

# Function (3A)

---

## Resolution Function

Copyright (c) 2012 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

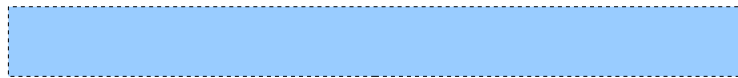
Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice and Octave.

# Resolved Signals (1)

```
entity mux is  
  port( i1, i2, a : in fourval; q : out fourval );  
end mux2;
```

architecture arch of mux is



```
signal nota : fourval;
```



```
begin  
  u1 : inv  port map (a, nota);  
  u2 : and2 port map (i1, a, intq);  
  u3 : and2 port map (i2, nota, intq);  
  q <= intq;  
end arch
```

```
component and  
  port( a, b : in fourval; c : out fourval);  
end component;  
component inv  
  port( a : in fourval; b : out fourval);  
end component;
```

```
signal intq : resolve fourval := x;
```

```
signal intq : resfour := x;
```

# Resolution Function Declaration

```
package fourpack is
  type fourval is (X, L, H, Z);
  type fourvalvector is array(natural range <>) of fourval;
  function resolve( s: fourvalvector ) return fourval;
  subtype resfour is resolve fourval;
end fourpack;
```

# Resolution Function Definition (1)

```
package body fourpack is
function resolve( s: fourvalvector) return fourval is
  variable result : fourval :=Z;
begin
  for i in s'range loop
    case result is
      when Z =>   ●   ●   ●
      when L =>   ●   ●   ●
      when H =>   ●   ●   ●
      when X =>   ●   ●   ●
    end case;
  end loop;
  return result;
end resolve;
end fourpack;
```

# Resolution Function Definition (2)

```
when Z =>  
  case s(i) is  
    when H => result := H;  
    when L => result := L;  
    when X => result := X;  
    when others => null;  
  end case;
```

```
when L =>  
  case s(i) is  
    when H => result := X;  
    when X => result := X;  
    when others => null;  
  end case;
```

```
when H =>  
  case s(i) is  
    when L => result := X;  
    when X => result := X;  
    when others => null;  
  end case;
```

```
when X =>  
  result := X;  
end case;
```

	Z	L	H	X
Z	Z	L	H	X
L	L	L	X	X
H	H	X	H	X
X	X	X	X	X

# Example (1)

package body mvl4\_pkg is



function **tristate\_rf**( v: logic4\_vector) return logic4 is

variable **result** : logic4 := 'Z';

begin

for i in v'range loop

**result** := **tristate\_rf\_table**(**result**, v(i));

exit when **result** = 'X';

end loop;

return **result**;

end **resolve**;

end mvl4\_pkg;

type logic4\_table is

array (logic4, logic4) of logic4;

Constant **tristate\_rf\_table**

: logic4\_table := (('X', 'X', 'X', 'X'),  
('X', '0', 'X', '0'),  
('X', 'X', '1', '1'),  
('X', '0', '1', 'Z') );

# Example (2)

package body mvl4\_pkg is



function **wireand\_rf**( v: logic4\_vector) return logic4 is

variable **result** : logic4 := 'Z';

begin

for i in v'range loop

**result** := **wireand\_rf\_table**(**result**, v(i));

exit when **result** = 'X';

end loop;

return **result**;

end **resolve**;

end mvl4\_pkg;

type logic4\_table is

array (logic4, logic4) of logic4;

Constant **wireand\_rf\_table**

: logic4\_table := (('X', 'X', 'X', 'X'),  
('X', '0', 'X', '0'),  
('X', 'X', '1', '1'),  
('X', '0', '1', 'Z') );



## References

- [1] <http://en.wikipedia.org/>
- [2] J. V. Spiegel, VHDL Tutorial,  
[http://www.seas.upenn.edu/~ese171/vhdl/vhdl\\_primer.html](http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html)
- [3] J. R. Armstrong, F. G. Gray, Structured Logic Design with VHDL
- [4] Z. Navabi, VHDL Analysis and Modeling of Digital Systems
- [5] D. Smith, HDL Chip Design
- [6] <http://www.csee.umbc.edu/portal/help/VHDL/stdpkg.html>
- [7] VHDL Tutorial - VHDL online [www.vhdl-online.de/tutorial/](http://www.vhdl-online.de/tutorial/)
- [8] [compgroups.net](http://compgroups.net)
- [9] J. Pick, VHDL Techniques, Experiments, and Caveats
- [10] D. L. Perry, VHDL 2<sup>nd</sup> Ed