skal følge navnreglene for vhdl-filer og retningslinjene for indentering som beskrevet i kokeboken.

Oppgave 1.

Vi skal nå benytte ekstern SRAM på testkortet til å lagre sekvenser av setpoint, altså et helt bevegelsesmønster.
Vi benytter SW7 til å velge mellom lagringsmodi og avspillingsmodi. Når SW7 er 1 er vi i lagringsmodi og setpoint innstillt på SW6-SW0 lagres i SRAM adresse 0 ved et trykk på BTN2. Vi stiller inn nytt setpoint på SW6-SW0 og dette lagres i neste adresse i SRAM etter at BTN2 er trykket. Vi kan fortsette på denne måten inntil SW7 går til 0. Etter å ha trykket på reset (hvorfor?) kan vi starte avspilling av lagrede setpoint fra SRAM ved et trykk på BTN2. Neste trykk på BTN2 avspiller setpoint i neste adresse osv.

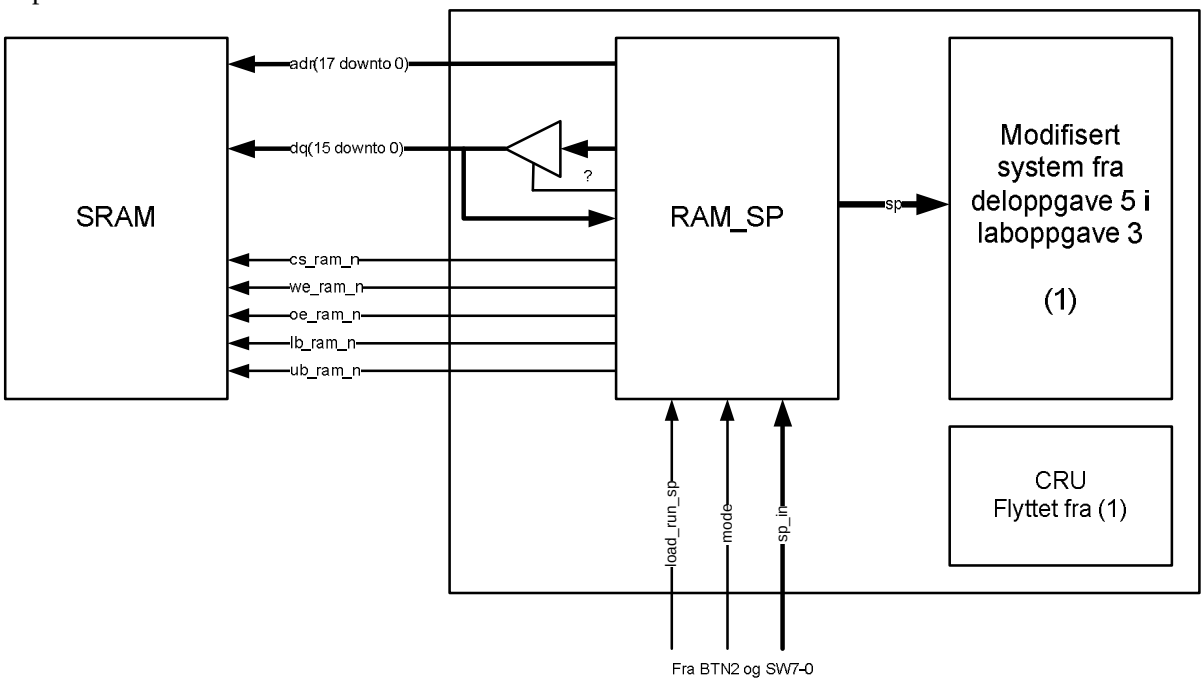

Figure 1. Struktur over system for lagring av sp til ram og avspilling av sp fra SRAM

I denne oppgaven skal dere ta utgangspunkt i den ferdige modulen ram_sp_rtl/ent.vhd. Den fungerer på den måten at man kan lagre en og en sp i etterfølgende ramadresser ved å trykke på BTN2, når SW7 =’1’. Når SW7 =’0’ kan man hente ut de lagrede sp’ene en og en ved å trykke på BTN2. Før de avspilles må tilstandsmaskinen resettes. Merk at data lagret i SRAM blir ikke slettet av en reset, bare etter power up. Sync_rst fra lab2 kobles til ’0’, mens BTN1 og BTN0 benyttes til å tvangskjøre motoren den ene eller andre veien.

For å simulere ram_sp benytter dere vedlagt simuleringsmodell for SRAM som består av filene: async256kx16.vhd, package_timing.vhd og package_utility.vhd.

Vi må regne med at det er mye prell på trykknappen, BTN2. Dette vil føre til feilfunksjon ved at det blir lagret til flere adresser enn en pr. trykk på BTN2 Det er instantiert en prellbeskyttelses-komponent, debouncer. Den tilhørende arkitekturen gir imidlertid ingen prellbeskyttelse. Den må dere lage selv.

Integrer ram_sp sammen med systemet fra deloppgave 4 i laboratorieoppgave 3. Benytt vedlagte entitet i `ram_pos_ctr_ent.vhd`. CRU modulen må flyttes opp i toppnivået.
Koble til SRAM grensesnittet på testkortet, se figuren over (benytt bare en av SRAM'ene). Tristate buffere legges inn i top nivå arkitekturen. Hvordan skal vi styre dette?

Hint 1.1: se forelesningsslider 2.9.2013 for eksempel på tristatebuffer.

Lag en prellbeskyttelse for signalet `load_run_sp`.
Lag en testbenk for dette systemet. Bruk vedlagte simuleringsmodell for SRAM.

Systemet skal implementeres i FPGA.

Oppgave 2.

Lag et XPS prosjekt i EDK verktøyet. Bruk Base System Builder (BSB) til å lage et enkelt prosessorsystem til Spartan3 testkortet (pass på å velge riktig "Board Name"!! Spartan-3 Starter Board-2).

Prosessorsystemet skal innholde en MicroBlaze prosessor, **16 KByte** med prosessor BRAM minne, en UART modul for RS232 Putty(eller annet terminalemuleringsprogram) forbindelse og en GPIO modul til LEDene. Andre moduler som er default valgt **skal** fjernes før generering. Velg bare testprogrammet TestApp_Pheripheral.

Putty må settes opp til 9600 bit/s, 8 data bit, ikke paritet, 1 stopp bit og ikke flow control.

Generer prosessorsystemet og deretter bit fil i XPS med LED testprogrammet. Bruk Impact til å generere flash fil og for å laste ned den genererte laste filen til testkortet for uttesting.

Kjør programvaren slik at blinkesekvensen synes på LEDene når man har trykket på BTN3 (dvs. reset knappen).

Lag ett nytt "Software Application Prosjekt" og kopier inn test.c fra oppgaven. Merk dette prosjektet "Mark to init BRAM". Legg til xgpio_tapp_example.c og test.c til prosjektet. Under Generate Linker script verifiser at programmet ikke blir lagt i sram, men internt BRAM.

Hint 2.1:
For å lage en ferdig bit file må følgende gjøres: Hardware => Generate Netlist og deretter Hardware => Generate Bitstream. Deretter lages software ved å velge "Test_App_Peripheral" og så velge Software => Generate Libraries and BSPs og så Software => Build_all_User_Applications. Nå som både HW og SW er generert legger vi SW ".elf" filen inn i ".bit" filen ved å velge Device Configuration => Update Bitstream.

Hint 2.2:
Deretter lages en ".mcs" fil ved å bruke Xilinx Impact verktøyet. Nå er det "download.bit" filen i "implementation" katalogen som skal velges pga. det er resultatet etter Update Bitstream operasjonen. Lag for eksempel en mylab4_del3.mcs fil som deretter lastes på testkortet. Trekk ut og inn strømforsyningen for å få rekonfigurert testkortet. Nå skal LED'ene blinke og tekst komme frem i Putty.

Oppgave 3.

Med utgangspunkt i deloppgave 2 skal det legges til en GPIO PushButton modul.

Den genererte prosessorsystem modulen som finnes under hdl katalogen og har fått lagt til "_stub" i navnet skal deretter legges inn i et ISE prosjekt. Heretter skal ISE brukes til å implementere hele systemet, men XPS brukes fortsatt for utvikling av programvaren.

Det er bare nødvendig å endre c koden i programmet xgpio_tapp_example.c som ligger i Sources under Applications/Project:TestApp_Pheripheral.

Programvaren skal endres slik at når BTN1 trykkes skal programmet kjøre en blinkesekvens med 2 blink og når BTN2 trykkes kjøres en sekvens med 4 blink. MicroBlazen kjører programmet en gang til hvis BTN1 eller BTN2 trykkes igjen, og eksekveringen avsluttes med lysblink i alle LEDene når BTN0 trykkes så det synes at programmet er avsluttet.

Det skal i tillegg stå i Putty vinduet hva programmet har utført og om det venter på ny input fra en trykk knapp.

Hint 3.1:
I XPS bør Hardware => Clean Hardware utføres før nytt prosessorsystem skal lages som i hint 2.1.

Hint 3.2:
Sjekk at ".mhs" filen sine eksterne pinner til Push_Button_3bit modulen har samme navn som i ".ucf" filen for å unngå feil.

Hint 3.3:
Når GPIO modulen er laget ferdig lukkes XPS. Deretter lages et nytt ISE prosjekt hvor for eksempel modulene mylab4_stub.vhd, mylab4.xmp og mylab4.ucf legges inn. Deretter kan XPS igjen startes opp ved å åpne mylab4.xmp.

Hint 3.4: I tillegg til Synthesis – XST og Implement Design MÅ nå også Update Bitstream with Processor Data i kjøres i ISE før lasting av testkortet med Impact. Husk at det er en bit fil med "_download" i navnet som har blitt generert med SW innhold!!

Hint 3.5:
Det kun nødvendig å utføre XPS Software => Build_All_User_Applications og deretter ISE Update Bitstream with Processor Data hvis det bare er gjort endringer i en .c fil.

Hint 3.6:
Push_Button_3Bit går rett inn på de 3 laveste bit. Dermed returnerer funksjonen "XGpio_DiscreteRead(&Push,1)" verdien 0 når det ikke trykkes noe og ellers 1,2 eller 4. Her er Push deklarert med setningen "XGpio Push".

Oppgave 4.

Nå skal systemet som ble laget i deloppgave 3 integreres sammen med VHDL koden fra deloppgave 1.

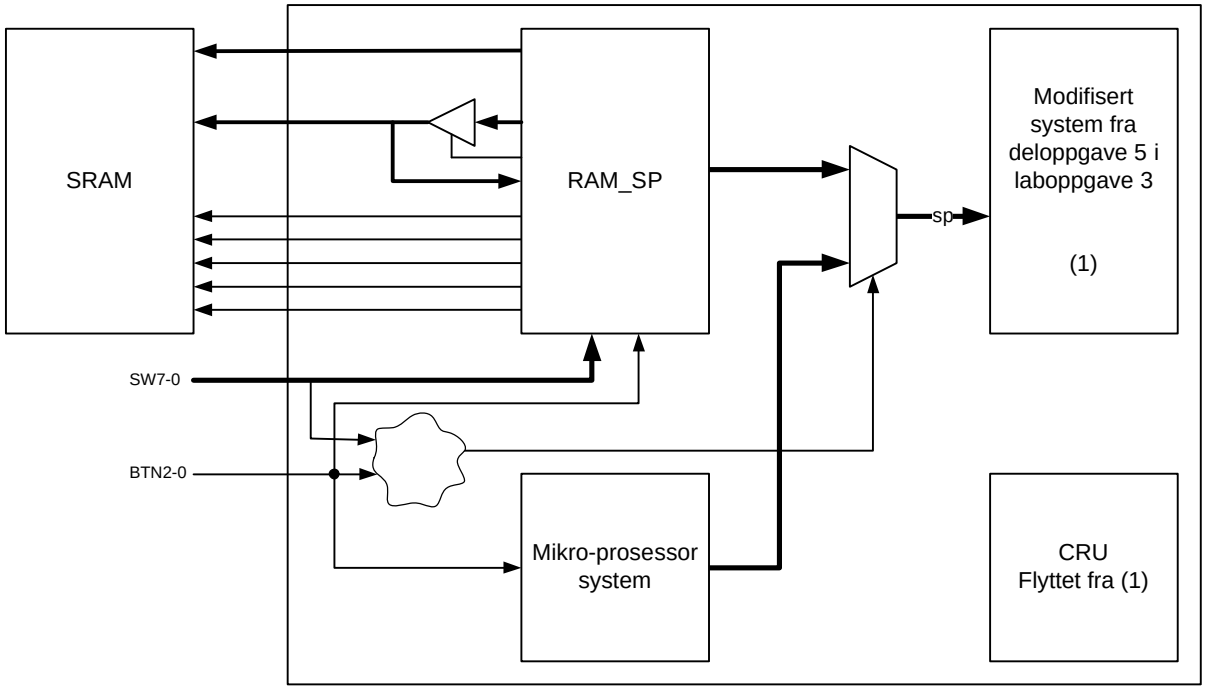

Figure 2

Her skal SW7='0' og SW6='0' velge at vi skal avspille fra RAM'en når BTN2 trykkes akkurat som i deloppgave 1, og når SW7='0' og SW6='1' kjøres MicroBlaze programmet når BTN2 trykkes. MicroBlazen kjører programmet en gang til hvis BTN2 trykkes igjen, og eksekveringen avsluttes med lysblink i alle LEDene når BTN0 trykkes så det synes at programmet er avsluttet. Ved å trykke på BTN3 skal microBlaze og "alt" annet starte opp igjen fra reset tilstand.

Det skal her også stå i Putty vinduet hva MicroBlaze programmet eventuelt har utført og om det venter på ny input fra en trykk knapp.

Når SW7=’1’ blir BTN2 brukt til til å lagre sp sammen med BTN0 og BTN1 for force signalene akkurat som i deloppgave. 1.

Implementer et MicroBlaze program som styrer SP i en valgt sekvens når SW7=’0’ og SW6=’1’.

Hint 4.1: Legg merke til at prosessorsystemet bruker ”x to y” notasjon istedenfor det mer vanlige ”x downto y”. Det betyr at bit 0 blir MSB bit istedenfor LSB bit som vi er vant med ved ”downto” notasjonen. Dette er noe ustandard og litt forvirrende, men har ellers liten betydning.

LYKKE TIL!