8. Részletes tervek

54 – *Override*

Konzulens:
dr. László Zoltán

Csapattagok:

Kriván Bálint	CBVOEN	balint@krivan.hu
Jákli Gábor	ONZ5G1	j_gab666@hotmail.com
Dévényi Attila	L1YRH0	devenyiat@gmail.com
Apagyi Gábor	X8SG3T	apagyi.gabooo@gmail.com
Péter Tamás Pál	N5ZLEG	falconsaglevlist@gmail.com

2011. április 4.

Tartalomjegyzék

Ábrák jegyzéke

8. Részletes tervek

8.1. Osztályok és metódusok tervei

8.1.1. Osztály1

- Felelősség
  [Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]
- Ősosztályok
  [Mely osztályokból származik (öröklési hierarchia)
  Legősebb osztály → Ősosztály2 → Ősosztály3...]
- Interfészek
  [Mely interfészeket valósítja meg.]
- Attribútumok
  [Milyen attribútumai vannak]
  - attribútum1: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
  - attribútum2: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
- Metódusok
  [Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]
  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása, láthatósága (UML jelöléssel)
  - int bar(Osztály5 o1): metódus leírása, láthatósága (UML jelöléssel)

8.1.2. Osztály2

- Felelősség
  [Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]
- Ősosztályok
  [Mely osztályokból származik (öröklési hierarchia)
  Legősebb osztály → Ősosztály2 → Ősosztály3...]
- Interfészek
  [Mely interfészeket valósítja meg.]
- Attribútumok
  [Milyen attribútumai vannak]
  - attribútum1: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
  - attribútum2: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
- Metódusok
  [Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]
  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása, láthatósága (UML jelöléssel)
  - int bar(Osztály5 o1): metódus leírása, láthatósága (UML jelöléssel)

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokat, amelyekkel a program működése ellenőrizhető. Minden bemenő adatsorozathoz definiálni kell, hogy az adatsorozat végrehajtásától a program mely részeinek, funkcióinak ellenőrzését várjuk és konkrétan milyen eredményekre számítunk, ezek az eredmények hogyan vethetők össze a bemenetekkel.]

8.2.1. Áramkörök betöltése

Minden teszteset elején betöltjük a megfelelő fájból az áramkört. Ezt mindig meg kell tenni, azonban csak egy esetben mutatjuk meg az egyszerűség és átláthatóság kedvéért.

Alap áramkör

loadCircuit test1.ovr

load successful

8.2.2. Alap áramkör

- Leírás
  Olyan áramkör, melyben 2 kapcsolóval állíthatjuk egy ÉS kapu bemeneteit, melyet egy LED jelenít meg.
- Ellenőrzött funkcionalitás, várható hibahelyek
  Ellenőrízzük a kapcsoló helyes váltását,az ÉS kapu kimenetének helyes kiszámítását és a LED működését
- Áramkör létrehozása

```
kapcs1=TOGGLE()
kapcs2=TOGGLE()
es=AND(kapcs1,kapcs2)
led=LED(es)
```

- Bemenet és kimenet

Bemenet	*Kimenet*
step switch kapcs1 step check -all switch kapcs2 step	kapcs1: 1 simulation successful kapcs1: 1 kapcs2: 0 led: 0 kapcs2: 1 simulation successful kapcs1: 1 kapcs2: 1 led: 1

8.2.3. MPX-es áramkör

- Leírás
  Olyan áramkört hozunk létre, melyben egy 7 szegmenses kijelzőt hajtunk meg kapcsolókkal és egy MPX-xel. A 7szegmenses kijelző [2]-[7] bemeneteire kapcsolókat kötünk, a [1] bemenetét egy MPX adja, mely 4 kapcsolóból választja ki az egyiket, tehát egy 4/1 es MPX.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Ellenőrízzük a MPX helyes működését, és a 7 szegmenses kijelzőt. Hiba a MPX kiválasztása során történhet, hogy rossz jelet juttat a kimenetére.

- Áramkör létrehozása

```
inmpx1=TOGGLE()
inmpx2=TOGGLE()
inmpx3=TOGGLE()
inmpx4=TOGGLE()
selmpx1=TOGGLE()
selmpx2=TOGGLE()
mux=MPX(inmpx1,inmpx2,inmpx3,inmpx4,selmpx1,selmpx2)
seg7=TOGGLE()
seg6=TOGGLE()
seg5=TOGGLE()
seg4=TOGGLE()
seg3=TOGGLE()
seg2=TOGGLE()
display=7SEG(mux,seg2,seg3,seg4,seg5,seg6, seg7)
```

- Bemenet és kimenet

Bemenet	*Kimenet*

`switch inmpx1`
`switch inmpx3`
`step`
`switch selmpx2`
`switch seg2`
`step`
`switch selmpx2`
`switch selmpx1`
`step`
	`copy-paste from netbeans`

8.2.4. Visszacsatolt stabil áramkör

- Leírás
  Egy olyan áramkört hozunk létre, melyben egy VAGY kapu szerepel, aminek egyik bemenete egy kapcsoló, kimenetét pedig visszakötjük a második bemenetére, illetve egy csomóponton keresztül egy LED-re is eljuttatjuk.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Ellenőrízzük, hogy az áramkör helyesen stabilnak érzékeli e a kapcsolást, illetve a VAGY kapu helyes működését is ellenőrízzük. Hibát a visszakötés okozhat.

- Áramkör létrehozása

```
kapcs=TOGGLE()
vagy=OR(kapcs,node[2])
node=NODE(vagy,2)
led=LED(node[1])
```

- Bemenet és kimenet

Bemenet	*Kimenet*

`step`
`switch kapcs`
`step`
	`copy-paste from netbeans`

8.2.5. Visszacsatolt nem stabil áramkör

- Leírás
  Egy olyan áramkört hozunk létre, melyben egy ÉS kapu szerepel, aminek egyik bemenete egy kapcsoló, kimenetét pedig visszakötjük egy inverteren keresztül a második bemenetére, illetve egy csomóponton keresztül egy LED-re is eljuttatjuk.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Ellenőrízzük, hogy az áramkör helyesen instabilnak érzékeli e a kapcsolást. Továbbá, hogy a hálózat helyesen egy bizonyos lépésszám után instabillá nyilvánítja e a hálózatot. Hibás működést ez okozhat, tehát ha az áramkör ezt rosszul állapítja meg, és nem jelzi.

- Áramkör létrehozása

```
kapcs=TOGGLE()
inv=INV(node[2])
es=AND(kapcs,inv)
node=NODE(es,2)
led=LED(node[1])
```

- Bemenet és kimenet

Bemenet	*Kimenet*
`switch kapcs` `step`	`copy-paste from netbeans`

8.2.6. Flip-flop-os áramkör

- Leírás
  Egy olyan áramkört hozunk létre, melyben egy JK flipflop szerepel, J és K bemenetére kapcsolókat kötünk, órajelét egy jelgenerátorból kapja, és a kimenetét egy oszcilloszkóp kapja meg.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Ellenőrízzük a jelgenerátort, hogy megfelelő jelet ad e ciklikusan, ellenőrízzük a JK flipflop működését, illetve, hogy megfelelelően lép e az órajelre, továbbá ellenőrízzük az oszcilloszkóp helyes működését. Hiba lehetséges a jelgenerátor működésében, a JK flipflop működésében illetve számolásában, és az oszcilloszkóp működésében.

- Áramkör létrehozása

```
j=TOGGLE()
k=TOGGLE()
```

```
seqgen=SEQGEN()
jk=FFJK(seqgen,j,k)
scope=SCOPE(jk)
```

- Bemenet és kimenet

Bemenet	*Kimenet*
switch k step step switch j step step switch j switch k step step	copy-paste from netbeans

8.2.7. Kompozitos áramkör

- Leírás
  Egy olyan áramkört valósítunk meg, melyben egy kompozit szerepel. Ez a kompozit egy 2 bites balról tölthető shiftregisztert valósít meg. A kompozitnak két bemenete van egy kapcsoló ami a balról bejövő értéket adja, és egy jelgenerátor, amely az órajelet. Belül 2 D flipflop található összekötve. Az első flipflop kimenetét kiadja a kompozit kimenetén is, és a 2-ik flipflop bemenetére is adja, ezért NODE is kell. Kompozit kimenete a 2 bit és a carry.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Kompozit helyes működését ellenőrízzük.

- Áramkör létrehozása

```
input=TOGGLE()
seqgen=SEQGEN()
composite SHR(clk,in){
    nodeclk=NODE(clk,2)
    d1=FFD(nodeclk[1],in)
    node1=NODE(d1,2)
    d2=FFD(nodeclk[2],node1[2])
    node2=NODE(d2,2)
}(node1[1],node2[2],node2[2])
myshr=SHR(seqgen,input)
led1=LED(myshr[1])
led2=LED(myshr[2])
ledcarry=LED(myshr[3])
```

- Bemenet és kimenet

Bemenet	*Kimenet*
switch input step step switch input step step step step	copy-paste from netbeans

8.2.8. Kompoziton belüli kompozitos áramkör

- Leírás
  Egy olyan áramkört hozunk létre melyben egy kompozit szerepel ami egy 4bites shiftregiszter. Ezt shiftregisztert úgy hozzuk létre, hogy a kompoziton belül 2db 2 bites shiftregiszter szerepel mint kompozitok. Kívülről csak a 4 bites shiftregisztert látjuk, ami belül 4 kompozittal jön létre. 4 bit és carry kimeneteket leden jelezzük, míg az input és órajel bemenetét kapcsolóval és jelgenerátorral adjuk.

- Ellenőrzött funkcionalitás, várható hibahelyek
  Leteszteljük, hogy működik e a kompozit elem, ha belül bonyolultabb áramköri hálózat szerepel, egy kompozit, illetve jelen esetben több kompozit.

- Áramkör létrehozása

```
composite SHR2BIT(clk,in){
    nodeclk=NODE(clk,2)
    d1=FFD(nodeclk[1],in)
    node1=NODE(d1,2)
    d2=FFD(nodeclk[2],node1[2])
    node2=NODE(d2,2)
}(node1[1],node2[2],node2[2])
input=TOGGLE() \newline
seqgen=SEQGEN()
composite SHR4BIT(clk,in){
    nodeclk=NODE(clk,2)
    shr2bit_1=SHR2BIT(nodeclk[1],in)
    shr2bit_2=SHR2BIT(nodeclk[2],shr2bit[3])
}(shr2bit_2[2],shr2bit_2[1],shr2bit_1[2],shr2bit_1[1],shr2bit_2[3])
my4bitshr=SHR4BIT(clk,in)
ledbit1=LED(my4bitshr[1])
ledbit2=LED(my4bitshr[2])
ledbit3=LED(my4bitshr[3])
ledbit4=LED(my4bitshr[4])
ledcarry=LED(my4bitshr[5])
```

- Bemenet és kimenet

Bemenet	*Kimenet*

`switch input`
`step`
`step`
`step`
`step`
`switch input`
`step`
`step`
`switch input`
`step`
`step`
	`copy-paste from netbeans`

8.3. A tesztelést támogató programok tervei

[A tesztadatok előállítására, a tesztek eredményeinek kiértékelésére szolgáló segédprogramok részletes terveit kell elkészíteni.]

8.4. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2011.04.01. 15:00	2,5 óra	**Péter T.**	Tesztesetek megtervezése, leírása, felépítésük megadása a bemeneti nyelvnek megfelelően
2011.04.02. 10:00	3 óra	**Apagyi G.**	Tesztesetek felhasználói interakciójának, illetve várt kimeneteinek megtervezése.
...	...	**...**	...