A feladat:

Laboratórium 1 felkészülési feladat

Hallgató:
Mérés sorszáma: 10

Készítsen egy olyan Verilog modult, ami Z kimenetén 1-el jelzi, ha az egyetlen X bemenetére ciklikusan érkező 4 bites számok az alább közölt feltételnek megfelelnek! A közbülső ütemekben a kimenet 0 legyen! Az X bemenetre az adatok sorosan érkeznek, a legnagyobb helyiértékű bit (MSB) jön elsőnek.
A vizsgált feltétel: **A bemenet bináris súlya (1 bitek száma) 1**
A mérés kezdetén be kell mutatni a működő Verilog kódot és a működést igazoló szimulációt. A modulnak legyen órajel engedélyező bemenete is! (Ez a tesztelhetőség szempontjából fontos.)

Példa: (A feladatot a fent megadott feltételre oldja meg, ez csak egy értelmezést segítő példa!)
Feltétel: a bemenet osztható 7-tel
X: 0010 0111 1100 1110 0000 0110...
Z: 0000 0001 0000 0001 0001 0000...

A megoldásomról röviden:

Egy olyan modult hoztam létre amely 4-esével vizsgálja a sorosan érkező biteket, és az alapján jelez, ha a számnégyesnek a bináris súlya 1. Az MSB érkezik először, ezért a bit fogadásakor a meglévő biteket balra shifteltem (bár a kód helyesen működne akkor is, ha megcserélném ezt a sorrendet). A modul úgy működik, hogy számolja a biteket, folyamatosan vizsgálja az eddigi értékeket, hogy 1-e a bináris súlyuk, és amikor elér a 4. bithez, akkor az eredményt kiírja Z-re.

A modulnak van ce (engedélyező), clk (órajel), és rst (reset) bemenete is. “Reset”-kor a belső regisztereket 0-ba állítjuk, ezért számoláskor a negyedik bit érkezésekor 0-án fog állni a számláló. A számláló is csak akkor kezd el számolni amikor engedélyezve van a modul.

A modul kódja:

```
21  module sorozat(
22     input X, rst, clk, ce,
23     output Z
24      );
25
26  //az elozo 4 erteket tarolom
27  reg [3:0] prev;
28  //belso szamlalo regiszter
29  reg [1:0] cntr;
30
31  //ebben a valtozoban azt latjuk, hogy az eppen tarolt negyes binaris sulya 1-e
32  wire cout;
33
34  always @(posedge clk)
35  begin
36     if(rst)
37     begin
38        prev<=0;
39        cntr <=0;
40     end
41
42     else if(ce)
43     begin
44     //msb erkezik eloszor, ezert balra shiftelek
45     prev <= {prev[2:0],X};
46
47     if(cntr == 3)
48        cntr <=0;
49     else
50        cntr <=cntr+1;
51
52     end
53
54  end
55
56  //ezeknek a szamoknak a binaris sulya 1
57  assign cout = (prev==1)|(prev == 2)|(prev==4)|(prev==8);
58
59  //amikor megkapom a 4. bitet, akkor kiirom a kimenetre hogy az eppen tarolt negyes binaris sulya 1-e
60  assign Z = cout & (cntr ==0);
61
62
63  endmodule
```

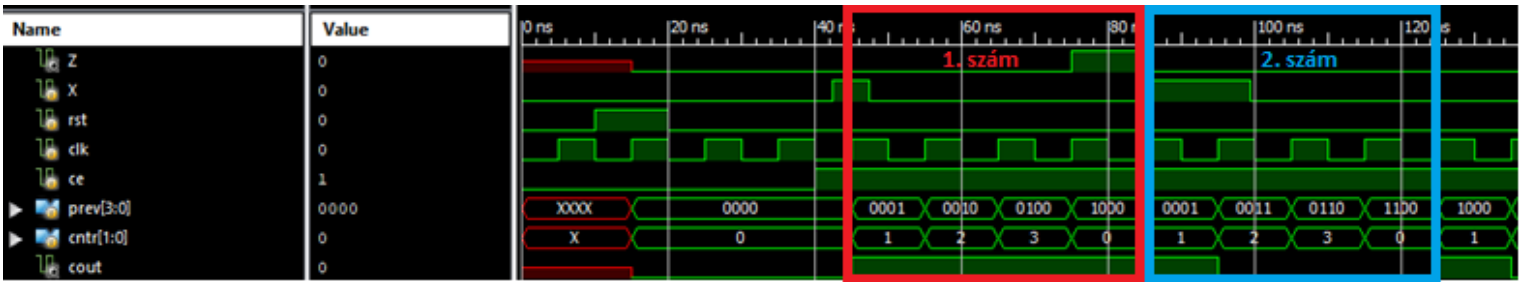
A kódon látszik, hogy a reset,ce mind szinkron. A számláló 0-tól 3-ig számol (jobban mondva 1-től 0-ig). Cout wire-ben található a vizsgálandó feltételnek (bináris súly = 1) az értéke, és ezt ugyan minden köztes állapotra is kiértékelem, a kimeneten csak akkor látszik, amikor cntr == 0, azaz beérkezett az utolsó bit is a vektorból.

A teszteléshez írtam egy rövid testbench-et, amellyel resetelés, és engedélyezés után 2 db 4 bites számot vizsgálunk meg.

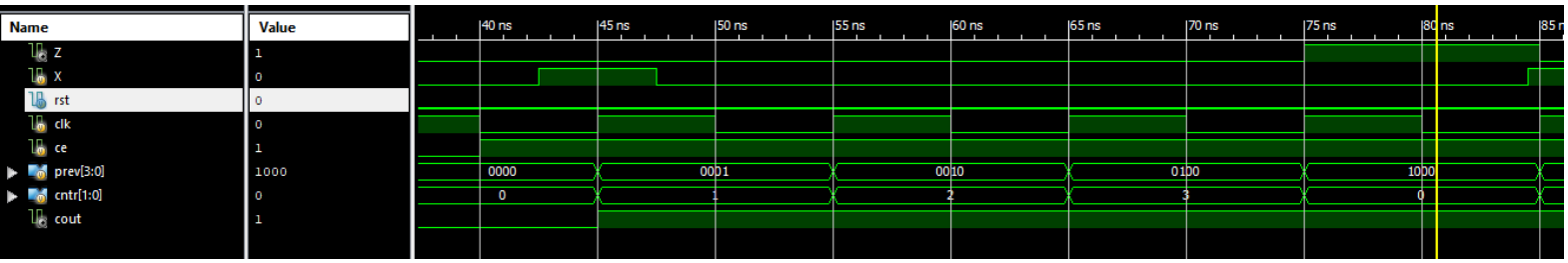
A testbench kódja:

```
25  module tb_sorozat;
26
27      // Inputs
28      reg X;
29      reg rst;
30      reg clk;
31      reg ce;
32
33      // Outputs
34      wire Z;
35
36      // Instantiate the Unit Under Test (UUT)
37      sorozat uut (
38          .X(X),
39          .rst(rst),
40          .clk(clk),
41          .ce(ce),
42          .Z(Z)
43      );
44
45      initial begin
46          // Initialize Inputs
47          X = 0;
48          rst = 0;
49          clk = 0;
50          ce = 0;
51
52
53          // Add stimulus here
54          #10 rst<=1;
55          #10 rst<=0;
56          #20 ce<=1;
57          #2.5 X<=1;
58          #5 X<=0;
59          #37 X<=1;
60          #15 X<=0;
61
62      end
63
64      always #5 clk <=~clk;
65
66  endmodule
```

Ez nem túl sokat mondó, lássuk a hullámformát:

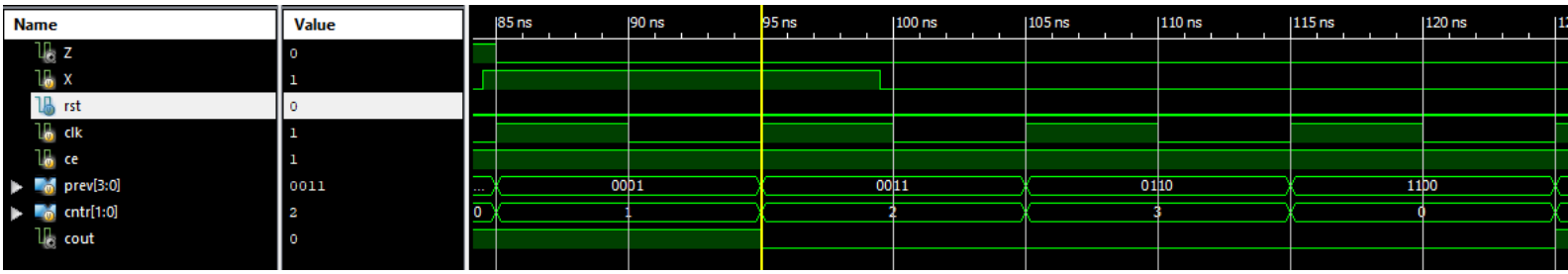


A szimulációt futtatva a fentebbi hullámformát kaptuk. “prev”, “cntr”, és “cout” a modul belső változói. 15 ns-nél a modult reseteljük, a várt működésnek megfelelően a modul belső változói 0-ba állnak. 40ns-nál engedélyezzük a modult, és nagyon helyesen csak innen kezdi el számolni az érkező mintákat. Nézzük meg a mintákat közelebbről:



Az első minta (4 bit) egy egyessel kezdődik, majd 3 db 0 követi. Ezt az 1-est fogjuk folyamatosan shiftelgetni. A számlálás 1-gyel kezdődik, és a 4. bit beérkezésekor 0-án áll. A feltételt cout wire-ön folyamatosan kiértékeljük, de csak akkor engedjük megjelenni a kimeneten, amikor megkaptuk az utolsó bitet. A kimeneten az eredményt abban az órajelperiódusban látjuk, amelyikben mintavételeztük az utolsó bitet.

A következő mintában már nem teljesül a feltétel:



Ebben a sorozatban a két első bit 1-es, ezért helyesen a negyedik bit fogadásakor a belső regiszterből “1100”-t olvasunk. Látjuk, hogy cout folyamatosan kiértékelődik, és az első bit után azt mondja, hogy teljesül a feltétel, de nem látjuk a kimeneten, hiszen nem kapjuk meg a teljes vektort.

A feladatokat önállóan, meg nem engedett segítség igénybevétele nélkül oldottam meg