IEEE Std 1149.1-2013

This is an IEEE 1149.1 working group member’s attempt in presenting changes and new features defined in IEEE 1149.1-2013. Slides not reviewed or approved by the working group.
Outline

Changes to existing concepts

New concepts

New instructions

New data registers

New BSDL constructs

New: Procedural Description Language (PDL)
Changes to existing concepts

- Previously deprecated BC_6 no longer supported
- Boundary-scan register can have excludable segments
- All (except TAP) pins may have additional observe-only cells
- Additional BSDL constructs and new options for existing constructs
- Defined interface for design-specific TDR (recommended)
Outline

Changes to existing concepts

New concepts

New instructions

New data registers

New BSDL constructs

New: Procedural Description Language (PDL)
New Concepts

- Test-Mode Persistence (TMP) Controller *(optional)*
- Electronic Chip Identification (ECIDCODE) *(optional)*
- Initialization *(optional)*
- IC Reset *(optional)*
- *(excludable) Register segments *(optional)*
- Power domain control *(optional)*
- Register fields, mnemonics, and assembly *(optional)*
- Procedural Description Language (PDL) *(optional)*
Test-Mode Persistence (TMP) Controller

- New, optional, synchronous finite state machine
- Assert test mode regardless of active instruction
- Instructions CLAMP_HOLD, CLAMP_RELEASE, and TMP_STATUS
- Two-bit TMP Status register

IEEE Std 1149.1-2013 Clauses 6.2, 8.20, and 16
Electronic Chip Identification (ECID)

- Optional
- Individual chip’s ECID value is unique
- “Serial number” for the component
- Permits tracking the history of the component through its lifetime
- ECIDCODE instruction, ECID register
- Extraction defined in PDL

IEEE 1149.1-2013 Clauses 8.15 and 13
Focusing on devices with programmable I/O

Optional method to initialize component before test

INIT_SETUP and INIT_SETUP_CLAMP, and/or INIT_RUN instructions

Initialization process to be defined in PDL

Initialized system state is retained as long as the device stays in test mode (or “persistence” is on)

IEEE 1149.1-2013 Clauses 8.17, 8.18, 8.19, 14, and 15
IC Reset

- Provide test control of **system** reset and related inputs through TAP
- Does **not** control reset signals to the test logic
- IC_RESET instruction and Reset Selection register

*IEEE 1149.1-2013 Clauses 8.21 and 17*
Register segments

- Registers may be segmented (other than Bypass, ID, and TMP status register)

- Register segment length must fixed (1 bit or longer)

- Segments can be included / excluded

- Segments must not overlap and must not be contained within another segment*

- Excludable segments are initially excluded

- Segment-select cell controls inclusion of segment

- Segments may be in different power domains

- Mutually exclusive segments in “broadcast” config

* = this restriction is true for registers defined in the standard

Power domain control

- Power domains may be powered down
- Segments in un-powered domains must be excluded
- Domain-control cell for on-chip control through TAP

"Ready to scan" (from domain controller; 1 = ready)

From TDI

- Domain-control Cell
- Segment-selector Cell
- Excludable Segment
- SegMux

Captures, shifts, and updates <TDR>

To TDO
Register fields, mnemonics, and assembly

- Registers can be segmented, requiring “assembly” defined in BSDL
- Registers / segments can be described with fields
- Patterns / values to be loaded into fields may be defined with mnemonics

IEEE 1149.1-2013 Clauses B.8.18, B.8.19, B.8.20, and B.8.21
Outline

Changes to existing concepts

New concepts

New instructions

New data registers

New BSDL constructs

New: Procedural Description Language (PDL)
New Instructions

- CLAMP_HOLD
- CLAMP_RELEASE
- TMP_STATUS
- ECIDCODE
- INIT_SETUP
- INIT_SETUP_CLAMP
- INIT_RUN
- IC_RESET

(all optional)
CLAMP_HOLD

- Test-mode instruction
- Required if TMP controller is provided
- Selects TMP Status register
- Sets TMP controller to Persistence-On
- Enforces CLAMP-behavior
- Chip designer selects opcode (anything but all-0)
CLAMP_RELEASE

- Test-mode instruction
- Required if TMP controller is provided
- Selects TMP Status register
- Sets TMP controller to Persistence-Off
- Enforces CLAMP-behavior
- Chip designer selects opcode (anything but all-0)
Normal-mode instruction

Required if TMP controller is provided

Selects TMP Status register

Reports status of TMP controller and value of Bypass-Escape bit

Chip designer selects opcode
Normal-mode instruction

Used to retrieve vendor-defined public electronic chip identification code (ECID register)

- All '1' or all '0' in ECID register indicates failure to retrieve ECID value

Chip designer selects opcode
INIT_SETUP

To supply parameters required for component initialization, through Init Data register.

Such information is component and board (instance) dependent.

Not a test-mode instruction.

If provided, must be run before any test-mode instructions.

Chip designer selects opcode.
INIT_SET-_UP_CLAMP

- Required whenever the INIT_SET_UP is provided
- Enforces CLAMP-behavior (test mode instruction)
- Would be run in place of INIT_SET_UP
- Selects Init Data register
- Chip designer selects opcode
INIT_RUN

Test-mode instruction

Selects Init Status register

Executes sequential initialization process

Completion of initialization determined based on:
  - time delay (defined as clock cycles or actual time), or
  - polling status

Chip designer selects opcode
IC_RESET

- Normal-mode instruction
- Selects Reset Selection register
- Control of "system reset" functions through the TAP
- No affect on test logic
- Does not change TMP controller state
- Chip designer selects opcode (should avoid all-0)
Outline

- Changes to existing concepts
- New concepts
- New instructions
- New data registers
- New BSDL constructs
- New: Procedural Description Language (PDL)
New Data Registers

- TMP Status Register
- Electronic Chip ID (ECID) Register
- Init Data Register
- Init Status Register
- Reset Selection Register

(all optional)
TMP Status Register

- Required if TMP Controller is implemented
- Selected by CLAMP_HOLD, CLAMP_RELEASE, and TMP_STATUS
- Two-bit register
  - Bit 0: Bypass-Escape bit (1 for transition to Persistence-Off)
  - Bit 1: TMP-status bit (reports status of TMP Controller)
Figure 16 shows a possible implementation of the TMP control register that meets the requirements of this clause.

The connection of the "TMP_state" input signal and the "Bypass_Escape" output signal are to Figure 6.

Figure 6 shows the generation of the TAP_POR* signal from the TRST* on-component power generation circuit.

Deleted: shows the use of the "Bypass_Escape" output signal.
Electronic Chip ID (ECID) Register

- Selected by ECIDCODE instruction
- User-(designer-)defined length
- Value is unique to each chip of a specific type
- All-1 and all-0 indicate error during retrieval
- Details of retrieval process not defined in standard (extraction sequence described in PDL)
Init Data Register

- Provides parameters for initializing programmable I/O and other circuits requiring initialization
- Selected by INIT_SETUP
- Length depends on chip design
- Value depends on component instance and board design
Init Status Register

- Observes status of the initialization process
- Selected by INIT_RUN
- Two or more bits long:
  - Bit 0: busy (1) or done (0)
  - Bit 1: successful/pass (1) or unsuccessful/fail (0)
  - (additional bits may capture further status / failure details)
Reset Selection Register

- Selects one or more possible functional reset operations to be performed
- Selected by IC_RESET
- Length: $1+(N*2)$ bits, with $N \geq 1$
  - LSB = reset-hold bit (0 allows reset control w/ Reset-Select)
  - N 2-bit pairs, one for each system logic reset signal to be controlled:
    - MSB (closer to TDI): reset-control bit (0 asserts reset signal when enabled)
    - LSB (closer to TDO): reset-enable bit (0 enables reset-control bit for use)
- Documented with REGISTER_FIELDS, etc.
Outline

Changes to existing concepts

New concepts

New instructions

New data registers

New BSDL constructs

New: Procedural Description Language (PDL)
New BSDL Constructs

- New port types for additional information
- Power Port Association
- Register Fields
- Register Mnemonics
- Register Assembly
- Register Association
- Boundary Scan Register segments
- System Clock description
New port types

- in
- out
- buffer
- inout

**LINKAGE**

**LINKAGE**_INOUT, **LINKAGE**_BUFFER, **LINKAGE**_OUT, **LINKAGE**_IN, **LINKAGE**_MECHANICAL, **POWER**_0, **POWER**_POS, **POWER**_NEG, **VREF**_IN, **VREF**_OUT
Optional new pin descriptions for use in pin map

OPEN, TIE0, or TIE1 for non-connected package pins

TIE0 / TIE1 only allowed for pin types IN, LINKAGE_IN, POWER_0, POWER_POS, or VREF_IN

constant SM:PIN_MAP_STRING:=
"CLK:9, Q:(open,open,open,open,16,17,18,19), " &
"D:(open,open,open,open,2,1,26,25), " &
"GND1:15, VCC1:8, GND2:open, VCC2:open, OC_NEG:7, " &
"TDO:20, TMS:21, TCK:23, TDI:24";
Boundary-scan register may now be segmented

**REGISTER_ASSEMBLY** would be used to define the construction of the full boundary-scan register

**ASSEMBLED_BOUNDARY_LENGTH** specifies

- reset length (all excludable segments excluded), and
- maximum length (all excludable segments included)

Attribute **ASSEMBLED_BOUNDARY_LENGTH** of myIC : entity is (40,46);
Defines always-on and/or excludable segments of the boundary-scan register

REGISTER_ASSEMBLY is used to combine segments into the full boundary-scan register

Also new for the boundary-scan register description: new input-spec elements:

OPEN0, OPEN1, OPENX, EXTERN0, EXTERN1
SYSCLOCK_REQUIREMENTS

- Describe the use of system clocks for various instructions (for use in PDL)
- Defines the required minimum and maximum frequency
- \texttt{RUNBIST}, \texttt{INTEST}, \texttt{INIT\_SETUP}, \texttt{INIT\_SETUP\_CLAMP}, \texttt{INIT\_RUN}, \texttt{ECIDCODE}, \texttt{IC\_RESET}, or design specific
- Pin type: \texttt{in}, \texttt{inout}, \texttt{LINKAGE\_IN}, or \texttt{LINKAGE\_INOUT}

\begin{verbatim}
attribute SYSCLOCK_REQUIREMENTS of myIC is "(myCLK, 10, 20, RUNBIST)";
\end{verbatim}
Register Mnemonics

- Provide meaningful text names / descriptions for values to be loaded into a TDR / part of a TDR
- Allowed in BSDL and/or package files
attribute REGISTER_MNEMONICS of INIT_Example : entity is
"SerDes_Protoocol ( "&
  " off (0b000) <Powered down>, ", ",&
  " SATA (0b010) <Serial Advanced Technology Attachment>, ", ",&
  " SRI0 (0b011) <Serial RapidIO>, ", ",&
  " XAUI (0b101) <10 Gbps Attachment Unit Interface>, ", ",&
  " Resvd1 (0b100) <Reserved for future use>, ", ",&
  " Resvd2 (0b11X) <Undefined behavior - Do Not Use> )";
Register Fields

- Defining possibly hierarchical construction of a TDR
- Allowed in BSDL and/or package files
- Identifying fields within a TDR and characteristics such as:
  - the type of TDR cell used in each field,
  - how the fields are reset and to what value, and
  - what values should be written / expected from the fields
- Register Fields may be addressed by PDL
Defines and names fields within a register / segment

Total length of register / segment is stated explicitly

Support for hierarchy through PREFIX keyword

Register field definition includes:

Field name, field length, bit list, and possibly field options (value, type, or reset)

```
attribute REGISTER_FIELDS of MEMD_example : package is
  "MBist [6] ( "&
  " (Algorithm [3] IS (5 DOWNTO 3 )), "&
  " (Command [1] IS (2) ), "&
  " (Status [2] IS (1 DOWNTO 0) ) "&
  " )";
```
REGISTER_FIELDS

Register field options:

- Value: CAPTURES, DEFAULT, SAFE, or RESETVAL
- Type: NOPI, NOPO, NOUPD, MON, PULSE0, PULSE1, SHARED
- Reset: PORRESET, TRSTRESET, TAPRESET, or CHRESET
Register Assembly

- Defines a register / segment by concatenating register segments and fields in the order listed
- Listed segments may be defined with REGISTER_FIELDS or a REGISTER_ASSEMBLY
- May contain in-line field definitions
- Package hierarchy can be added to reference registers / segments in specific package files
- DOMCTRL, SEGSEL, SEGSTART, and SEGMUX support excludable segments
attribute REGISTER_ASSEMBLY of INIT_Example : entity IS
  "init_data" (
    "init_tail IS init_seg), "&
    "(SerDesChannel_00 IS Channel), "&
    "(SerDesChannel_01 IS Channel), "&
    "(SerDesChannel_02 IS Channel), "&
    "(SerDesClk_0 IS ChClock) );";
attribute REGISTERCONSTRAINTS of init_example : entity is "init_data (" & "( (PDA=={Override})&& ( PDB == { Override } ) ) "& " ERROR <Domain A & B cannot both be ON at the same time> "& ")";
Register and Power Port Association

- Providing information that may point to causes of incorrect behavior
- To support test, diagnostics, debug
REGISTER_ASSOCIATION

Allows pairing register bits / fields with specific info

Register field / instance

Port, info, sysclk, or user

Attribute REGISTER_PORT_ASSOCIATION of init_example : entity is

"VSEL_bits (4) : (PwrUp_IO_VSEL(4)), ", &
"VSEL_bits (3) : (PwrUp_IO_VSEL(3)), ", &
"VSEL_bits (2) : (PwrUp_IO_VSEL(2)), ", &
"VSEL_bits (1) : (PwrUp_IO_VSEL(1)), ", &
"VSEL_bits (0) : (PwrUp_IO_VSEL(0)), ", &
"SerDesChannel(0) : (IO_TXP(0), IO_TXN(0), IO_RXP(0), IO_RXN(0)), ", &
"SerDesChannel(1) : (IO_TXP(1), IO_TXN(1), IO_RXP(1), IO_RXN(1)), ", &
...
"SerDesChannel(16) : (IO_TXP(16), IO_TXN(16), IO_RXP(16), IO_RXN(16)), ", &
"SerDesChannel(17) : (IO_TXP(17), IO_TXN(17), IO_RXP(17), IO_RXN(17))";
Associates reference voltages with dependent I/Os

```vhdl
attribute POWER_PORT_ASSOCIATION of myIC : entity is
  "DDR_REF1 : ( DDR_DATA(7), "&
   DDR_DATA(6), "&
   DDR_DATA(5), "&
   DDR_DATA(4), "&
   DDR_DATA(3), "&
   DDR_DATA(2), "&
   DDR_DATA(1), "&
   DDR_DATA(0) ), "&
"IO_REF1 : ( SERDES(0), SERDES(1) ), "&
"IO_REF2 : ( SERDES(2), SERDES(3) ");
```
Register segments, initialization example

from IEEE Std 1149.1-2013 Annex D.1
Outline

Changes to existing concepts

New concepts

New instructions

New data registers

New BSDL constructs

New: Procedural Description Language (PDL)
New: Procedural Description Language

- Based on TCL
- Aligned with PDL in IEEE P1687
- Purpose 1149.1 PDL is to load and unload registers or register fields independent of package level
- TDR-centric language
- Dependent on BSDL and package file(s)
- Normative Annex C
New: Procedural Description Language

PDL Level 0:
- intended to support “load-and-go” ATE
- procedures that operate on test data registers (writing to and comparing expected values from registers)
- does not return the data captured in the registers
- provides very limited flow control

PDL Level 1:
- intended to support diagnostic, debug, and test procedures where interactive operation is needed
- defines all PDL commands as extensions to Tcl
Thank you

Heiko Ehrenberg

h.ehrenberg@goepelusa.com