IEEE 1149.1 JTAG
Boundary Scan Standard
Motivation

- Bed-of-nails printed circuit board tester gone
  - We put components on both sides of PCB & replaced DIPs with flat packs to reduce inductance
    - Nails would hit components
  - Reduced spacing between PCB wires
    - Nails would short the wires
  - PCB Tester must be replaced with built-in test delivery system -- JTAG does that
  - Integrate components from different vendors
    - Test bus identical for various components
    - One chip has test hardware for other chips
Purpose of Standard

- Allows test instructions and test data to be serially fed into a component-under-test (CUT)
  - Allows reading out of test results
  - Allows RUNBIST command as an instruction
    - Too many shifts to shift in external tests

- JTAG can operate at chip, PCB, & system levels
- Allows control of tri-state signals during testing
- Allows other chips collect responses from CUT
- Allows system interconnects be tested separately from components
History

- 1985
  - Joint European Test Action Group (JETAG, Philips)

- 1986
  - VHSIC Element-Test & Maintenance (ETM) bus standard (IBM et al.)
  - VHSIC Test & Maintenance (TM) Bus structure (IBM et al.)

- 1988
  - Joint Test Action Group (JTAG) proposed Boundary Scan Standard

- 1990
  - Boundary Scan approved as IEEE Std. 1149.1-1990
  - Boundary Scan Description Language (BSDL) proposed by HP

- 1993
  - 1149.1a-1993 approved to replace 1149.1-1990
- **1994**
  - 1149.1b BSDL approved
- **1995**
  - 1149.5 approved
## Overview of P1149 Family

<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1149.1</td>
<td>Testing of digital chips and interconnections between chips</td>
<td>Std. 1149.1-1990&lt;br&gt;Std. 1149.1a-1993&lt;br&gt;Std. 1149.1b-1994 (BSDL)</td>
</tr>
<tr>
<td>1149.2</td>
<td>Extended Digital Serial Interface</td>
<td>Near completion</td>
</tr>
<tr>
<td>1149.3</td>
<td>Direct Access Testability interface</td>
<td>Discontinue</td>
</tr>
<tr>
<td>1149.4</td>
<td>Mixed-Signal Test Bus</td>
<td>Started Nov. 1991</td>
</tr>
<tr>
<td>1149.5</td>
<td>Standard Module Test and Maintenance (MTM) Bus Protocol</td>
<td>Std. 1149.5-1995</td>
</tr>
<tr>
<td>1149</td>
<td>Unification</td>
<td>Not yet started</td>
</tr>
</tbody>
</table>
Basic Chip Architecture for 1149.1

- **Boundary Scan Cell**
- **I/O Pins**
- **TAP Controller**
- **Internal Logic**
- **Miscellaneous Registers**
- **Instruction Register**
- **Bypass Register**

**Input/Output Pins:**
- TDI (Test Data In)
- TRST* (Test Reset)
- TMS (Test Mode Select)
- TCK (Test Clock)
- TDO (Test Data Out)

**Boundary Scan Path:**
- Boundary Scan Cells connected through I/O Pins.
Boundary Scan Circuitry in a Chip

- Data Registers
- Device-ID Reg.
- BS Register
- Bypass Reg. (1 bit)
- TDO
- TDI
- TMS
- TCK
- TRST*
- TAP
- TAPC
- ClockDR
- ShiftDR
- UpdateDR
- ClockIR
- ShiftIR
- UpdateIR
- Reset*
- IR decode
- Instruction Register
- Select
- TCK
- Enable
Instruction Register Loading with JTAG
Elementary Boundary Scan Cell
Serial Board / MCM Scan

Chip 1
- TMS
- TCK

Chip 2
- TMS
- TCK

Chip 3
- TMS
- TCK

Chip 4
- TMS
- TCK

Boundary-scan cell

Serial test interconnect

System interconnect

TMS Serial data in

TMS Serial data out

TCK TMS
Parallel Board / MCM Scan
Independent Path Board / MCM Scan
Bus Protocol

- Signals
  - TDI: Test Data In
  - TDO: Test Data Out
  - TMS: Test Mode Selection
  - TCK: Test Clock
  - TRST* (optional): Test Reset

- Basic operations
  - Instruction sent (serially) over TDI into instruction register.
  - Selected test circuitry configured to respond to instruction.
  - Test instruction executed.
  - Test results shifted out through TDO; new test data on TDI may be shifted in at the same time.
**Tap Controller Signals**

- **Test Access Port (TAP)** includes these signals:
  - *Test Clock Input (TCK)* -- Clock for test logic
    - Can run at different rate from system clock
  - *Test Mode Select (TMS)* -- Switches system from functional to test mode
  - *Test Data Input (TDI)* -- Accepts serial test data and instructions -- used to shift in vectors or one of many test instructions
  - *Test Data Output (TDO)* -- Serially shifts out test results captured in boundary scan chain (or device ID or other internal registers)
  - *Test Reset (TRST)* -- *Optional* asynchronous TAP controller reset
Tap Controller State Diagram
States of TAP Controller

- Test-Logic-Reset: normal mode
- Run-Test/Idle: wait for internal test such as BIST
- Select-DR-Scan: initiate a data-scan sequence
- Capture-DR: load test data in parallel
- Shift-DR: load test data in series
- Exit1-DR: finish phase-1 shifting of data
- Pause-DR: temporarily hold the scan operation (allow the bus master to reload data)
- Exit2-DR: finish phase-2 shifting of data
- Update-DR: parallel load from associated shift registers
Timing of instruction scan

- **TCK**
- **TMS**
- **Control State**
  - Test-Logic
  - Run-Test/Idle
  - Select-DR-Scan
  - Select-IR-Scan
  - Capture-IR
  - Shift-IR
  - Exit-IR
  - Exit-IR
  - Update-IR
  - Run-Test/Idle
- **TDI**
- **Data input to IR**
- **IR shift-register**
- **Parallel output of IR**
- **IDCODE** = New instruction
- **Data input to TDR**
- **TDR shift-register**
- **Parallel output of TDR**
- **Register selected**
- **Instruction register**
- **TDO enable**
  - Inactive
  - Active
  - Inactive
  - Active
  - Inactive
- **TDO**

= Don't care or undefined
Timing of data scan

- TCK
- TMS
- Control State
  - Run-Test/Idle
  - SelectDR-Scan
  - Capture-DR
  - Shift-DR
  - Pause-DR
  - Exit1-DR
  - Update-DR
  - Exit2-DR
  - Shift-DR
  - Exit1-DR
  - Update-DR
- TDI
- Data input to IR
- IR shift-register
- Parallel output of IR
- Instruction
- IDCODE
- Data input to TDR
- TDR shift-register
- Parallel output of TDR
- Old data
- New data
- Instruction Register
- Test data register
- Inactive
  - Active
  - Inactive
  - Active
  - Inactive
- TDO
- Inactive
- Active
- Inactive
- Active
- Inactive

= Don’t care or undefined
Boundary Scan Instructions
## Optional / Required Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BYPASS</strong></td>
<td>Mandatory</td>
</tr>
<tr>
<td><strong>CLAMP</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>EXTEST</strong></td>
<td>Mandatory</td>
</tr>
<tr>
<td><strong>HIGHZ</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>IDCODE</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>INTEST</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>RUNBIST</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>SAMPLE / PRELOAD</strong></td>
<td>Mandatory</td>
</tr>
<tr>
<td><strong>USERCODE</strong></td>
<td>Optional</td>
</tr>
</tbody>
</table>
SAMPLE / PRELOAD Instruction -- SAMPLE

Purpose:
1. Get snapshot of normal chip output signals
2. Put data on boundary scan chain before next instr.

Data transfer paths activated in SAMPLE mode.
SAMPLE / PRELOAD Instruction -- PRELOAD

INPUT CELL

OUTPUT CELL

Component boundaries

Data transfer paths activated in PRELOAD mode.
EXTEST Instruction

- **Purpose:** Test off-chip circuits and board-level interconnections

Diagram:

- **INPUT CELL**
  - ShiftDR
  - SO
  - Mode
  - DQ \( \rightarrow \) CLK
  - SI \( \rightarrow \) ClockDR \( \rightarrow \) UpdateDR

- **OUTPUT CELL**
  - ShiftDR
  - SO
  - Mode
  - DQ \( \rightarrow \) CLK
  - SI \( \rightarrow \) ClockDR \( \rightarrow \) UpdateDR

Component boundaries:

- Data transfer paths activated in EXTEST mode.
1. Shift-DR (Chip1)

2. Update-DR (Chip1)
3. Capture-DR (Chip2)

4. Shift-DR (Chip2)
**INTEST Instruction**

- **Purpose:**
  1. Shifts external test patterns onto component
  2. External tester shifts component responses out

---

**Diagram:**

- **INPUT CELL**
  - ShiftDR
  - SO
  - Mode
  - SI
  - ClockDR
  - UpdateDR

- **OUTPUT CELL**
  - ShiftDR
  - SO
  - Mode
  - SI
  - ClockDR
  - UpdateDR

**Component boundaries:**
- Used for first phase of INTEST
- Used for second phase of INTEST
- Not used in the INTEST testing mode
INTEST Instruction Clocks

- Control of applied system clock during INTEST

- Use of TCK for on-chip system logic clock
**RUNBIST Instruction**

- **Purpose:** Allows you to issue BIST command to component through JTAG hardware
- **Optional instruction**
- Lets test logic control state of output pins
  1. Can be determined by pin boundary scan cell
  2. Can be forced into high impedance state
- BIST result (success or failure) can be left in boundary scan cell or internal cell
  - Shift out through boundary scan chain
- May leave chip pins in an indeterminate state (reset required before normal operation resumes)
CLAMP Instruction

- Purpose: Forces component output signals to be driven by boundary-scan register
- Bypasses the boundary scan chain by using the one-bit Bypass Register
- Optional instruction
- May have to add RESET hardware to control on-chip logic so that it does not get damaged (by shorting 0’s and 1’s onto an internal bus, etc.)
**IDCODE Instruction**

- Purpose: Connects the component device identification register serially between *TDI* and *TDO*
  - In the *Shift-DR* TAP controller state
- Allows board-level test controller or external tester to read out component ID
- Required whenever a JEDEC identification register is included in the design
### Device ID Register --JEDEC Code

<table>
<thead>
<tr>
<th>MSB</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 28</td>
<td>27 12</td>
<td>11 1</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Version</td>
<td>Part Number</td>
<td>Manufacturer Identity</td>
<td></td>
<td>'1'</td>
<td>(4 bits) (16 bits) (11 bits) (1 bit)</td>
</tr>
</tbody>
</table>
USERCODE Instruction

- **Purpose:** Intended for user-programmable components (FPGA’s, EEPROMs, etc.)
  - Allows external tester to determine user programming of component
- **Selects the** device identification register as serially connected between TDI and TDO
- User-programmable ID code loaded into device identification register
  - On rising TCK edge
- Switches component test hardware to its system function
- Required when Device ID register included on user-programmable component
**HIGHZ Instruction**

- **Purpose:** Puts all component output pin signals into high-impedance state
- **Control chip logic to avoid damage in this mode**
- **May have to reset component after **HIGHZ** runs**
- **Optional instruction**
**BYPASS Instruction**

- Purpose: Bypasses scan chain with 1-bit register
BYPASS

TAP Controller

Internal Logic

Bypass Register (1 bit)

TAP Controller
Test Bus Configuration

Ring configuration

Star configuration
A Printed Circuit Board with 1149.1
(Ring configuration, test controller on board)
Summary

- Boundary Scan Standard has become absolutely essential --
  - No longer possible to test printed circuit boards with *bed-of-nails* tester
  - Not possible to test multi-chip modules at all without it
  - Supports BIST, external testing with Automatic Test Equipment, and boundary scan chain reconfiguration as BIST pattern generator and response compacter
  - Now getting widespread usage
Boundary Scan Description Language (BSDL)

- Now the IEEE 1149.1b standard
- Purposes:
  - To provide a standard description language for boundary scan devices.
  - To simplify the design work for boundary scan---automated synthesis is possible.
  - To promote consistency throughout ASIC designers, device manufacturers, foundries, test developers and ATE manufacturers.
  - For easy incorporation into software tools for test generation, analysis and failure diagnosis.
  - To reduce possibility of human error when employing boundary scan in a design.
Features of BSDL

- BSDL describes the testability features of boundary scan devices which are compatible with 1149.1.
- It's a subset of VHDL.
- Elements of a design which are absolutely mandatory for the 1149.1 and system-logic are not included in the language.
  - Examples: BYPASS register, TAP controller, etc.
- BSDL may be used in a full or in a partial VHDL environment.
Structure of BSDL

- BSDL contains three sections: Entity, Package, Package Body

```
entity demo is
    ...
    ...
    use STD_1149_1_1990.all;
end demo;
```

```
package STD_1149_1_1990 is
    ...
    constant BC_1:CELL_INFO;
end STD_1149_1_1990;
```

```
package body STD_1149_1_1990 is
    ...
    constant BC_1:CELL_INFO:= ...
    ...
end STD_1149_1_1990;
```

- As long as 1149.1 is unchanged, the Package need not be modified.
- If designers create their own packages, they can place complete cell descriptions in associated BSDL Package Body and only list the cell names in the Package.
Identifiers are made up of alpha, numeric, and underscore "_" characters, with the first character being alpha.

The Backus-Naur Format (BNF) is used to describe the syntax.

Case insensitive, free form, single- or multi-line statements, statements terminated with semicolons.

Comments are enclosed between a "--" and an EOL (end of line) character.

For long string a concatenation character "&" is used to break the strings in arbitrary but readable form.
Entity

- Describe a device's I/O ports and attributes
- Format:

```plaintext
entity <device_id> is
    [ generic parameter ]
    [ logical port description ]
    [ usage statement(s) ]
    [ package pin mapping ]
    [ scan port identification ]
    [ TAP description ]
    [ Boundary Register description ]
end <device_id>;

† Note: the order shown above must be followed
```
Generic Parameter

- Used to select a packaging option by name.
- The generic allows an external application to select one package.
- Format:
  ```
  generic(PHYSICAL_PIN_MAP:string:="undefined");
  ```

- In BSDL, the generic is a string with a name PHYSICAL_PIN_MAP.
- The word "undefined" means a default string and it can be pressed in from outside (by an application).
- The default string is arbitrary.
Logical Port Description

- Give meaningful symbolic names to I/O pins of core logic; the names will be used in subsequent description.
- Non_digital pins such as power or ground should be included.

Format:
```
port( <PinID>;<PinID>;...<PinID> );
where
  <PinID>::=<IdentifierList>:<Mode> <PinType>
  <IdentifierList>::=<Identifier>|<IdentifierList>,<Identifier>
  <Mode>::=in|out|inout|buffer|linkage
  <PinType>::=<PinScaler>|<PinVector>
  <PinScaler>::=<Identifier>
  <PinVector>::=<Identifier>(<Range>)
  <Range>::=<number> to <number>|<number> downto <number>
```

Example:
```
port(D:in vector(8 downto 1); CLK:in bit; ENA:in bit; Q_OUT:out
  vector(8 downto 1));
```
Usage Statement(s)

- The statement refers to external definitions found in packages and package bodies.
- The following use statement is mandatory in BSDL:

  ```
  use STD_1149_1_1990.all;
  ```

  - The statement must appear before any other use statement.
  - If a designer invents new cell definitions, he may additionally use `use New_Cell.all`
  - The ".all" suffix means to use all components of the package.
The statement is used to list the package pin mapping (using attribute & constant).

Format:

```plaintext
attribute PIN_MAP of <device_id>:entity is PHYSICAL_PIN_MAP;
custom <package_name>: PIN_MAP_STRING:=<MapString>;
```

Example:

```plaintext
constant DW_PACKAGE:PIN_MAP_STRING:=
"CLK:1,Q(2,3,4,5,7,8,9,10)," &
"D:(23,22,21,20,19,17,16,15)," &
"GND:6,VCC:18,OC_NEG:7," &
"TDO:20,TMS:21,TCK:23,TDI:24";
```
Scan Port Identification

- Define the scan port of the device.
- There are four mandatory and one optional TRST pins.
- Format:
  
  ```
  attribute TAP_SCAN_IN of TDI:signal is true;
  attribute TAP_SCAN_OUT of TDO:signal is true;
  attribute TAP_SCAN_MODE of TMS:signal is true;
  attribute TAP_SCAN_RESET of TRST:signal is true;
  attribute TAP_SCAN_CLOCK of TCK:signal is (freq,state)
  
  --The signal TAP_SCAN_CLOCK has two fields:
  <freq>: the maximum TCK clocking frequency in Hertz
  <state>: one of the two values, BOTH or LOW,
  ```
TAP Description

Define the attributes of instructions

Format:

attribute INSTRUCTION_LENGTH of <device_id>:entity is <integer>;  
attribute INSTRUCTION_CAPTURE of <device_id>:entity is <Pattern>;  
attribute INSTRUCTION_OPCODE of <device_id>:entity is <OpTable>;

Example

attribute INSTRUCTION_LENGTH of demo:entity is 4;  
attribute INSTRUCTION_CAPTURE of demo:entity is "0101";  
attribute INSTRUCTION_OPCODE of demo:entity is "Extest(0000)," &  
"Bypass(1111)," &  
"Sample(1100,1010)," &
The standard requires the EXTEST to have an all-zero opcode and BYPASS an all-one opcode.

SAMPLE/PRELOAD does not have a prescribed opcode bit pattern, so it must be given in a BSDL description.

The character "X" may appear in any bit indicating "don't care" bit positions.
Boundary Register Description

- Format:
  - attribute BOUNDARY_CELLS of <device_id>:entity is <CellList>;
  - attribute BOUNDARY_LENGTH of <device_id>:entity is <integer>;
  - attribute BOUNDARY_REGISTER of <device_id>:entity is <CellTable>;

- Example:
  - attribute BOUNDARY_CELLS of demo:entity is "BC_1";
  - attribute BOUNDARY_LENGTH of demo:entity is 3;
  - attribute BOUNDARY_REGISTER of demo:entity is
    -- num (cell, port, function, safe,
    [ccell, disval, rslt])
    "0  (BC_1,IN,input,X)," &
    "1  (BC_1,*,control,0)," &
    "2  (BC_1,D(1) input X)" ;
Boundary Register Description (cont.)

- **function**: The cell function is identified from an enumeration consisting of the symbols input, clock, output2, output3, internal, control, controlr and bdir.

- **safe**: May be 0, 1 or X. This subfield specifies what an Update flip-flop should be loaded with when software might otherwise choose a value at random.
Package

- A collection of declarations to describe 1149.1 standard information, or user-specified design information.

- The definitions related to Std. 1149.1 may come from a pre-written standard package, and is only expected to change when the standard itself changes.

- For a user-specified package, the name of cells created by designers should be listed using the following format:

```plaintext
package New_Cells is
    constant NC_1:CELL_INFO;
    constant NC_2:CELL_INFO;
end NEW_CELL;
```

where `New_Cells` must appear in a Usage statement of the Entity description

    and all cells must be named in the BOUNDARY CELLS attribute string.
Package

A typical example of the 1149.1 standard package:

```pascal
package STD_1149_1_1990 is
    attribute PIN_MAP:string;
    subtype PIN_MAP_STRING is string;

    type CLOCK_LEVEL is (LOW, BOTH);
    type CLOCK_INFO is record
        FREQ:real;
        LEVEL:CLOCK_LEVEL;
    end record;

    attribute TAP_SCAN_IN: boolean;
    attribute TAP_SCAN_OUT:boolean;
    attribute TAP_SCAN_CLOCK:CLOCK_INFO;
    attribute TAP_SCAN_MODE:boolean;
    attribute TAP_SCAN_RESET:boolean;
```
attribute INSTRUCTION_OPCODE:string;
attribute INSTRUCTION_CAPTURE:string;
attribute INSTRUCTION_DISABLE:string;
attribute INSTRUCTION_PRIVATE:string;
attribute INSTRUCTION_USAGE:string;

ID_BITS is ('0', '1', 'X');
type ID_STRING is array (31 downto 0) of ID_BIT;
attribute IDCODE_REGISTER:ID_STRING;
attribute USERCODE_REGISTER:ID_STRING;

attribute REGISTER_ACCESS:string

BSCAN_INST is (EXTEST, SAMPLE, INTEST, RUNBIST);
CELL_TYPE is (INPUT, INTERNAL, CLOCK, CONTROL, OUTPUT2, OUTPUT3, BIDIR, IN
type CELL_DATA is record
   CT:CELL_TYPE;
   I:BSCAN_INST;
   CD:CAP_DATA;
end record;

type CELL_INFO is array(positive range<> ) of CELL_DATA;
constant BC_1:CELL_INFO;
constant BC_2:CELL_INFO;
constant BC_3:CELL_INFO;
constant BC_4:CELL_INFO;
constant BC_5:CELL_INFO;
constant BC_6:CELL_INFO;

attribute BOUNDARY_CELLS:string;
attribute BOUNDARY_LENGTH:integer;
attribute BOUNDARY_REGISTER:string;
Describe the operations of boundary scan cell defined in package

Format

```plaintext
package body STD_1149_1_1990 is
  constant BC_1:CELL_INFO:=(
    (INPUT,EXTEST,PI), (OUTPUT2,EXTEST,PI),
    (INPUT,SAMPLE,PI), (OUTPUT2,SAMPLE,PI),
    (INPUT,INTEST,PI), (OUTPUT2,INTEST,PI),
    (INPUT,RUNBIST,PI), (OUTPUT2,RUNBIST,PI),
    (OUTPUT3,EXTEST,PI), (INTERNAL,EXTEST,PI),
    (OUTPUT3,SAMPLE,PI), (INTERNAL,SAMPLE,PI),
    (OUTPUT3,INTEST,PI), (INTERNAL,INTEST,PI),
    (OUTPUT3,RUNBIST,PI), (INTERNAL,RUNBIST,PI),
    (CONTROL,EXTEST,PI), (CONTROL,EXTEST,PI),
    (CONTROL,SAMPLE,PI), (CONTROL,SAMPLE,PI),
    (CONTROL,INTEST,PI), (CONTROL,INTEST,PI),
    (CONTROL,RUNBIST,PI), (CONTROL,RUNBIST,PI),
  )

  constant BC_2:CELL_INFO:=....... 
end STD_1149_1_1990
```
A Complete Example
entity demo is
    generic(PHYSICAL_PIN_MAP:string:="UNDEFINED");
    port(CLK:in,bit;Q:out,bit_vector(1 to 6);D:in,bit_vector(1 to 6);
         GND,VCC:linkage,bit;TDO:out,bit;TMS,TCK,TDI:in,bit);
    use STD_1149_1_1990.all;
    attribute PIN_MAP of demo:entity is PHYSICAL_PIN_MAP;
    constant DW_PACKAGE:PIN_MAP_STRING:="CLK:1," &
        "Q(6,7,8,9,10,11),D(12,13,14,15,16,17),GND:18,VCC:19," &
        "TDO:5,TMS:4,TCK:3,TDI:2";
    attribute TAP_SCAN_IN of TDI:signal is true;
    attribute TAP_SCAN_MODE of TMS:signal is true;
    attribute TAP_SCAN_OUT of TDO:signal is true;
    attribute TAP_SCAN_CLOCK of TCK:signal is (20e6,BOTH);
    attribute INSTRUCTION_LENGTH of demo:entity is 4;
    attribute INSTRUCTION_OPCODE of demo:entity is
        "BYPASS (11111)," &
        "EXTEST(0000)," &
        "SAMPLE(1100,1010)," &
        "INTEST(1010)";
attribute INSTRUCTION_CAPTURE of demo/entity is "0101";
attribute BOUNDARY_CELLS of demo/entity is "BC_1";
attribute BOUNDARY_LENGTH of demo/entity is 12;
attribute BOUNDARY_REGISTER of demo/entity is

-- num cell port function safe [ccell disval rslt]
"12 (BC_1,CLK,input,X)," &
"11 (BC_1,D(1),input,X)," &
"10 (BC_1,D(2),input,X)," &
"9 (BC_1,D(3),input,X)," &
"8 (BC_1,D(4),input,X)," &
"7 (BC_1,D(5),input,X)," &
"6 (BC_1,D(6),input,X)," &
"5 (BC_1,Q(1),output3,X,000,1,Z)," &
"4 (BC_1,Q(2),output3,X,000,1,Z)," &
"3 (BC_1,Q(3),output3,X,000,1,Z)," &
"2 (BC_1,Q(4),output3,X,005,1,Z)," &
"1 (BC_1,Q(5),output3,X,005,1,Z)," &
"0 (BC_1,Q(6),output3,X,005,1,Z);" 

end demo;
package STD_1149_1_1990 is
attribute PIN_MAP:string;
subtype PIN_MAP_STRING is string;

type CLOCK_LEVEL is (LOW, BOTH);
type CLOCK_INFO is record
  FREQ:real;
  LEVEL:CLOCK_LEVEL;
end record;

attribute TAP_SCAN_IN: boolean;
attribute TAP_SCAN_OUT: boolean;
attribute TAP_SCAN_CLOCK:CLOCK_INFO;
attribute TAP_SCAN_MODE: boolean;
attribute TAP_SCAN_RESET: boolean;

attribute INSTRUCTION_LENGTH: integer;
attribute INSTRUCTION_OPCODE: string;
attribute INSTRUCTION_CAPTURE: string;
type ID_BITS is ('0', '1', 'X');
type ID_STRING is array (31 downto 0) of ID_BIT

attribute REGISTER_ACCESS:string

type BSCAN_INST is (EXTEST, SAMPLE, INTEST, RUNBIST);
type CELL_TYPE is (INPUT, INTERNAL, CLOCK, CONTROL, OUTPUT2, OUTPUT3, BIDIR_IN, BIDIR_OUT);
type CAP_DATA is (PI, PO, UPD, CAP, X, ZRRO, ONE);
type CELL_DATA is record
  CT:CELL_TYPE;
  I:BSCAN_INST;
  CD:CAP_DATA;
end record;
type CELL_INFO is array(positive range<>) of CELL_DATA;
constant BC_1:CELL_INFO;

attribute BOUNDARY_CELLS:string;
attribute BOUNDARY_LENGTH:integer;
attribute BOUNDARY_REGISTER:string;
attribute DESIGN WARING:string;
constant BC_1:CELL_INFO:=(
  (INPUT,EXTEST,PI), (OUTPUT2,EXTEST,PI),
  (INPUT,SAMPLE,PI), (OUTPUT2,SAMPLE,PI),
  (INPUT,INTEST,PI). (OUTPUT2,INTEST,PI),
  (OUTPUT3,EXTEST,PI),
  (INTERNAL,EXTEST,PI),
  (OUTPUT3, SAMPLE,PI),
  (INTERNAL,SAMPLE,PI),
  (OUTPUT3, INTEST,PI), (INTERNAL,INTEST,PI),
  (CONTROL,EXTEST,PI),(CONTROL,EXTEST,PI),
  (CONTROL,SAMPLE,PI),(CONTROL,SAMPLE,PI),
  (CONTROL,INTEST,PI),(CONTROL,INTEST,PI)
);