

# Tutorial – Writing Combinational and Sequential Code

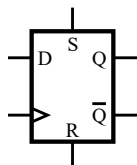
## Using VHDL Process or Verilog Always Blocks

This tutorial shows how to write blocks of either VHDL or Verilog that are contained in either a **Process** or an **Always Block** respectively. Processes (in VHDL) and Always Blocks (in Verilog) are fundamental and they need to be well understood. They behave in exactly the same way, so both are introduced here for you, if you are learning just one language right now, pay attention to the examples focused at that particular language. Processes or Always Blocks are used in two main scenarios:

1. To define a block of **combinational** logic
2. To define a block of **sequential** logic

The first scenario is what is commonly seen in textbooks when introducing Processes or Always Blocks to a new student. It is presented here for you, to make you aware of its existence. But in reality, a Process/Always Block used to define a block of combinational logic is seen far less often in “real-world” code than a Process/Always Block used to define sequential logic.

The first question you might be asking yourself is what is the difference between combinational and sequential logic? Combinational (or combinatorial) logic is logic that does not require a clock to operate. The example of the and-gate previously is a combinational example. Sequential logic is logic that does require a clock to operate. The most fundamental building block of sequential logic is the D Flip-Flop (pictured below).



The D Flip-Flop!

If you are unaware of how a D Flip-Flop works, stop reading this immediately! You need to understand [how flip-flops are used inside FPGAs](#) before you continue reading. All set? Good.

## Combinational process in VHDL:

## Learn Verilog

Verilog Tutorials	^
Introduction To Verilog for beginners with code examples	
Always Blocks for beginners	
Introduction to Modelsim for beginners	
Your First Verilog Program: An LED Blinker	
Recommended Coding Style for Verilog	
Verilog Reserved Words (Keywords)	v

## Modules

Verilog & VHDL Modules	v
------------------------	---

## Learn VHDL

VHDL Tutorials	v
VHDL Reserved Words (Keywords)	v

```

1 | process (input_1, input_2)
2 |   begin
3 |     and_gate <= input_1 and input_2;
4 |   end process;

```

## Combinational Always Block in Verilog:

```

1 | always @ (input_1 or input_2)
2 |   begin
3 |     and_gate = input_1 & input_2;
4 |   end

```

In the both the VHDL and Verilog code above, input\_1 and input\_2 are in what is called a **sensitivity list**. The sensitivity list is a list of all of the signals that will cause the Process/Always Block to execute. In the example above, a change on either input\_1 or input\_2 will cause the Process/Always Block to execute. This process/always block takes the two inputs, perform an “and” operation on them, and stores the result in the signal and\_gate. This is the exact same functionality as this code:

```

1 | -- VHDL:
2 | and_gate <= input_1 and input_2;

1 | // Verilog:
2 | assign and_gate = input_1 & input_2;

```

Both examples of code serve the same purpose: to assign the signal and\_gate. The difference is that one is in a combinational Process/Always Block and the other is not. Therefore since the same result can be achieved without the use of a combinational Process/Always Block, I do not recommend that the beginning digital designer use these statements in this way.

The second way that a Process or Always Block can be used (and the far more interesting example) is to define a block of sequential logic. Again, sequential logic is logic that is clocked.

### Using Process/Always Blocks for Sequential Code

## 3 Comments



**Teccho** January 13, 2023 at 9:19 pm - Reply

The top ad bar makes reading the article impossible! Please fix it, this is by far the best tutorial I've found and you deserve people to use it.



**Eric** January 17, 2023 at 10:41 pm - Reply

Very good intro and it is a great idea to discuss VHDL and Verilog simultaneously! In fact, it seems you have performed a functional simulation and to see the effect of propagation delay, a timing simulation would be required.



**MinhUyen** February 25, 2024 at 3:22 am - Reply

Sequential logic is logic that does require a clock to operate. How about a D-Latch circuit? I think Sequential logic is logic requires a feedback signal.

## Leave A Comment

Comment...

Name (required)

Email (required)



I'm not a robot

reCAPTCHA  
[Privacy](#) - [Terms](#)

Post Comment

# NANDLAND



[ORDER NOW »](#)

© 2022. Content cannot be re-hosted without author's permission. ([contact me](#)) web design by DS