sciPROVE

sci-worx' C++ Based Verification Environment for IP and SoC Design

Richard Hoffer
richard-peter.hoffer@sci-worx.com
www.sci-worx.com

sciworx

Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

Motivation of Transaction-based Verification

- Person hours spent on verification ≈ 2 x design effort
- Reduction of verification effort through re-use
  - Requires company standard for verification
- More productivity gain through transaction-based verification
  - Allows debugging at a higher level of abstraction resulting in easier and more efficient debugging
  - On-the-fly result checking reduces time to find errors with simulation

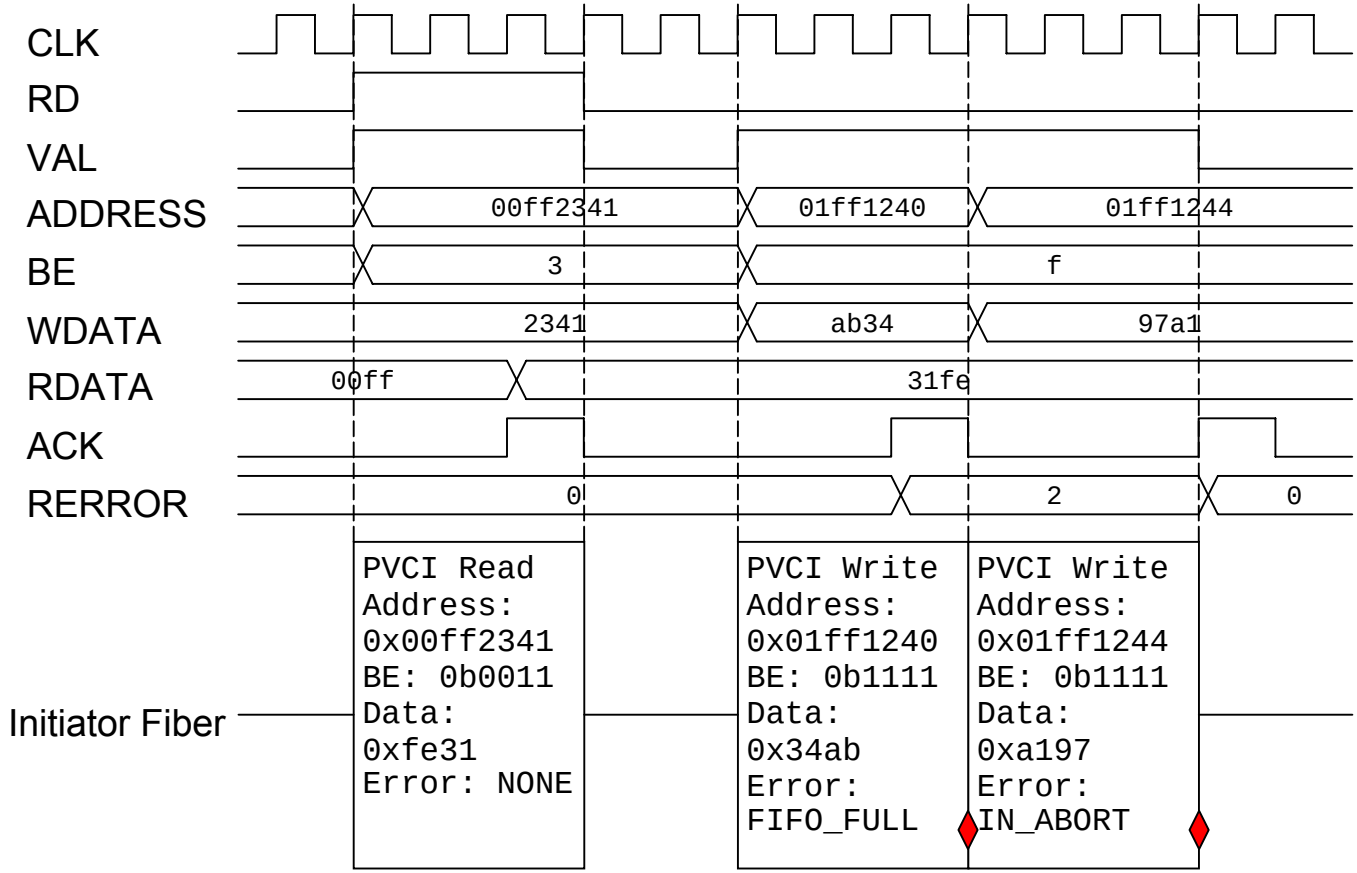
sciPROVE

Transaction-Based Verification

- What is transaction-based verification -
  or better, what is it not?
  - debugging at signal level
  - stimulating single signals or busses directly
- So what is it now?
  - debugging at protocol level
  - considering related signals together
  - using functions to create stimulus
  - using recorded transactions and their attributes for debugging

Debugging using Transactions

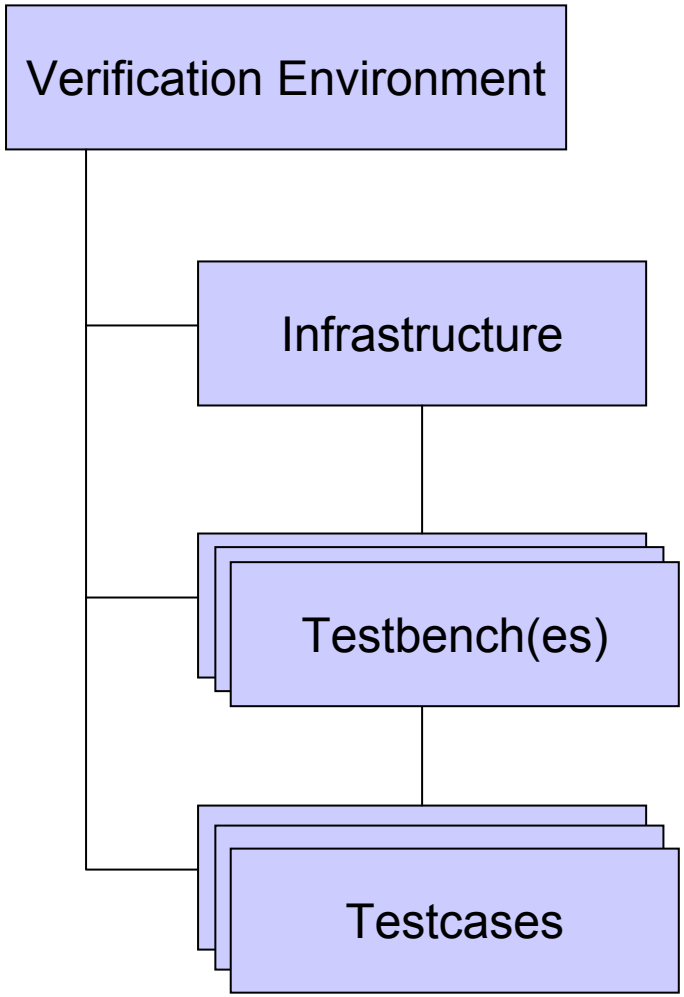
- What is easier to debug in terms of data mismatches: signals and busses, or transactions?



Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

sciPROVE Overview



- Testbuilder by Cadence
  - connection to HDL simulator via Transaction Verification modules (TVM)
  - transaction mechanisms
- Testbench Elements (TBEs)
  - reusable building blocks
  - standard interface protocols
- Test Description Language (TDL)
  - testcase description

sciPROVE Building Blocks

- TVM: Transaction Verification Module
  - introduced by TestBuilder
  - interface from C++ testbench to HDL testbench and DUV
  - contains methods/tasks to perform transactions on interfaces
- TBE: Testbench Element
  - introduced by **sciPROVE**
  - testbench building block containing
    - command interface to manager
    - container-based interface to other TBEs
  - may contain a TVM to interface to HDL

sciPROVE Test Description Language

- TDL: Testcase Description Language
  - TDL scripts for description of testcases
    - Script controlled tests
    - powerful script language featuring loops, conditionals, mathematical and logical expressions, subroutines
    - sequential and parallel threads of execution
    - synchronization facilities
  - TDL script parser and scheduler: part of sciPROVE

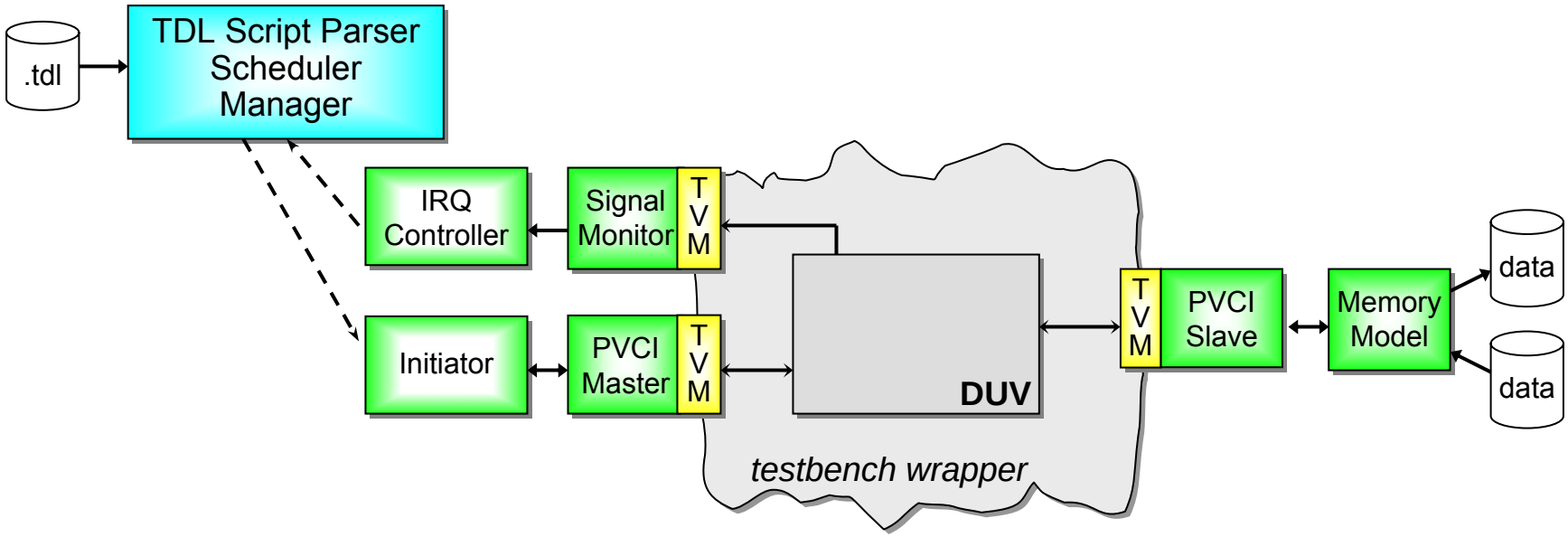
=> No C/C++ know-how necessary for writing test cases

Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

Testbench Structure

- interface specific TBEs
  - PVCI Initiator and Target
- **sciPROVE** base classes
- TBEs from **sciPROVE** library
  - Initiator TBE
  - Memory Model
  - Signal Monitor
  - IRQ controller



Testcase Construction

- tbv_main()
  - references to TVMs
  - TBE connections
  - registration of TBEs to SVE manager
- HDL toplevel
  - call of tbv_main()
  - TVM instances
  - DUV instance

=> can be generated by script

Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

Testcase Description Language

- TDL uses best of Perl and C
  - no variable declaration necessary

```
rdata = -1;
read(DWORD, ADDR, rdata);
```

  - provides arrays and hashes

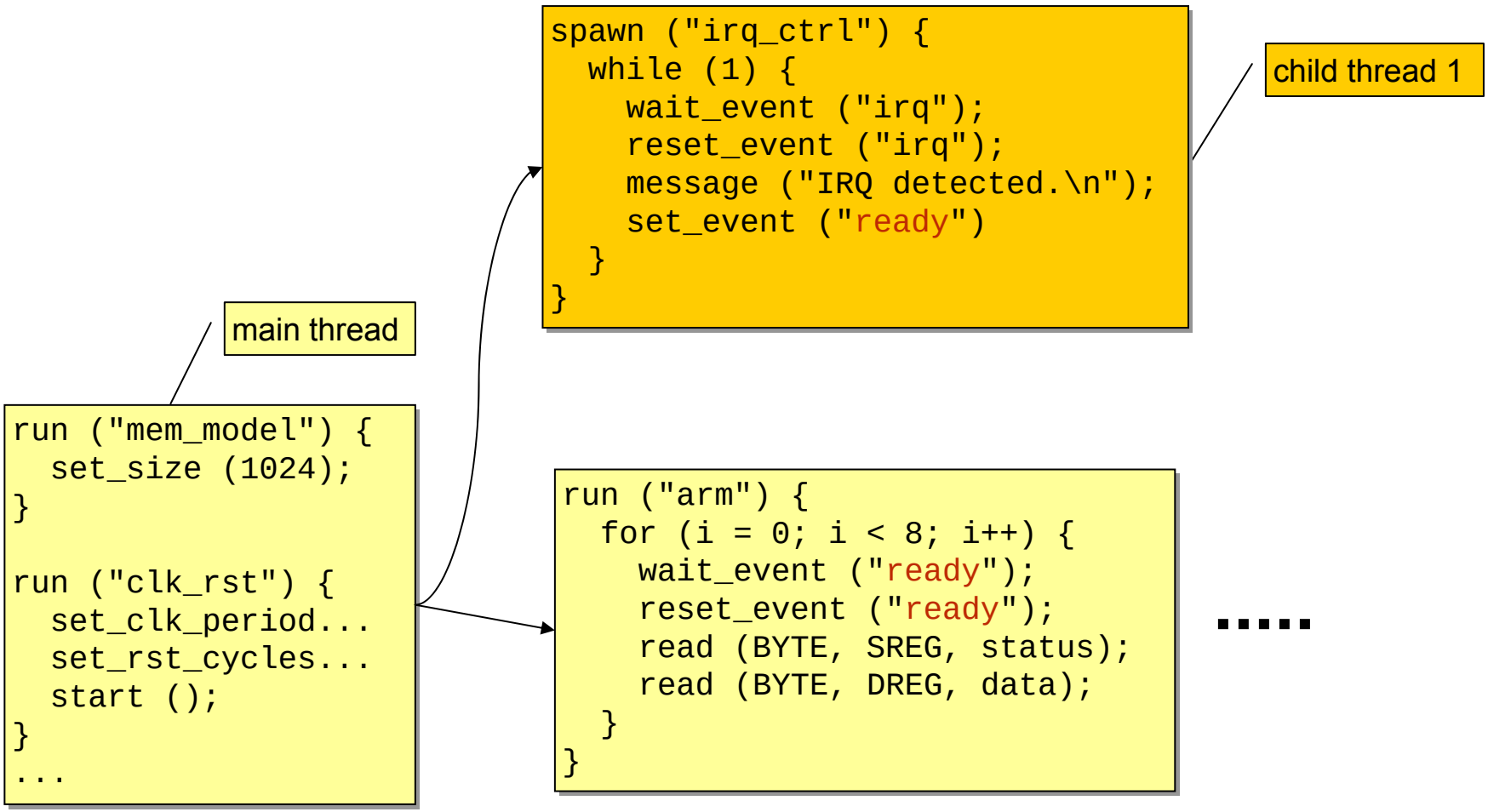
```
data0[0] = 123;                                  // 1 dim.
data1    = {0, 1, {10, 11}, ‘b0100, 0xf3};  // 2 dim.
data2{"name"} = "testcase 2";
```

  - C-style parameters in subroutines

```
sub my_write (addr, data) {...}
```

=> comfortable language for test scripts

Threads for Interaction with Hardware



sciworx

Description of Concurrency in TDL

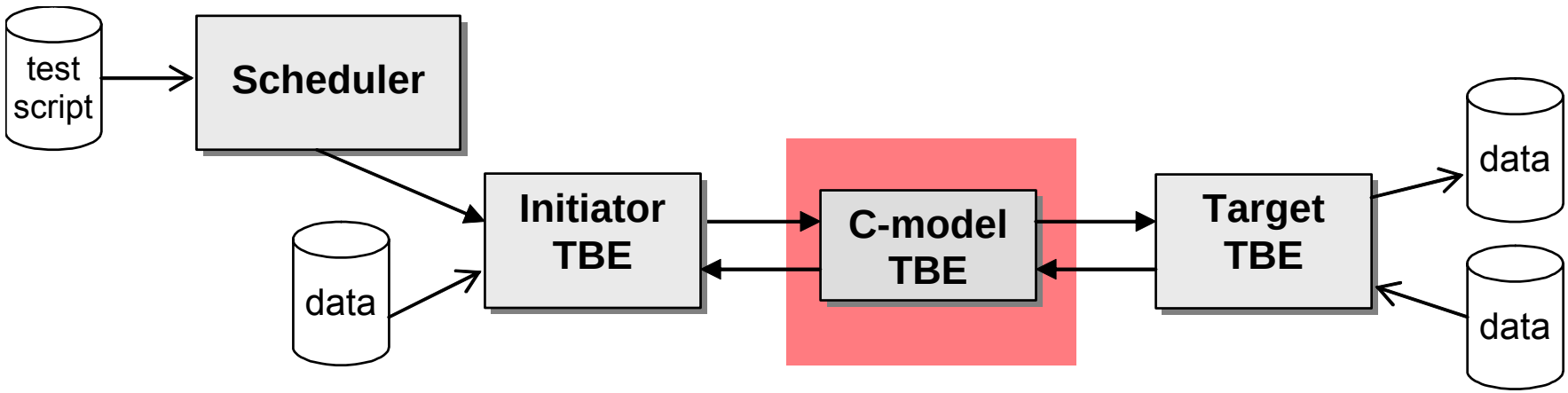
- Threads are used to run commands concurrently
- Threads can be synchronized using events or semaphores
- Scopes set the TBE environment for command execution
  - e.g. for output functions the message logger of the TBE selected by the scope is used
  - *run* statements continue executing in the same thread, using the defined TBE scope
  - *spawn* statements create a new thread for execution, using the defined TBE scope
- Test script execution is finished after exit() or if all threads are finished

Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

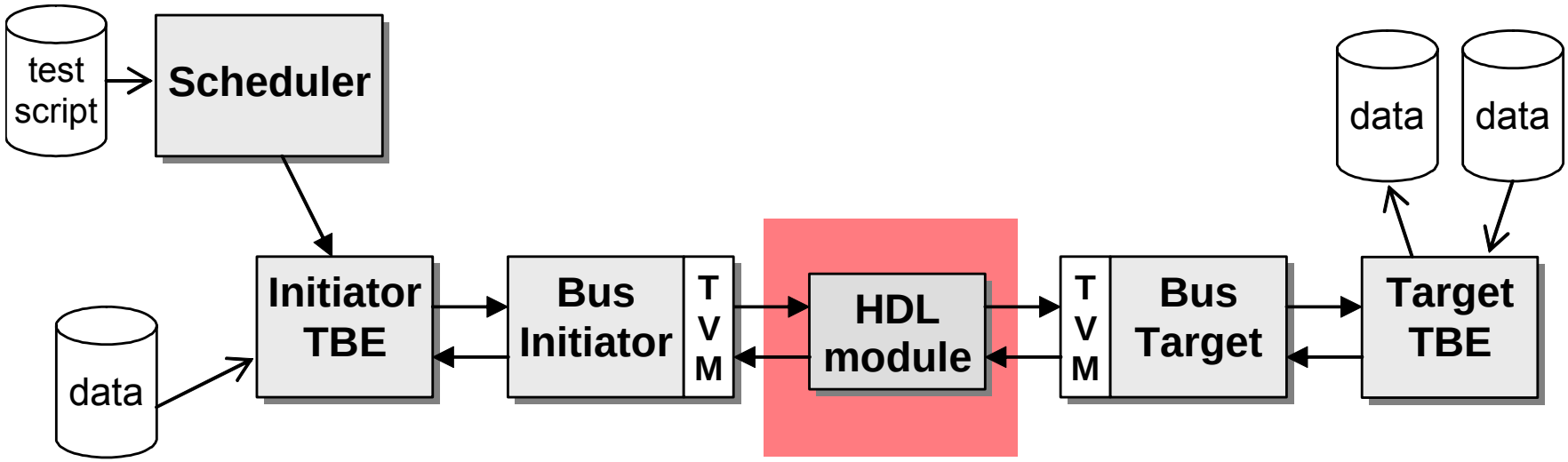
Use Case: C Model Development

- IP design, sub module design



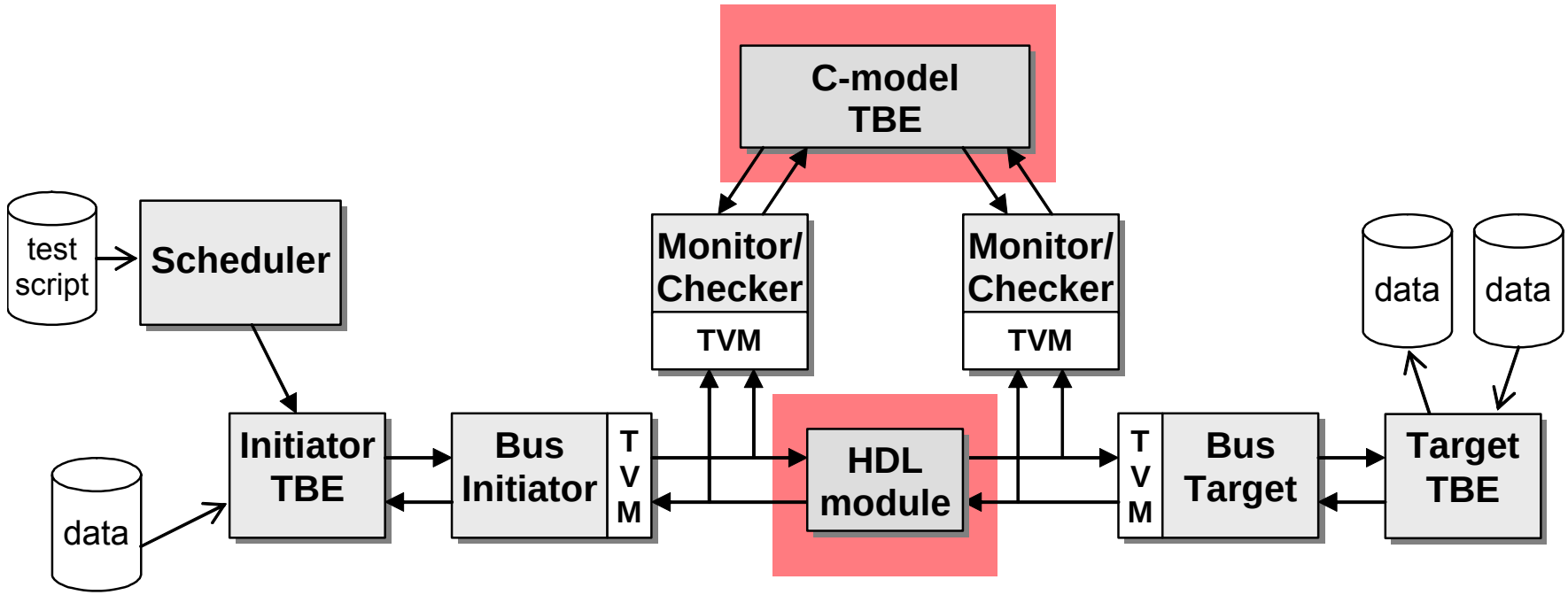
Use Case: HDL Development

- IP design, sub module design



Use Case: HDL Verification against C-model

- IP design, sub module design



sciworx

Outline

- Transaction-Based Verification
- **sciPROVE** Concept
- Testbench Structure
- Testcase Description Language
- Use Cases
- Conclusion

Conclusion

- Company wide verification environment
  - part of sci-worx' design flow
- Support for many verification scenarios
- Language independent testcase description
  - use of TDL for testcase description
- Efficient verification
  - TBE reuse
  - higher level of abstraction (transactions)
- Easy provision to customers
  - only TestBuilder is required
- Proven in several SoC design projects