IEEE P1687.1 Extending the Network Boundaries for Test

Michael Laisne - Dialog Semiconductor - A Renesas Company
Alfred Crouch – Amida Technology Solutions
Michele Portolan - Univ Grenoble Alpes, CNRS
Martin Keim – Siemens Digital Industries Software
Hans Martin von Staudt - Dialog Semiconductor - A Renesas Company
Bradford G. Van Treuren - VT Enterprises Consulting Services
Jeff Rearick - Advanced Micro Devices
Songlin Zuo – Facebook
Disclaimer

• Although I am presenting the paper submitted by the IEEE P1687 Working Group
  – I am not representing the Working Group
  – I am not speaking of on behalf of the Working Group

• All statements and comments are my own
Motivation of IEEE P1687.1

- IEEE 1687-2014, aka IJTAG, achieved great popularity
- But limited to IEEE 1149.1 Test Access Port (TAP)
  - Large number of designs feature a TAP
  - But not all
- Other type of interfaces, e.g. I2C, SPI, in-house, ...
- Still want to use IJTAG internal to the device
Motivation of IEEE P1687.1

Brief history

• Study Group formed ITC’15, Working Group approved in 2016

Title

• Standard for the Application of Interfaces and Controllers to Access 1687 IJTAG Networks Embedded Within Semiconductor Devices

Objectives

• Extends or expands application of 1687
• Allow use/description of range of IC interfaces and their controllers
• Use I²C, SPI, …, “any” future (synchronous) slow interface
Outline

• Principles
• Recap IEEE P1687.1
• Solutions outline
• Summary
Principles

• Don’t change IEEE 1687 unless *really, really* necessary
  – Coordinate with IEEE P1687 Refresh Working Group

• Enable the work of IEEE P2654
  – Create a P1687.1 solution what will integrate in P2654

• Remain a descriptive standard
  – Do not prescribe the what & how to implementers or users
  – Describe interfaces and access points ← This presentation!
Recap P1687.1

- Access & operation of an IEEE 1687 network through a non-Tap interface
- How to describe the “Transformation Engine”?
- How to interface to *tools*?
Towards the Solution

- P1687.1 must not care!
  - Will not prescribe IP
  - Will not prescribe function

- This is property and responsibility of the owner of the transformation engine

What is in the box?

This could be I2C, SPI, MDIO, ...

External Host Protocol Interface & Controller

Transformation Engine

Functional circuitry

1687 serial Network

1687 Instruments

ScanIn, CaptureEn, ShiftEn, UpdateEn, Select, TCK, Reset, ScanOut

ScanIn, TMS, Select, TCK, TRST, ScanOut
Towards the Solution

- P1687.1 only describes the interfaces to / from the Transformation Engine box
- Follow-up question: How?
- Some detailed questions
  - How to interface to 1687 EDA tool?
  - Who writes the patterns on the left?

This could be I2C, SPI, MDIO, ...

Only standardize IO / API

Functional circuitry
Towards the Solution

1687 AccessLink like construct

Hook of ICL description to design

Container or place of reference for Transformation Engine “code”
- ICL/PDL cannot describe behavior
- Code = User piece of software that transforms the retargeted PDL data to the EHPIC

Requires domain specific language

Only standardize IO / API

Transformation Domain

Retargeting Domain

This could be I2C, SPI, MDIO, ...

External Host Protocol Interface & Controller

Functional circuitry

1687 serial Network

1687 Instruments

ScanIn, CaptureEn, ShiftEn, UpdateEn, Select, TCK, Reset, ScanOut (or)

ICL/PDL cannot describe behavior

Code = User piece of software that transforms the retargeted PDL data to the EHPIC

Requires domain specific language
Domain specific languages are nothing new

- **TMS Domain**
  - Defines the method to navigate the FSM
  - Only certain sequences of TMS values are meaningful

- **Scan Domain**
  - Composed by connecting specific elements like the Boundary Scan Cells

- **Standard document & specific protocol**
  - Serial Vector Format (SVF)
  - Boundary Scan Description Language (BSDL)
Example System

- Two-Pin Serial Port (TPSP) interface
- TPSP controls the TAP
- Bit-banging the interface based on three cycles
  - controlled by SPCLK
  - two cycles are used to push TDI and TMS data to the TAP
  - the third one is used to retrieve TDO from it
Example System

- Transformation does not include TAP

- API to encode
  - Scan-in / out
  - TMS
  - Select
  - TCK
  - TRST
Example System

• Transformation includes TAP

• API to encode
  – Scan-in / out
  – Capture / Shift / Update Enable
  – Select
  – TCK
  – Reset
Example System

1687 like AccessLink

EHPI

Transforming

Retargeting

SPIO

SPIO

TPSP

TAP

CSU

en

tck

tms

tdi

tdo_en

tdo

spio_in

spio_en

spio_out

spio_clk

spio_en

tck

SIB

IJJAG Register

SIB

IJJAG Register

ScanIn

CaptureEn

ShiftEn

UpdateEn

Select

TCK

Reset

ScanOut

Fully retargeted PDL to this IJTAG scan interface
Example System

- An API can present the data to the TPSP transformation
  
- This API can be standardized
  
- It is a small list
  - CSU, PI, PO, PIO
  - Wait, Note, Idle,
  - …

1687 like AccessLink

EHPIC  |  Transforming  |  Retargeting

- spio_in
- spio_en
- spio_out
- spio_clk

- tck
- tms
- tdi
- tdo_en
- tdo

- CSU en
- rst

- SIB
- IJTAG Register
- SIB
- IJTAG Register

Fully retargeted PDL to this IJTAG scan interface
Example System

- This API can be standardized
- P1687.1 is looking into Google’s Protocol Buffers (protobuf) as a universal solution
- Message
  - E.g. CSU event
  - Data

```protobuf
Message CSU_Request {
  bool is_ir = 1;
  string interface_name = 2;
  int32 chain_id = 3;
  int32 length = 4;
  PDLNumber si = 5;
  PDLNumber so = 6;
  bool is_stable = 7;
}

Message IRunLoop_Request {
  int32 cycle_count = 1;
  bool tck = 2;
}

Message iReset_Request {
  bool sync = 1;
}
```
What are protobufs?

• Protobuf = Protocol Buffers
• Google: https://developers.google.com/protocol-buffers
• A language-neutral, platform-neutral, extensible mechanism for serializing structured data (think of XML kind of idea)
• Protocol buffers currently support generated code in Java, Python, Objective-C, and C++
P1687.1 Standardization Opportunities

Transforming

Retargeting

EDA tool support
EDA provided
User provided

Retargeter

ICL
PDL
comp data

Transforming

Retargeting
P1687.1 Standardization Opportunities

Transforming

Retargeting

Retargeter

ICL

PDL

comp data

1. 1687 domain primitive operation grammar: CSU, PI, PO, PIO, Wait, Note, Idle,…,

EDA tool support

EDA provided

User provided
1. 1687 domain primitive operation grammar: CSU, PI, PO, PIO, Wait, Note, Idle,…,

2. RVF
   Messages: Request, Response
   Delegation: to TransferProcs
   Error handling: bad message, bad operation (metadata);

EDA tool support
EDA provided
User provided
4. Translator actions:
- Recognize message types
- Delegate handling of RVF
- Handle errors

5. Transfer Proc structure:
- Written in terms of the primitive operation grammars

1. 1687 domain primitive operation grammar: CSU, PI, PO, PIO, Wait, Note, Idle,…
   {payload of serialized data}

2. RVF
   Messages: Request, Response
   Delegation: to TransferProcs
   Error handling: bad message, bad operation (metadata);

3. Protobuf:
   {ID}{Op (CSU, PI, PO, PIO, Wait, Note, Idle,…)}
   {payload of serialized data}

EDA tool support
EDA provided
User provided
Relocatable Vector Format (RVF)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data/Mask</td>
<td>Binary</td>
<td>Binary representation of the vector to be sent to the SUT</td>
</tr>
<tr>
<td>Callback_idf</td>
<td>String</td>
<td>Unique Callback identifier</td>
</tr>
<tr>
<td>Optional Data</td>
<td>binary</td>
<td>Callback-specific data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Binary</td>
<td>Binary representation of the vector received from the SUT</td>
</tr>
<tr>
<td>Status</td>
<td>Binary/String?</td>
<td>Status information about last Request</td>
</tr>
</tbody>
</table>

- Abstract Information exchanged between translators
- Can be implemented as needed
  - Protobuf for EDA exchange
  - Binary data structure for internal processing
  - …etc…
- Exact content yet TBD
P1687.1 Standardization Opportunities

Provides e.g.
- Debug data / dump / log
- Write user pattern format

iWrite RegA 0x55AA

<table>
<thead>
<tr>
<th>Proc</th>
<th>EDA tool support</th>
<th>EDA provided</th>
<th>User provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compl data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EDA tool support
EDA provided
User provided

Transforming

Retargeting

Procs

RVF

Other

Proto
1. For P2654 alignment

2. Alternative path for pattern
   - Use second Transforming box, just for the pattern syntax
   - Return data to EDA tool
Pattern debug and diagnosis

Transfer Procs must maintain tracking information:
Where every bit comes from / goes to
Summary

• IEEE P1687.1 WG has made great progress identifying what needs to be standardized and how this can be achieved

• Current focus:
  – Run these ideas through detailed examples with focus on what to standardize, what not, and how
  – These examples are planned to be added to the draft
Any Questions?

- Want to join the IEEE P1687.1 Working Group?
- Contact the Secretary at Martin_Keim@Mentor.com