Sistemes Digitals
Examen Final. 27 de juny de 2012

Temps per a la resolució: 3 hores. Publicació de qualificacions: 2 de Juliol de 2012.

1 Barra de LEDs

Volem construir una barra de 16 LEDs on, en cada instant, sempre hi ha un únic LED encès. L'objectiu és aconseguir efectes lluminosos variats. Disposem d'un senyal de rellotge, `clk`, de 100 kHz.

El disseny s'estructurarà en dos blocs. El primer d'ells, `core`, indicarà quin LED s'ha d'encendre, codificant el resultat amb un bus de 4 bits. El segon mòdul, `decoder` ha d'encendre el LED adequat en funció d'aquesta informació.

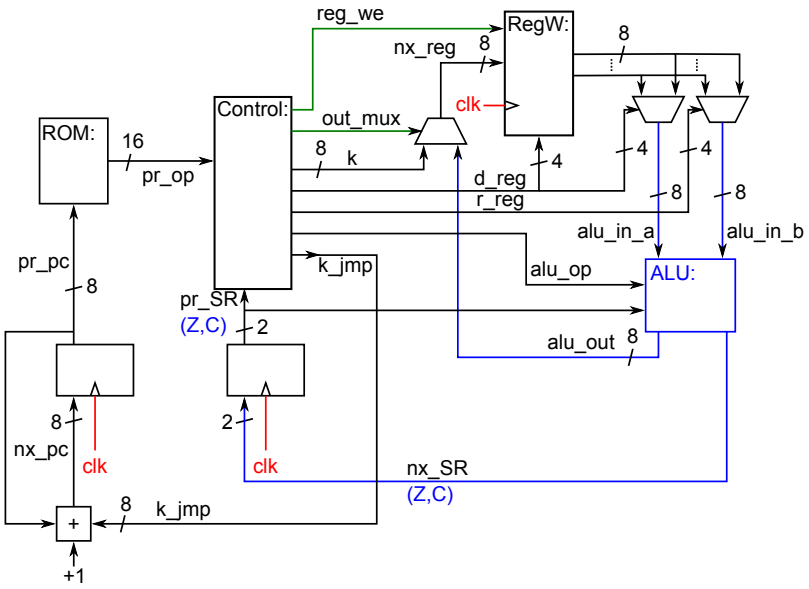
1. Escriviu el codi VHDL corresponent al `decoder` complet.
2. Escriviu un testbench que permeti avaluar el `decoder` i indiqueu el resultat esperat de la simulació.
3. Implementeu el codi VHDL corresponent a un `core` que encén els LEDs consecutivament d'esquerra a dreta, torna a començar per l'esquerra i es repeteix indefinidament. Cada combinació de LEDs s'ha de mantenir exactament 0.2 s.
4. Implementeu el codi VHDL corresponent a un `core` que encén els LEDs consecutivament d'esquerra a dreta, després de dreta a esquerra i a partir d'aquí es repeteix indefinidament. Cada combinació de LEDs s'ha de mantenir exactament 0.2 s, incloent les que tenen només el primer i el darrer LED encès.
5. Implementeu un `core` similar a l'anterior però en el que el canvi de direcció d'esquerra a dreta no es faci cada vegada en el LED 0 (el de més a l'esquerra) sino que cada vegada vagi situant-se més a la dreta: el primer cop al LED 0, el segon al LED 1 i així successivament fins acabar: 16, 15, 14, 15, 16, 15 ,16, i tornar a començar per 0. També en aquest cas, cada combinació s'ha de mantenir durant exactament 0.2 s.
6. Implementeu un `core` similar al més simple, però aconseguint que cada repetició de la seqüència sigui més ràpida: La primera ha de durar un total de 0.2×16 segons, la segona la meitat, la tercera una quarta part i així successivament fins que la darrera seqüència duri 0.1 s. Aleshores es torna a començar el procés.
7. Escriviu un testbench que permeti avaluar el `core` més senzill i indiqueu el resultat esperat de la simulació.

2 AVR

Considereu el processador de la figura. La ROM del programa té les següents ordres:

```
case pr_pc is
when x"00" => LDI & "1000" & cr16 & "0000"; --LDI r16, x80
when x"01" => ADC & "----" & cr16 &  cr16 ; --ADC r16, r16
```

- Dibuixeu un cronograma detallat del valor que prenen els senyals significatius durant l'execució d'aquestes ordres. Suposeu que els flags són zero inicialment. Emfatitzeu les relacions de causa i efecte (podeu fer-ho introduint petits retards entre l'efecte i la causa acompanyats de fletxes indicatives)



Un processador AVR inclou, a més de les instruccions explicades a classe, la instrucció `RCALL`, que fa una crida a un subprograma (una seqüència d'instruccions que fa una funció concreta i que seria útil poder cridar des de diversos llocs). La seva sintaxi és `RCALL k` i fa dues operacions: Desa la que hauria de ser la següent instrucció (`PC+1`) i continua l'execució del programa en la posició de memòria `PC+k+1`, de forma similar a un `RJMP`. El opcode és `1101 kkkk kkkk kkkk`.

Una vegada executat el subprograma desitjat, existeix la instrucció `RET`, que, essencialment restaura el comptador de programa al valor que hauria tingut si mai no s'hagués cridat la instrucció `CALL`.

Els AVR implementen aquesta funcionalitat amb una estructura que permet fer `CALL`s dins d'un subprograma, oferint múltiples nivells de crida a subprogrames. De moment, no pensem en aquesta funcionalitat.

- Quines modificacions caldria introduir al miniAVR per aconseguir implementar un únic nivell de `CALL` i `RET`? Podeu desar el comptador de programa en un registre addicional reservat a aquesta funció. Indiqueu els canvis necessaris a nivell de diagrama de blocs i també a nivell de VHDL dels mòduls que considereu que estarien afectats.

3 Qüestions

1. Quins valors pot prendre `a` si està definit com a `signal a : std_logic;` ? Quin significat tenen aquests valors?

2. Si `a` està definit com a l'apartat anterior, quin és el cronograma resultant d'executar el següent codi?

```
a <= 'Z', '1' after 100 ns, '0' after 200 ns;
a <= '1';
```

3. Quina és la causa més freqüent per la qual les eines de síntesi avisen de la creació d'un latch no intencionat? Doneu-ne un exemple indicant el codi erroni, les implicacions que tindria aquest codi erroni i un codi correcte.

4. Explain (in English -100%- or Catalan -50%-) the differences between the following codes:

```
RAM_process : process(clk)
begin
  if rising_edge(clk) then
     if mem_we ='1' then
        RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
            regs(to_integer(unsigned(r_reg)));
     end if;
     ram_q <= RAM(to_integer(unsigned(mem_adr(9 downto 0))));
  end if;
end process;
```

```
RAM_process : process(clk)
begin
  if rising_edge(clk) then
     if mem_we ='1' then
        RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
            regs(to_integer(unsigned(r_reg)));
     end if;
  end if;
end process;
ram_q <= RAM(to_integer(unsigned(mem_adr(9 downto 0))));
```

```
RAM_process : process(clk)
begin
   if rising_edge(clk) then
      if mem_we ='1' then
         RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
             regs(to_integer(unsigned(r_reg)));
      end if;
      mem_read_adr <= mem_adr;
   end if;
end process;
ram_q <= RAM(to_integer(unsigned(mem_read_adr(9 downto 0))));
```

5. Describe how a synchronizer is used to input asynchronous signals to a synchronous design. Discuss the metastability problem (in English -100%- or Catalan -50%-)

6. Write a list of 30 English terms related to Digital Systems together with a translation to Catalan or Spanish.