

## A Preview of the IEEE P1687 (IJTAG) Languages

- **Jeff Rearick (AMD / P1687 Editor) and the P1687 Working Group**

*As general background on IEEE P1687 and as a basis for comparison of the language offered in the main article, we asked the IJTAG Working Group for a high-level summary of their language proposal, which they have provided in this sidebar.*

- *Erik Jan Marinissen, Editor*

The IJTAG (Internal JTAG, or Instrument JTAG) Working Group was created to address the problem of making on-chip instruments (loosely, test and debug features) available for off-chip usage. The IEEE embraced this effort and launched P1687, entitled “Draft Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device.” The team has produced a hardware architecture that is compatible with but goes beyond IEEE 1149.1 for the physical access to the instruments, along with a software framework for communicating with the instruments.

Though the IEEE P1687 Working Group is still in the process of completing the draft standard, many details of the language content have been engineered and are presented here, along with the reasoning for the strategy chosen by the working group. P1687 uses two different languages for two different purposes: a hardware description language (HDL) that contains the hierarchical connectivity of the scan network between the IEEE 1149.1 Test Access Port (TAP) and the instruments, and a procedure description language (PDL) that contains the patterns used to interact with the instruments.

The working group has separated the syntaxes of HDL and PDL in light of their very different purposes: HDL is similar in concept to a hierarchical RTL netlist (inspired by Synopsys’ Liberty file grammar), and PDL is effectively a programming language (in fact, PDL has been designed to be an extension of Tcl – Tool command language). For both languages, a minimalist design approach has been used that includes only those elements necessary for the task and nothing more; for this reason, more open-ended and feature-laden languages like Verilog, VHDL, and CTL were rejected.

The easiest way to illustrate the P1687 language content is with a small example: writing to and reading from an instrument in an embedded core of a chip.

The HDL for the structure shown in the figure below is a simple hierarchical instantiation of an instrument connected to a Test Data Register (TDR), with identification of the ports (by name and *function*) connected to the scan network’s P1687 interface.

```

Module MyCore {
  Ports {
    SI      { Function : ScanIn; }
    R       { Function : Reset; }
    SL      { Function : Select; }
    SE      { Function : ShiftEn; }
    CE      { Function : CaptureEn; }
    UE      { Function : UpdateEn; }
    SO      { Function : ScanOut; Source: MyLogic.SO; }
  }
  Instances {
    MyLogic { Modulename : Logic;
              InputSource { D : MyInstrument.Out; } }
    // all other inputs use default source
    MyInstrument { Modulename : Instrument;
                  InputSource { In : MyLogic.Q; } };
  }
}

Module Logic {
  Ports {
    SI      { Function : ScanIn; }
    R       { Function : Reset; }
    SL      { Function : Select; }

```

```

    SE      { Function : ShiftEn; }
    CE      { Function : CaptureEn; }
    UE      { Function : UpdateEn; }
    SO      { Function : ScanOut; Source: MyTDR; }
    Q[3:0] { Function : ToDataBit; Source: MyTDR[3:0]; }
    D[3:0] { Function : FromStatus; }
}
Registers {
    MyTDR[3:0] { ScanIn : SI;
                 CaptureSource : D[3:0]; }
}
}
Module Instrument {
    Ports {
        In[3:0] { Function : DataBit; }
        Out[3:0] { Function : Status; }
    }
}

```

The nine-command core of PDL is intended to be a concise and intuitive language: setup commands (like `iWrite` and `iRead`) are queued, and then executed by the `iApply` command (which is an action command that gets interpreted immediately). These core commands may be utilized in a Tcl program that performs more complex flow control functions if needed. The following PDL procedure writes a four-bit value to the instrument then reads an expected four-bit result.

```

iScope Mycore.MyInstrument
iWrite In 0x7
iApply
iRead Out 0xF
iApply

```

The software which processes the HDL and PDL is envisioned to first build an internal data structure representation of the scan network described by the HDL, and then use that data structure to automatically map the PDL commands from the instrument ports through the scan network to the TAP.