PyVHDL Documentation

Release 0.0.1

Vern Muhr

October 26, 2016

Contents

1	I Introduction:	1
	2 Documentation Contents:	3
	2.1 Quickstart	 3
	2.2 Write a Python testbench	 11
	2.3 License	 12
3	3 Indices and tables	13

Introduction:

PyVHDL is an open source project for simulating VHDL hardware designs. It cleanly integrates the general purpose Python programming language with the specialized VHDL hardware description language. You can write the testbench for your VHDL design in Python. You can use all the great features of Python to quickly create your testbench. Some advantages of using Python include:

- It's an easy to learn, modern mainstream language.
- Object oriented for better organization of code.
- Simple and powerful formatting of of text output.
- Easy file read and write of formatted data.
- Quality modules available for just about any programming task.

Python can be used for more than just testbenches. You can also create architectures in your VHDL design that are written in Python. Standard VHDL syntax is used to link the architecture to a Python file. Concurrent Python processes can wait for signal transitions, wait for a specified period of time, read values of signals in the design, or update the values of signals with an optional delay. This capability is useful for quickly prototyping the behavior of an entity, or emulating complex behavior like a memory device or UART.

PyVHDL features a unified execution environment. VHDL is translated to Python bytecode, and runs on the same interpreter as the Python code. There is no overhead wasted on passing data and maintaining synchronization between separate Python and VHDL environments. An advantage of using Python is that PyVHDL can utilize the work of other projects in the very active Python community. For example, using the Cython optimising compiler, the speed of the simulator event queue can be improved by more than a factor of seven.

An important goal of PyVHDL is to create a VHDL simulator that the open source community can easily make contributions to. The simulator is written in Python, a language familiar to many programmers. Python is much easier to read and modify compared to most languages. Also, the hurdle of setting up an obscure tool chain environment is not an issue. The PyVHDL project is maintained on GitHub.

The excellent zamiaCAD Eclipse based IDE is packaged with PyVHDL. The zamiaCAD IDE is an open source Eclipse based platform for advanced VHDL hardware design. PyVHDL is tightly integrated with the zamiaCAD IDE. Features provided by zamiaCAD include:

- An environment for creating and managing PyVHDL projects.
- VHDL editing with syntax highlighting and checking.
- Graphical tools for visualizing and navigating the VHDL hierarchy.
- Configuration and running of simulations.
- A waveform viewer for displaying the results of a simulation.
- A VHDL parser used by PyVHDL for analysis and elaboration.

	Window Help										
3 • 🛛 🖻 🚔 🛛 • 🔗 •	🖗 • 🖓 • 🗫 🔇	• = = =								Quick Access	😰 🖉 za
🕈 zamiaCAD Navigator 🛛 🛛 🔳	3 🔩 🔻 🗖 🚺	peripheral.py	reg_bank.vhd	🔛 py_periphera 🛛	plasma_ram.vt	d ³³ 1	- 0	🖳 Console 🛛		R.	🚮 📑 🗉 - 📬
) (E 11 11 11 11 11 11 11 11 11 11 11 11 11	9 DESCRIP Impl Pyth Pyth Pyth Pyth 	ware 'as is' w' 'iTON: lements the RAM ion FLI module. see; itd_logic_1164.; peripheral is lk : on set : of :ity py_peripher ure fli of py_pu NUTE foreign : :	nt std_logic; nt std_logic ral eripheral <mark>is</mark>	ug port, clk	e tor noth	-	May 20 11:06:53 May 20 11:06:53	<pre>www.MEM_BYTE; www.MEM_BYTE; www.MEMORY RE www.MEMORY RE www.MEMORY RE www.MEMORY RE www.MEMORY RE www.MEMORY RE www.MEMORY RE www.MEMORY RE simulation don Elapsed time = simulation tim</pre>	SEL: 1000 e: 0A AD: value 0X34 0000000 fr zer 0000000 fr zer 0000000 fr sdu ND: value 0X62 to v0 @ 332 SEL: 1000 AD: value 0X06 AD: value 0X0	102001C, addres o @ 33125.00 @ 33125.00 8200001, addres 225.00 ns 80000E9, addres 00000E9, addres 9000000, addres 9000000, addres 9000000, addres 90000E9, addres
	o o ø ø m: 🗄 🔬		12 21	🗶 @ @ @ @							
ignal Value	300	1900	2000	2100	22	00	2300	2400	2500	2600	2700
							<u> </u>		· · · · · ·		
WRITE ENABLE 0					Variation						
	0000" X"A295	0000"	X"000000C X 34020	06 XX"A2820000"	XX''34	21DI XX"340300		XX 0043001 XX	2401FFF XX*14610004"	X"3C01800	XX.00001015.
WRITE_ENABLE 0	0000" X"A295 X"0"	50000" \{X"1"	X"000000C XX"34020 XX"0"	06 XX"A2820000"	X"34 X"1" X"0"	021D[X"340300	J7XX-14600002-	χχ-0043001 χχ-	2401FFFXX"14610004"	XX"3C01800	XX-00001012"
WRITE_ENABLE 0 OPCODE_REG (31 downto 0) X'0000			Xx*000000CXx*34020 Xx*0*	06 <u>X</u> "A2820000"		021D[X"340300	// <u>///////////////////////////////////</u>	χx 0043001 χx	2401FFF X * 14610004" (X * 0000003 (X		X× 00001012
WRITE_ENABLE 0 OPCODE_REG (31 downto 0) X"0000 MEM_STATE_REG (1 downto 0) X"0"			X"000000CXX"3402C XX"0"	06 <u>X</u> X"A2820000"		021DI XX''340300	// <u>/</u> /////////////////////////////////	<u>χ</u> χ -0043001 <u>χ</u> χ	~		~
WRITE_ENABLE 0 OPCODE_REG (31 downto 0) X'00000 MEM_STATE_REG (1 downto 0) X'0" REG_B (63 downto 0) X'00000			<u>X"000000C XX"34020</u> XX"0"	06 XX"A2820000"		021DLXX"340300	// <u>///////////////////////////////////</u>	<u>XX 0043001 XX</u>	~		~
WRITE_ENABLE 0 OPCODE_REG (31 downto 0) X"0000 MEM_STATE_REG (1 downto 0) X"0" REG_B (31 downto 0) X"0000 REG_A (31 downto 0) X"0000	00000" X"0000 X"0000		<u>X:'0000000 X:'34020</u> X:''0'' X:''0''	06 <u>(</u> X"A2820000" (X"D"		021D(XX''340300	<u>77 XX 14600002"</u>	<u>X* 0043001 X*</u>	X"0000003)(X X"00001DDE"		<u>}{x:0000000</u> {x:000000
WRITE_ENABLE 0 OPCODE_REG (31 downto 0) X'00000 MEM_STATE_REG (1 downto 0) X'0' REG_B (63 downto 0) X'0000 REGA (31 downto 0) X'0000 COUNT_REG (5 downto 0) X'00'	00000000000000000000000000000000000000		_X:"0000000 XX"34020 XX"0" XX"0"		<u>)(x·1·)(x·0·</u>)/ <u>XX 14600002"</u> Di}X::0000007}		X"0000003)(X X"00001DDE"		<u>}{x:0000000</u> {x:000000
WRITE_ENABLE D OPCODE_REG (31 downto 0) X*0000 MEM_STATE_REG (1 downto 0) X*0'* REG_B (63 downto 0) X*0000 REG_A (31 downto 0) X*0000 COUNT_REG (5 downto 0) X*000 COUNT_REG (5 downto 0) X*00* MEM_SOURCE (3 downto 0) X*00*	00000" X*0000 X*000 X*000 X*000 X*0000 X*0000 X*0000 X*0000 X*0000 X*0000	00000"	X**0000000 X**34020 X**0* X**0* X**0*	X:"D" XX:"D"	<u>)(x:1,)(x:0,</u>)(x:1,)(x:0,		DI { X:'0000007 }/	(*)00000008" XX"	<mark>Хх°0000003)</mark> (х Хх°00001DDE" Хх°00" Х х	"0000001)(X"0000000 "01" (X"02" "00000010"	X::0000000:X::000000 X::03* X::04*
WRITE_ENABLE 0 OPCODE_REG (31 downte 0) X*0000 MEM_STATE_REG (1 downte 0) X*0" REG 8 (43 downte 0) X*0000 COUNT_REG (5 downte 0) X*00" COUNT_REG (5 downte 0) X*00" MEM_STATE_COURCE (3 downte 0) X*00" REJSURG (3 downte 0) X*00"	00000" X"00" X"0000 X"0000 X"0000 X"0000 X"0000 X"0000	00000"	<u>Xx"0"</u> XX"0"	<u>X*D*</u> <u>X*D*</u>	<u>)(x:1,)(x:0,</u>)(x:1,)(x:0,	X×"00001E	DI { X:'0000007 }/	(*)00000008" XX"	<u>Х*'0000003)(</u>))/X*'00001DDE'' <u>Х*'00'' Х</u> Х	"0000001)(X"0000000 "01" (X"02" "00000010"	(<u>), x., 0000000 (), x., 00000</u> (), x., 003. (), X., 04. (), X., 04. (), X., 00000
OPTER 0 OPCODE_REG (31 downto 0) X'0000 MEM_STATE_REG (1 downto 0) X'0' REG # (63 downto 0) X'0' REG # (63 downto 0) X'0000 COUNT_REG (5 downto 0) X'0000 REM_STATE_G (3 downto 0) X'000 REM_SUBC (3 downto 0) X'0' R_BUSO (31 downto 0) X'000 X_0000 (31 downto 0) X'0000	00000" X"00" X"0000 X"0000 X"0000 X"0000 X"0000 X"0000	00000"	<u>Xx"0"</u> XX"0"	<u>X*D*</u> <u>X*D*</u>	<u>)(x:1,)(x:0,</u>)(x:1,)(x:0,	X×"00001E	DI { X:'0000007 }/	(*)00000008" XX"	<u>Х*'0000003)(</u>))/X*'00001DDE'' <u>Х*'00'' Х</u> Х	"0000001)(X"0000000 "01" (X"02" "00000010"	(<u>), x., 0000000 (), x., 00000</u> (), x., 003. (), X., 04. (), X., 04. (), X., 00000
WRITE_ENABLE 0 OPC:ODE_REG (31 downto 0) X*0000 MEM_STATE_REG (1 downto 0) X*0" REG_8 (43 downto 0) X*0000 COUNT_REG (5 downto 0) X*00" MEM_STATE_REG (1 downto 0) X*00" REG_8 (3 downto 0) X*00" REM_STATE_G (3 downto 0) X*00" All Monte_SOURCE (3 downto 0) X*00" A_BUSO (31 downto 0) X*000 REISET 1	x*0* x*0000000000* x*0000 x*000 x*000 x*000 x*000 x*000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*00000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*0000 x*00000 x*00000 x*00000 x*00000 x*00000 x*00000 x*00000 x*00000000 x*0000000000	00000"	<u>Х</u> х:0" Хх:00 Д	<u>X*D*</u> <u>X*D*</u>	<u>)(x:1,)(x:0,</u>)(x:1,)(x:0,	<u>X*'00001</u> X*'000000	DI { X:'0000007 }/	(*)00000008" XX"	<u>Х*'0000003)(</u>))/X*'00001DDE'' <u>Х*'00'' Х</u> Х	"0000001)(X"0000000 "01" (X"02" "00000010"	(<u>), x., 0000000 (), x., 00000</u> (), x., 003. (), X., 04. (), X., 04. (), X., 00000
WHTE ENABLE 0 OPCC0FE_ENABLE 0 OPCC0FE_ERG { 31 downto 0} X°00° MEM_STATE_REG { 1 downto 0} X°00° REG_B { 43 downto 0} X°00° COUNT_REG { 54 downto 0} X°00° COUNT_REG { 54 downto 0} X°00° REM_SUBJECT { downto 0} X°00° REM_SUBJECT { downto 0} X°00° R_BUS0 { 31 downto 0} X°000° R_SUST 1 MEM_WWTE 0	x*0* x*00000000000000000000000000000000	X*'1" 0000000000000" 00000" 00000" XX'0A0A0	<u>Х</u> х:0" Хх:00 Д	X*D* X*000000£X*000000 000* X*200000	<u>) (x:1:</u>) (x:0:)(x:0:)(x:6565656 (x:00)	X*'00000 X*'000000	DI { X:'0000007 }/	C'0000008° XX° C'00000040° XX°	<u>Х*'0000003)(</u> X)X*'00001DDE'' <u>Х*'00'' Х</u> Х	**************************************	(<u>), x., 0000000 (), x., 00000</u> (), x., 003. (), X., 04. (), X., 04. (), X., 00000
WRITE_ENABLE 0 OPC:ODE_REG (31 downto 0) X*0000 MEM_STATE_REG (1 downto 0) X*0" REG_8 (43 downto 0) X*0" REG_4 (31 downto 0) X*0000 COUNT,REG (5 downto 0) X*000 MEM_STATE_C 3 downto 0) X*0" RESS (21 downto 0) X*0" RESS (21 downto 0) X*0000 RESS (21 downto 0) X*0000 RESS (21 downto 0) X*0000 MEM_OATA_W (31 downto 0) X*0000 X*0000 X*0000	x*0" x*0000000000000 x*0000 x*000 x*000 x*000 x*000 x*2000 x*2000 x*2000 x*2000 x*2000 x*2000 x*2000	X*'1" 0000000000000" 00000" 00000" XX'0A0A0)xro" xro" xrooooooooooooooooooooooooooooo	X*D* X*000000£X*000000 000* X*200000	<u>) (x:1:</u>) (x:0:)(x:0:)(x:6565656 (x:00)	X*'00000 X*'000000	DIX**0000007X 000***X	C'0000008° XX° C'00000040° XX°	(x*0000003)(X /x*0001101e* (x*00*)(X 00000007)(x*FFFFFF)(X 00000101)(x*0000000)(x	**************************************	[X="0000000"] X="000000" X="0000000" X="00000000" X="00000000"
WRITE_ENABLE 0 OPC:ODE_REG (31 downto 0) X*0000 MEM_STATE_REG (1 downto 0) X*0" REG_8 (43 downto 0) X*0" REG_4 (31 downto 0) X*0000 COUNT,REG (5 downto 0) X*000 MEM_STATE_C 3 downto 0) X*0" RESS (21 downto 0) X*0" RESS (21 downto 0) X*0000 RESS (21 downto 0) X*0000 RESS (21 downto 0) X*0000 MEM_OATA_W (31 downto 0) X*0000 X*0000 X*0000	X.0. X.0. 0000. X.000 X.000 X.0000 X.000 X.0000	Xx.8. Xx.7. 00000, 0000, 0000, Xx.090900 0000, Xx.09090 Xx.09090 Xx.09090 Xx.8.)xro" xro" xrooooooooooooooooooooooooooooo	X:"D" X:"000000 X:"000000 0000" X:"200000 000 X:"34021DDE"	<u>) (x:1:</u>) (x:0:)(x:0:)(x:6565656 (x:00)	X**00001E X**000000 000000** 13007 X**146000	DI <u>X</u> *'0000007 <u>X</u> 1000'' X 1000'' X	//////////////////////////////////////	(x*0000003)(X /x*0001101e* (x*00*)(X 00000007)(x*FFFFFF)(X 00000101)(x*0000000)(x	x	[X="0000000"] X="000000" X="0000000" X="00000000" X="00000000"
WRITE SHARLE 0 OPC.ODE.REG (31 downte 0) X'0000 MEM_STATE.REG (1 downte 0) X'0' REG_8 (43 downte 0) X'0' REG_8 (43 downte 0) X'0' NEM_STATE.REG (1 downte 0) X'0' REG_8 (3 downte 0) X'0' REM_STATE.G. (3 downte 0) X'0' REMSD (31 downte 0) X'0' RESET 1 MEM_VATE 0 MEM_VATE 0 MEM_VATE 0 MEM_VATE X'0000 MEM_OTATAK (31 downte 0) X'0000 MEM_VATE, (3 (31 downte 0)) X'0000 MEM_VATE, (31 downte 0) X'0000	x '0' x '0' x '00' x '0' x '	Xx.8. Xx.7. 00000, 0000, 0000, Xx.090900 0000, Xx.09090 Xx.09090 Xx.09090 Xx.8.	Xx*00* Xx*00000 Xx*0000000* Xx*340200eXx*A2822 Xx*0*	X:"D" X:"000000 X:"000000 0000" X:"200000 000 X:"34021DDE")(X'1")(X'0"))(X'0"))(X'0"))(X'6565656)(X''00))(X'8")(X'34) (X'8")(X'0")	X**00001E X**000000 000000** 13007 X**146000	DI <u>X</u> *'0000007 <u>X</u> 1000'' X 1000'' X	//////////////////////////////////////	X**0000003 (X X**0001 DDE X**00*** XX 0000007 X**FFFFF (X 000001 D1 X**0000000 (X 000000 D1 X**0000000 (X 1461000 (X**3C018000**	x	X*************************************

Fig. 1.1: The zamiaCAD+PyVHDL IDE

Caution: PyVHDL 0.0.1 is an alpha release! Not all VHDL syntax is supported. Un-supported VHDL syntax includes:

- Functions and Procedures.
- Type statements.
- Arrays and structures.
- Libraries written in VHDL (std_logic and std_logic_vectors are supported natively in Python).

Despite these limitations, PyVHDL can simulate the 32-bit, MIPs compatible, plasma CPU design, and pass plasma's instruction set test suite!

Documentation Contents:

2.1 Quickstart

Quickstart table of contents:

• Quickstart

- Requirements
- Install PyVHDL
- zamiaCAD
 - * Initial zamiaCAD setup
- Run the plasma demo
 - * Setup a new project
 - * Import the archived project
 - * Configure a simulation
 - * Run the simulation

2.1.1 Requirements

Note: At this time, zamiaCAD+PyVHDL has been tested on Windows 7 and Windows 10.

- Java Runtime Environment version 1.8. Install the 64 bit version of the JRE.
- Stackless Python 2.7 choose the amd64 version.

Note: Be sure python.exe in on your path.

2.1.2 Install PyVHDL

Download PyVHDL-0.0.1.zip from GitHub here. Extract the contents of the zip file to a location of your choice. I use C:/Projects/PyVHDL

2.1.3 zamiaCAD

As noted above, PyVHDL runs alongside the zamiaCAD Eclipse IDE. zamiaCAD provides an environment for creating VHDL design projects, editing VHDL files, analyzing and elaborating a VHDL design, controlling a simulation, and displaying waveform files generated by a simulation. If you have used the Eclipse framework before, you will feel fairly at home using zamiaCAD. zamiaCAD has many features that facilitate working with VHDL designs. Read this zamiaCAD tutorial to learn more about zamia-CAD's features.

Zamiacad is split into an IDE component, and a VHDL processing component. It is possible to run simulations without the IDE in command line mode using just the VHDL processing component.

Initial zamiaCAD setup

Navigate to the **PyVHDL-0.0.1** directory. Double click on **run_giu.bat**. The zamiaCAD+PyVHDL splash screen should appear:

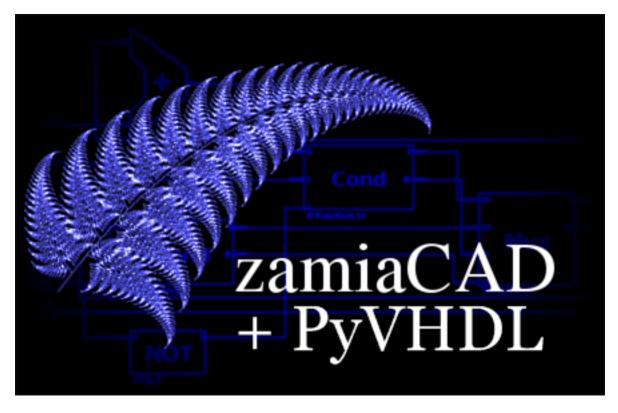


Fig. 2.1: zamiaCAD+PyVHDL Splash Screen

Next a **Workspace Launcher** dialog should appear. Accept the default workspace that appears in the Workspace textbox. It should be a folder named workspace in the PyVHDL installation folder:

Click on the Use this as the default and do not ask again checkbox, then click on OK. The initial zamia-CAD+PyVHDL IDE should appear.

Workspace	Launcher	×
	kspace ores your projects in a folder called a workspace. rkspace folder to use for this session.	
<u>W</u> orkspace:	G:\Test\PyVHDL-0.0.1\workspace	▼ <u>B</u> rowse
🔲 <u>U</u> se this as	the default and do not ask again	
		OK Cancel

Fig. 2.2: Workspace Launcher Dialog

💉 zamiaCAD - zamiaCAD - G:\Test\PyVHI	DL-0.0.1\workspace				a state of the		• X
File Edit Navigate Search Project	Run Window Help						
▶	<mark>∻ •</mark> ☆ • ☆ • ☆ • ♂			Quick Ac	cess	😫 🌠 :	zamiaCAD
💉 zamiaCAD Navigator 🛛 🗖 🗖					🗄 Outline 🖾		
□ 🕏 🎽					An outline is not	t <mark>availabl</mark> e.	
	📳 Problems 🖾 🌈 RTL Graph Viewer 🛒 Sir	nulator 📮 Console	🔗 Search 🔤 Proc	gress			~
	0 items						
	Description	Resource	Path	Locatio	on Type		

Fig. 2.3: Initial zamiaCAD+PyVHDL IDE

2.1.4 Run the plasma demo

The plasma demo is a zamiaCAD+PyVHDL project for simulating the plasma CPU design. Plasma is a 32-bit MIPs compatible processor. The design is a set of RTL style VHDL files.

Setup a new project

Before the plasma project can be simulated, a new project must be setup in the zamiaCAD Navigator view. Make sure the zamiaCAD perspective is active. It should appear highlighted in the upper right corner of the IDE window. On the top menu select **File > New > zamiacad Project**. The **Create New zamiaCAD Project** dialog will appear.

r		
	roject name, saving directory and its toplevel entity	
Project nan	ne: plasma	
	ault location	
Location:	G:\Test\PyVHDL-0.0.1\workspace\plasma	Browse
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> ir	nish Cancel

Fig. 2.4: Create New zamiaCAD Project Dialog

In the Project name textbox enter plasma. Click Finish. A **Do Full Build?** dialog will appear. Click **No**. A project folder labled plasma will now appear in the IDE **zamiaCAD Navigator** view.

Import the archived project

The zamiaCAD+PyVHDL IDE allows projects to be archived, and later imported back into the IDE workspace. The folder at **PyVHDL-0.0.1/share/saved-projects** contains the plasma project archived as plasma-PyVHDL.zip.

Make sure the plasma project is highlighted in the IDE **zamiaCAD Navigator** view. Right click on the plasma project folder. In the context menu that appears select **Import...** The Import dialog will appear. Double click on the **General** folder, and select **Archive File**, then click the **Next** > button.

🕐 Import	
Select Import resources from an archive file into an existing project.	Ľ
Select an import source:	
type filter text	
 General Archive File Existing Projects into Workspace File System Preferences Preferences Run/Debug Team Other 	
(?) < <u>Back</u> <u>Next ></u> <u>Finish</u>	Ca

Fig. 2.5: Import Dialog

Next a dialog appears where the import is setup. Click on the **Browse...** button to the right of the **From archive file:** text box. Navigate to **PyVHDL-0.0.1/share/saved-projects**. Select the **plasma-PyVHDL.zip** file. Click the **Open** button. Now the completed Import dialog should look like this:

[*] Import Archive file	in zip or tar format from the local file system.
	share\saved_projects\plasma-PyVHDL.zip
▷ ☑ ि /	Image: Second state st
Into fo <u>l</u> der: plasma	Bro <u>w</u> se
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

Fig. 2.6: Completed Import Dialog

Click on the **Finish** button. In the Question dialogs that appears, click the **Yes** button. In the **Do Full Build?** dialog, click **Yes**. zamiaCAD will analyze and elaborate the plasma design. A log of that process will be in the **Zamia Console** window. Click on the plasma project folder. The project navigation information will appear below the folder. The red and blue rectangular icons can be opened to view the project design hierarchy. The files that make up the project are listed below the icons. You are now ready to setup and run a PyVHDL simulation.

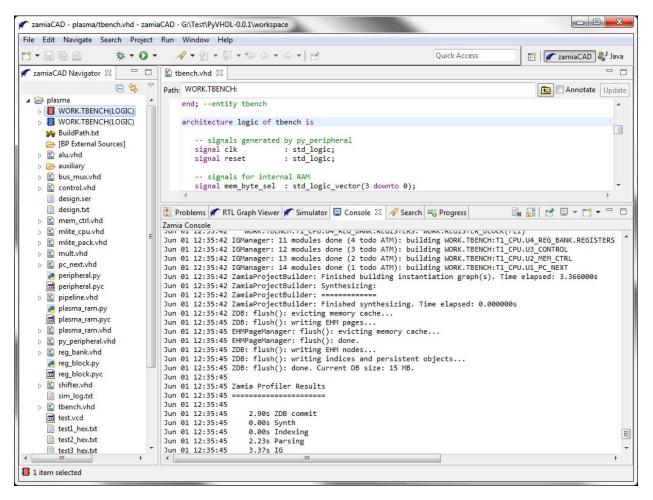


Fig. 2.7: The IDE After Plasma Project is Built

Configure a simulation

Make sure the plasma project is highlighted. On the IDE menu click **Run > Run Configurations...**. The **Run Configurations** dialog appears. Right click on **zamiaCAD Simulation**, and select **New**.

The right side of the dialog now displays the settings for the new configuration.

- In the Name: text box type plasma PyVHDL.
- Click the Browse... button next to the Project text box. Select the plasma project folder, and click OK.
- Click on the Simulator: dropdown, and select Python Simulator.
- Click on the **Browse...** button next to the **Toplevel:** textbox. **TOPLEVEL WORK.TESTBENCH** should be highlighted. Click **OK**.

- Click the **Browse...** button next to the **Flle:** textbox. Navigate to the **PyVHDL-0.0.1\workspace\plasma** folder. Select the **test.vcd** file. Click the **Open** button.
- The Signal path prefix: textbox should be empty.

Check that the fields in the Run Configurations dialog match the figure below:

Create, manage, and run con	figurations		
* 🖶 🗶 🕒	Name: plasma PyVH	łDL	
type filter text	💉 Main 🔗 Refr	resh 🚾 Environment 🔲 <u>C</u> ommon	
Java Applet	Project		
Java Application	plasma		Browse
Mew_configuration	Simulator		
	Simulator:	Python Simulator 🔹	
	Toplevel:	WORK.TBENCH	Browse
	File:	G:\Test\PyVHDL-0.0.1\workspace\plas	Browse
	Signal path prefix:		
Filter matched 4 of 4 items		Apply	Revert

Fig. 2.8: Completed Run Configurations Dialog

Run the simulation

Click the **Run** button. If a **Do Full Build ?** dialog appears, click **Yes**. Open the **Zamia Console** window if it is not open. The text at the bottom of the console window will look similar to this, without the coloring:

```
Jun 03 18:00:47 >>> RD_DATA1 0000000 fr zero @ 33325.00
Jun 03 18:00:47 **** MEMORY READ: value 0x0000000, address 0x00003A8 @ 33325.00
Jun 03 18:00:47 **** MEMORY READ: value 0x080000E9, address 0x000003A4 @ 33375.00
Jun 03 18:00:47 **** MEMORY READ: value 0x080000E9, address 0x000003A8 @ 33425.00
Jun 03 18:00:47 **** MEMORY READ: value 0x080000E9, address 0x000003A4 @ 33475.00
Jun 03 18:00:47 **** MEMORY READ: value 0x080000E9, address 0x000003A4 @ 33475.00
Jun 03 18:00:47 Simulation done (StopEvent)
Jun 03 18:00:47
Jun 03 18:00:47 Elapsed time = 2.8060
Jun 03 18:00:47 Simulation time = 33500.00 ns
```

Jun	03	18:00:47		
Jun	03	18:00:47		
Jun	03	18:00:47	Zamia Prof	filer Results
Jun	03	18:00:47		
Jun	03	18:00:47		
Jun	03	18:00:47	2.71s	ZDB commit
Jun	03	18:00:47	0.00s	Synth
Jun	03	18:00:47	0.03s	Indexing
Jun	03	18:00:47	2.24s	Parsing
Jun	03	18:00:47	3.42s	IG

Make sure the Simulator window is visible. The IDE will look similar to this:

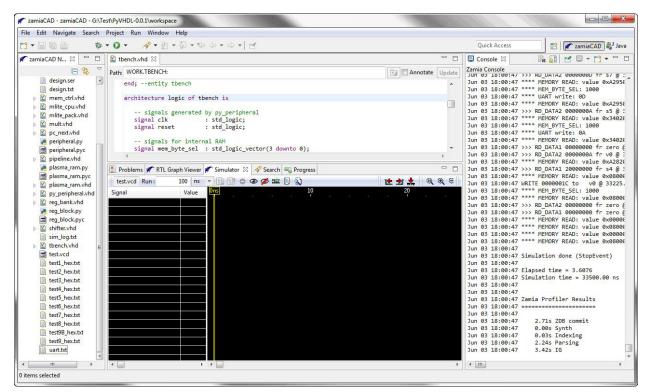


Fig. 2.9: IDE After Simulation Run

To select the signal waveforms to be displayed in the **Simulator** window, click on the **Eye** icon on the **Simulator** menu. The **Select signals to trace** dialog appears. In the **Select signals to be traced** textbox enter *. A list of signals will appear. Select all the signals. Click **OK**. The names and waveforms of the signals will now appear:

The IDE windows can be resized to show more of the waveforms. You can click on the **Simulator** window magnifying glass menu icons to zoom in, zoom out, or zoom full.

This completes the zamiaCAD+PyVHDL Quickstart tutorial. Read this zamiaCAD tutorial to learn more about the very useful features of the IDE.

2.2 Write a Python testbench

This part of the documentation needs more work! For now, look at the Python files in the **plasma** project. Briefly:

• Setup for the simulation is in the file: PyVHDL-0.0.1\workspace\plasma\peripheral.py

ile Edit Navigate Search Proj	ect Run Window	Help							
••• • • • • • •) • 🛷 • 🖞 ·	• 🖓 • 🕸 🔶 • 🖓	\$ • et				Quic	k Access	🖹 💉 zamiaCAD 🐉
🕈 zamiaCAD Navigator 🕴 🛛 👔	3 🔹 🗸 🗖 🗍	📓 tbench.vhd 🖾				- 0	🔄 Console 🛛		🛃 🗐 🕶 📫 🗖 🗖
WORK.TBENCH(LOGIC)	*	Path: WORK.TBENCH	4:		(A)	Annotate Update	Zamia Console		
b 🗁 Locals						Annotate [opdate]	Jun 07 12:40:3	5 >>> RD_DATA1 @	00000000 fr zero @ 33:
Globals	E		Plasma CPU co		e public domain b	A the puther			EAD: value 0x080000E9, 20000000 fr s4 @ 3317
T1_CPU: WORK.MLITE,	CPU(LOGIC)				ty. Author liabl				to v0 @ 33225.00 r
T2_MEMORY: WORK.P	LASMA_RAM(EMI	DESCRIPTI			.,			5 **** MEM BYTE	
T3_PERIPHERAL: WORL	C.PY_PERIPHERAL				ch for testing th				EAD: value 0x080000E9
⊿ 🗁 Locals									00000000 fr zero @ 33:
CLK: STD LOGI	C	library ieee	; logic 1164.a	11					EAD: value 0x00000000
→ RESET: STD_LO	SIC	use leee.std		;					00000000 fr zero @ 33: EAD: value 0x080000E9
Globals		USE WORKINII	.ce_pack.aii,						EAD: value 0x000000000
WORK.TBENCH(LOGIC)		entity then	h is						EAD: value 0x080000E9
H BuildPath.txt	-	end;entit	y tbench			-	Jun 07 12:40:3	5	· · · · · · · · · · · · · · · · · · ·
	+	4		III		•	· ·		· · · · · · · · ·
1 1 0 100				14 June 19 June					
	▼ EL EF ● @ § Value		7300		👷 🔍 🔍 🔍 🔍 🔍 🔍		7500	7550	7600
	Look Loof -		7300			7450	7500	7550) 7600
ignal CLK	Look Loof -			7350		7450		7550) 7600
lignal	Value 1						7500	7550	0 7600
ignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL (3 downto 0)	Value 1	<u>X"00</u> {X"00000 X"0"	00D4" ()	7350 ("000000D8"		7450 X"000000E8") 7600
ignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL (3 downto 0) MEM_DATA_R (31 downto 0)	Value 1 X"00000000" X"0"	x*00 X*00000 x*0* x*24 X**14610	00D4" ()	7350		7450	XX"000000EC") 7600 .
ignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL (3 downto 0)	Value 1 X*00000000" X*0 X*00 X*000000000"	<u>X"00</u> {X"00000 X"0"	00D4" ()	7350 ("000000D8"		7450 X"000000E8"	XX"000000EC"	7550) 7600
ignal CLK MEM_ADDRESS (31 downto 0) MEM_DATE_SEL (3 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_W (31 downto 0)	Value 1 X"00000000" X"0" X"00000000" X"00000000"	x*00 X*00000 x*0* x*24 X**14610	00D4" ()	7350 ("000000D8"		7450 X"000000E8"	XX"000000EC"	7550)
ignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL(3 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_W (31 downto 0) MEM_WRITE	Value 1 X"00000000" X"0" X"00000000" X"00000000"	x*00 X*00000 x*0* x*24 X**14610	0004" (λ 0004" (λ	7350 ("000000D8"		7450 X"000000E8"	XX"000000EC"	7550)
Signal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE SEL (3 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_W (31 downto 0) MEM_WRITE RESET	Value 1 X"00000000" X"0" X"00000000" X"00000000" 0 1	x*00 X*00000 x*0* x*24 X*14610 x*00000000*	0004" X3	7350 ("000000D8" ("3C018000"	7400	7450 X"000000E8"	Xx"000000EC")
iignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SE(3 downto 0) MEM_DATA, R (31 downto 0) MEM_DATA, W (31 downto 0) MEM_WRITE RESET A_BUSD (31 downto 0)	Value 1 x*00000000* X*00* X*00000000* 0 1 x*00000000* 1 x*00000000*	x~00 /x~0000 x~2 x~24 /x~14610 x~00000000	0004" X3	7350 ("000000D8" ("3C018000"	7400 	7450 X"000000E8"	X**0000000C** XX**00000000* XX**00000000*) 7600
iignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL (3 downto 0) MEM_DATA, R (31 downto 0) MEM_DATA, W (31 downto 0) MEM_WRTE RESET A_BUSD (31 downto 0) B_BUSD (31 downto 0)	Value 1 X*00000000* X*00000000* X*00000000* 0 1 X*00000000* x*00000000* x*00000000*	X:00 X:00000 X:00 X:24 X:14610 X:00000000 X:00 X:00000 X:00 X:7FFFF	0004" X3 0004" X3 0061C" X3 FFE9" X3	7350 ("000000D8" ("3C018000"	7400 	7450 X"000000E8"	X**0000000C** XX**00000000* XX**00000000*)
ignal CLK MEM_DORESS (31 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_R (31 downto 0) MEM_QATA_W (31 downto 0) MEM_VSUB (31 downto 0) B_BUSD (31 downto 0) MEM_SOURCE (3 downto 0)	Value 1 X*0000000* X*00 X*0000000* X*000	X:00 X:0000 X:0° X:24 X:14610 X:00000000° X:00 X:00000 X:00 X:FFFFF	D004" X3 D004" X3 D004" X3 D061C" X3 FFE9" X3 X3	7350 C*000000D8* C*000000D8* C*00000000* C*00000000* C*FFFFFFFF*	7400 X**00000008* X**00000010*	7450 Xx*0000068* Xx*00001012*	X:"000000EC" X:"00000000" X:"00000000" X:"FFFFFFFF	<u>)</u> <u>(</u> X*00000000*	
iignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SE (3 downto 0) MEM_DATA, R (31 downto 0) MEM_WRITE RESET A, BUSD (31 downto 0) B_BUSD (31 downto 0) MEM_SOURCE (3 downto 0) COUNT_REG (5 downto 0)	Value 1 X'00000000' X'00 X'00000000' X'00000000' I X'00000000' X'00000000' X'00000000' X'0000000' X'000 X'00'	x*00) X*00000 x*00 x*24 X*14610 x*00000000 x*00 X*0000 x*00 X*0000 x*00 X*0000 x*20* x*20*	2004" (2) 2004"	7350 .:	7400 XX*000000D8* XX*00000000 XX*01*	7450 X=00000068* Xx=00001012* Xx=00001012*	X**00000000* X**00000000* X**FFFFFFF* X**03*	<u>)</u> (<u>x</u> *000000000*)(<u>x</u> *04*	
iignal CLK MEM_DDRESS (31 downto 0) MEM_DYTE_SE. (3 downto 0) MEM_DATA, R (31 downto 0) MEM_DATA, R (31 downto 0) MEM_WRTE RESET A_BUSD (31 downto 0) MEM_SOURCE (3 downto 0) MEM_SOURCE (3 downto 0) COUNT_REG (5 downto 0) REG.A (31 downto 0)	Value 1 x*0000000* X*0* X*0000000* 0 1 x*0000000* X*0000000* X*0000000* X*00* X*00* X*00* X*00* X*00* X*00*	x:00) X: 00000 x:01 x:24 X: 14610 x:00000000 x:00 X: 00000 x:00 X: 0000 x:00 X: 0000 x:20 X: 00000000	2004" (2) 2004"	7350 .:	7400 XX*000000D8* XX*00000000 XX*01*	7450 X=00000068* Xx=00001012* Xx=00001012*	X**00000000* X**00000000* X**FFFFFFF* X**03*	<u>)</u> (<u>x</u> *000000000*)(<u>x</u> *04*	<u>\X"05"</u>
ignal CLK MEM_DORESS (31 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_R (31 downto 0) MEM_QATA_W (31 downto 0) MEM_VSUB (31 downto 0) B_BUSD (31 downto 0) B_BUSD (31 downto 0) REG_A (31 downto 0) REG_B (34 downto 0)	Value 1 X*00000000* X*00 0 0 1 X*00000000* 0 1 X*00000000* X*0000000* X*00* X*00* X*00* X*00* X*00* X*00* X*00*	x00/x00000 x0 x0 x00/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x14610 x00/x74/x1400 x00/x74/x1400 x00/x74/x1400 x00/x74/x1400 x00/x74/x1400 x00/x74/x1400 x00/x74/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x1400 x00/x000 x00/x000 x00/x000 x00/x000 x00/x00 x00/x	0004" (2) 0004" (2) 00 00 00 00 00 00 00 00 00 0	7350 .:	7400 XX*000000D8* XX*00000000 XX*01*	7450 X=00000068* Xx=00001012* Xx=00001012*	X**00000000* X**00000000* X**FFFFFFF* X**03*	<u>)</u> (<u>x</u> *000000000*)(<u>x</u> *04*	<u>\X"05"</u>
ignal CLK MEM_ADDRESS (31 downto 0) MEM_BYTE_SEL (3 downto 0) MEM_DATA_R (31 downto 0) MEM_DATA_W (31 downto 0) MEM_VATE RESET A, BUSD (31 downto 0) MEM_SOURCE (3 downto 0) MEM_SOURCE (3 downto 0) REG_B (6 downto 0) REG_B (6 downto 0) REG_B (6 downto 0) MEM_STE_REG (1 downto 0)	Value 1 1 x*0000000* X*00* 0 0 1 x*0000000* 0 x*0000000* x*0000000* x*0000000* x*0000000* x*00* x*0000000* x*00* x*00* x*00* x*00* x*00* x*00* x*00* x*00* x*00* x*0*	x ¹⁰⁰ /x ¹⁰⁰⁰ /x ¹⁰⁰⁰⁰ x ¹⁰ /x ¹⁰⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰⁰⁰⁰ x ¹⁰⁰ /x ¹⁰⁰⁰⁰⁰⁰⁰⁰	0004" (2) 0004" (2) 00 00 00 00 00 00 00 00 00 0	7350 C00000008* C3C018000* C00000000* C00000000* CFFFFFFF C00* C0000001C* C0000008500000	7400 XX*000000D8* XX*00000000 XX*01*	7450 Xx*000000E8" Xx*00001012" Xx*022" 00000 (X*0000002E000000	Хх [.] 00000000 [°] Хх [.] 10000000 [°] Хх [.] 190000000 [°] Хх [.] 19FFFFFF [°] Хх [.] 193 [°]	<u>)</u> (<u>x</u> *000000000*)(<u>x</u> *04*	<u>\X"05"</u>

Fig. 2.10: IDE Displaying Waveforms

- The plasma registers are implemented in the file: PyVHDL-0.0.1\workspace\plasma\reg_block.py. Note the print statements which print register activity messages to the zamiaCAD console.
- 8K of memory for the plasma CPU, and a UART data register are implemented in the file: PyVHDL-0.0.1\workspace\plasma_plasma_ram.py. Note how the initialize(self) method is used to set up the contents of the memory from a file.

The **Process** class implements a **concurrent** process. The class has 2 important methods:

- initialize(self) is called before the simulation starts to do any setup that is needed.
- **run(self, P, S)** is called to start the process running. The while true: loop causes the process to continuously run through its code, yielding to other processes when a wait is encountered. The **P** argument is a class whose attributes are the ports defined in the VHDL entity declaration. Signals are accessed as **P.signal_name**. The **signal_name** must be in **UPPER CASE**. the **S** argument is a list containing references to all the signals in the design (*avoid using this for now*).

Each Python file that implements a VHDL architecture has a **setup**(\mathbf{tb} , \mathbf{P}) function that adds the processes defined in the file to the simulation execution environment. The \mathbf{tb} argument is a reference to the simulation execution environment. The \mathbf{P} argument is a string identifying the process.

2.3 License

PyVHDL is an open source project. This means that it is not only available for download free of charge, but users have access to the source code and may contribute to the project.

PyVHDL is copyright 2016 by Vern Muhr and is covered by the GNU General Public License Version 3.

Here is the ZamiaCAD project license.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search