

# Signals & Variables (3A)

---

Synthesis

Copyright (c) 2012 Young W. Lim.

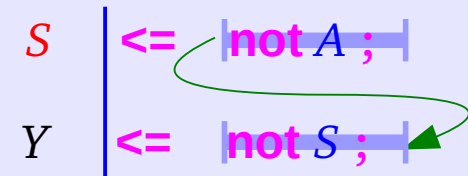
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is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

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

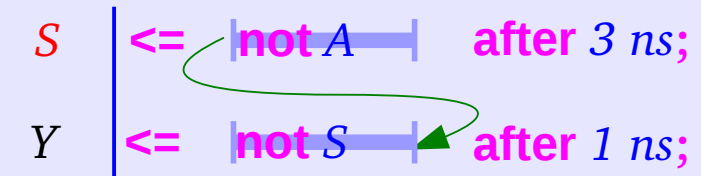
This document was produced by using OpenOffice and Octave.

# Sequential Assignment (1)

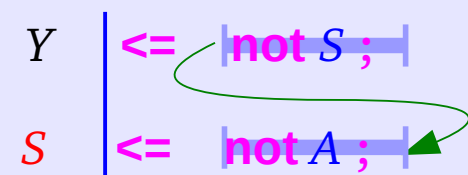
```
process (A)
  signal S: std_logic
begin
  S <= not A;
  Y <= not S;
end process;
```



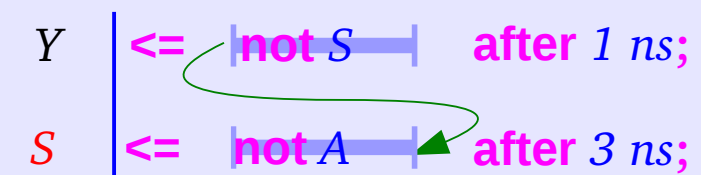
```
process (A)
  signal S: std_logic
begin
  S <= not A after 3 ns;
  Y <= not S after 1 ns;
end process;
```



```
process (A)
  signal S: std_logic
begin
  Y <= not S;
  S <= not A;
end process;
```

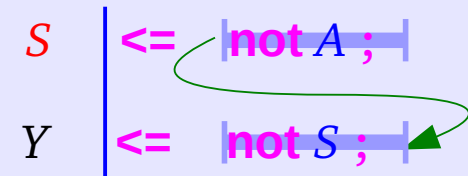


```
process (A)
  signal S: std_logic
begin
  Y <= not S after 1 ns;
  S <= not A after 3 ns;
end process;
```

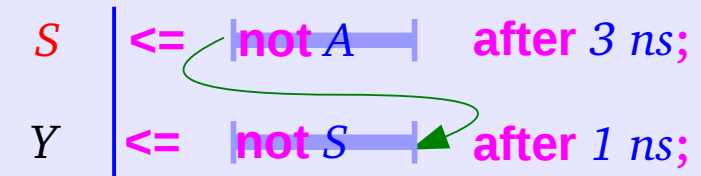


# Sequential Assignment (2)

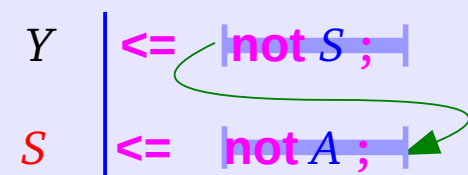
```
process (A, S)
  signal S: std_logic
begin
  S <= not A;
  Y <= not S;
end process;
```



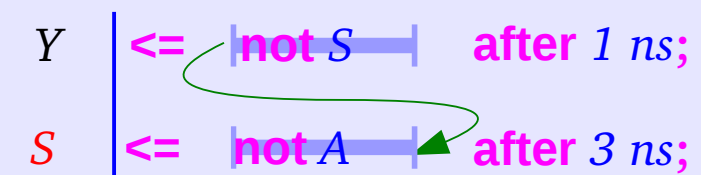
```
process (A, S)
  signal S: std_logic
begin
  S <= not A after 3 ns;
  Y <= not S after 1 ns;
end process;
```



```
process (A, S)
  signal S: std_logic
begin
  Y <= not S;
  S <= not A;
end process;
```

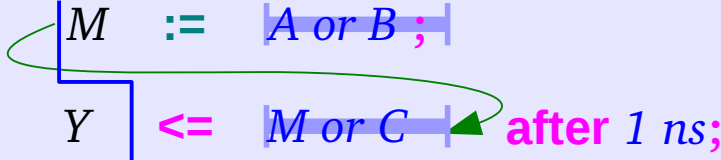


```
process (A, S)
  signal S: std_logic
begin
  Y <= not S after 1 ns;
  S <= not A after 3 ns;
end process;
```

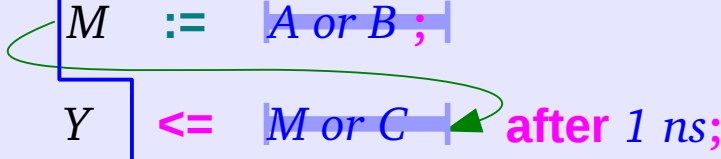


# Ex 1

```
process (A, B, C)
  variable M: std_logic
begin
  M := A or B ;
  Y <= M or C after 1 ns;
end process;
```

A diagram illustrating the execution flow of the first code block. A vertical line on the left represents the process timeline. A horizontal line from the first statement 'M := A or B ;' goes to the right, then turns down, then right again, and finally up to the second statement 'Y <= M or C after 1 ns;'. A green arrow points from the end of the second statement back to the start of the first statement, indicating a loop.

```
process (A, B, C)
  variable M: std_logic
begin
  M := A or B ;
  Y <= M or C after 1 ns;
end process;
```

A diagram illustrating the execution flow of the second code block, which is identical to the first. A vertical line on the left represents the process timeline. A horizontal line from the first statement 'M := A or B ;' goes to the right, then turns down, then right again, and finally up to the second statement 'Y <= M or C after 1 ns;'. A green arrow points from the end of the second statement back to the start of the first statement, indicating a loop.

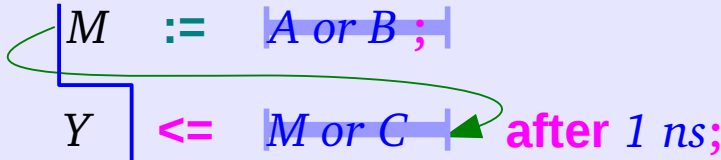
# Ex 2

```
process (A, B, C, M)
  signal M: std_logic
begin
  M <= A or B ; after 3 ns;
  Y <= M or C after 1 ns;
end process;
```

```
process (A, B, C, M)
  signal M: std_logic
begin
  M <= A or B ; after 3 ns;
  Y <= M or C after 1 ns;
end process;
```

# Ex 3

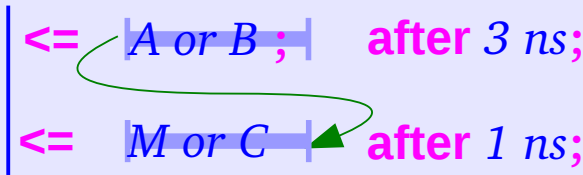
```
process (Clock)
  variable M: std_logic
begin
  if rising_edge(Clock) then
    M := A or B;
    Y <= M or C after 1 ns;
  end if;
end process;
```



end

# Ex 4

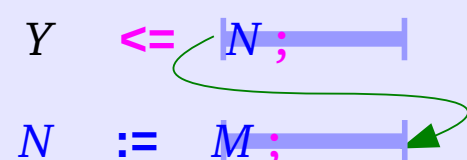
```
process (Clock)
  signal M: std_logic
begin
  if rising_edge(Clock) then
    M <= A or B ; after 3 ns;
    Y <= M or C ; after 1 ns;
  end if;
end process;
```



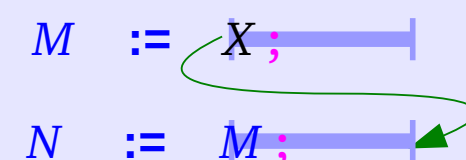


# Variable & FlipFlop

```
process (Clock)
  variable M, N: std_logic
begin
  if rising_edge(Clock) then
    Y <= N;
    N := M;
    M := X;
  end if;
end process;
```

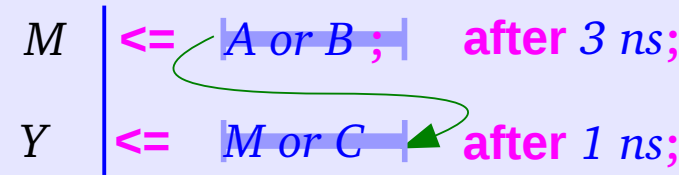


```
process (Clock)
  variable M, N: std_logic
begin
  if rising_edge(Clock) then
    M := X;
    N := M;
    Y <= N;
  end if;
end process;
```



# Ex 4

```
process (Clock)
  signal M: std_logic
begin
  if rising_edge(Clock) then
    M <= A or B ; after 3 ns;
    Y <= M or C ; after 1 ns;
  end if;
end process;
```



## 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/)