



[Store](#) [Blog](#) [Forum](#) [Projects](#)  
[Documentation](#)

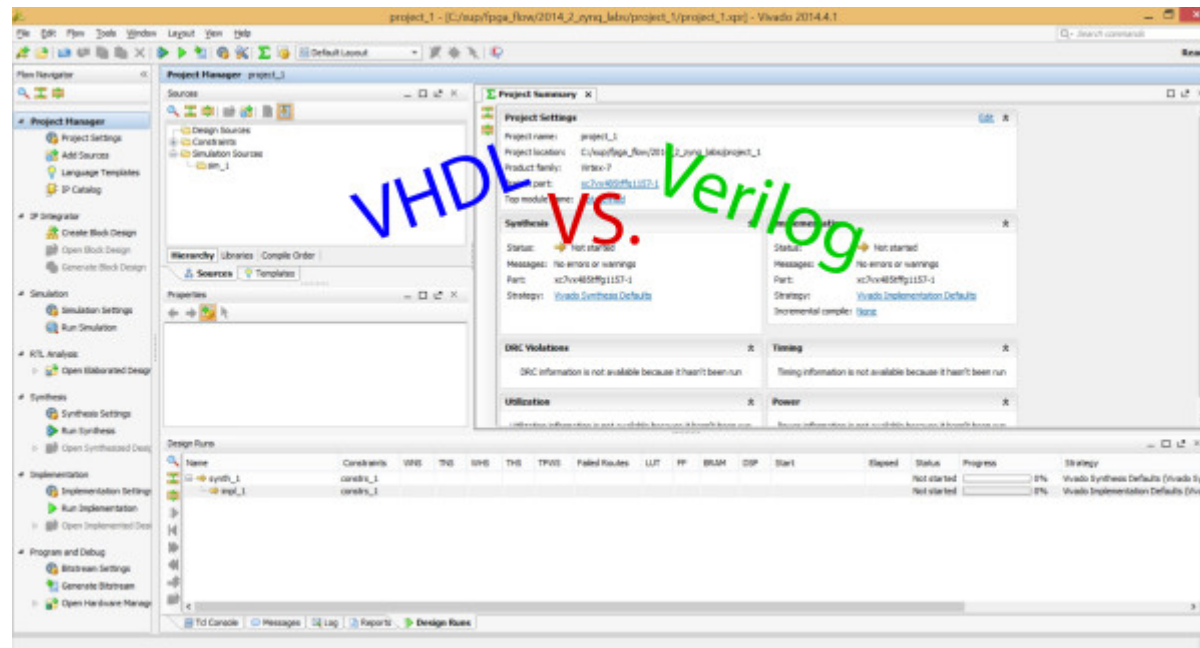
## GUIDE

# Battle Over the FPGA: VHDL vs Verilog! Who is the True Champ?

- by Robert Prew - 10 Comments.

VHDL:	Verilog:
<pre>2 process ({S0,S1},A,B,C,D) 3 begin 4     case {S0,S1}, is 5         when "00" =&gt; Y &lt;= A; 6         when "01" =&gt; Y &lt;= B; 7         when "10" =&gt; Y &lt;= C; 8         when "11" =&gt; Y &lt;= D; 9         when others =&gt; Y &lt;= A; 10    end case; 11 end process;</pre>	<pre>1 2 3 always @({S0,S1}, A, B, C, D) 4     case ({S0,S1}) 5         2'b00: Y = A; 6         2'b01: Y = B; 7         2'b10: Y = C; 8         2'b11: Y = D; 9     endcase 10</pre>

With the intense tune of *Eye of the Tiger* playing in the background, it is time for this showdown to begin. In the left corner we have [Verilog](#). Born in 1984, it didn't make it to the big scene until 1995. In the right corner we have [VHDL](#), the language created by the US Department of Defense in the 1980's.



Many people have different opinions on which language is better, but it really comes down to which language you prefer. I should also mention that there is of course System Verilog, but it is very closely related to Verilog so we'll just leave that for another day. Let's take a look at these languages and see what the differences are.

VHDL stands for VHSIC Hardware Description Language and VHSIC stands for Very High Speed Integrated Circuit. So on the whole VHDL actually stands for Very High Speed Integrated Circuit Hardware Description Language. Now, that's a mouthful if I ever saw one! One of the key features of VHDL is that it is a strongly typed language, which means that each data type (integer, character, or etc.) has been predefined by the language itself. All values or variables defined in this language must be described by one of the data types.

VHDL is more verbose than Verilog and it also has a non-C like syntax. With VHDL, you have a higher chance of writing more lines of code. VHDL can also just seem more natural to use at times.

When you're coding a program with VHDL, it can seem to flow better. But maybe that's just my personal opinion.

In Verilog, the language is more compact, as the Verilog language is more of a hardware modeling language. You will end up typing few lines of code and it draws similarities to the C language. Verilog has a better grasp on hardware modeling, but has a lower level of programming constructs. Verilog is not as verbose as VHDL so that's why it's more compact. All in all, Verilog is pretty different from VHDL. There are some similarities, but they are overshadowed by their differences.

Looking at this example code, we can compare at the how a MUX can be programmed through VHDL and Verilog.

VHDL:	Verilog:
2 process ((S0,S1),A,B,C,D)	1
3 begin	2
4 case {S0,S1}, is	3 always @((S0,S1), A, B, C, D)
5 when "00" => Y <= A;	4 case ((S0,S1))
6 when "01" => Y <= B;	5 2'b00: Y = A;
7 when "10" => Y <= C;	6 2'b01: Y = B;
8 when "11" => Y <= D;	7 2'b10: Y = C;
9 when others => Y <= A;	8 2'b11: Y = D;
10 end case;	9 endcase
11 end process;	10


The layout of these programs are very similar; you can reasonably follow what each version of the code is doing. The VHDL version is longer than the Verilog, but it can be understood better.

The last match of this competition is something that can't be decided easily... it's the personal preference challenge. This rides on the readers. Weigh in on why you like each programming language, and which one you prefer the most. This is a good opportunity to share some insight on to

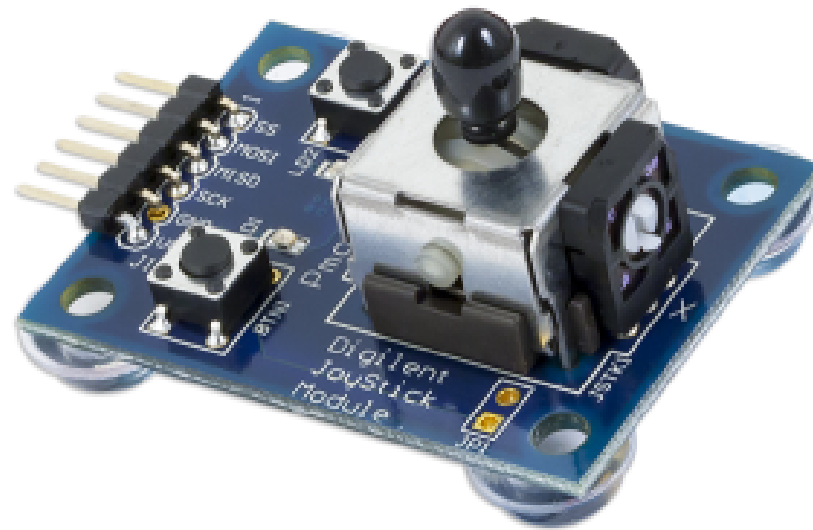
what beginners should look out for when starting to pick up VHDL or Verilog. If you want to see either language in action, check out some of the learn projects we have on [VHDL](#) and [Verilog](#)!

Also check out some of our popular projects that use each language. Kaitlyn, a longtime supporter of Verilog details on Instructables some of the awesome things you can do with this powerful tool.

This includes controlling servos on your FPGA! [Here](#) she demonstrates how she used the [Basys 3](#) and Verilog to control the motion required for the [Claw Game Demo](#).

 Picture of Put all the Pieces Together

However despite a strong showing from Verilog, VHDL is not so quick to be outdone. [Here](#) you can see it in action being used to program the popular [PmodJSTK](#).



This Pmod can then be used for a variety of applications...some even battle-related.

 Picture of Jousting Robot (Wiring Tutorial)

Check out more on the PmodJSTK powered Joustier [here!](#)

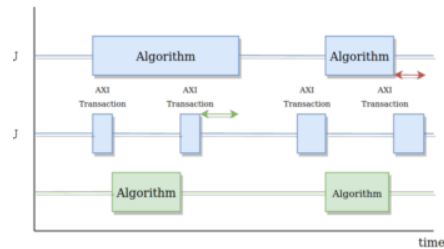
+4  
0

TAGGED BASYS3 BETTER C++ COMPARISON OPINION PMODJSTK SHOWDOWN VERILY VHDL

## RELATED POSTS



Navigating the Difference  
Between Pmod, SYZYGY,  
and FMC



Custom AXI for IP  
Acceleration using Petalinux



Creating a Genesys ZU Vitis  
Acceleration Platform

[PREVIOUS ARTICLE](#)[NEXT ARTICLE](#)[Aaaand We're Off! The Pmod Racing Ruler  
Roars into Action!](#)[New Video: Digilent Forum Tips, Tricks and Best  
Practices!](#)

### About Robert Prew

I am studying Electrical Engineer at the University of Idaho. I have always been interested in technology, video games, science and learning new things. I always believed in the saying “if you can't explain it simply you don't understand it well enough” by Albert Einstein. I have a thirst for self discovery and learning new things about myself. Bringing joy or making others feel happy gives me a great pleasure. You can usually get me interested in an activity that has a more hands on approach. I will always try and strive to be the best person I can be.

[View all posts by Robert Prew →](#)

## 10 Comments on “Battle Over the FPGA: VHDL vs Verilog! Who is the True Champ?”



**James Colvin** at 2:48 pm

I was a little confused at first when I read your statement that while you are using VHDL you will have a tendency to write more lines of code but have it seem to flow better since that's somewhat counter intuitive—instead of being quick and to the point, being long-winded is better?

But knowing that the reverse tends to be true (more lines of code equating to more clarity) for coding and looking that the VHDL to Verilog comparison for a MUX, I can definitely see how the VHDL makes a little more sense, even to the untrained eye (at least I think it does).

REPLY



**Gena Djiga** at 2:46 pm

hey,

i am my self mostly programming in VHDL.

I would like to point out that the MUX example showed in the article, in Verilog can be very similar to VHDL only by adding 2 more lines that are not most in the example, BUT CAN BE(!!)

always@(...)

begin//the first line, like 'begin' in VHDL

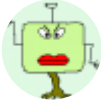
.

.

.

end // like 'end process' in VHDL.

REPLY



**Ludwig Schreier** at 10:45 am

Hi,  
thanks for the article.

You might also want to read the article from Mr Gaisler called 'A structured VHDL design method'

(<http://www.gaisler.com/doc/vhdl2proc.pdf>)

Mr Gaisler is not comparing the two languages, rather proposing a design methods compared to 'dataflow' design for VHDL which is interesting.

Kind regards,  
LS

REPLY



**Sid Ray** at 5:44 pm

I have used both. I am one of the few who prefers VHDL over verilog. VHDL provides more in when you are creating generic designs compared to Verilog. Please don't argue on this point because it is a fact. Verilog has some strange quirks that individual vendors try to correct in their implementations. VHDL is better defined and you are less likely to get bitten because you understood something wrong. It just is better



defined than Verilog. In fact Systemverilog was created to make Verilog like VHDL. Instead SystemVerilog is a hodge-podge mess.

REPLY



**Joachim** at 12:05 am

In the mux example, in Verilog you should write “always @\*”. This gives you two things:

1. The inputs to the process are implicit and allows the tools to find them. No more risk of missing stating an input causing latches.
2. You indicate to the tool(s) that the process is combinational. This makes it easier for the tool to find problems and warn you.

This concept was further developed in SystemVerilog into the “always\_comb” and “always\_ff” statements.

REPLY



**Joachim** at 12:08 am

Since you are talking about syntax. It might be worth pointing out that VHDL is case insensitive. Which makes it the more weirder that it is basically a standard in VHDL to write statements in all capital letters. This makes the code harder to read.

REPLY

---



**Oleksandr** at 2:05 am

Fix your VHDL example code.

VHDL does not have {A, B} syntax. In the process sensitivity list you can use just (S0, S1, A, B, C, D).

The concatenation would look like S0 & S1. However, you cannot use it in the case statement. Create a variable before, say S, and assign it the value S := S0 & S1. Then do “case S is”.

REPLY

---



**Steve VanderLeest** at 6:23 pm

We have a reference app that provides all the keywords of both VHDL and Verilog, notes the equivalent in the other language (when they exist), and gives a simple usage example. <http://www.squishlogic.com/VHDLRef/>

REPLY

---



**Matt Davies** at 4:20 am

I have to completely disagree that VHDL reads better than Verilog in your MUX example. There is double the punctuation.

REPLY

---

Pingback: [Verilog or VHDL? – FPGA Coding](#)

---

## Leave a Reply

Your email address will not be published. Required fields are marked \*

COMMENT

NAME \*

EMAIL \*

WEBSITE

POST COMMENT

Search ...

SEARCH

## RECENT COMMENTS

The SYZYGY Origin Story – Digilent Blog on Embracing a New Standard

Zoinks! What's a Zmod? – Digilent Blog on Embracing a New Standard

olivia william on Display Tech – OLED Displays

Digilent's 2020 Holiday Gift Guide – Digilent Blog on Five Projects for Your Arty S7

Valmik Patel on Introducing the PmodDPG1 – Differential Pressure Gauge

## CATEGORIES

Select Category



## **SUBSCRIBE**

Email\*

SUBMIT

## **EXPLORE MORE POSTS**

Select Month



### [Our Partners](#)

Xilinx University Program  
Technology Partners  
Distributors

### [Help](#)

Technical Support Forum  
Reference Wiki  
Contact Us

## Customer Info

Videos

FAQ

Store Info

## Company Info

About Us

Shipping & Returns

Legal

Jobs

Internships

## Connect With Us



Copyright © 2021 Digilent Blog.

---