Examples

Seven segment display decoder

A decoder of BCD code driving seven segment display with common anode is shown in this example. The decoder owns additional 'test' input which is used to check the display.

Description using logical equations

```
"Header
module BCD27seg
title 'seven segment display decoder'
"declaration part
  a,b,c,d,e,f,g pin istype 'com';"outputs
  x3,x2,x1,x0,test pin;          "inputs
  bcd = [x3,x2,x1,x0];
  H,L,X =1,0,.X.
"circuit description part
equations
  a = !test & (!x3 & !x2 & !x1 & x0 # x2 & !x1 & !x0);
  b = !test & (x2 & !x1 & x0 # x2 & x1 & !x0);
  c = !test & (!x2 & x1 & !x0);
  d = !test & (x2 & x1 & x0 # !x3 & !x2 & !x1 & x0 # x2 & !x1 & !x0);
  e = !test & (x2 & !x1 # x0);
  f = !test & (!x2 & x1 # !x3 & !x2 & x0 # x1 & x0);
  g = !test & (!x3 & !x2 & !x1 # x2 & x1 & x0);
"simulation part
test_vectors([test,bcd] -> [a,b,c,d,e,f,g])
             [0,0]      -> [0,0,0,0,0,0,1];
             [0,1]      -> [1,0,0,1,1,1,1];
             [0,2]      -> [0,0,1,0,0,1,0];
             [0,3]      -> [0,0,0,0,1,1,0];
             [0,4]      -> [1,0,0,1,1,0,0];
             [0,5]      -> [0,1,0,0,1,0,0];
             [0,6]      -> [0,1,0,0,0,0,0];
             [0,7]      -> [0,0,0,1,1,1,1];
             [0,8]      -> [0,0,0,0,0,0,0];
             [0,9]      -> [0,0,0,0,1,0,0];
             [1,.x.]    -> [0,0,0,0,0,0,0];
end
```

Description using truth table

Description of the same decoder as above using truth table is shown in this example. Only circuit description part is shown because header, declaration and simulation parts don't change. When you compare this example with that above you can notice that, in decoder circuit case, description using logical equations is little shorter but description using truth table is much easier.

```
"circuit description part
equations
@dcset
truth_table([test,bcd]  -> [a,b,c,d,e,f,g])
             [0,0]      -> [0,0,0,0,0,0,1];
             [0,1]      -> [1,0,0,1,1,1,1];
```

```
              [0,2]      ->[0,0,1,0,0,1,0];
              [0,3]      ->[0,0,0,0,1,1,0];
              [0,4]      ->[1,0,0,1,1,0,0];
              [0,5]      ->[0,1,0,0,1,0,0];
              [0,6]      ->[0,1,0,0,0,0,0];
              [0,7]      ->[0,0,0,1,1,1,1];
              [0,8]      ->[0,0,0,0,0,0,0];
              [0,9]      ->[0,0,0,0,1,0,0];
             [1,.x.]     ->[0,0,0,0,0,0,0];
```

Counter

Two-directional counter counting in BCD code is shown in this example. Counter has tree-state outputs and asynchronous clear input.

Description using logical equations

```
module counter
  q3,q2,q1,q0 pin istype 'reg';                              "counter outputs
  dir pin;                                 "input which choose counting direction
  clear pin;
  oe pin;                               "input driving three state output buffers
  clk pin;
  count = [q3,q2,q1,q0];
equations
  count.oe=!oe;
  count.clk=clk;
  count.ar=!clear;
  when dir then
    count:=(count<9) & (count+1)
  else
    when (count==0) then
      count:=[1,0,0,1]
    else
      count:=count-1;

test_vectors( [clk, clear, dir,oe] -> count )
              [0,   0,     0,  0]  ->  0;
              [0,   1,     0,  1]  ->  .z.;
@const i=1;
@repeat 9 {
              [.c., 1,     1 , 0]  ->  i;
@const i=i+1;
}
              [1,   0,     0,  0]  ->  0;
@const i=9;
@repeat 10 {
              [.c., 1,     0,  0]  ->i;
@const i=i-1;
}
end
```

Description using state diagram

Description of the same counter as above using state diagram is shown in this example. Header and declaration and simulation parts don't change so they are omitted.

```
equations
  count.oe=!oe;
  count.clk=clk;
  count.ar=!clear;
  state_diagram (count)
  state 0:
    if dir then 1
    else 9;
  state 1:
    if dir then 2
    else 0;
  state 2:
    if dir then 3
    else 1;
  state 3:
    if dir then 4
    else 2;
  state 4:
    if dir then 5
    else 3;
  state 5:
    if dir then 6
    else 4;
  state 6:
    if dir then 7
    else 5;
  state 7:
    if dir then 8
    else 6;
  state 8:
    if dir then 9
    else 7;
  state 9:
    if dir then 0
    else 8;
```