FPGA Essential Glossary

Descriptions from Wikipedia and other technical sources

Contributors: Jan Cumps, Enrico Miglino

Version 1 - Oct 2021

FPGA

FPGA is the acronym of **field-programmable gate array**; it is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term field-programmable.

The FPGA configuration is specified using a hardware description language (HDL).

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnections allowing blocks to be wired together.

Logic blocks can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

Verilog

Verilog is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits. In 2009, the Verilog standard (IEEE 1364-2005) was merged into the SystemVerilog standard, creating IEEE Standard 1800-2009.

VHDL

The VHSIC Hardware Description Language (VHDL) is a hardware description language (HDL) that can model the behaviour and structure of digital systems at multiple levels of abstraction, ranging from the system level down to that of logic gates, for design entry, documentation, and verification purposes.

VHDL is named after the United States Department of Defense program that created it, the Very High-Speed Integrated Circuits Program (VHSIC).

In the early 1980s, the VHSIC Program sought a new HDL for use in the design of the integrated circuits it aimed to develop, including FPGA.

VHDL Keywords Essentials

The following are some of the most frequently used keywords in the VHDL sources

`library IEEE;`

`use IEEE.std_logic_1164.all;`

: Library and their use are like library import in Python: they give access to standard and extended functionality.

`entity xxxxxxx`

: It is the interface of a VHDL object you create

`port`

: The parameters you can pass to an object

`architecture`

: The implementation of a VHDL object you create. Here you write how the object behaves

`process`

A sequential process that executes step by step. Everything outside a process runs in parallel. You mention all the signals that this process will react to as parameters. (sensitivity list)

`package`

Allows to create your own namespace. Like a lib or namespace in a program language.

`component xxxxxx`

Allows to reuse your design, having more than one instance of the same thing, and make bigger designs by combining objects

`signal`

Represents a logic state true or false and can be set/read/stored. It has a different behaviour than a variable.

`variable`

A software only element that also represents true or false, but not as a hardware state, but a utility to write your code.

`STD_LOGIC`

A single bit of information

`STD_LOGIC_VECTOR`

A bus of bits

Assignment Operators

`:=`

`<=`

Example:

```
signal QuadA_Delayed: std_logic_vector(2 downto 0) := (others=>'0');
Count <= (others=>'0');
QuadA_Delayed <= (others=>'0');
```

Filter Events

```
clocked: process(clk_i, n_reset_i)
  begin
    -- async reset
    if n_reset_i = '0' then
        -- reset
    elsif rising_edge(clk_i) then
      -- logic
```

Declarative Language

In computer science, declarative programming is a programming paradigm – a style of building the structure and elements of computer programs—that expresses the logic of a computation without describing its control flow.

This is in contrast with imperative programming, which implements algorithms in explicit steps.

Declarative programming often considers programs as theories of a formal logic, and computations as deductions in that logic space.

MicroBlaze

The MicroBlaze is a soft microprocessor core designed for Xilinx field-programmable gate arrays (FPGA). As a soft-core processor, MicroBlaze is implemented entirely in the general-purpose memory and logic fabric of Xilinx FPGAs.

With few exceptions, the MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances.

The MicroBlaze has a versatile interconnect system to support a variety of embedded applications.

MicroBlaze's primary I/O bus, the AXI interconnect, is a system-memory mapped transaction bus with master–slave capability. Older versions of the MicroBlaze used the CoreConnect PLB bus.

Vivado

Xilinx's Vivado Design Suite is the development environment for building current MicroBlaze (or ARM - see Zynq) embedded processor systems in Xilinx FPGAs.

Designers use the Vivado IP Integrator to configure and build the hardware specification of their embedded system (processor core, memory-controller, I/O peripherals, etc.)

The IP Integrator converts the designer's block design into a synthesizeable RTL description (Verilog or VHDL), and automates the implementation of the embedded system (from RTL to the bitstream-file)

For the MicroBlaze core, Vivado generates an encrypted (non human-readable) netlist.

The SDK handles the software that will execute on the embedded system. Powered by the GNU toolchain (GNU Compiler Collection, GNU Debugger), the SDK enables programmers to write, compile, and debug C/C++ applications for their embedded system. Xilinx's tools provides the possibility of running software in simulation, or using a suitable FPGA-board to download and execute on the actual system.

IP

In electronic design, a semiconductor intellectual property core (SIP core), IP core, or IP block is a reusable unit of logic, cell, or integrated circuit layout design that is the intellectual property of one party.

IP cores can be licensed to another party or owned and used by a single party.

The term comes from the licensing of the patent or source code copyright that exists in the design. Designers of application-specific integrated circuits (ASIC) and systems of field-programmable gate array (FPGA) logic can use IP cores as building blocks.