

```
1  -----
2  --      D-FlipFlip      --
3  -----
4  entity FF is
5      port(D:      in bit;
6            Q:      out bit;
7            CLK:    in bit);
8  end FF;
9
10 architecture behv of FF is
11 begin
12     process(CLK)
13     begin
14         if CLK'event and CLK = '1' then
15             Q <= D;
16         end if;
17     end process;
18 end behv;
19
20 -----
21 --      CRC decoder      --
22 -----
23 entity RANDOM_6BIT is
24     port(V:      in bit;
25           CLK:    in bit;
26           RAND:   out bit_vector(5 downto 0));
27 end RANDOM_6BIT;
28
29 architecture behv of RANDOM_6BIT is
30
31     component FF
32     port(D:      in bit;
33           Q:      out bit;
34           CLK:    in bit);
35 end component;
36
37 signal D: bit_vector(5 downto 0);
38 signal Q: bit_vector(5 downto 0);
39 signal Z: bit;
40
41 begin
42     D(0) <= Z XOR V;
43     D0:FF port map (D(0), Q(0), CLK);
44     d(1) <= Q(0);
45     D1:FF port map (D(1), Q(1), CLK);
46     D(2) <= Z XOR Q(1);
47     D2:FF port map (D(2), Q(2), CLK);
48     D(3) <= Z XOR Q(2);
49     D3:FF port map (D(3), Q(3), CLK);
50     D(4) <= Q(3);
51     D4:FF port map (D(4), Q(4), CLK);
52     D(5) <= Z XOR Q(4);
53     D5:FF port map (D(5), Q(5), CLK);
54     Z <= Q(5);
55
56     RAND <= D;
57 end behv;
58
59 -----
60 --      7-Bit counter      --
61 -----
62
63 library ieee;
64 use ieee.std_logic_1164.all; -- Beschreibung der std_logic Datentypen
65 use ieee.std_logic_unsigned.all; -- Konvertierungsfunktionen std_logic
66
67 entity RANDOM_CT is
68     port(RESET: in bit;
69           CLK:   in bit;
```

```
70     CNT: out bit_vector (6 downto 0));
71 end RANDOM_CT;
72
73 architecture counter of RANDOM_CT is
74
75 signal CNT_INT: std_logic_vector (6 downto 0);
76
77 begin
78     process(CLK, RESET)
79     begin
80         if (CLK'event and CLK='0') then
81             if (RESET = '1') then
82                 CNT_INT <= "0000001";
83             else
84                 CNT_INT <= CNT_INT + 1;
85             end if;
86         end if;
87     end process;
88 CNT <= to_bitvector(CNT_INT);
89 end counter;
90
91
92 -----
93 -- SET FOR EVERYTHING --
94 -----
95
96 entity RANDOM_SET is
97     port(RAND: out bit_vector(5 downto 0);
98          INIT: in bit;
99          CLK: in bit;
100          MAX: out bit_vector(6 downto 0));
101 end RANDOM_SET;
102
103 architecture behv of RANDOM_SET is
104
105 component RANDOM_6BIT is
106     port(V: in bit;
107          CLK: in bit;
108          RAND: out bit_vector(5 downto 0));
109 end component;
110 component RANDOM_CT is
111     port(RESET: in bit;
112          CLK: in bit;
113          CNT: out bit_vector (6 downto 0));
114 end component;
115
116 signal RAND_INT: bit_vector(5 downto 0);
117 signal RST_INT: bit;
118 signal CNT_INT: bit_vector (6 downto 0);
119
120 begin
121 RND0:RANDOM_6BIT port map(INIT, CLK, RAND_INT);
122 CNT0:RANDOM_CT port map(RST_INT, CLK, CNT_INT);
123
124 process (RAND_INT, CLK)
125 begin
126     if (CLK'event and CLK='1') then
127         RST_INT <= '0';
128         if (RAND_INT = "001000") then
129             RST_INT <= '1';
130             MAX <= CNT_INT;
131         end if;
132     end if;
133 end process;
134
135 RAND <= RAND_INT;
136
137 end behv;
```