

10. mérés

Sorrendi hálózat vizsgálata

A mérés célja

A mérés célkitűzése (1) a Digitális technika I-II. tantárgyakban tanult funkcionális elemekkel tervezés elmélyítése és gyakorlati vonatkozásainak bemutatása, (2) ismerkedés a CAD-rendszerrel való logikai tervezés alapjaival.

Felkészülés a mérésre

Ez a mérés épít a korábbi „Digitális alapeszközök” mérésben tanultakra, illetve az ott átismételt korábbi anyagokra, így ezeket ismétlje át!

*Házi feladat*ként egy egyszerű sorrendi hálózatot kell terveznie. Ezt elvégezheti a Digitális technika tárgyban tanult módszerekkel, de kihasználhatja azt is, hogy a rendelkezésre álló CAD rendszer számos feladatot (pl.: állapotkódolás, állapotminimalizálás) képes automatikusan elvégezni. Bármilyen kreatív mérnöki megoldás elfogadható, ha a specifikált funkciót megvalósítja. Ugyanakkor elvárjuk, hogy a feladat megoldása során hozott döntéseit indokolni tudja. A bevezető mérésen szerzett ismereteit felhasználva szimulációval igazolja a tervezett eszköz helyes működését!

Olvassa el és gondolja végig a *Mérési feladatokat*!

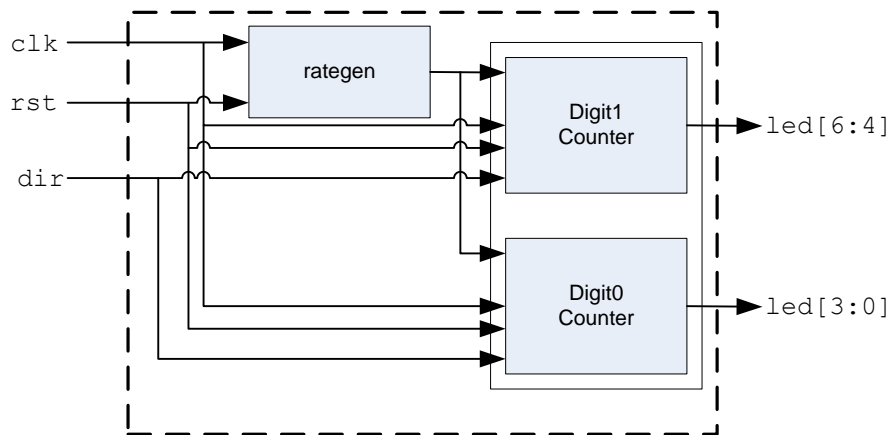
Válaszolja meg a (mérési leírás végén található) *Ellenőrző kérdéseket*!

Mérési feladatok

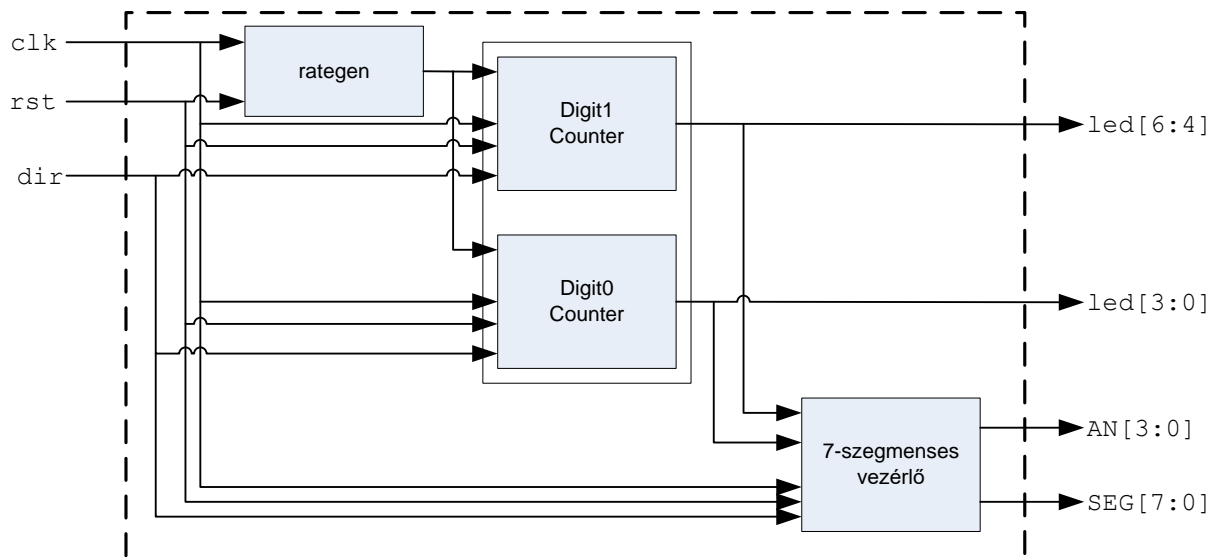
1. Tervezési feladat: Számláló megjelenítése a 7 szegmenses kijelzőn

A bevezető mérés (kiegészítő feladata) során az alábbi blokkvázlatnak megfelelő rendszert hoztuk létre, mely egy 2 digités BCD másodpercszámlálót valósít meg.

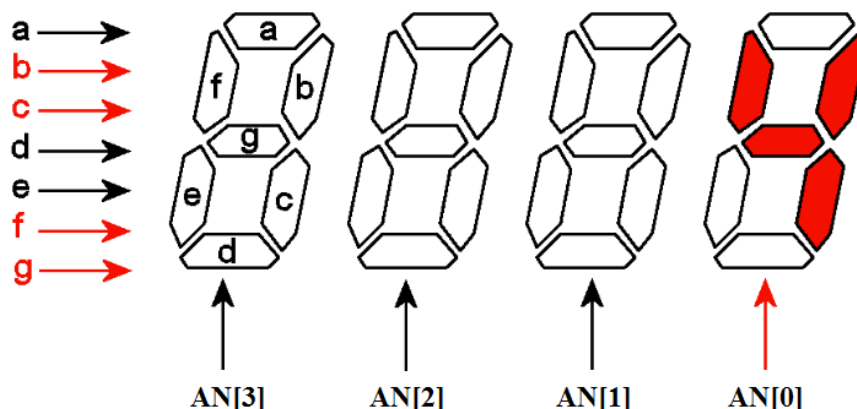
Labor 1. Hallgatói segédlet



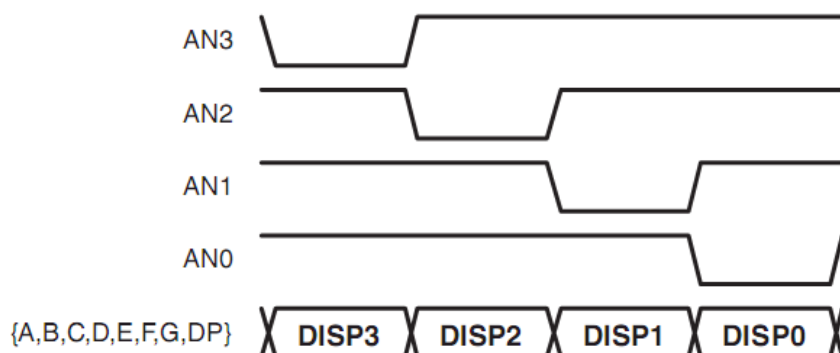
Ezt a rendszert egészítjük ki úgy, hogy a számláló értékét nem csak a LED-eken, hanem a fejlesztői panelen található 7-szegmenses kijelzőn is megjelenítjük, azaz az alábbi blokkvázlatot hozzuk létre.



A fejlesztői panelen található 4-digites 7-szegmenses kijelző idő-multiplexált vezérléssel rendelkezik. A négy darab 7-szegmenses kijelző szegmens jelei (a, b, c, d, e, f, g szegmensek és a tizedespon) közösítve vannak, ezen kívül minden digithez tartozik egy alacsony-aktív anód jel, mely az adott digitet „engedélyezi”. A szegmens és anód jeleket a következő ábra szemlélteti.



Mivel a szegmens jelek közösek mind a négy digitre, ha különböző értéket szeretnénk ezeken megjeleníteni, azt csak időosztásban tudjuk megtenni. Azaz először meghajtjuk a szegmens vonalakat azzal az értékkel, amit a jobb oldali digiten szeretnénk viszontlátni, és ezzel egy időben kiválasztjuk a digitet az anód jelének 0-ba vezérlésével (mindeközben persze a másik 3 digit anód jele 1 értékű, azaz azok nincsenek engedélyezve). Ezt periódikusan ismételve a négy digitre mindegyiken ki tudjuk jelezni a kívánt értéket. Hullámforma ábrán összefoglalva az elmondottakat:



Ha a fenti megjelenítési ciklus túl gyors, akkor a 7-szegmenses kijelző LED-jeinek „nincs idejük kigyulladni”, ha túl lassú, akkor pedig folyamatosnak tűnő megjelenítés helyett villogást tapasztalunk. **A megfelelő frekvencia kHz nagyságrendű.**

Gondoljuk át, hogy mi szükséges ahhoz, hogy másodpercszámlálónkat a 7-szegmenses kijelzőn meg tudjuk jeleníteni! A megfelelő ciklus gyakorisághoz szükséges egy órajel osztó, amely az engedélyező jelet generálja a 7-szegmenses vezérlő többi elemének; ez – az osztási aránytól eltekintve – nem különbözik a másodperc számláló 1 Hz frekvenciájú órajel osztójától.

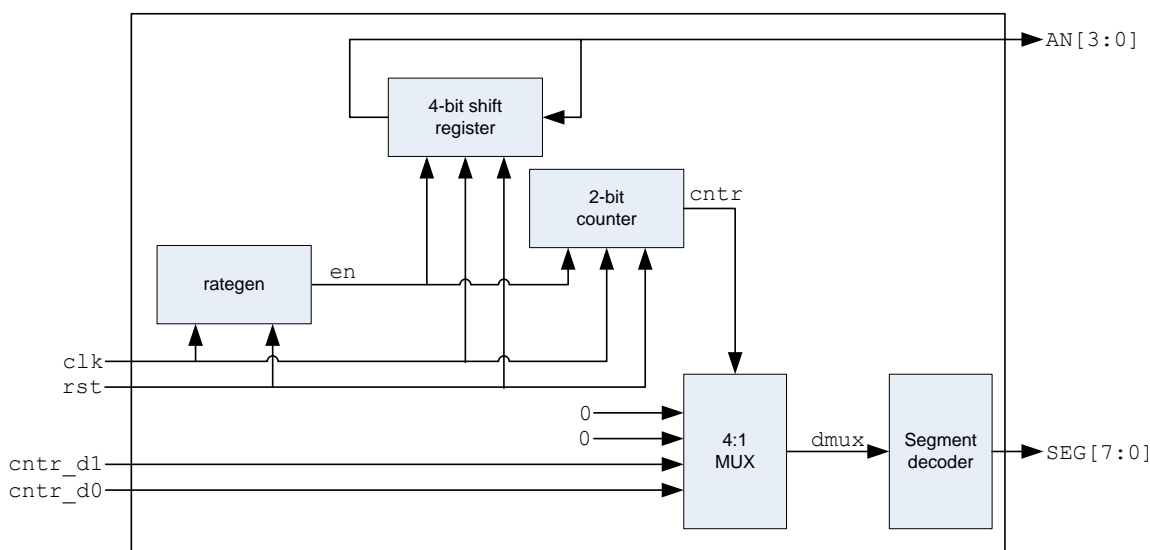
Az anódvezérlő jelet legegyszerűbben egy visszacsatolt, engedélyezhető shift regiszterrel állíthatjuk elő, mely reset hatására az „1110” értéket veszi fel, egyébként pedig balra shiftel. Az LSB helyiértékre az MSB értékét csatoljuk vissza. Így a generált értékek: 1110 → 1101 → 1011 → 0111 → 1110...

Az anód vezérlő jellel szinkronban a megjelenítendő értéket is ki kell választani. A 0. digit esetén ez a másodpercszámláló alsó helyiértéke, az 1. digit esetén a másodperc számláló felső helyiértéke, míg a 2. és 3. digit esetében 0. A kiválasztást egy 4:1 multiplexerrel oldhatjuk meg, melynek vezérlőjele egy 2 bites számláló, mely az anód shift regiszterrel szinkronban

számol (mindig azt mutatja, hogy hányadik anód aktív). *Megj.: természetesen az anód vezérlő jelet generálhatnánk ezen számláló alapján is egy dekóderrel.*

A szegmens multiplexer kimenetén 4 bites bináris értékek fordulhatnak elő, így ezekből még elő kell állítani a szegmensek vezérlőjeleit, azaz egy olyan dekóderre van szükség, amely minden egyes bináris értékhez meghatározza, hogy mely szegmens LED aktív. (A szegmens vezérlőjelek is aktív alacsonyok!) Ez a dekóder megtalálható a panel dokumentációjában, illetve az ISE Language Templates-ek között (*Verilog/Synthesis Constructs/Coding Examples/Misc/*).

A fentiek alapján tehát a 7-szegmenses megjelenítő blokkvázlata az alábbi.



1.1. Project létrehozása

Töltse le a tárgy honlapjáról a forráskódokat tartalmazó ZIP fájlt! A letöltött Verilog fájlok használatával hozzon létre egy új ISE project-et. A letöltött fájlok az alábbiak:

- *wpbevtop1*: a megegyező nevű legfelső szintű modult tartalmazó Verilog fájl. Az alábbi három modulból 1-1 példányt tartalmaz.
- *rategen*: a másodperc számláló számára generál 1 Hz-es frekvenciájú engedélyező jelet.
- *count_sec*: 2 digités másodpercszámláló.
- *cntrl_7seg*: a 7-szegmenses vezérlő modulja, ennek Verilog leírását kell megadni.
- *counter_pins.ucf*: a szükséges láb-bekötéseket tartalmazó fájl.

1.2. A 7-szegmenses kijelző vezérlőjének implementálása

A fentiek alapján készítse el a 7-szegmenses kijelző vezérléséért felelős modult. A modul neve legyen *cntrl_7seg*. A Verilog kód *ebből az egyetlen modulból álljon*, a blokkvázlatban feltüntetett funkcionális elemeket ezen belül, külön always blokkokkal (illetve assign értékadásokkal) valósítsa meg. A letöltött fájl comment-ben tartalmazza a megvalósítandó funkcionális elemeket.

A módosítandó *cntrl_7seg* modulnak az alábbi portjai vannak:

- *clk*: Az 16 MHz-es rendszer órajel
- *rst*: Külső reset jel
- *din0*, *din1*: 4 bites adatbemenetek, melyekre majd az alsó két digiten megjelenítendő értékek kerülnek.
- *AN*: 4 bites kimenet a digit engedélyező jelek meghajtásához (anód jel).
- *SEG*: 8 bites kimenet a szegmensek vezérléséhez.

A fenti blokkvázlatból a szegmens dekóder kódja az alábbi, ez megtalálható a letöltött fájlban (a 8 bites *SEG_DEC* változó és a szegmensek összerendelése: {A,B,C,D,E,F,G,P}):

```
reg [7:0] SEG_DEC;
always @(dmux)
  case (dmux)
    4'h0: SEG_DEC <= 8'b00000011;
    4'h1: SEG_DEC <= 8'b10011111;
    4'h2: SEG_DEC <= 8'b00100101;
    4'h3: SEG_DEC <= 8'b00001101;
    4'h4: SEG_DEC <= 8'b10011001;
    4'h5: SEG_DEC <= 8'b01001001;
    4'h6: SEG_DEC <= 8'b01000001;
    4'h7: SEG_DEC <= 8'b00011111;
    4'h8: SEG_DEC <= 8'b00000001;
    4'h9: SEG_DEC <= 8'b00001001;
    default: SEG_DEC <= 8'b11111111;
  endcase
```

A 7-szegmenses vezérlő kódjának elkészítése után ellenőrizze, hogy a kód szintaktikailag helyes-e!

1.3. Példányosítás

A top-level modulban a jelzett helyen példányosítsa az elkészített 7-szegmenses vezérlőt!

A korábbi mérésen elkészített rendszerhez képest a jelenlegi terv két további portot tartalmaz (*AN[]*, *SEG[]*), melyeket a 7-szegmenses kijelző megfelelő bemeneteire kell kötni. A láb – port hozzárendelést az UCF fájl már tartalmazza, így a terv szintetizálható. A konfigurációs fájl letöltése után győződjön meg a helyes működésről!

2. Futófény

Készítse el az alábbi specifikációnak megfelelő rendszer Verilog leírását

A rendszer kétirányú futófényt („Knight Rider effektus”) valósít meg a fejlesztő panel 8 LED-jét használva. A kiindulási állapot „0000001”, mely után a világító LED balra lép, kb.

másodpercenként. Az „1000000” végértéket elérve a fény haladási iránya megváltozik, s jobbra lép, egészen a „0000001” állapot eléréséig, amikor is újra balra léptetés kezdődik.

A másodpercenkénti ütemezést a már meglévő ratgen modul biztosítja, így a tervezendő blokknak lesz engedélyező bemenete.

Az alábbi megoldások közül válasszon ki egyet és implementálja ezt.

2.1 Megvalósítás állapotgéppel (1)

A megvalósítandó rendszernek $2^8=256$ állapota van: a „00000001” és „10000000” érték csak egyszer fordul elő, a többi kétszer. Egy lehetséges megoldás, hogy az állapotaink kódolása megegyezik a kimenet értékével, így az állapotregiszter közvetlenül az állapotgép kimenete.

Verilog-ban a leírás egy nagyobb case szerkezettel egyszerűen megoldható, lásd Verilog PPT.

2.2 megvalósítás állapotgéppel (2)

Észrevehetjük, hogy az állapotgép állapot átmenetei nagyon egyszerűek, egy adott állapotból mindig egy adott következő állapotba kerülünk feltétel nélkül. Ez igazándiból egy számláló, ami a 0...13 tartományban számol, a számláló maga az állapotregiszter. A kimeneti logika pedig egy dekóder, ami az adott bináris értékből előállítja a LED-eken megjeleníteni kívánt bitmintát (pl. „0000” \rightarrow „00000001”).

2.3 Megvalósítás funkcionális elemekkel

Funkcionális elemek használatával is megvalósíthatjuk a tervet, ha észrevevesszük, hogy igazándiból egy kétirányú shift regiszterről van szó. A shift regiszter mellett szükségünk lesz még egy irány regiszterre, amely meghatározza a shift-elés irányát. Figyeljen rá, hogy az irány regisztert mikor módosítja: adott adott irány végértékét elérve már a másik irányban kell shift-elni, így az irány módosítását a megelőző állapotban kell megtenni.

3. Ellenőrző kérdések

1. Adja meg egy időmultiplexált, 4 digitos 7-segmenses kijelző vezérlésének hullámformáját (AN és DIG jelek). Milyen gyakorisággal kell változtatni az AN jelet? Mi a hatása, ha nagyon lassan (~ 1 sec) változik ez a jel?
2. Készítsen 3 bemenetű és/vagy kaput 2 bemenetű és/vagy kapuk (and2/or2) példányosításával! A 2-bemenetű kapuk portjai: i0 és i1 bemenetek, o kimenet. A 3 bemenetű kapu portjai i0, i1, i2 bemenetek, o kimenet.
3. Adja meg egy 4 bites, visszacsatolt shift regiszter Verilog kódját. A regiszter kezdeti értéke legyen 4'b1000, a shiftelés balra történjen. A modul bemenetei: órajel (clk), reset (rst); a kimenete a shift regiszter aktuális állapota (shr).
4. Adja meg egy 4 bites, visszacsatolt shift regiszter Verilog kódját. A regiszter kezdeti értéke legyen 4'b1000, a shiftelés jobbra történjen. A modul bemenetei: órajel (clk), reset (rst); a kimenete a shift regiszter aktuális állapota (shr).

5. Adja meg egy engedélyező jel generátor modul (rategen) Verilog forráskódját, amely 16 MHz-es rendszer órajel használatával 800 kHz frekvenciájú, egy órajel hosszúságú engedélyező impulzus sorozatot állít elő. A modul bemenetei: órajel (clk), reset (rst); kimenete: engedélyező jel (en).
6. Adja meg egy engedélyező jel generátor modul (rategen) Verilog forráskódját, amely 50 MHz-es rendszer órajel használatával 1 kHz frekvenciájú, egy órajel hosszúságú engedélyező impulzus sorozatot állít elő. A modul bemenetei: órajel (clk), reset (rst); kimenete: engedélyező jel (en).
7. Adja meg egy 1 digites felfelé számláló BCD számláló Verilog kódját. A számláló reset állapota legyen 0. A modul bemenetei: órajel (clk), reset (rst); a kimenete a számláló aktuális állapota (cnt).
8. Adja meg egy 1 digites lefelé számláló BCD számláló Verilog kódját. A számláló reset állapota legyen 0. A modul bemenetei: órajel (clk), reset (rst); a kimenete a számláló aktuális állapota (cnt).
9. Adja meg azt a Verilog Test Fixture kódrészletet (jel deklarációkkal együtt), ami 10 MHz frekvenciájú órajelet biztosít a vizsgálandó modul számára!
10. Adott az alábbi Verilog kód. Egészítse ki az időzítési diagramot.

```

reg [7:0] cntr=0;
reg en; reg c_dl;
always @ (posedge clk)
begin
    cntr <= cntr + 1;
    c_dl <= cntr[7];
    en <= ~cntr[7] & c_dl;
end

reg [3:0] shr=4'b1110;
always @ (posedge clk)
    shr <= {shr[0], shr[3:1]};

```

