The Protocol Aware (PA) Problem

• Highly integrated SOC devices, used in everyday products, communicate with each other using protocols – defined structures with a common sequence which can be non deterministic in terms of length, delays and content.

• To replicate the true mode of operation of a given device, ATE needs to be able to communicate with the DUT using specific protocols and to stress the device in “real life” conditions...

• ATE needs to understand and speak the protocol and handle non-determinism (if any)
What is Driving ATE Change?

- Highly integrated, sub-100nm SOC are becoming increasingly difficult to test...
  - Designers are implementing on-chip BIST techniques to overcome challenges with timing, delays and effects of temperature.

- Current ATE generally expects that engineers work with low level flat-pattern based setups to access device registers.

- The next generation of ATE solutions must...
  - Enable engineers to access the device registers via standard interfaces and protocols.
  - Provide easy to use tools, similar to the design environment, working at a higher abstraction level rather than using patterns.
  - Emulate “chip to chip” communication and handle non-determinism
  - Address the requirements to validate the electrical specifications of all external connections from the device.
ATE Change: Motivation

• Lack of harmony between ATE and bench
  – Enable full device characterization on ATE from first silicon

• My personal goal: a designer can walk up to ATE and make changes to the setup with almost no instruction

• ATE needs to be “smarter” and not rely on fixed timing pattern based simulations
  – Focus on payload and programming at a higher level of abstraction... not 1’s and 0’s
  – Handle non-determinism to detect faults and work within the boundaries of many protocols

• Take advantage of the broad range of resources available on ATE to stress devices
  – Emulation of real device environment ➔ system like test to cover structural test gaps
Goal of PA: Bench Utilization Model

- Communication Protocol Definition provided as a library.
  - Device control predefined setups which incorporate:
    - Timing
    - Levels
    - Frame Definition
    - Synchronization processing
- Engineer provides transaction information
  - Must still understand messaging component of protocol
    - i.e. write, read, idle
  - Provide sequence of commands, addresses and data
Structural Test Gaps → System-Like Test

- Coverage gaps of structural test must be filled by tests under system-like conditions.
  - Synchronization logic between today’s many clock domains of complex SOCs is not well covered structurally.
  - Subtle path delay faults in 65 nm must be tested under controlled voltage, frequency and thermal stress conditions.
  - Support for power management with adaptive voltage and frequency is not covered by structural test.
- Dedicated system test stations are not generic and not worst case.

- ATE should support system-like DUT activity under realistic and controlled voltage, frequency, timing, and thermal stress conditions.
Protocol Aware ATE
Easier test program generation and better test coverage

Tolerate Non-Determinism
Detect Payload
PRBS Gen & Check
Parametric loopback
Clock-Per-Pin

DUT

Mem Ctrl
CPU
PCle
JTAG
USB

Digital I/O Cards

Digital I/O
Card with
Memory
Emulation

Digital I/O Cards

Digital I/O
Cards

Digital I/O Cards

Digital I/O
Cards

DPS

PA SW

Transactions

Protocol Based DUT Access
- write register
- read register

AWG

DTZ

ATE
Protocol Based DUT Access - Key Values

- Protocol transaction level communication with the chip CPU and BIST IP.
  - The user deals with logical transactions (e.g. write register, read register) instead of thousands of ‘unreadable’ vectors
  - Reduced need for chip-wide simulations for test generation
  - Abstraction to protocol transaction level significantly simplifies and accelerates test program development and debug
Motivation: Typical Debug Scenario with an ATE Pattern

**How to change just one register value?**

**With ordinary flat vectors:**
register write transactions are buried in a long pattern
...how to locate them?
...how to change them?
Typical result: I need new pattern!

But re-simulation is a lengthy process...

**On transaction level it is easy:**
write (JTAG, 0x1a72, 0x12)
-> write (JTAG, 0x1a72, 0x13)
## PA Essentials

### Protocol Definition

<table>
<thead>
<tr>
<th>Signal</th>
<th>tck (SC)</th>
<th>tdi (SC)</th>
<th>tdo (SC)</th>
<th>trst (SC)</th>
<th>tAG_port (TA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Mode Area</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td>JTAG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Protocol Debug

```cpp
// C++ code

void testProtocol() {
    // Protocol transactions
    // Protocol data
    // Protocol debug
}
```

![Protocol Debug Graph](image-url)
The protocol display in the Protocol Editor matches the presentation in spec sheets:

- easy to understand
- easy to compare
- easy to adapt

Example: MDIO 2-wire Protocol
Direct modification of all protocol elements

Example: change MSB/LSB order of a 16 bit data word

data[15..0] → data[0..15]
Interactive Protocol Level Debugging

- Pattern tool supports side by side display of transactions and bit-level vectors. It also shows result data after execution.
- Values can be changed interactively in both representations → fast debug cycle without need to recompile test methods
- Normal patterns that contain protocol transactions can be viewed and debugged on protocol level
Benefits of Software-Based PA

- Runs on all digital channel cards (current and next generation)
- Fully flexible resource assignment
  - each digital channel can take any protocol’s pin role
- Supports proprietary protocols with any number of channels and any channel type needed
- No restrictions for multi-site count
- Multi-purpose pins can run different protocols during a test flow without added time for protocol switching
- Execution of transactions with best HW performance equal to plain pattern execution
- In concurrent testing each IP block can “talk” its own protocol
- Full set of digital channel capabilities available for PA (timing/level spec changes, tolerate non-determinism, ...)

PA Introduction
Slide 14
Hardware PA: Non-Determinism

- Under realistic test conditions, complex SOCs exhibit non-deterministic behavior.
  - Asynchronous clock domains, delay drift (thermal)
  - Adaptive voltage and frequency for power management
  - Fractional-N PLL, no synchronous reset in HSIO

- Functional vectors become difficult to setup on deterministic ATE.
  - Lengthy re-simulations → TTM
  - Error masking → Coverage
  - Timing and pattern searches → Test time, cost of test

- ATE must tolerate non-deterministic DUT response.
  - Non-repeatable timing
  - Timing drift
  - Non-deterministic start-up latency
  - Non-deterministic insertion of idle or maintenance packets
PCIe Functional Test Challenges

• **Non-deterministic behavior**
  - Training sequences required to initiate links
    - Example: Non-deterministic transmission start times

 Lane 0
 Training-Sequence  Training-Sequence  Training-Sequence

 Lane 1
 Training-Sequence  Training-Sequence  Training-Sequence

 Lane 2
 Training-Sequence  Training-Sequence  Training-Sequence

$T_0 = \text{Power-up}$

- Lanes aligned based on detectable sequence
- Tester needs to align and handshake

$1100000101$
PCIe Functional Test Challenges

- Asynchronous events:
  - skip ordered set inserted in Tx data stream at an interval between 1180 and 1538 symbol times (non-deterministically). Allows PPM offsets between points
  - skip ordered set is inserted between packets and other ordered sets
Tolerating Non-Deterministic Response in Hardware

- Source Synchronous Busses
- Serial Protocols
- Per Pin
- At-speed HW

Clock forwarding from clock channel

Deterministic payload packets with non-deterministic arrival times

Expected
Packet 1 | Packet 2 | xxxx | xxxx | xxxx | Packet 3

Actual
Packet 1 | Idle   | Packet 2 | Idle | Packet 3 | Idle

Signature of inter-payload duration

Optional Token size 8, 9, 10, 12, 14,…
Memory Emulation

- Emulates missing memory to enable system-like cooperative test, including booting from ATE.

- 64 channel card can switch to Memory Emulation mode

- Can emulate one memory (DDR2, DDR3, LPDDR1/2, Flash, ...)

- Unused channels can be used in vector mode

- 1 MByte size of addressable emulated memory

- >400 MByte of sequentially accessed memory, e.g. for booting

- CAS latency of up to 38 clocks, must be tolerated by DfT in DUT memory controller

- Source synchronous timing in emulation mode
  - DQS to DQ input and output skew with 30~60 ps resolution
  - Timing calibration invisible to user

- Logic Analyzer trace capture during emulation
Summary

• Protocol Aware ATE
  - Significantly reduces time-to-market
  - Enables modern test methodologies for lower cost-of-test

• Protocol Aware is an integral part of the 93k platform
  - Fully integrated, highly flexible PA SW solution for all current and next Generation digital channel cards
  - Hardware handles non-determinism

• and it’s here now...
Thank You