Solutions to Quiz 2

There were two versions of each question. The values and the answers for both versions are given below.

Question 1

(a) A signal named $\overline{\texttt{wet}}$ (or $\overline{\texttt{dry}}$) is at a high (or low) logic level. Is it wet (or dry) (yes or no)?

(b) What logic level would you expect if an alarm signal was indicating a warning and the name of an output was `alarm_n` (or `warning_n`) (high or low)?

Answers

The names of each of these signals indicates they are active-low.

(a) If the signal is low it is true (yes), if it is high it is false (no):

- No, since $\overline{\texttt{wet}}$ is high it is not wet.
- Yes, since $\overline{\texttt{dry}}$ is low it is dry.

(b) For an active-low signal, when a condition is true (in this case, the alarm or warning is true) then the logic level should be low.

Question 2

Fill in the blank boxes in the table below so that all values in each row are consistent (agree with each other). The first row is an example.

signal name	truth value (T/F)	truth value in an expression (0/1)	logic level (H/L)	Verilog value when input (0/1)
$\overline{\texttt{active}}$	T	1	L	0
`smoke*`				1
`hot`	T			

signal name	truth value (T/F)	truth value in an expression (0/1)	logic level (H/L)	Verilog value when input (0/1)
$\overline{\texttt{running}}$	T	1	L	0
`clear`				0
`cold*`	F			

Answers

We can work from the given values and the signal name to fill in the remaining columns.

The input value and logic level must agree (0 is L(low) and 1 is H(igh)) regardless of whether the signal is active-high or active-low.

Similarly the truth value and the truth value used in an expression must agree (0 is F(alse) and 1 is T(rue)).

However, the relationship between truth value and logic level is determined by whether the signal is active-high (T(rue) is H(igh)) or active-low (T(rue) is L(low)).

Following these rules we can fill in the tables:

signal name	truth value (T/F)	truth value in an expression (0/1)	logic level (H/L)	Verilog value when input (0/1)
$\overline{\texttt{active}}$	T	1	L	0
`smoke*`	F	0	H	1
`hot`	T	1	H	1

Or:

signal name	truth value (T/F)	truth value in an expression (0/1)	logic level (H/L)	Verilog value when input (0/1)
$\overline{\texttt{running}}$	T	1	L	0
clear	F	0	L	0
cold*	F	0	H	1

Question 3

A module named **m** is declared as:

```
module m
   #(parameter i=4)
   ( input logic [7:0] x, y, c ) ;

endmodule
```

Another module, named **top** is declared as:

```
module top ;

   logic [7:0] a, b, c ;

   assign a = 1 ;
   assign b = 2 ;
   assign c = 3 ;

   m m0 (a,b,c) ;
   m #(1) m1 (.y(b),.x(a),.*) ;
   m #(.i(2)) m2 (c-a,b,a+1) ;

endmodule
```

Or:

```
module top ;

   logic [7:0] a, b, c ;

   assign a = 3 ;
   assign b = 2 ;
   assign c = 1 ;

   m m0 (a,b,c) ;
   m #(2) m1 (.y(b),.x(a),.*) ;
   m #(.i(1)) m2 (a-c,b,a+1) ;

endmodule
```

Fill in the following table with the values of **i**, **x**, **y** and **c** in each instance of the **m** module in the **top** module:

instance	i	x	y	c
m0				
m1				
m2				

Answers

Instance m0 A parameter value is not given so the default value (4) is used. The signals are matched to ports according to order. In the first version the values are 1, 2, 3. In the second they are 3, 2, 1.

Instance m1 The parameter value is specified as 1 (or 2) and the first two signals are matched to ports by name (**y** is set to **b**, **x** to **a**) and the remaining signal(s) are matched by the wildcard according to matching names so **c** is matched with **c**. In the first version this sets **y** to 2, **x** to 1 and **c** to 3. In the second version this sets **y** to 2, **x** to 3 and **c** to 1.

Instance m2 The parameter value is set using an explicit name (**i**) to 2 (or 1). The signals are matched by position again. In the first version the values of the expressions are $c - a = 3 - 1 = 2$, $b = 2$, and $a + 1 = 1 + 1 = 2$. In the second they are $a - c = 3 - 1 = 2$, $b = 2$, and $a + 1 = 3 + 1 = 4$.

Version 1

instance	i	x	y	c
m0	4	1	2	3
m1	1	1	2	3
m2	2	2	2	2

Version 2

instance	i	x	y	c
m0	4	3	2	1
m1	2	3	2	1
m2	1	2	2	4

As a check, the following simulation:

```
module m
   #(parameter i=4)
   ( input logic [7:0] x, y, c ) ;

   initial #1 $display(i, x, y, c) ;

endmodule

module top1 ;

   logic [7:0] a, b, c ;

   assign a = 1 ;
   assign b = 2 ;
   assign c = 3 ;

   m m0 (a,b,c) ;
   m #(1) m1 (.y(b),.x(a),.*) ;
   m #(.i(2)) m2 (c-a,b,a+1) ;

endmodule

module top2 ;

   logic [7:0] a, b, c ;

   assign a = 3 ;
   assign b = 2 ;
   assign c = 1 ;

   m m0 (a,b,c) ;
   m #(2) m1 (.y(b),.x(a),.*) ;
   m #(.i(1)) m2 (a-c,b,a+1) ;

endmodule

module top ;
   top1 t1 () ;
   top2 t2 () ;
endmodule
```

outputs the following[1]:

```
#             4   1   2   3
#             1   1   2   3
#             2   2   2   2
#             4   3   2   1
#             2   3   2   1
#             1   2   2   4
```

[1]The order of execution of the **`initial`** statements in each instance of **`m`** can be determined from the values of **`i`**.